

University Course

MATH 127
Mathematical and Computational
Methods in Molecular Biology

UC BERKELEY
Fall 2002

My Class Notes
Nasser M. Abbasi

Fall 2002

Contents

1	Introduction	1
1.1	A bit about UC Berkeley	1
1.2	Course description	2
2	Study notes	3
2.1	collected notes	4
2.2	FFT project	66
2.3	Small note on terfoil_combinations	71
3	HWs	77
3.1	HW 1	78
3.2	HW 2	105
3.3	HW 3	138
3.4	HW 4	177
3.5	HW 5	203

Chapter 1

Introduction

Local contents

1.1 A bit about UC Berkeley	1
1.2 Course description	2

I took Mathematics 127, Mathematical and Computational Methods in Molecular Biology at UC Berkeley in fall 2002 while I was still working at AppliedBiosystems. Used to have to drive from San mateo to Hayward, then take Bart to get to Berkeley. I took 1.5 hrs just to get to Berkeley one way.

Instructor: Name:	Lior Pachter
Position:	Assistant Professor
E-mail:	lpachter@math.berkeley.eduTo reduce spam, this address is javascript encoded.
Phone:	+1 (510) 642-2028
Office:	1081 Evans Hall
Research:	Applications of statistics and combinatorics to problems in biology

Homepage: <http://www.math.berkeley.edu/~lpachter>

1.1 A bit about UC Berkeley

This below is a picture of Evans hall. It is a big tall building full of very smart people. The math department is on the 9th floor. The course was in room 3, which is on the ground floor on Evans hall



1.2 Course description

description Introduction to mathematical and computational problems arising in the context of molecular biology. Theory and applications of combinatorics, probability, statistics, geometry, and topology to problems ranging from sequence determination to structure analysis. Units 3 (Semester system) University UC Berkeley, CA..

Chapter 2

Study notes

Local contents

2.1	collected notes	4
2.2	FFT project	66
2.3	Small note on terfoil_combinations	71

2.1 collected notes

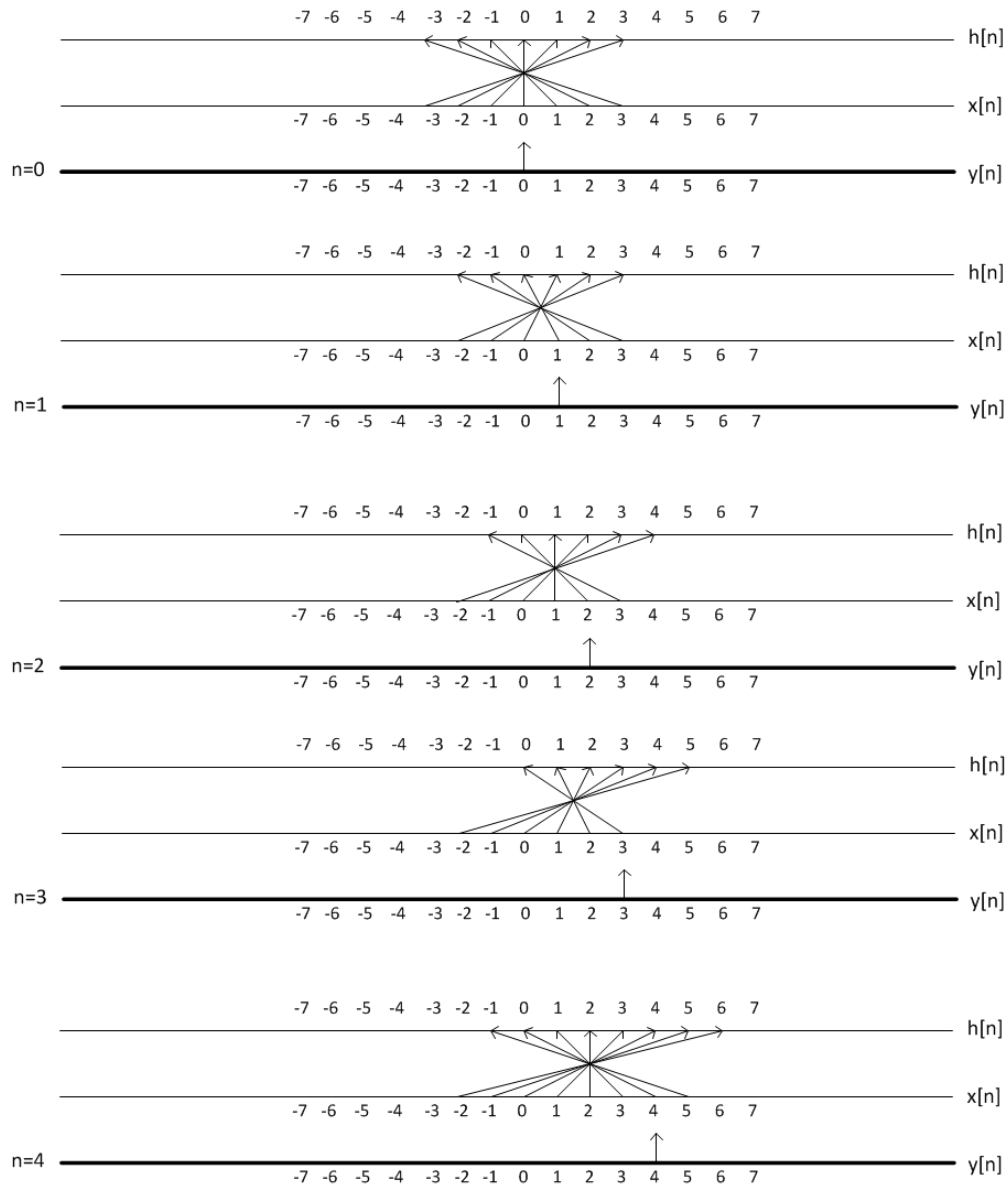
[up](#)

Study notes for Math 127.

Mostly collected from the net by Nasser M. Abbasi

These are misc. class notes wrote during taking math 127 at UC Berkeley.

Convolution diagram I did (need to check this)



Graphical representation of the convolution sum. 'y' is the response of the system. 'x' is the input, and 'h' is the response of the system to a unit impulse

$$y[n] = \sum_{k=-\infty}^{k=+\infty} x[k]h[n-k]$$

Nasser Abbasi
oct 4, 2002.
convolution.vsd

From <http://mathworld.wolfram.com/LinkingNumber.html>

Formally, a link is one or more disjointly embedded [circles](#) in three-space. More informally, a link is an assembly of [knots](#) with mutual entanglements.

A [link invariant](#) defined for a two-component oriented [link](#) as the sum of ⁺¹ crossings and -1 crossing over all crossings between the two links divided by 2.

For components α and β ,

$$Lk(\alpha, \beta) \equiv \frac{1}{2} \sum_{p \in \alpha \cap \beta} \epsilon(p),$$

where $\alpha \cap \beta$ is the set of crossings of α with β , and $\epsilon(p)$ is the sign of the crossing. The linking number of a splittable two-component link is always 0.

[Calugareanu Theorem](#), [Gauss Integral](#), [Jones Polynomial](#), [Link](#), [Twist](#), [Writhe](#)

SEE ALSO:

The twist of a ribbon measures how much it twists around its axis and is defined as the integral of the incremental twist around the ribbon. A formula for the twist is given by

$$\text{Tw}(K) = \frac{1}{2\pi} \int_K ds \varepsilon_{\mu\nu\alpha} \frac{dx^\mu}{ds} n^\nu \frac{dn^\alpha}{ds}, \quad (1)$$

where K is parameterized by $x^\mu(s)$ for $0 \leq s \leq L$ along the length of the knot by parameter s , and the [frame](#) K_f associated with K is

$$y^\mu = x^\mu(s) + \varepsilon n^\mu(s), \quad (2)$$

where ε is a small parameter and $n^\mu(s)$ is a unit [vector field](#) normal to the curve at s (Kaul 1999).

Letting Lk be the linking number of the two components of a ribbon, Tw be the twist, and Wr be the [writhe](#), then the [calugareanu theorem](#) states that

$$Lk(R) = Tw(R) + Wr(R) \quad (3)$$

(Adams 1994, p. 187).

Calugareanu Theorem

SEE ALSO: Letting Lk be the [linking number](#) of the two components of a ribbon, Tw be the [twist](#), and Wr be the [writhe](#), then

$$Lk(K) = Tw(K) + Wr(K).$$

(Adams 1994, p. 187).

[Gauss Integral](#), [Linking Number](#), [Twist](#), [Writhe](#)

Gauss Integral

Consider two closed oriented [space curves](#) $f_1 : C_1 \rightarrow \mathbb{R}^3$ and $f_2 : C_2 \rightarrow \mathbb{R}^3$, where C_1 and C_2 are distinct [circles](#), f_1 and f_2 are differentiable C^1 functions, and $f_1(C_1)$ and $f_2(C_2)$ are disjoint loci. Let $Lk(f_1, f_2)$ be the [linking number](#) of the two curves, then the Gauss integral is

$$Lk(f_1, f_2) = \frac{1}{4\pi} \int_{C_1 \times C_2} dS.$$

From the net

Here are some references having to do with DNA and Differential Geometry that may be of interest here.

DNA and Differential Geometry, William Pohl in Mathematical Intelligencer (ca. 1991)

Pohl, W. F. "Some Integral Formulas for Space Curves and their Generalization",
Amer. J. of Math., 90, 1321-1345 (1968)

Pohl, W. F. "The Self Linking Number of a Closed Space Curve",
J. of Math. and Mech. 17, 975-986 (1968)

White, J. H. "Self Linking and the Gauss Integral in Higher Dimensions",
Amer. J. of Math. 91, 693-728 (1969)

Fuller, F. B. "The Writhing Number of a Space Curve",
Proc. Natl. Acad. Sci USA, 68, 815-819 (1971)

Fuller, F. B. "Decomposition of the Linking Number of a Closed Ribbon:
A Problem from Molecular biology", Proc. Natl. Acad. Sci USA,
75, 3557 (1978)

Some of these are pretty heavy reading, but some are fairly accessible.

-- Jeff Horn

Calculation of link number

The number of times the two strands of DNA double helix are interwound, i.e., the link number Lk , is a topologic invariant quantity for closed DNA

When is a DNA strand considered supercoiled?

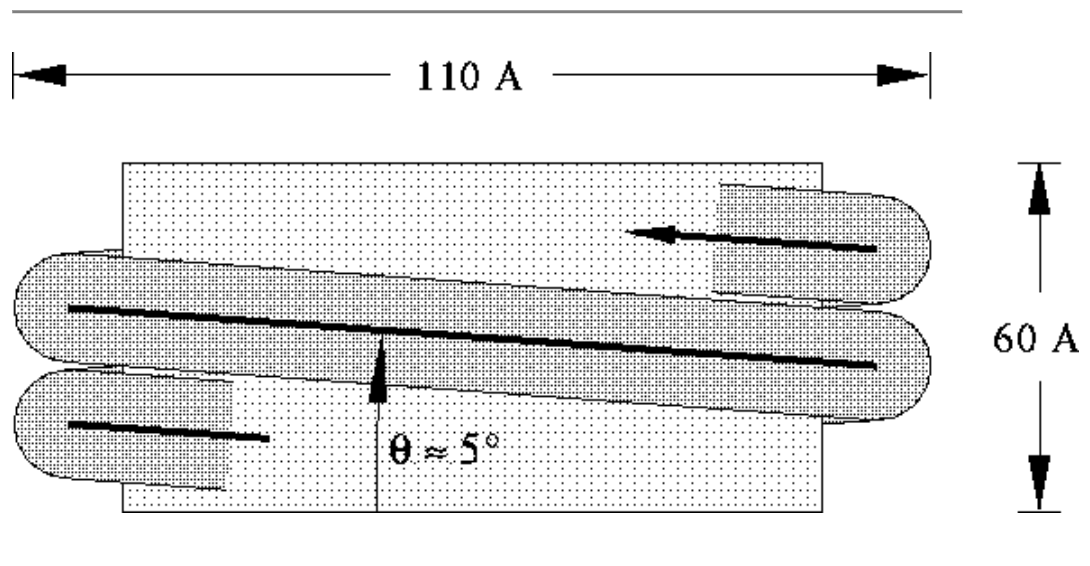
from <http://www.scripps.edu/case/nab5/NAB-sh-7.5.html>
hints for HW1 problem 4:

Nucleosome Model

While the DNA duplex is locally rather stiff, many DNA molecules are sufficiently long that they can be bent into a wide variety of both open and closed curves. Some examples would be simple closed circles, supercoiled closed circles that have relaxed into circles with twists, and the nucleosome core fragment, where the duplex itself is wound into a short helix.

The overall strategy for wrapping DNA around a curve is to create the curve, find the points on the curve that contain the base pair origins, place the base pairs at these points, oriented so that their helical axes are tangent to the curve, and finally rotate the base pairs so that they have the correct helical twist. In the example below, the simplifying assumption is made that the rise is constant at 3.38 \AA .

The nucleosome core fragment is composed of duplex DNA wound in a left handed helix around a central protein core. A typical core fragment has about 145 base pairs of duplex DNA forming about 1.75 superhelical turns. Measurements of the overall dimensions of the core fragment indicate that there is very little space between adjacent wraps of the duplex. A side view of a schematic of core particle is shown below.



Computing the points at which to place the base pairs on a helix requires us to spiral an inelastic wire (representing the helical axis of the bent duplex) around a cylinder (representing the protein core). The system is described by four numbers of which only three are independent. They are the number of base pairs n , the number of turns its makes

around the protein core t , the "winding" angle θ (which controls how quickly the helix advances along the axis of the core) and the helix radius r . Both the number of base pairs and the number of turns around the core can be measured. This leaves two choices for the third parameter. Since the relationship of the winding angle to the overall particle geometry seems more clear than that of the radius, this code lets the user specify the number of turns, the number of base pairs and the winding angle, then computes the helical radius and the displacement along the helix axis for each base pair:

$$d = 3.38 \sin(\theta); \quad \phi = 360t/(n-1)$$

$$r = \frac{3.38(n-1) \cos(\theta)}{2\pi t}$$

where d and ϕ are the displacement along and rotation about the protein core axis for each base pair.

These relationships are easily derived. Let the nucleosome core particle be oriented so that its helical axis is along the global Y-axis and the lower cap of the protein core is in the XZ plane. Consider the circle that is the projection of the helical axis of the DNA duplex onto the XZ plane. As the duplex spirals along the core particle it will go around the circle t times, for a total rotation of $360t^\circ$. The duplex contains $n-1$ steps, resulting $360t/(n-1)^\circ$ of rotation between successive base pairs.

```

1 // Program 10. Create simple nucleosome model.
2 #define PI 3.141593
3 #define RISE 3.38
4 #define TWIST 36.0
5 int b, nbp; int getbase();
6 float nt, theta, phi, rad, dy, ttw, len, plen, side;
7 molecule m, ml;
8 matrix matdx, matrx, maty, matry, mattw;
9 string sbase, abase;
10
11 nt = atof( argv[ 2 ] ); // number of turns
12 nbp = atoi( argv[ 3 ] ); // number of base pairs
13 theta = atof( argv[ 4 ] ); // winding angle
14
15 dy = RISE * sin( theta );
16 phi = 360.0 * nt / ( nbp-1 );
17 rad = (( nbp-1 ) * RISE * cos( theta )) / ( 2 * PI * nt );
18
19 matdx = newtransform( rad, 0.0, 0.0, 0.0, 0.0, 0.0 );
20 matrx = newtransform( 0.0, 0.0, 0.0, -theta, 0.0, 0.0 );
21
22 m = newmolecule();
23 addstrand( m, "A" ); addstrand( m, "B" );
24 ttw = 0.0;
25 for( b = 1; b <= nbp; b = b + 1 ){
26     getbase( b, sbase, abase );
27     ml = wc_helix( sbase, "", "dna", abase, "", "dna",
28                 2.25, -4.96, 0.0, 0.0 );
29     mattw = newtransform( 0., 0., 0., 0., 0., ttw );
30     transformmol( mattw, ml, NULL );
31     transformmol( matrx, ml, NULL );
32     transformmol( matdx, ml, NULL );
33     maty = newtransform( 0., dy*(b-1), 0., 0., -phi*(b-1), 0. );
34     transformmol( maty, ml, NULL );
35
36     mergestr( m, "A", "last", ml, "sense", "first" );
37     mergestr( m, "B", "first", ml, "anti", "last" );
38     if( b > 1 ){
39         connectres( m, "A", b - 1, "O3'", b, "P" );
40         connectres( m, "B", 1, "O3'", 2, "P" );
41     }
42     ttw += TWIST; if( ttw >= 360.0 ) ttw -= 360.0;
43 }
44 putpdb( "nuc.pdb", m );

```

Finding the radius of the superhelix is a little tricky. In general a single turn of the helix will not contain an integral number of base pairs. For example, using typical numbers of 1.75 turns and 145 base pairs requires ≈ 82.9 base pairs to make one turn. An approximate solution can be found by considering the ideal superhelix that the DNA duplex is wrapped around. Let L be the arc length of this helix. Then $L \cos(\theta)$ is the arc length of its

projection into the XZ plane. Since this projection is an overwound circle, L is also equal to $2\pi r n$, where n is the number of turns and r is the unknown radius. Now L is not known but is approximately $3.38(n - 1)$. Substituting and solving for r gives Eq. (1).

The resulting `nab` code is shown in Program 2. This code requires three arguments--the number of turns, the number of base pairs and the winding angle. In lines 15-17, the helical rise (`dy`), twist (`phi`) and radius (`rad`) are computed according to the formulas developed above.

Two constant transformation matrices, `matdx` and `matrx` are created in lines 19-20.

`matdx` is used to move the newly created base pair along the X-axis to the circle that is the helix's projection onto the XZ plane. `matrx` is used to rotate the new base pair about the X-axis so it will be tangent to the local helix of spirally wound duplex. The model of the nucleosome will be built in the molecule `m` which is created and given two strands "A" and "B" in line 23. The variable `ttw` will hold the total local helical twist for each base pair.

The molecule is created in the loop in lines 25-43. The user specified function `getbase()` takes the number of the current base pair (`b`) and returns two strings that specify the actual nucleotides to use at this position. These two strings are converted into a single base pair using the `nab` builtin `wc_helix()`. The new base pair is in the XY plane with its origin at the global origin and its helical axis along Z oriented so that the 5'-3' direction is positive.

Each base pair must be rotated about its Z-axis so that when it is added to the global helix it has the correct amount of helical twist with respect to the previous base. This rotation is performed in lines 29-30. Once the base pair has the correct helical twist it must be rotated about the X-axis so that its local origin will be tangent to the global helical axes (line 31). The properly-oriented base is next moved into place on the global helix in two stages in lines 32-34. It is first moved along the X-axis (line 32) so it intersects the circle in the XZ plane that is projection of the duplex's helical axis. Then it is simultaneously rotated about and displaced along the global Y-axis to move it to final place in the nucleosome. Since both these movements are with respect to the same axis, they can be combined into a single transformation.

The newly positioned base pair in `m1` is added to the growing molecule in `m` using two calls to the `nab` builtin `mergestr()`. Note that since the two strands of a DNA duplex are antiparallel, the base of the "sense" strand of molecule `m1` is added *after* the last base of the "A" strand of molecule `m` and the base of the "anti" strand of molecule `m1` is *before* the first base of the "B" strand of molecule `m`. For all base pairs except the first one, the new base pair must be bonded to its predecessor. Finally, the total twist (`ttw`) is updated and adjusted to remain in the interval [0,360) in line 42. After all base pairs have been

created, the loop exits, and the molecule is written out. The coordinates are saved in PDB format using the nab builtin putpdb().

[\[Contents\]](#) [\[Previous\]](#) [\[Next\]](#)

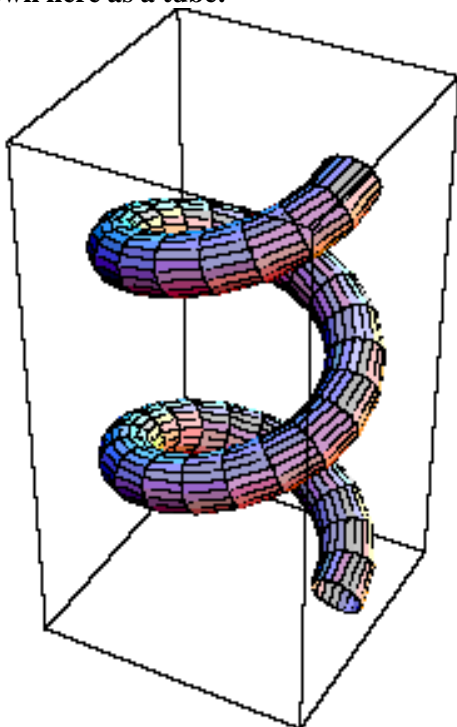
Updated on November 10, 2000. Comments to case@scripps.edu

from www.ma.umist.ac.uk/kd/geomview/geometry.html

curvature---which is the rate of change of angular direction per unit arc length s .

The Fundamental Theorem of Space Curves states that space curves are classified, up to a rotation and translation, by their curvature and torsion. Torsion measures the departure of a curve from a plane.

So, given a starting point and two functions of arc length representing curvature and torsion, the curve can be determined. We know that constant curvature in the plane gives a circle; if we have constant torsion as well then we obtain a circular helix---shown here as a tube:



White's Theorem states that the topological link number Lk for a pair of closed curves is the sum of their twist number Tw and their writhe number Wr :
 $Lk = Tw + Wr$

From <http://www.rwgrayprojects.com/Lynn/HelixKnot/helixknot01.html>
 parameters of the helix: (length, radius and number of cycles around the cylinder

from <http://members.tripod.com/vismath8/malkovsky/Section5.htm>

This below shows the equation for the helix in parametric form. Same as HW 1, problem 4. replace r by a , and b by h .

Example 7. *The vectors of the trihedra of a helix with a parametric representation*

$$\mathbf{x}(s) = (r \cos(\omega s), r \sin(\omega s), h \omega s) \quad \text{with } s \in \mathbf{R} \text{ where } r > 0, h \in \mathbf{R} \text{ and } \omega = \frac{1}{\sqrt{r^2 + h^2}} \text{ arc constants.}$$

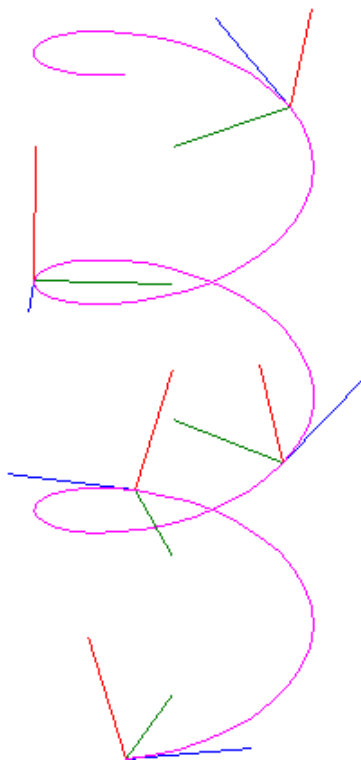


Figure 19. The vectors of trihedra of a helix

From <http://www.cco.caltech.edu/~brokawc/Suppl3D/TTW.htm>

The change in this local coordinate system as it moves along the curve can be described by the Frenet-Serret equations:

$$\begin{aligned} d\mathbf{t}/ds &= \kappa \mathbf{n}, \\ d\mathbf{n}/ds &= -\kappa \mathbf{t} + \tau \mathbf{b}, \\ d\mathbf{b}/ds &= -\tau \mathbf{n}. \end{aligned} \tag{4}$$

The new scalar variable, τ , is the torsion of the curve. Equation (4) tells us that the two scalar functions $\kappa(s)$ and $\tau(s)$ are sufficient to describe the change in the local coordinate system as it moves along the curve, and therefore are sufficient to describe the shape of the curve.

Below from <http://www.math.umd.edu/users/jmr/241/curves2.htm>

We are now interested in the derivative of the unit normal with respect to arclength. By

differentiating the equation $T \cdot N = 0$ with respect to s , we obtain $\frac{dT}{ds} \cdot N + T \cdot \frac{dN}{ds} = 0$,

from which it follows that $\frac{dN}{ds} \cdot T = -\kappa$. Since N is a unit vector, we know that $\frac{dN}{ds}$ is

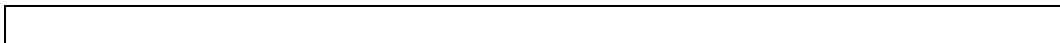
perpendicular to N . The torsion τ is defined to be $\frac{dN}{ds} \cdot B$. Note that since the direction

of B is determined independently of $\frac{dN}{ds}$, the torsion, unlike the curvature, is signed. Notice also that for a plane curve, the binormal is identically perpendicular to the plane in which the curve lies and the torsion is 0. Thus we have the Frenet-Serret formulae:

$$\frac{dT}{ds} = \kappa N$$

$$\frac{dN}{ds} = -\kappa T + \tau B$$

$$\frac{dB}{ds} = -\tau N$$



from <http://planetmath.org/encyclopedia/Torsion.html>

torsion

(Definition)

The torsion of a [space curve](#) is a [complement](#) to the [curvature](#) of a curve; curvature [measures](#) in the osculating plane what torsion measures in the [normal](#) plane. Thus

$$\tau(t) = \frac{1}{\sigma(t)}$$

where σ is the [radius](#) of torsion (analogous to radius of curvature). Because torsion is measured orthogonally to curvature, curves in a plane can have a non-zero curvature, but they will always have zero torsion. An example of a curve with both non-zero torsion and curvature is a helix.

A more analysis friendly way to find torsion is by the equation

$$\tau(t) = -\vec{N}'(t) \cdot \vec{B}'(t)$$

where \vec{N} and \vec{B} are the [unit](#) normal and binormal [vectors](#), respectively. Since

$$\vec{N} = \frac{1}{\kappa} \vec{T}' = \rho \vec{T}'$$

, the above can also be written in [terms](#) of the radius of curvature.

From <http://math.stanford.edu/courses/math51h/51htext2.pdf>

Example. Consider the helix

$$\mathbf{r} = (R \cos t, R \sin t, ct), \quad t \in [0, \infty).$$

The arc-length function is given by the formula

$$s(t) = \int_0^t \sqrt{R^2 + c^2} du = t\sqrt{R^2 + c^2},$$

and hence in the natural parameterization the helix is defined by the equations

$$\mathbf{r} = \left(R \cos \frac{s}{\sqrt{R^2 + c^2}}, R \sin \frac{s}{\sqrt{R^2 + c^2}}, c \frac{s}{\sqrt{R^2 + c^2}} \right), \quad s \in [0, \infty).$$

Then

$$\mathbf{r}'(s) = -\frac{R}{R^2 + c^2} \left(\cos \frac{s}{\sqrt{R^2 + c^2}}, \sin \frac{s}{\sqrt{R^2 + c^2}}, 0 \right)$$

and

$$k(s) = \|\mathbf{r}''(s)\| = \frac{R}{R^2 + c^2}.$$

The vector $\mathbf{b}(s) = \mathbf{T}(s) \times \mathbf{n}(s)$ is called the *binormal*. It is a unit normal vector to the osculating plane $P(s)$. Notice that the derivative $\mathbf{b}'(s)$ is proportional to the principal normal $\mathbf{n}(s)$. Indeed, $\mathbf{b}'(s) \perp \mathbf{b}(s) = 0$ because $\mathbf{b} \cdot \mathbf{b} = 1$ (see the above argument that $\mathbf{T}' \perp \mathbf{T}$ and $\mathbf{b}' \cdot \mathbf{T} = 0$, because $\mathbf{T} \cdot \mathbf{b} = 0$ and hence

$$0 = (\mathbf{T} \cdot \mathbf{b})' = \mathbf{T}' \cdot \mathbf{b} + \mathbf{T} \cdot \mathbf{b}' = k\mathbf{n} \cdot \mathbf{b} + \mathbf{T} \cdot \mathbf{b}' = \mathbf{T} \cdot \mathbf{b}'.$$

The **Reidemeister** manipulations suffice to completely untie any knot that is equivalent to the unknot,

From: [Mark-Jason Dominus \(mjd@op.net\)](mailto:mjd@op.net)

Subject: Re: Knot Question

Newsgroups: [sci.math](https://www.ietf.org/mail-archives/sci.math)

Date: 1996/03/19

View: [Complete Thread \(7 articles\)](#) | [Original Format](#)

In article <4iif1b\$71p@news1.radix.net>, Jim Ward <jfw@radix.net> wrote:

>the normalized bracket polynomial is invariant under the
>Reidemeister moves. Does this imply that if A and the unknot have
>the same polynomial, then A is the unknot?

Alas, no.

Here's what you can do instead:

Project the knot and list the over and undercrossings as before. Name each crossing with a different letter, and write $X+$ for an overcrossing at X and $X-$ for an undercrossing at X .

You now have a character string that looks like (for example)

$A+B-C+A-B+C-$ (trefoil knot)
 (empty) (unknot)
 $A+A-$ (unknot with a twist in it)
 $A+B+B-A-$ (unknot with two twists in it)
 $D+B-A-C+B+D-C-A+$ (tangled version of trefoil)

These 'descriptors' are circular, meaning that they have no particular beginning or end. The following descriptors are all identical:

$B-C+D-B+A-D+C-A+$
 $C+D-B+A-D+C-A+B-$
 $D-B+A-D+C-A+B-C+$
 $B+A-D+C-A+B-C+D-$ knot)
 $A-D+C-A+B-C+D-B+$ eight
 $D+C-A+B-C+D-B+A-$ figure-
 $C-A+B-C+D-B+A-D+$ a
 $A+B-C+D-B+A-D+C-$ (it's

Also, we shouldn't care whether we read a descriptor forwards or backwards. (Unless for some reason we need to worry about knot chirality, which we don't for this problem, since the unknot is not chiral.)

Now:

1. Reidemeister move I is applicable iff the descriptor contains a sequence of the form $P+P-$. To perform RMI on the knot, just delete the $P+P-$ from the descriptor.
2. Reidemeister move II is applicable iff the descriptor contains sequences of the form $P+Q+$ and $P-Q-$. ($P+Q+$ and $Q-P-$ will also work.) To perform RMII on the knot, just delete the two sequences from the descriptor. Note that if you have $P+Q-$ and $P-Q+$, you *can't* apply RMII.

3. Reidemeister move III is applicable iff the descriptor contains sequences of the form 'P+Q+' , 'P-R-' , and 'Q-R+' . (One or more of these may be reversed, as in paragraph 2.) To perform RMIII on the knot, reverse them, replacing the three sequences with 'Q+P+' , 'R-P-' , and 'R+Q-' , respectively.

The Reidemeister manipulations suffice to completely untie any knot that is equivalent to the unknot, and therefore, if D is the descriptor of such a knot, there is some sequence of operations 1, 2, and 3 which will reduce D to the empty string.

Here, then, is an algorithm for determining whether or not a given descriptor D represents the unknot:

You need a stack, S , and a list L . L will record the knots that you have analyzed already, so that you don't analyze anything twice. Both start out empty.

Push D onto the stack. Append D to list L .

TOP:

If the stack is empty, halt and report failure. Otherwise...

Pop the top of the stack into X .

If X is the empty string, then D was equivalent to the unknot. Halt and report success. Otherwise...

If X appears in list L , we have already analyzed it, so goto TOP without doing anything further. Otherwise...

Append X to list L , so that we don't analyze it again.

If X has the form described in (1) or (2) above, then
 Apply the appropriate Reidemeister transformation.
 The result is configuration Y .
 Push Y onto the stack.
 Goto TOP.
 Otherwise...

If X has the form described in (3) above, then,
 for each such P, Q, R ,
 apply the indicated transformation to X , yielding $Y_{\{P, Q, R\}}$.

```
    Push Y_{P,Q,R} onto the stack
    Goto TOP.
Otherwise...

    Goto TOP.
```

This is a very simple algorithm: It's just a depth-first search of the space of knots which are simpler than the original and reachable from it by a sequence of Reidemeister moves. If you want a BFS instead of a DFS, just replace stack S with a queue. The worst-case running time is exponential in the number of crossings, but I don't believe anyone knows anything better.

This algorithm works fine for knots with more than one component. You just need to maintain one descriptor for each component.

From <http://www.freelearning.com/knots/intro.htm#num1>

One invariant is the **minimal crossing number**. The minimal crossing number of a knot is the least number of crossings that appear in any projection of the knot. For example, the unknot has a minimal crossing number of 0. The trefoil knot has a minimal crossing number of 3.

From <http://members.tripod.com/vismath5/bor/bor6.htm>

For example, by duplicating one ring in Borromean rings, we obtain 4-Borromean link, and continuing in the same manner, n -Borromean links ($n=5,6,7\dots$). Different links of that infinite series follow from other choices of rings that will be duplicated.

We can define an evolutionary distance between two sequences as the number of point mutations that are necessary to evolve one sequence to other. (More specifically, the distance is the minimal number of mutations.)

From http://www.wi.mit.edu/nap/2002/nap_press_02_arachne.html on arachne

Various assembly programs have been previously reported, including SEQAID, CAP, PHRAP, TIGR and the Celera assembler, among others, but this is the first of its kind that is publicly available whole genome sequence assembler.

On nucleosomes

From http://www.rcsb.org/pdb/molecules/pdb7_1.html

Each nucleosome is composed of eight "histone" proteins bundled tightly together at the center (shown here in blue), encircled by two loops of DNA (shown here in orange). The histone proteins, however, are not completely globular like most other proteins. They have long tails, which comprise nearly a quarter of their length.

From <http://www.web-books.com/MoBio/Free/Ch3D1.htm>

A chromosome contains five types of histones: H1 (or H5), H2A, H2B, H3 and H4. H1 and its homologous protein H5 are involved in higher-order structures. The other four types of histones associate with DNA to form nucleosomes. H1 (or H5) has about 220 residues. Other types of histones are smaller, each consisting of 100-150 residues.

See <http://www.accessexcellence.org/AB/GG/nucleosome.html> for diagram of nucleosomes

In fact chromosomal DNA is packaged into a compact structure with the help of specialized proteins called **histones**. The complex DNA plus histones in eucaryotic cells is called **chromatin**.

The fundamental packing unit is known as a **nucleosome**. Each nucleosome is about 11nm in diameter. The DNA double helix wraps around a central core of eight histone protein molecules (an octamer) to form a single nucleosome. A second histone (H1 in the illustration) fastens the DNA to the nucleosome core. The total mass of this complex is about 100,000 daltons.

Nucleosomes are usually packed together, with the aid of a histone (H1,) to form a 30nm large fiber. As a 30nm fiber, the typical human chromosome would be about 0.1cm in length and would span the nucleus 100 times. This suggests higher orders of packaging, to give a chromosome the compact structure seen in a typical [karyotype](#) (metaphase) cell.

From <http://opbs.okstate.edu/~melcher/MG/MGW1/MG11226.html>

At the first level of structure, eight histone molecules are clustered on 146 bp of DNA. These clusters, called **nucleosomes**, are separated from one another by 20 to 100 bp (depending on cell type, organism and physiological status) of spacer DNA.

- Extensive characterization of nucleosomes by neutron diffraction, crystallization, X-ray diffraction, protein crosslinking, immuno electron microscopy and other techniques revealed further properties.
 - The DNA is wrapped left-handed around a 3.2 nm radius core of histones;
 - There are 1.8 turns of DNA per nucleosome.
 - DNase nicking of nucleosomes shows an average periodicity of 10.7 bp in center; 10.0 near ends.
 - The DNA is kinked; kinks are found at +/- 1 and +/- 4.
 - Alpha helical regions of histones have basic residues that contact the major groove of the DNA.
- The folding of DNA in nucleosomes results in a compaction factor of about 7.5. Higher levels of chromatin structure involve histone H1 and non-histone chromosomal proteins.

nucleosome

Repeating units of organization of chromatin fibres in chromosomes, consisting of around 200 base pairs, and two molecules each of the [histones](#) H2A, H2B, H3 and H4. Most of the DNA (around 140 base pairs) is believed to be wound around a core formed by the histones, the remainder joins adjacent nucleosomes, thus forming a structure reminiscent of a string of beads.

<http://www.average.org/~pruss/Nucleosomes/othersites.html> have links on nucleosome.

- nucleosome consists of:
 - core particle
 - histone octamer (2 copies each of H2A, H2B, H3 and H4)
 - 146 bp of DNA wrapped around octamer
 - linker DNA
 - 54 bp of DNA
 - histone H1
 - not found in all "beads" on the string
 - is found in all nucleosomes in the chromatin fibre
 - bound to linker DNA and core particle
 - responsible for higher order folding of chromatin structure

from <http://www.lbl.gov/Science-Articles/Research-Review/Magazine/1997-fall/vr/DNA.html>

DNA is typically wrapped around a mixture of proteins called "histone" that provide the giant molecule with structural support. Repeating segments of DNA, some 165 to 240 base pairs in length, combine with eight histone molecules to form a distinct unit called a "nucleosome."

Nucleosomes are in turn organized into even larger units called "chromatin."

From <http://aesop.rutgers.edu/~molgen/nucleosomes.htm>

6. Model was derived partly from the dimensions of the components (Fig 19.5), 6 nm x 11 nm, length of DNA is ~2 x the circumference of the particle 34 nm. Height of the particle is slightly larger than 2 x the diameter of DNA ~ 4 nm.

The nucleosome is ~10 nm diameter, produces the 10 nm fiber (Fig 19.17)

From <http://www.mbg.cornell.edu/biobm332/exams/exams97-00/00finalkey.html>

1. **L = linking number, it is the number of times one strand winds around the other**

T = twisting number, it is the number of base pairs in a B form DNA divided by 10.5

W = writhing number, it is the number of supercoils

1. The topology of a circular B-form DNA molecule has the following values; L = 395, T = 400 and W = -5. How many nucleosomes are there on this DNA molecule? (Assume that each nucleosome generates one negative supercoil.)

5

check this below. Nice review.

<http://www.cmb.uab.edu/courses/Lectures/harvey.pdf>

<http://www.mindfully.org/GE/DNA.htm>

see <http://www.mpip-mainz.mpg.de/~heli/chromatin.html> for best diagram I saw of nucleosome

from <http://intranet.siu.edu/~mbmb/sample%20test2.html>

5. Nucleosomes package DNA within the cell by wrapping DNA around the exterior of the protein's cylindrical structure. About 200 base pairs of DNA is wrapped around the nucleosome in a left-handed helical fashion and makes 1.7 wraps per nucleosome. Starting with a relaxed 2 kilobase pair closed circular DNA, determine what the linking number, writhe and twist are after 10 nucleosomes are bound per DNA molecule. The relaxed plasmid is B-DNA with 10 base pairs per helical turn (10 points).

From <http://web.uvic.ca/sciweb/Courses/B300/DNA%20supercoiling.html>

DNA supercoiling - an outline

The strands of topologically closed DNA, for example, covalently closed circular plasmid DNA, or DNA that loops out of the scaffolding structure in eukaryotic cells (cf. Figure 24-28), cannot change the number of times they cross each other: this is called their linking number

If the twist (the basic double helical turns of the two strands about each other) of the double helix in topologically closed DNA changes, there is supercoiling ("writhing"; the coiled-coil) to keep the linking number constant

Most cellular DNA is underwound (917-918)

A sample of circular DNA isolated from a cell, e.g. plasmid DNA, if constrained to lie flat, would have fewer twists of one strand about the other than B DNA does - this is caused by intracellular conditions that favour partial unwinding (for example, parts of the DNA that are being transcribed into RNA are unwound)

This is called underwinding, and is characteristic of natural DNAs

When underwound DNA is isolated, it tends to wind into a normal B DNA helix (10.5 bases per turn); in doing so, it generates negative supercoils because the linking number cannot change (Figure 24-12)

The partially unwound state, and the fully wound-up, supercoiled state, are equivalent in linking number (Figure 24-13)

DNA underwinding is defined by topological linking number (918-921)

Topologically closed DNA is characterised by its linking number, Lk, which is an integer and which cannot change as long as no covalent bonds are broken

Lk is related to the twist (Tw) and the writhe (Wr) numbers by

$$Lk = Tw + Wr$$

Tw is about the same as the basic, B DNA winding number (10.5 bp per turn in relaxed DNA), but may differ from this due to topological stress

Writhe is the supercoiling number, the coiling of the DNA coil

Tw and Wr need not be integers

If Tw changes, then Wr must change correspondingly, to keep Lk constant

For relaxed, closed circular B DNA (no supercoils), Lk and Tw are equal to the number of base pairs divided by 10.5, and Wr is zero

The superhelical density σ is defined as the number of turns that have been added or subtracted in the supercoiled DNA, compared to the relaxed state, divided by the total number of turns in the DNA if it were relaxed (normally bp/10.5)

Typically, σ is between -.05 and -.07 (5-7% underwinding) in isolated natural DNA

Supercoils in isolated plasmid DNA are in the form shown in Figure 24-16; this is called the plectonemic form of supercoiled DNA

Right-handed, plectonemic DNA has negative supercoils

Underwound DNA facilitates the formation of cruciforms (Figure 24-18), because both have unpaired bases

Topoisomerases catalyze changes in the linking number of DNA (921-922)

Topoisomerases allow the linking number to change

Type I topoisomerases cause a transient break in one strand of duplex DNA, allow the open end to rotate around the other strand, and then re-seal the broken strand, to change the linking number by 1

Type II topoisomerases change the value of Lk by 2, by creating a double-stranded break and allowing *two* strands to pass through before re-sealing the break (Figure 24-20)

An unusual example of a Type II topoisomerase is *E. coli* gyrase, which can add negative supercoils to an already negatively supercoiled DNA, using the energy of ATP hydrolysis to drive the reaction; eukaryotic cells do not have gyrase activity, but they do have Type II topoisomerases which can relax supercoiled DNA

Eukaryotic topoisomerases of Type II can relax positive and negative supercoils, as can eukaryotic topoisomerases of Type I

Plasmid DNAs differing in numbers of supercoils have different degrees of compactness, and will have different mobilities on gel electrophoresis, as shown for isolated plasmid treated with Type I Topoisomerase (Figure 24-19)

DNA compaction requires a special form of supercoiling (922-923)

The plectonemic form of supercoiling causes only a moderate compaction of DNA

The alternative, solenoidal, form is more compact (Figure 24-22)

The solenoidal form exists in natural eukaryotic DNA, which is wound around histone cores in the nucleosomes in a left-handed (negative) supercoil (Figure 24-23)

This negative supercoiling of the DNA as the nucleosome forms introduces positive supercoils in the segments of DNA linking nucleosomes, and these must be relaxed by topoisomerases

From

<http://216.239.33.100/search?q=cache:VtcfbuDe32YC:www.uovs.ac.za/faculties/nat/mkbc/biochem/Super.htm+twist+nucleosomes&hl=en&ie=UTF-8>

Supercoiling

Introduction

Supercoiling simply means coiling of a coil. Supercoiling and topology, although perhaps at first glance abstract mathematical concepts, have very relevant application in molecular biology. The DNA molecule is subject to topological constraints, and these have very real effects on the function of DNA. Negative supercoiling can stabilise secondary DNA structures such as hairpin loops, cruciforms, and also facilitate the formation of a melted region in the transition of a transcriptional pre-initiation complex (PIC) to a elongating complex. Also, the DNA in both pro- and eukaryotes are naturally negatively supercoiled. In prokaryotes this is due to the action of gyrases (these are enzymes like topoisomerases, but induce supercoiling in an ATP-dependent manner). In eukaryotes, the packaging of DNA into chromatin causes the DNA template (after removal of proteins) to be supercoiled. In addition, the passage of the RNA polymerase along the DNA molecule generates a twin supercoiled domain. The region behind the polymerase is negatively supercoiled, and the region in front of the polymerase is positively supercoiled. This superhelical stress is normally relaxed by topoisomerases. In yeast, there are two types: topoisomerase I (topo I) and topoisomerase II (topo II). Topo I induces a single strand nick, and will relax the DNA molecule in units of 1. Topo II, predictably, cuts both strands, and changes the superhelicity by units of 2. These enzymes appear to be required for normal DNA function, and are involved in the relaxing of superhelical stress that accumulates during transcription and replication of DNA. Thus, it is clear that an understanding of many of the processes that can influence DNA function, requires some understanding of supercoiling.

Classic Linking Theory (CLT).

The essential concept that is used in a theoretical study of supercoiling is the ribbon. The ribbon has two sides (which can represent the phosphodiester backbones of the DNA duplex), and it has an axis, equidistant from the ribbon edges, equivalent to the helix axis. There are three parameters that are important when considering supercoiling: the linking number (L_k), the **twist** (T_w) and the writhing (W_r). The L_k and T_w is a function of the edge of the ribbon, and has no meaning for a one-dimensional line, such as an axis. The W_r , on the other hand, is a function of the ribbon (or helix) axis, and describes the shape of the axis in space. These three parameters are related by the function

$$L_k = T_w + W_r$$

This function simply states that the L_k of a molecule is the sum of the T_w and W_r parameters, and that if the L_k of a molecule is kept constant, but the shape of the axis (the W_r) is changed, the T_w must change by an equal amount of opposite sign.

The linking number, roughly speaking, is the number of times the one DNA strand (or ribbon edge) crosses the other in space. This is a topological property, since the smooth deformation of the molecule does not change the linking number. In this sense a doughnut and a coffee cup is topologically equivalent: both has a single hole, and, were it made from soft clay, the one can be deformed into the other without breaking the clay or introducing additional holes into it.

The linking number of a closed (the ends of the ribbon or DNA molecule meet, forming a circle) can be calculated by inspection by viewing the entire ribbon from any orientation at an infinite distance (watch those taxi fares). If one concentrates on only one edge, and count the number of times this edge crosses in front of the other, one is, in effect, calculating the linking number. Note that the linking number has a sign. If the edge one is inspecting crosses the other in a right-handed (clockwise) manner, the sign is positive. Thus, in viewing the entire ribbon, if the one edge crossed in the one direction and then crosses back in the other direction, the net L_k is zero, since the one crossing contributes +1, and the reverse crossing

Figure 1. The linking number of a DNA molecule. The schematic shows a small circular DNA molecule where the one stand crosses the other a total of 6 times. The L_k of this molecule is thus 6. Since the helix is right-handed, the sign of the linking number is positive, and $L_k = +6$.

On a plane, the **twist** is defines as 1 if the ribbon rates about the ribbon axis by 360° .

Thus, a flat ribbon that is bent in a plane, does not have **twist**. The T_w is not a topological property, since the deformation of the ribbon (or DNA molecule) in space can change the **twist**. When the ribbon is wound flat onto a cylinder, the **twist** of the ribbon is given by the equation:

$$T_w = N \sin \alpha$$

Where N is the number of times (in units of radians, i.e. one rotation is 2π) that the ribbon revolves around the cylinder, and α is the pitch angle of the helix. Note that T_w is normalised to 2π . When the winding of the ribbon onto a cylinder is very shallow (i.e., every gyre tends to a circle) the **twist** approaches 0 [$2\pi \times \sin(0)/2\pi = 1 \times 0 = 0$].

The wrythe is not an easy parameter to calculate. Generally, this parameter can be arrived at by knowing the L_k and T_w of a molecule

Virtual surface linking theory (VSLT)

Despite its impressive name, virtual surface linking theory provides a more rigorous and more easily understood theoretical frame within which to calculate supercoiling.

The L_k in VSLT is calculated relative to a surface, where the orientation of the surface is defined as shown in Figure 2.

A normal vector to the surface is defined, with the direction of the vector deduced by the "right-hand rule". Thus, in A, the direction of the vector is up. The direction of this vector becomes important when calculating the sign of the L_k . The L_k itself is defined as the number of times the helix axis crosses the spanning surface of the ribbon. The sign of L_k is positive if the axis direction (arbitrarily defined) crosses the spanning surface in the same direction as the normal vector. The L_k is negative in the inverse situation, as shown in Figure 3.

Figure 3. Calculation of the L_k and the association sign.

One can also calculate L_k in VSLT by looking at the whole structure from infinite distance, or looking at the shadow that is produced by illuminating with a light from infinite distance. In this case, if the strand on top is rotated in a clockwise direction by less than 180° to point in the same direction as the helix axis, the crossing (node) is assigned a negative value. If the rotation is anti-clockwise, the node is positive. Each such node contributes $\frac{1}{2} L_k$ unit. Note the similarity of this calculation with that given for CLT, above. The determination of the sign of the node is illustrated in Figure 4.

Figure 4. Calculation of the sign of a crossing node

In VSLT the W_r is defined as the number of times that the helix axis crosses itself in a plane projection, with the sign of the node calculated as shown in Fig. 3. However, careful consideration will reveal that the number of times that the axis crosses itself is dependent on the specific orientation of the projection. Thus, the W_r is the average of the sum of the individual W_{rp} over all possible projections. Thus, W_r is not necessarily an integer. It also follows directly that $W_r = 0$ for any molecule constrained to a plane, irrespective of the complexity of the helix axis in the plane. This is so because there is no possible orientation in which the axis will cross itself in any projection.

It is obvious that W_r is not a quantity that can be easily calculated for complex shapes. However, again, W_r can be arrived at by measuring other more tenable parameter such as L_k which is a function of W_r .

Twist

The definition of **twist** in VSLT is mathematically complex to calculate, but relatively simple to understand. **Twist** is not simply the number of times the one DNA strand crosses the helix axis, since the helix axis itself can contribute to T_w . The definition is as follows: A plane, perpendicular to the helix axis, transects the helix at a given point \mathbf{A} . From this point \mathbf{a} , a vector runs to a point \mathbf{c} of the one DNA strand, represented by curve \mathbf{C} . As the plane is moved forward by an infinitesimal amount, the vector \mathbf{ac} will rotate since the DNA strand revolves around \mathbf{A} . If the helix axis \mathbf{A} is planar, the T_w is simply the number of times that \mathbf{ac} rotates about \mathbf{A} . However, when \mathbf{A} is not planar, the orientation of the coordinate system xyz , defined at a point \mathbf{a}_0 , must be taken into consideration. This Cartesian coordinate system xyz is defined so that the z axis coincides with the infinitesimal length of \mathbf{A} . As the plane is now moved along the axis \mathbf{A} , the component (or projection) of \mathbf{ac} is the important quantity. The **twist** is the sum of the xy component angles \mathbf{ac} describes in the coordinate system of \mathbf{a}_0 as it is moved along the entire length of \mathbf{A} .

Figure 5. Definition of T_w . \mathbf{A} is the helix axis, \mathbf{C} is the phosphodiester backbone, and \mathbf{ac} is a vector connecting \mathbf{A} to \mathbf{C} at points \mathbf{a} and \mathbf{c} . T_w is the sum of the xy component of the rotation of \mathbf{ac} as the transecting plane, which stays perpendicular to \mathbf{A} , is moved through the DNA circle.

Figure 6. Definition of the virtual surface. The surface coincides with \mathbf{A} , and has a vector \mathbf{v} perpendicular to the surface at point \mathbf{a} .

This definition of T_w can be further refined by using a [virtual] surface as reference. Although this may seem unnecessarily complex, it actually provides a very physical framework within which to analyze real DNA molecules. Take the framework described above, and place the axis \mathbf{A} on a surface (that may be curved). Another vector \mathbf{v} is now defined that is perpendicular to this plane, and in the plane of the original \mathbf{ac} vector. As this plane is moved along \mathbf{A} , the winding number (Φ) is defined as the number of times that \mathbf{ac} rotates past \mathbf{v} . Since we are considering a closed circular molecule (L_k has no physical meaning in linear molecules), Φ must necessarily be an integer. The helical period h of the DNA is simply the length of the DNA divided by Φ ($h = N/\Phi$). The helical repeat of DNA is often measured by adsorbing the molecule to a surface, and examining the number of nuclease cleavage sites over the length of the molecule, where it is assumed that maximal cleavage will occur where the DNA molecule is farthest from the surface. The relation between the helical period so measured and Φ is now immediately obvious.

What is the relation between Φ and T_w ?

Although Φ is a component of T_w , Φ certainly does not equal T_w . An additional parameter, surface **twist** (ST_w), which measures the way in which the reference vector \mathbf{v} changes, and therefore the virtual surface on which the axis \mathbf{A} lies, is required to fully define T_w . This is most clearly seen when considering an observer that walks along \mathbf{A} on the surface. This observer can count how many times the vector \mathbf{ac} crosses his or her field of vision. This is simply Φ . However, the observer will not be able to tell whether he or she has rotated around \mathbf{A} without reference to something else. This something else is a displacement curve. This displacement curve is simply the trace of the observer's head as he or she walks along the surface. This displacement curves measures the number of times (or fraction thereof) that the observer rotated about \mathbf{A} . Forward and backward movements do not contribute to this curve. For instance, if the axis \mathbf{A} lies on the equator of a sphere, the observer can walk around the sphere on the equator, yet not rotate around the axis \mathbf{A} . Thus, in this case, ST_w is 0. In physical terms, ST_w measures the shape of the DNA molecule in space.

This immediately takes us to the surface linking number SL_k which measures the linking of \mathbf{A} with the displacement curve.

$$SL_k = ST_w + W_r \quad 3$$

SL_k for a helix formed on a sphere (interwound or plectonemic supercoiling) is 0. This is because the displacement curve is not linked to \mathbf{A} at all: it can be the central axis of the spheroid. Thus, in this case $ST_w = -W_r$. For toroidal supercoiling (the type found in chromatin), SL_k is simply the number of toroidal turns, defined in the same way as for the Φ .

By combining equations 1 and 3, we arrive at

$$\begin{aligned} L_k &= ST_w + \Phi + W_r \quad 4 \\ &= SL_k + \Phi \quad 5 \end{aligned}$$

Thus, the L_k of a molecule can be determined from Φ which can be measured as the helical period relative to a surface, and SL_k , which can be calculated from the shape of the molecule.

For any closed DNA on a spheroid, $SL_k = 0$ and therefore $L_k = \Phi$. For toroidal DNA, SL_k is simply the number of times that the helix winds about the super helix axis (n). Usually, a DNA that is totally relaxed (for instance nicked and then religated) is assigned a linking number N/Φ , since the relaxed molecule will be nearly planar and have little contributions from ST_w and W_r . The linking number of such a molecule is often written as L_{k0} . It is possible to measure the difference in linking number between this molecule and another in an agarose gel, since the degree of supercoiling compacts the DNA, and its migration through the gel matrix is different. Individual topoisomers can be so resolved, and the ΔL_k of a specific topoisomer calculated by counting the number of topoisomers between the L_{k0} species, and the band (topoisomer) of interest.

A length independent number, the specific linking difference (σ), is calculated as

$$\sigma = \Delta L_k / L_{k0}$$

For bacterial plasmids, this is usually -0.06 .

Nucleosomes and the linking number paradox

When viewing the structure of the nucleosome, it is seen that the helix winds almost two times around the histone octamer over a length of 168bp. Yet, when the linking difference of a nucleosome is measured, it is found to be ~ -1.1 . This became known as the *linking number paradox*. Armed with our understanding of supercoiling theory, we can now investigate why this may be so.

In **nucleosomes**, we have ~ 81 bp per superhelical turn. Say we have a 4600bp circular molecule containing nucleosome cores, we will have 57 left-handed superhelical turns. The change in the winding number Φ for one supercoil in going from free DNA (which has a helical period, h , of ~ 10.5) to nucleosomal DNA (where $h = 10.0$) is

$$\begin{aligned} \Delta \Phi &= N/h_{\text{nuc}} - (N/h_0) \\ &= (81/10.0) - (81/10.5) \\ &= 0.39 \end{aligned}$$

Thus, for 57 supercoils $\Delta \Phi = 22$.

Now,

$$\begin{aligned} \Delta L_k &= SL_k + \Delta \Phi \\ &= -57 + 22 \\ &= -35 \end{aligned}$$

Thus

$$\Delta L_k = -0.61n$$

or, the linking number of the nucleosome is $0.61 \times -2 = -1.2$

Applications

One dimensional resolution of topoisomers

This is good below

<http://wine1.sb.fsu.edu/bch4053/Lecture21/Lecture21.htm>

from <http://searchlauncher.bcm.tmc.edu/help/AlignmentScore.html>

The **alignment score** is the sum of the scores specified for each of the aligned pairs of letters, and letters with nulls, in the alignment. The higher the alignment score, the better the alignment.

<http://www.cbil.upenn.edu/genlang/papers/fft.html>

Cheever, E.A., Overton, G.C., and Searls, D.B. (1991) "Fast Fourier Transform-Based Correlation of DNA Sequences using Complex Plane Encoding" Computer Applications in the Biosciences 7(2):143-159.

The detection of similarities between DNA sequences can be accomplished using the signal processing technique of cross correlation. An early method used the Fast Fourier Transform (FFT) to perform correlations on DNA sequences in $O(n \log n)$ time for any length sequence [Felsenstein et al, 1982]. However, this method requires many FFTs (nine), runs no faster if one sequence is much shorter than the other, and measures only global similarity, so that significant short local matches may be missed. We report that, through the use of alternative encodings of the DNA sequence in the complex plane, the number of FFTs performed can be traded off against (1) signal-to-noise ratio, and (2) a certain degree of filtering for local similarity via k-tuple correlation. Also, when comparing probe sequences against much longer targets, the algorithm can be sped up by decomposing the target and performing multiple small FFTs in an overlap-save arrangement. Finally, by decomposing the probe sequence as well, the detection of local similarities can be further enhanced. With current advances in extremely fast hardware implementations of signal processing operations, this approach may prove more practical than heretofore.

From

<http://hades.ph.tn.tudelft.nl/Internal/PHServices/Documentation/MathWorld/math/math/e/e350.htm>

Euler Graph

A [Graph](#) containing an [Eulerian Circuit](#). An undirected [Graph](#) is Eulerian [Iff](#) every [Vertex](#) has [Even Degree](#). A [Directed Graph](#) is Eulerian [Iff](#) every [Vertex](#) has equal [Indegree](#) and

Outdegree. A planar Bipartite Graph is Dual to a planar Euler graph and vice versa. The number of Euler graphs with n nodes are 1, 1, 2, 3, 7, 16, 54, 243, ... (Sloane's [A002854](#)).

Eulerian Circuit

An Eulerian Trail which starts and ends at the same Vertex. In other words, it is a Cycle which uses each Edge exactly once. The term Eulerian Cycle is also used synonymously with Eulerian circuit. For technical reasons, Eulerian circuits are easier to study mathematically than are Hamiltonian Circuits. As a generalization of the Königsberg Bridge Problem, Euler showed (without proof) that a Connected Graph has an Eulerian circuit Iff it has no Vertices of Odd Degree.

Eulerian Trail

A Walk on the Edges of a Graph which uses each Edge exactly once. A Connected Graph has an Eulerian trail Iff it has at most two Vertices of Odd Degree.

From <http://www.utc.edu/~cpmawata/petersen/lesson12.htm>

An Euler circuit on a graph G is a circuit that visits each vertex of G and uses every edge of G .

Vertex Degree

The degree of a Vertex of a Graph is the number of Edges which touch the Vertex, also called the Local Degree. The Vertex degree of a point A in a Graph,

denoted $\rho(A)$, satisfies

$$\sum_{i=1}^n \rho(A_i) = 2E,$$

where E is the total number of Edges. Directed Graphs have two types of degrees, known as the Indegree and the Outdegree.

Indegree

The number of inward directed [Edges](#) from a given [Vertex](#) in a [Directed Graph](#).

Outdegree

The number of outward directed [Edges](#) from a given [Vertex](#) in a [Directed Graph](#).

NP-Complete Problem

A problem which is both [NP](#) (solvable in nondeterministic [Polynomial](#) time) and [NP-Hard](#) (can be translated into any other [NP-Problem](#)). Examples of NP-hard problems include the [Hamiltonian Cycle](#) and [Traveling Salesman Problems](#).

In a landmark paper, Karp (1972) showed that 21 intractable combinatorial computational problems are all NP-complete.

See also [Hamiltonian Cycle](#), [NP-Hard Problem](#), [NP-Problem](#), [P-Problem](#), [Traveling Salesman Problem](#)

NP-Problem

A problem is assigned to the NP (nondeterministic [Polynomial](#) time) class if it is solvable in polynomial time by a nondeterministic [Turing Machine](#). (A nondeterministic [Turing Machine](#) is a "parallel" [Turing Machine](#) which can take many computational paths simultaneously, with the restriction that the parallel Turing machines cannot communicate.) A [P-Problem](#) (whose solution time is bounded by a polynomial) is always also NP. If a solution to an NP problem is known, it can be reduced to a single [P](#) ([Polynomial](#) time) verification.

[Linear Programming](#), long known to be NP and thought *not* to be P, was shown to be [P](#) by L. Khachian in 1979. It is not known if all apparently NP problems are actually [P](#).

A problem is said to be [NP-Hard](#) if an [Algorithm](#) for solving it can be translated into one for solving any other NP-problem problem. It is much easier to show that a problem is NP than to show that it is [NP-Hard](#). A problem which is both NP and [NP-Hard](#) is called an [NP-Complete Problem](#).

See also [Complexity Theory](#), [NP-Complete Problem](#), [NP-Hard Problem](#), [P-Problem](#), [Turing Machine](#)

P-Problem

A problem is assigned to the P ([Polynomial](#) time) class if the number of steps is bounded by a [Polynomial](#).

See also [Complexity Theory](#), [NP-Complete Problem](#), [NP-Hard Problem](#), [NP-Problem](#)

Complexity Theory

The theory of classifying problems based on how difficult they are to solve. A problem is assigned to the [P-Problem](#) ([Polynomial](#) time) class if the number of steps needed to solve it is bounded by some [Power](#) of the problem's size. A problem is assigned to the [NP-Problem](#) (nondeterministic [Polynomial](#) time) class if it permits a nondeterministic solution and the number of steps of the solution is bounded by some power of the problem's size. The class of [P-Problems](#) is a subset of the class of [NP-Problems](#), but there also exist problems which are not [NP](#).

However, if a solution is known to an [NP-Problem](#), it can be reduced to a single period verification. A problem is [NP-Complete](#) if an [Algorithm](#) for solving it can be translated into one for solving any other [NP-Problem](#). Examples of [NP-Complete Problems](#) include the [Hamiltonian Cycle](#) and [Traveling Salesman Problems](#). [Linear Programming](#), thought to be an [NP-Problem](#), was shown to actually be a [P-Problem](#) by L. Khachian in 1979. It is not known if all apparently [NP-Problems](#) are actually [P-Problems](#).

See also [Bit Complexity](#), [NP-Complete Problem](#), [NP-Problem](#), [P-Problem](#)

Math 455.1 • 1 April 2002

Algorithm to Find Euler trail in Eulerian graph

Input

A connected graph G in which each vertex has even degree.

Output

An Eulerian trail in G .

Begin

(0) Remove all loops from G .

(1) (1.1) Select an arbitrary vertex w_0 of G ;

(1.2) form some cycle C in G from w_0 to w_0 (use Cycle Lemma method);

and

(1.3) remove all edges in C , leaving a subgraph H of G .

(2) While H has edges . . .

(2.1) select some vertex v of H that is an end of some edge of C ;

(2.2) select some cycle S in H from v to v ;

(2.3) let C' be the new walk obtained by inserting S into C at vertex v

so that C' is a trail in G ; and

(2.4) let H be the subgraph of G obtained by removing from H all edges of S and all the isolated (that is, degree 0) vertices of H .

(3) At each vertex w of C , insert all loops of G at w .

(4) Return C .

End

Theorem 2. (Fleury's algorithm)

Let G be an Eulerian graph. Then the following construction is always possible, and produces an Eulerian trail of G .

Start at any vertex u and traverse the edges arbitrarily, except subject to 2 rules:

- (a) erase the edges as they are traversed, and the isolated vertices resulted (if any);
- (b) use a bridge only if there is no alternative.

From <http://www2.toki.or.id/book/AlgDesignManual/BOOK/BOOK4/NODE165.HTM>

There are well-known conditions for determining whether a graph contains an Eulerian cycle, or path:

- An *undirected* graph contains an Eulerian *cycle* iff (1) it is connected and (2) each vertex is of even degree.
- An *undirected* graph contains an Eulerian *path* iff (1) it is connected and (2) all but two vertices are of even degree. These two vertices will be the start and end points of the path.
- A *directed* graph contains an Eulerian *cycle* iff (1) it is connected and (2) each vertex has the same in-degree as out-degree.
- Finally, a *directed* graph contains an Eulerian *path* iff (1) it is connected and (2) all but two vertices have the same in-degree as out-degree, and these two vertices have their in-degree and out-degree differ by one.

From <http://www.cvl.ua.edu/math103/euler/howcanwe.htm>

How can we tell if a graph has an Euler path or circuit?

In solving the [Königsberg bridge problem](#), Euler proved three theorems which answer this question.

EULER'S THEOREMS**Euler's Theorem 1**

If a graph has any vertices of odd degree, then it CANNOT have an EULER CIRCUIT.

AND

If a graph is connected and every vertex has even degree, then it has

AT LEAST ONE EULER CIRCUIT (usually more).

Euler's Theorem 2

If a graph has more than 2 vertices of odd degree, then it CANNOT have an EULER PATH.

AND

If a graph is connected and has exactly 2 vertices of odd degree, then it has AT LEAST ONE EULER PATH (usually more). Any such path must start at one of the odd-degree vertices and end at the other.

Euler's Theorem 3

The sum of the degrees of all the vertices of a graph is an even number (exactly twice the number of edges).

In every graph, the number of vertices of odd degree must be even.

The implications of Euler's theorems are summarized in the table below:

# of ODD Vertices	Implication (for a connected graph)
0	There is at least one Euler Circuit.
1	THIS IS IMPOSSIBLE! (Try it yourself.)
2	There is no Euler Circuit but at least 1 Euler Path.
more than 2	There are no Euler Circuits or Euler Paths.

Keep in mind:

- An Euler Circuit IS a type of Euler Path but an Euler Path is not necessarily an Euler Circuit.
- The sum of the degrees of all the vertices of a graph is twice the number of edges.
- The number of vertices of odd degree must be even.

[Back](#)

From <http://www.cvl.ua.edu/math103/euler/ifagraph.htm>

(Remember, only connected graphs with no vertices of odd degree have Euler circuits.)

oct 10, 2002 notes

How many Euler circuits are there starting from one random vertex? Are there are the same number of Euler Circuits starting from any vertex?

To help me understand about Alu, these are quotes from the internet related to Alu.

From <http://www.accessexcellence.org/AE/AEPC/DNA/detection.html>

The Alu family of short interspersed repeated DNA elements are distributed throughout primate genomes. Over the past 65 million years, the Alu sequence has amplified via an RNA-mediated transposition process to a copy number of about 500,000--comprising an estimated 5% of the human genome

An estimated 500-2,000 Alu elements are mostly restricted to the human genome

From <http://www.sequenceanalysis.com/glossary.html#alu>

Alu
A family of approx. 300 bp repetitive sequences, found dispersed throughout the *human* genome. Almost any 100 kb human nucleotide sequence will have Alu sequences within it.

From the internet, found the definition of Alu sequence:

Alu sequences A family of 300-bp sequences occurring nearly a million times in the human genome.

<http://users.aber.ac.uk/obb0/pcr.htm>

The alu sequence has 900,000 copies throughout the human genome.

From http://www.iephb.nw.ru/labs/lab38/spirov/hox_pro/rare-alu.html

Over the past 30 to 60 million years these insertions have occurred repeatedly, leaving roughly a million copies of *Alu* scattered through the human genome and making up almost 10 per cent of all the DNA in each cell. During this time, the sequences of the various *Alus* have begun to diverge, so that four distinct subfamilies of *Alu* can now be recognised.

The results also provide the first clear evidence that most *Alus* could have the potential to regulate human genes. Apparently most *Alus* have little effect on nearby genes, perhaps because they are bundled deep within folds of DNA. But with a million *Alus* strewn

randomly through the genome during the course of primate evolution, at least a few are likely to have landed where they could regulate a nearby gene. When this occurred, the effect would be equivalent to randomly twisting a knob on an instrument panel. Usually the effect would be harmful, but once in a while it might produce an interesting and beneficial genetic novelty. It seems that over the last 30 to 50 million years, it would provide good evolutionary fodder.

This is reference paper for HW3, problem 2.

Roy-Engel AM, Salem AH, Oyeniran OO, Deininger L, Hedges DJ, Kilroy GE, Batzer MA, Deininger PL.

Tulane Cancer Center, SL-66, Department of Environmental Health Sciences, Tulane University-Health Sciences Center, New Orleans, Louisiana 70112, USA.

Long and short interspersed elements (LINEs and SINEs) are retroelements that make up almost half of the human genome. L1 and Alu represent the most prolific human LINE and SINE families, respectively. Only a few Alu elements are able to retropose, and the factors determining their retroposition capacity are poorly understood. The data presented in this paper indicate that the length of Alu "A-tails" is one of the principal factors in determining the retropositional capability of an Alu element.

The A stretches of the Alu subfamilies analyzed, both old (Alu S and J) and young (Ya5), had a Poisson distribution of A-tail lengths with a mean size of 21 and 26,

respectively. In contrast, the A-tails of very recent Alu insertions (disease causing) were all between 40 and 97 bp in length.

The L1 elements analyzed displayed a similar tendency, in which the "disease"-associated elements have much longer A-tails (mean of 77) than do the elements even from the young Ta subfamily (mean of 41). Analysis of the draft sequence of the human genome showed that only about 1000 of the over one million Alu elements have tails of 40 or more adenosine residues in length.

The presence of these long A stretches shows a strong bias toward the actively amplifying subfamilies, consistent with their playing a major role in the amplification process. Evaluation of the 19 Alu elements retrieved from the draft sequence of the human genome that are identical to the Alu Ya5a2 insert in the NF1 gene showed that only five have tails with 40 or more adenosine residues. Sequence analysis of the loci with the Alu elements containing the longest A-tails (7 of the 19) from the genomes of the NF1 patient and the father revealed that there are at least two loci with A-tails long enough to serve as source elements within our model. Analysis of the A-tail lengths of 12 Ya5a2 elements in diverse human population groups showed substantial variability in both the Alu A-tail length and sequence homogeneity. On the basis of these observations, **a model is presented for the role of A-tail length in determining which Alu elements**

are active. [The sequence data from this study have been submitted to GenBank under accession nos. AF504933-AF505511.]

retroposition

Simple backward **displacement** of a **structure** or **organ**, as the **uterus**, without **inclination**, bending, **retroversion**, or **retroflexion**.

From <http://www.dhushara.com/book/upd2/jan01/hgwww/hgp.htm>

In the human, coding sequences comprise less than 5% of the genome (see below), whereas repeat sequences account for at least 50% and probably much more. Broadly, the repeats fall into five classes:

- (1) transposon-derived repeats, often referred to as interspersed repeats;
- (2) inactive (partially) retroposed copies of cellular genes (including protein-coding genes and small structural RNAs), usually referred to as processed pseudogenes;
- (3) simple sequence repeats, consisting of direct repetitions of relatively short k-mers such as (A)_n, (CA)_n or (CGG)_n;
- (4) segmental duplications, consisting of blocks of around 10±300 kb that have been copied from one region of the genome into another region; and
- (5) blocks of tandemly repeated sequences, such as at centromeres, telomeres, the short arms of acrocentric chromosomes and ribosomal gene clusters. (These regions are intentionally under-represented in the draft genome sequence and are not discussed here.)

Classes of transposable elements. In mammals, almost all transposable elements fall into one of four types (Fig. 17), of which three transpose through RNA intermediates and one transposes directly as DNA. These are long interspersed elements (LINEs), short interspersed elements (SINEs), LTR retrotransposons and DNA transposons.

Three distantly related LINE families are found in the human genome: LINE1, LINE2 and LINE3. Only LINE1 is still active.

SINEs are wildly successful freeloaders on the backs of LINE elements. They are short (about 100±400 bp), harbour an internal polymerase III promoter and encode no proteins.

The promoter regions of all known SINEs are derived from tRNA sequences, with the exception of a single monophyletic family of SINEs derived from the signal recognition particle component 7SL. This family, which also does not share its 3' end with a LINE, includes the only active SINE in the human genome: the Alu element.

If the sequences above represent DNA sequences, then taking the max of f^*g corresponds to finding the maximum overlap between two DNA sequences

From http://ccrma-www.stanford.edu/~jos/mdft/Convolution_Theorem.html

Convolution Theorem

Theorem: For any $x, y \in \mathbb{C}^N$,

$$x * y \leftrightarrow X \cdot Y$$

Proof:

$$\begin{aligned} \text{DFT}_k(x * y) &\triangleq \sum_{n=0}^{N-1} (x * y)_n e^{-j2\pi nk/N} \\ &\triangleq \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} x(m)y(n-m) e^{-j2\pi nk/N} \\ &= \sum_{m=0}^{N-1} x(m) \underbrace{\sum_{n=0}^{N-1} y(n-m) e^{-j2\pi nk/N}}_{e^{-j2\pi mk/N} Y(k)} \\ &= \left(\sum_{m=0}^{N-1} x(m) e^{-j2\pi mk/N} \right) Y(k) \quad (\text{by the Shift Theorem}) \\ &\triangleq X(k) Y(k) \end{aligned}$$

This is perhaps the most important single [Fourier theorem](#) of all. It is the basis of a large number of applications of the [FFT](#). Since the FFT provides a [fast Fourier transform](#), it also provides [fast convolution](#), thanks to the convolution theorem. It turns out that using the FFT to perform convolution is really more efficient in practice only for reasonably long convolutions, such as $N > 100$. For much

longer convolutions, the savings become enormous compared with "direct" convolution. This happens because direct convolution requires on the order of N^2 operations (multiplications and additions), while FFT-based convolution

requires on the order of $N \lg(N)$ operations.

The following simple [Matlab](#) example illustrates how much faster convolution can be performed using the FFT:

```
>> N = 1024; % FFT much faster at this length
>> t = 0:N-1; % [0,1,2,...,N-1]
>> h = exp(-t); % filter impulse response = sampled exponential
>> H = fft(h); % filter frequency response
>> x = ones(1,N); % input = dc (any example will do)
>> t0 = clock; y = conv(x,h); t1 = etime(clock,t0); % Direct
>> t0 = clock; y = ifft(fft(x) .* H); t2 = etime(clock,t0); % FFT
>> sprintf(['Elapsed time for direct convolution = %f sec\n',...
    'Elapsed time for FFT convolution = %f sec\n',...
    'Ratio = %f (Direct/FFT)'],t1,t2,t1/t2)
```

ans =

```
Elapsed time for direct convolution = 8.542129 sec
Elapsed time for FFT convolution = 0.075038 sec
Ratio = 113.837376 (Direct/FFT)
```

From: [Martin \(martin.jonsson@cetus.se\)](mailto:martin.jonsson@cetus.se)

Subject: Re: ? Use FFT to do the **convolution** in finite interval

View: [Complete Thread \(17 articles\)](#)

Newsgroups: [comp.soft-sys.matlab](#)

[Original Format](#)

Date: 2001-12-13 00:38:35 PST

Convolution in finite interval can easily be done by correlation(xcorr) if you got Signal Processing Toolbox and the vectors are of equal length. $\text{conv}(a,b)=\text{xcorr}(a,\text{fliplr}(b))$

Example $\text{conv}(a,b)=\text{xcorr}(a,\text{fliplr}(b))$

```
a=[1 2 3 4 5 6 5 4 3 2 1 1 2 3 4 5];
```

```
b=[2 3 4 5 6 5 4 3 2 1 1 2 3 4 5 1];
```

```
k=xcorr(a,b);
```

```

figure,plot(k)
bb=fliplr(b);
c=conv(a,bb);%the same as xcorr(a,b)
hold on
plot(c,'r--')
%vector c and k will be the same

```

For finite conv interval use `xcorr(a,bb,lag)`

Example

```

lag=2;
convfinite=xcorr(a,bb,lag);%the same as conv(a,b) but with finite interval.
figure,plot(convfinite)

```

With lag you look at the center + 2(lag) samples backwards and 2(lag) forward.

If you don't have the signal processing toolbox or the vectors of different length you could just fix the vectors or write a little code snippet which does the **convolution**.

Example of finite **convolution**.

```

a=[1 1 1 2 2 2 3 3 3 4]; % length 10
b=[1 1 2 2 3 3 4]; % length 7
ct=conv(a,b);% length 10+7-1=16
%for finite conv.
start=3; %start of finite conv
stop=7; %end of finite conv
%bb=fliplr(b);%bb=[4 3 3 2 2 1 1];
for r=start:stop
    if (r > length(a))
        c(r-start+1)=sum( fliplr(b(r-length(a)+1:r)).*a );
    elseif (r > length(b))
        c(r-start+1)=sum( a(r-length(b)+1:r).*fliplr(b) );
    else
        c(r-start+1)=sum( a(1:r).*fliplr(b(1:r)) );
    end
end

```

%plots the whole **convolution** ct and the finite c in the same figure to compare.

```

figure,plot(ct)
hold on
plot([3 4 5 6 7],c,'r--')

```

from <http://www.nist.gov/dads/HTML/viterbiAlgorithm.html>

Viterbi algorithm

(algorithm)

Definition: An algorithm to compute the optimal (most likely) state sequence in a [hidden Markov model](#) given a sequence of observed outputs.

See also [Baum Welch algorithm](#).

Note: Also used to decode, i.e. remove noise from, linear error-correcting codes.

hidden Markov model

(data structure)

Definition: A variant of a [finite state machine](#) having a set of [states](#), Q , an output [alphabet](#), O , transition probabilities, A , output probabilities, B , and initial state probabilities, Π . The current state is not observable. Instead, each state produces an output with a certain probability, B . Usually the states, Q , and outputs, O , are understood, so an HMM is said to be a triple, (A, B, Π) .

Formal Definition: After [Michael Cohen](#).

- $A = \{a_{ij} = P(q_j \text{ at } t+1 \mid q_i \text{ at } t)\}$, where $P(a \mid b)$ is the conditional probability of a given b , $t \geq 1$ is time, and $q_i \in Q$.
Informally, A is the probability that the next state is q_j given that the current state is q_i .
- $B = \{b_{ik} = P(o_k \mid q_i)\}$, where $o_k \in O$.
Informally, B is the probability that the output is o_k given that the current state is q_i .
- $\Pi = \{p_i = P(q_i \text{ at } t=1)\}$.

Also known as HMM.

See also [Markov chain](#), [Baum Welch algorithm](#), [Viterbi algorithm](#).

Markov chain

(data structure)

Definition: A [finite state machine](#) with probabilities for each transition, that is, a probability that the next state is s_j given that the current state is s_i .

See also [hidden Markov model](#).

Note: Equivalently, a [weighted, directed graph](#) in which the weights correspond to the probability of that transition. In other words, the weights are nonnegative and the total weight of outgoing [edges](#) is positive. If the weights are normalized, the total weight, including [self-loops](#), is 1. After [\[Leda98\]](#).

Author: [PEB](#)

Baum Welch algorithm

(algorithm)

Definition: An algorithm to find [hidden Markov model](#) parameters A , B , and Π with the maximum likelihood of generating the given symbol sequence in the observation vector.

See also [Viterbi algorithm](#).

```
> interface(verboseproc=3) ;
> interface(warnlevel=4) ;
```

from <http://uirvli.ai.uiuc.edu/dugad/>

The Three Problems for HMMs

Most applications of HMMs are finally reduced to solving three main problems. These are :

Problem 1 : Given the model how do we compute $P(O)$, the probability of occurrence of the observation sequence $O = o_1, o_2, \dots, o_n$.

Problem 2 Given the model how do we choose a state sequence $I = i_1, i_2, \dots, i_n$ so that $P(O, I)$, the joint probability of the observation sequence $O = o_1, o_2, \dots, o_n$ and the state sequence given the model is maximized.

Problem 3 How do we adjust the HMM model parameters so that $P(O)$ (or $P(O, I)$) is maximized.

Problems 1 and 2 can be viewed as analysis problems while Problem 3 is a typical synthesis (or model identification or training) problem.

```
> restart;
Digits:= trunc(evalhf(Digits)):
n:= 100:
rnd01:= ()-> rand()/(10.^12-11):
X:= [seq(rnd01(), i= 1..n)]:
Y:= [seq(rnd01(), i= 1..n)]:
f:= (x,y)-> sqrt(-2*ln(x))*cos(2*Pi*y):
Z:= evalf(zip(f, X, Y)):
PLOT3D(POINTS([seq([X[i], Y[i], Z[i]], i= 1..n])),
SYMBOL(CIRCLE));
```

From <http://groups.google.com/groups?q=markov+entropy&hl=en&lr=&ie=UTF-8&oe=UTF-8&selm=3v3hrn%24kd8%40nntp5.u.washington.edu&rnum=4>

for an expository paper called "An Entropy Primer", or see the textbook "Elements of Information Theory" by Cover and Thomas, Wiley, 1991.

A related theorem (see "Primer"), the Equipartition Theorem, says that for ergodic text sources, the log of the probability of the particular message you observed is, for sufficiently large n , very close to the source entropy. (The source entropy is defined as you wrote it for a Bernoulli source, and in a slightly different way for Markov shifts; see "Primer" and other preprints linked to my home page, or see the textbook "An Introduction to Ergodic Theory" by Peter Walters, Springer, 1981.)

We should maximize $H(X) = -\sum\{pi*\log(pi)\}$ with the constraint that $\sum\{pi\} = 1$.
(We also have another constraint that $pi \geq 0$ for all i , but ignore it for the time.)

Lagrangian multiplier, say m , is introduced to solve the constrained maximization problem.

Now, $H_1(X) = -\sum\{pi*\log(pi)\} + m(\sum\{pi\}-1)$ should be maximized.

The fact that partial derivative of H_1 respective to p_i ($i = 1, 2, \dots, n$) and m should be 0 yields the desired solution.
(And, the solution also meets the constraint $p_i \geq 0$.)

For order-K **Markov** chain:

$$H = - \sum_{i=1}^m \dots \sum_{i_{K+1}=1}^m (p(a_{i1}, \dots, a_{iK+1}) \log(p(a_{iK+1} | a_{i1}, \dots, a_{iK}))),$$

where a_1, \dots, a_m are alphabet symbols, $p(x,y,z, \dots)$ is a probability of occurrence " x,y,z, \dots " in source output and $p(z | \dots, x,y)$ is a conditional probability of occurrence z after " \dots, x,y " in source output.

HIGH ORDER \implies LOW Entropy

So if a sequence is highly ordered, or in other words, highly predictable, then the entropy will be LOW.

So, random data will have high entropy.

From the net, example to exec program from java and get its output

> You might try something like this...

```
import java.io.*;

public class Test
{ public static void main (String args[])

    { try {

        Runtime myRunTime = Runtime.getRuntime();
        Process myProcess;
        InputStream theInputStream;
        byte[] theResultBytes;
        int numberResultBytes;
        int resultCode;
        String cmdString = "C:/test.bat";
        String theResultString;

        myProcess = myRunTime.exec(cmdString);
        theInputStream = myProcess.getInputStream();
```

```

myProcess.waitFor();
numberResultBytes = theInputStream.available();
System.out.println("numberResultBytes: " + numberResultBytes);
theResultBytes = new byte[numberResultBytes];
resultCode = theInputStream.read(theResultBytes);
theResultString = new String("");
while(resultCode != '\0' && resultCode != -1)
{ theResultString =
  new String(theResultString + new String(theResultBytes));
  resultCode = theInputStream.read(theResultBytes);
}
System.err.println(theResultString);
theInputStream.close();
}
catch(java.lang.InterruptedException e)
{ System.err.println("Something went wrong");
}

catch(java.io.IOException e)
{ System.err.println("Something went wrong");
}
}
}

```

from <http://www.asp.ucar.edu/colloquium/1992/notes/part1/node9.html>

The *mean* of a set of measurements $\{x_1, x_2, \dots, x_N\}$ is the average:

$$\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i. \quad (2.1)$$

The *expectation value* of a quantity is the value expected if averaged over the entire parent population, and will be denoted by angle brackets: $\langle \cdot \rangle$. For example, the mean in the parent population that corresponds to the sample mean \bar{x} is

$$\mu = \langle x \rangle = \lim_{N \rightarrow \infty} (\bar{x}).$$

So, the expected values is the AVERAGE, when done over the whole population.

When the sample size is smaller, then we called it the average. So, expectation is a stronger sense of the average.

- L. Allison. *Using Hirschberg's algorithm to generate random alignments*. Inf. Proc. Lett. **51** 251-255 1994.

- L. Allison and C. S. Wallace. *The posterior probability distribution of alignments and its application to parameter estimation of evolutionary trees and to optimization of multiple alignments*. *Jrnl. Molec. Evol.* **39** 418-430 1994.
- D. S. Hirschberg. *A linear space algorithm for computing maximal common subsequences*. *Inf. Proc. Lett.* **18** 341-343 1975.

D. S. Hirschberg.

Algorithms for the longest common subsequence problem.
J.ACM, 24:664-675, 1977.

<http://www.csse.monash.edu.au/~lloyd/tildeAlgDS/Dynamic/Hirsch/>

this below finds time and space complexity for hirsh. Algo

<http://www.math.tau.ac.il/~rshamir/algmb/98/scribe/html/lec02/node10.html>

gene finding and markov, time/space estimates:

<http://www.csse.monash.edu.au/~lloyd/tildeStrings/Notes/20001205HMMgene.html>

contains good gene struct

<http://www.fruitfly.org/GASP1/tutorial/presentation/sld009.htm>

BOGA Schematic gene structure

DNA:
Promoter Exon 1 Intron 1 Exon 2 Intron 2 Exon 3

transcription
E1 I1 E2 I2 E3

pre-mRNA:
E1 I1 E2 I2 E3

splicing
5'UTR ORF 3'UTR polyA

mRNA:
ATG IAA AAAAAAAAAA

translation
[k base open-reading frame]
ATG IAA
MPYCELTVGFL amino acid sequence

modification
[glycosylation site]

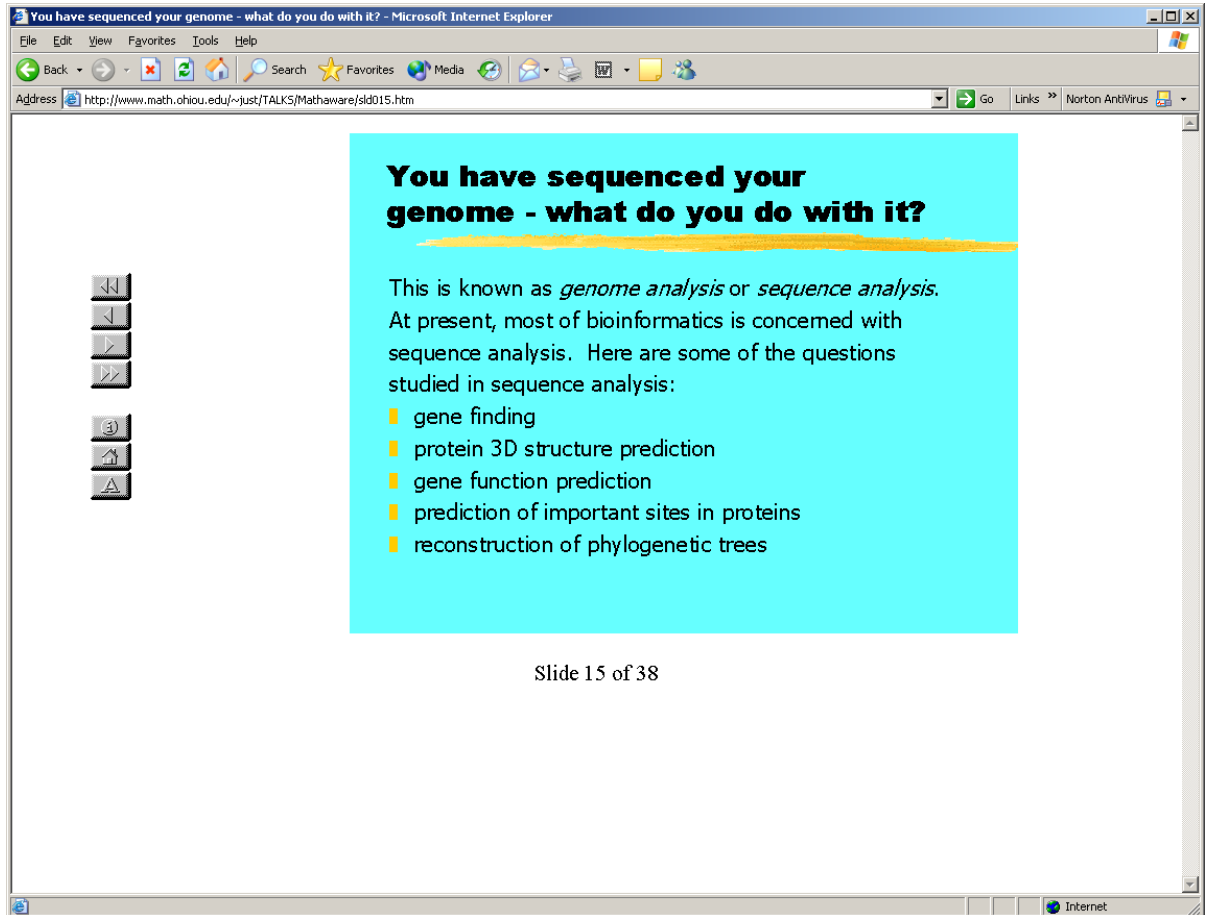
active protein:
CPLTVG

Reese et al., Tutorial #3, ISMB '99

Slide 9 of 182

from http://www.cs.mcgill.ca/~kaleigh/work/hmm/hmm_paper.html

A eukaryotic gene contains coding regions called exons which may be interrupted by non-coding regions called introns. The exons and introns are separated by splice sites. Regions outside genes are called intergenic. The goal of gene finding is then to annotate the sets of genomic data with the location of genes and within these genes, specific areas such as promoter regions, introns and exons.



You have sequenced your genome - what do you do with it?

This is known as *genome analysis* or *sequence analysis*. At present, most of bioinformatics is concerned with sequence analysis. Here are some of the questions studied in sequence analysis:

- gene finding
- protein 3D structure prediction
- gene function prediction
- prediction of important sites in proteins
- reconstruction of phylogenetic trees

Slide 15 of 38

Check this:

[10]

Kanungo, Tapas. HMM software learning toolkit. University of Maryland Institute for Advanced Computer Studies, Center for Automation Research, Language and Media Processing Lab - <http://www.cfar.umd.edu/kanungo/software/software.html>.

http://www.cs.ualberta.ca/~colinc/cmput606/606FinalPres.ppt - Microsoft Internet Explorer

File Edit Browse Go To Favorites Help

Back Forward Stop Home Search Favorites Media

Address http://www.cs.ualberta.ca/~colinc/cmput606/606FinalPres.ppt Go Links

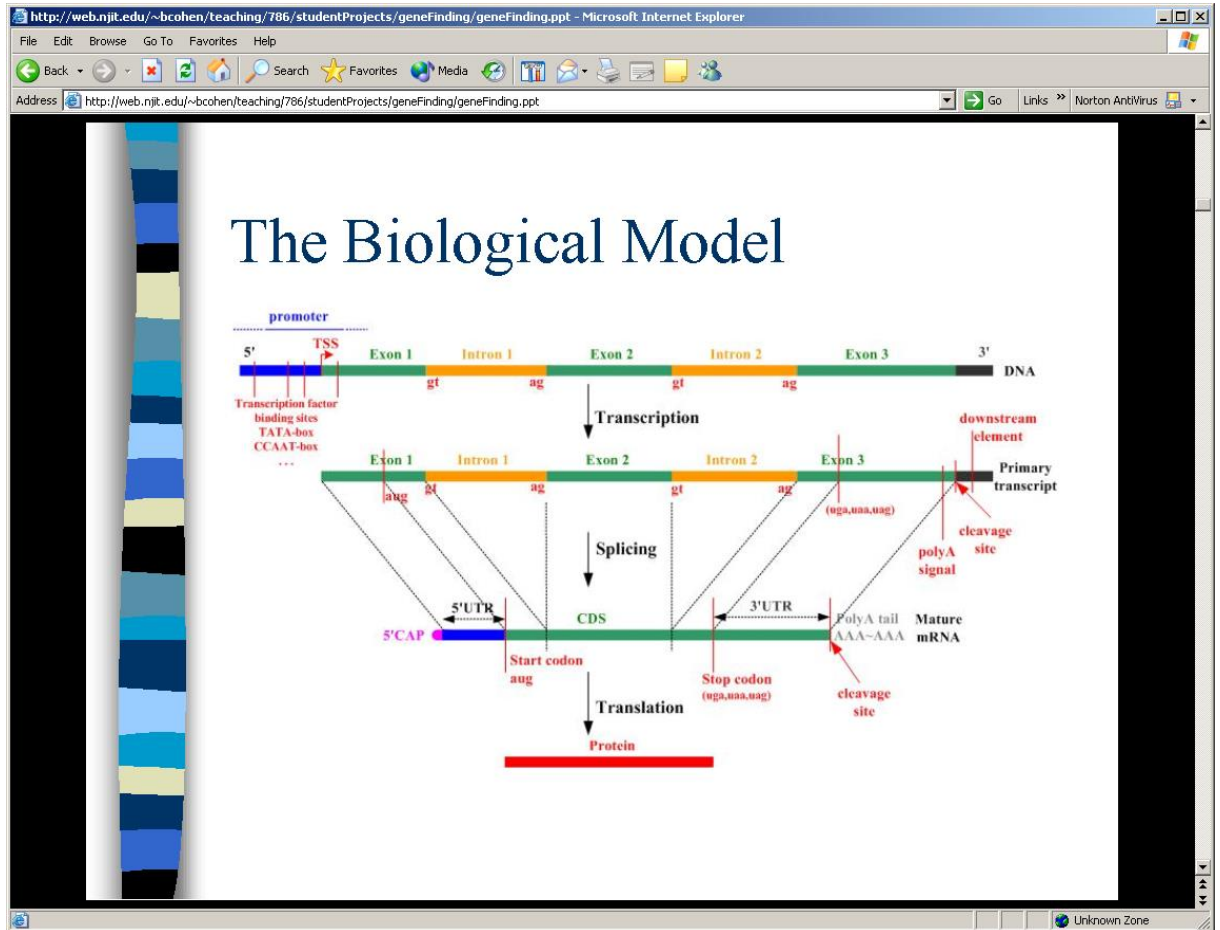
HMM Assumptions

- Observations are ordered
- Random process can be represented by a stochastic finite state machine with emitting states.

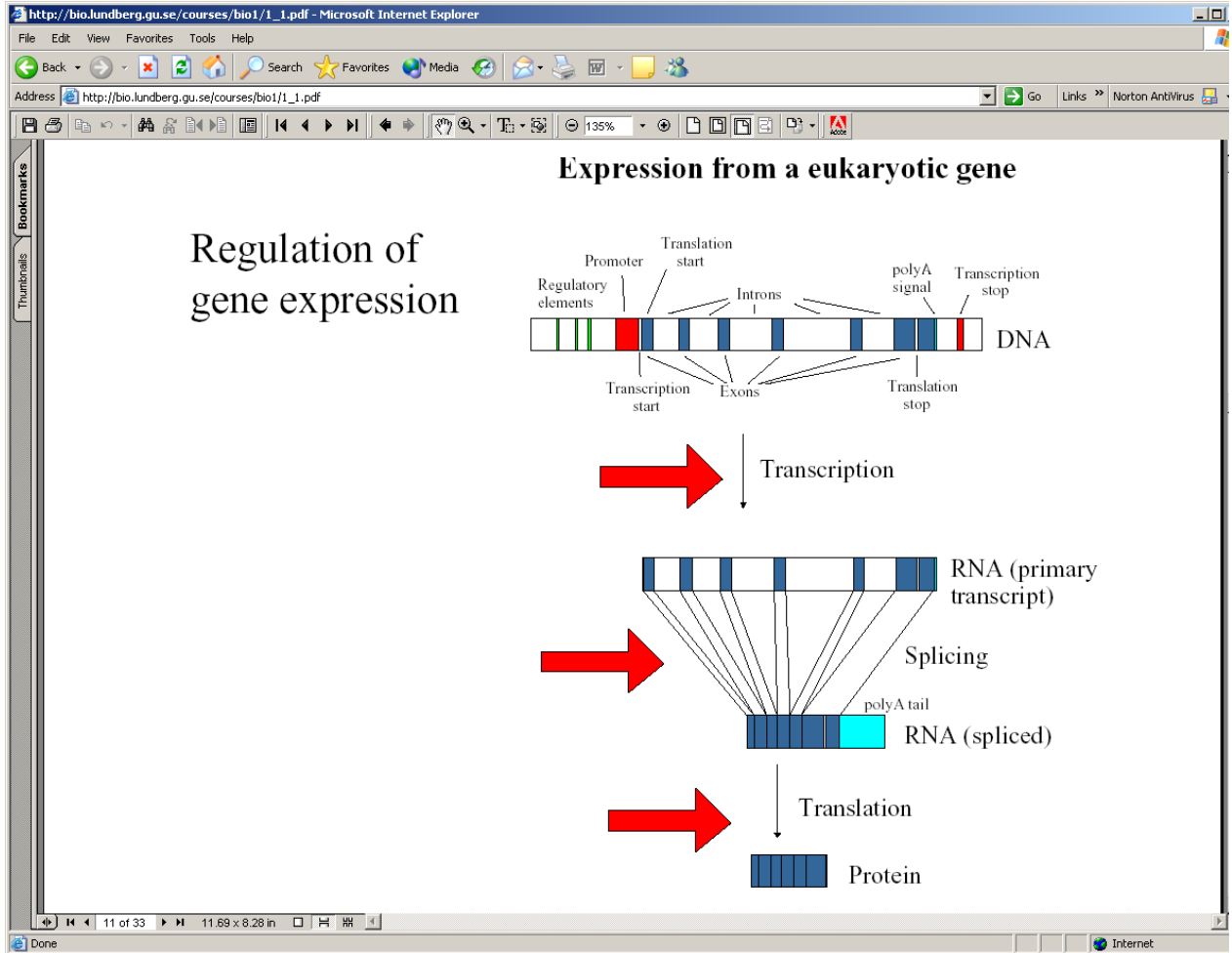
The diagrams illustrate various HMM models:

- Top Left:** A state transition diagram with states Start, a, and End, and emitting states A3, B1, B2, A4, B3, A5, B5.
- Top Right:** A complex state transition diagram with states Start Codon, 16 Backdrops, 3' Splice Site, and 5' Splice Site.
- Bottom Left:** A state transition diagram with states S, B, M1, M2, M3, M4, C, and T.
- Bottom Center:** A seasonal cycle diagram with states Spring, Summer, Fall, and Winter.
- Bottom Right:** A codon model diagram showing a sequence of codons: start codon (A-T-G), Codon model (three grey boxes), and stop codon (T-A-A).

from
<http://web.njit.edu/~bcohen/teaching/786/studentProjects/geneFinding/geneFinding.ppt>



this below from http://bio.lundberg.gu.se/courses/bio1/1_1.pdf



from http://www.genomicglossaries.com/content/proteins_glossary.asp

Poly A: A group of adenine ribonucleotides in which the phosphate residues of each adenine ribonucleotide act as bridges in forming diester linkages between the ribose moieties. [MeSH, 1976]

polyadenylation: The addition of a tail of polyadenylic acid (POLY A) to the 3' end of mRNA (RNA, MESSENGER). Polyadenylation involves recognizing the processing site signal, (AAUAAA), and cleaving of the mRNA to create a 3' OH terminal end to which poly A polymerase (POLYNUCLEOTIDE ADENYLTRANSFERASE) adds 60- 200 adenylate residues. The 3' end processing of some messenger RNAs, such as histone mRNA, is carried out by a different process that does not include the addition of poly A as described here. [MeSH, 2002]

During the maturation of messenger RNA, about 200 adenosine nucleotides are added in a polyadenylation reaction at the 3' end. These are not coded by the corresponding gene. In certain cases there are multiple alternative polyadenylation sites in the primary transcript. This was first observed in adenoviruses [127 - 131]. In cellular genes many alternative polyadenylation sites have also been found [see 132 for review]. Alternative polyadenylation sites usually involve the untranslated trailer sequence in the messenger RNA, but they can also involve translated sequences, and in this case they can affect the structure of the encoded protein. Thus multiple polyadenylation sites are one mechanism whereby a single gene can control the synthesis of more than one polypeptide. [Petter Portin in "The Origin, Development and Present Status of the Concept of the Gene: A Short Historical Account of the Discoveries" Univ. of Turku, Finland, 2000] <http://www.bentham.org/cg1-1/portin/P.Protin.htm>

translation: The unidirectional process that takes place on the **ribosomes** whereby the genetic information present in an **mRNA** is converted into a corresponding sequence of amino acids in a **protein**. [IUPAC Bioinorganic

transcription, genetic: The transfer of genetic information from DNA to messenger RNA by DNA- directed RNA polymerase. It includes reverse transcription and transcription of early and late genes expressed early in an organism's life cycle or during later development. [MeSH, 1973]

MAPLE:

evelyn wrote:

Just plot f,g,h with their ranges like

```
fplot := plot(f(x),x=a..b);
```

and then do

```
with(plots):
```

```
display({ fplot,gplot,hplot});
```

Cheers,

Raphael

> Hello,

>

> I want to plot 3 different functions f(x), g(x), h(x) in one plot and

> different colors. But every function has a different range. I will

> give an example:

>

```

> funcion 1: f in funcion of x in a range from x=0 to x=0.1
> funcion 2: g in funcion of x in a range from z=0.1 to x=2.5
> funcion 3: h in funcion of x in a range from x=2.5 to x=4
>
> Is that possible?
>
> Evelyn

```

CTG has been described as **start codon** for

start codon (initiation codon)

The triplet of nucleotides on a messenger [RNA](#) molecule (see [codon](#)) at which the process of [translation](#) is initiated. In eukaryotes the start codon is AUG (see [genetic code](#)), which codes for the amino acid methionine; in bacteria the start codon can be either AUG, coding for *N*-formyl methionine, or GUG, coding for valine. Compare [stop codon](#).

This below takes a little on splice sites

<http://www.fruitfly.org/GASP1/doc/format.html>

maple

```
>plot3d('findstable(1,1,1,1,x,y,1)', x=0..1, y=0..1);
```

```

use `=` Equals in
  x= 9; y= 9; z= 9;
  x= 9*6; x= x*3
  # Any number of statements can go in the use block
end use;

```

EXAM prep

Knots.

Q1. application to DNA supercoiling.

Q2. explain topoisomerases and how it works.

A2.

See <http://www.mun.ca/biochem/courses/3107/Lectures/Topics/supercoiling.html>

The degree of supercoiling in the cell must be and is carefully controlled by the action of topoisomerases.

These enzymes catalyse the transient breaking and rejoining of DNA strands.

Enzymes that change the degree of supercoiling in DNA by cutting one or both strands. Type I topoisomerases cut only one strand of DNA; type I topoisomerase of *E. coli* (*E. coli* ω protein) relaxes negatively supercoiled DNA and does not act on positively supercoiled DNA.

Type II topoisomerases cut both strands of DNA; type II topoisomerase of *E. coli* (DNA gyrase) increases the degree of negative supercoiling in DNA and requires ATP. It is inhibited by several antibiotics, including nalidixic acid and ovobiocin.

The major role of topoisomerases is to prevent DNA tangling.

As a result, the type I enzyme removes supercoils from DNA one at a time, whereas the type II enzyme removes supercoils two at a time. Although the type II topoisomerase is more efficient in removing supercoils, this enzyme requires the energy from ATP hydrolysis, but the type I topoisomerase does not.

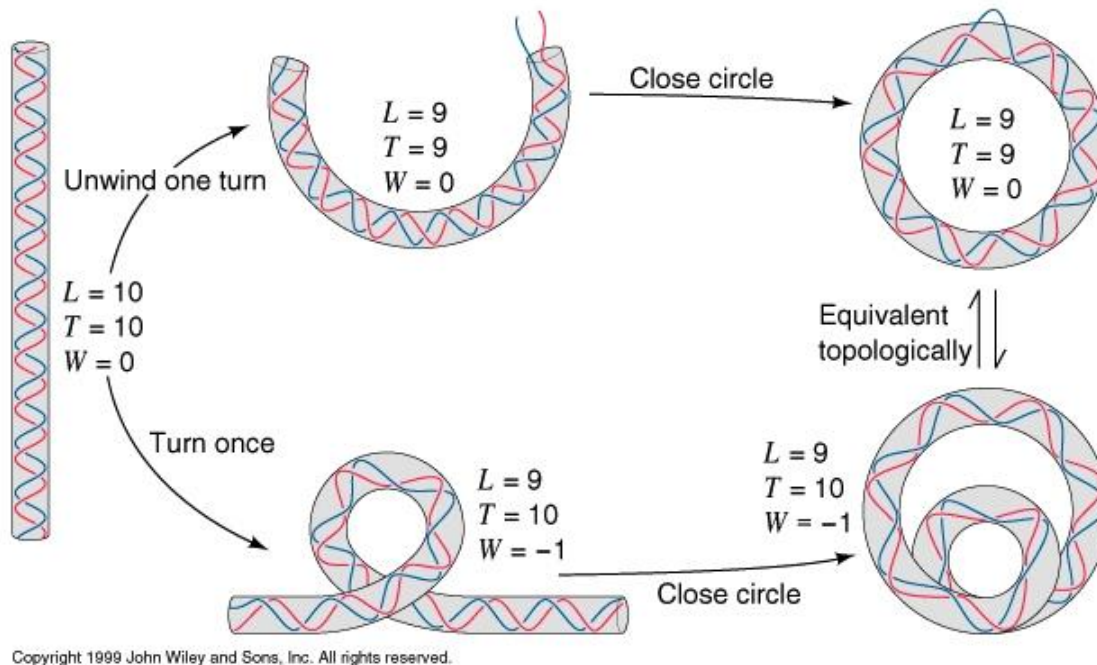
Without topoisomerases, the DNA cannot replicate normally.

Q3 what is a link number

A3 Number of times two DNA strands interwinds each others.

Another: This is only defined for two strands, not a single strand. Supercoiling in circular DNA molecules is described mathematically by the number of times the two phosphodiester backbones wrap around one another in a given distance. This quantity is the **Linking number** (Lk). If either one of the two DNA strands is nicked, this value has no meaning.

Because the entire molecule will be a closed circle, the linking number will always be an integer.

**Q4. what is the twist**

A4. One definition: number of times basepairs twist around the central axis.

Another: The twist of a ribbon measures how much it twists around its axis and is defined as the integral of the incremental twist around the ribbon

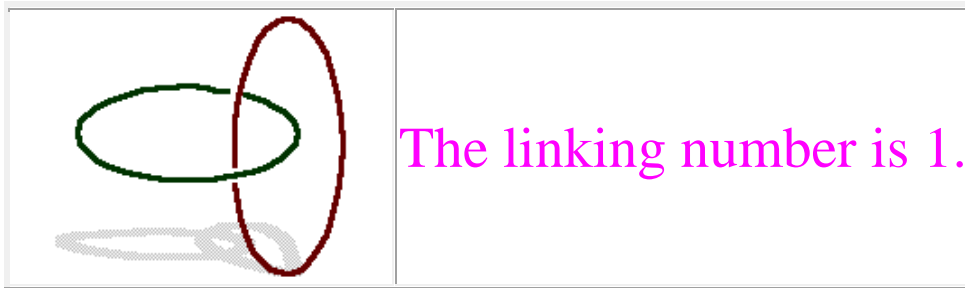
Q4.1 What is the Writhe number

Is a simple function of only the molecule axis vector \mathbf{r} .

$$Wr = \frac{1}{4\pi} \iint ds ds' \frac{\partial_s \mathbf{r}(s) \times \partial_{s'} \mathbf{r}(s') \cdot [\mathbf{r}(s) - \mathbf{r}(s')]}{|\mathbf{r}(s) - \mathbf{r}(s')|^3}. \quad (11)$$

Q4.2 How to find the link number

This is defined ONLY for 2 components. A KNOT does not have a link number, it has a crossing number. Linking number measures how many times one components go around the other.



either do the $-1, +1$ sum, and then divide by 2 the final result as shown below:

A link invariant defined for a two-component oriented link as the sum of $+1$ crossings and -1 crossing over all crossings between the two links divided by 2.

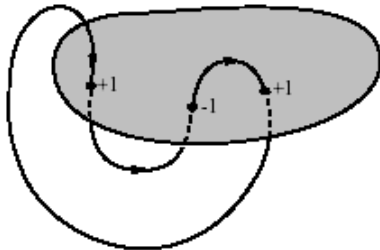
For components α and β ,

$$Lk(\alpha, \beta) \equiv \frac{1}{2} \sum_{p \in \alpha \cap \beta} \epsilon(p),$$

OR use this method below. I use this method to find the linking number between two components, (it is easier)

From paper: <http://www.biophysics.org/btol/img/Vologodskii.A.pdf>

A nice way to find the link number. Take one strand, and make it a closed surface. Then find how many times the other strand cuts the contour. If it cuts the contour twice from the same side one after the other, then they both cancel each other. Now simply add the number the other strand cuts the closed surface.



Just make one component shaded as above, and then add arrows to make sure the direction of one strand is kept constant, then count how many times the strands crosses the shaded plan, as above.

Q5. what is crossing number

The least number of crossings that occur in any projection of a [link](#). In general, it is difficult to find the crossing number of a given [link](#). Knots and links are generally tabulated based on their crossing numbers.

Q6. What is a knot

A knot is defined as a closed, non-self-[intersecting](#) curve embedded in three dimensions.

A knot is a single component [link](#)

i.e. a knot is a CLOSED curve in 3D space that does not intersect itself.

Teacher definition: a knot is a function which maps a unit circle into R^3 , and a link is a function which maps a collection of disjoint unit circles into R^3 .

This is another definition: A **knot** is a polygon embedded in threespace

Q7. what is a component

More informally, a link or component is an assembly of [knots](#) with mutual entanglements.

Q8. what does it mean that 2 knots are equivalent?

A8. This means there exist a number of Reidemeister moves that transform one knot to the other.

Q9 what is Jones polynomial

A9. It is a knot invariant in the form of a [polynomial](#)

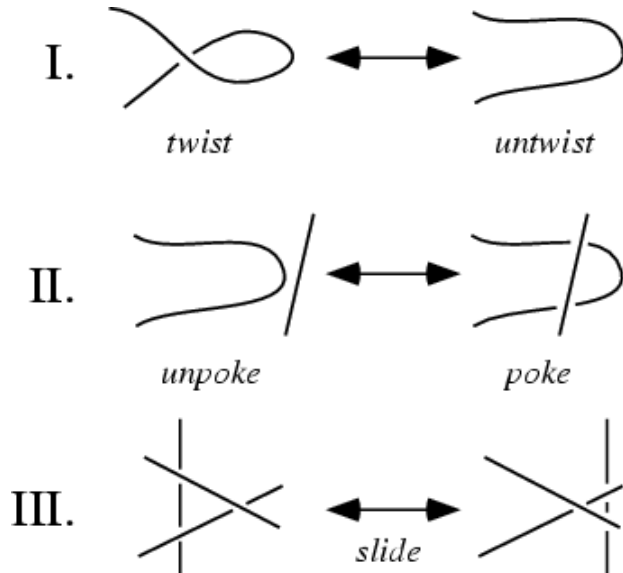
Q10. what are the Reidemeister moves

A10. twist, poke, slide.

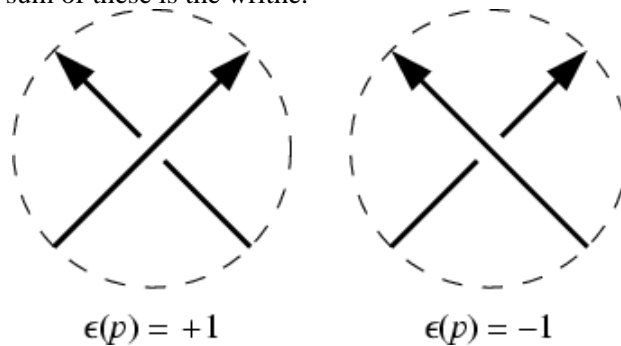
Reidemeister first rigorously proved that [knots](#) exist which are distinct from the [unknot](#).

He did this by showing that all [knot](#) deformations can be reduced to a sequence of three types of "moves,"

NOTE: do both sense for this. When up/down.

**Q11. What is the directed writhe ?**

A11. This is when we give an orientation to the knot projection, at each crossing we have a +1 or a -1. The sum of these is the writhe.



Also called the [twist number](#). The sum of crossings p of a [link](#) L ,

$$w(L) = \sum_{p \in C(L)} \epsilon(p),$$

Q12. What is the calugareanu formula.

A12. Link number = Twist number + writhe number

Writhe number can be found as above, by counting +1, -1, or by an integral

A formula for the writhe is given by

$$\text{Wr}(K) = \frac{1}{4\pi} \int_K ds \int_K dt e^\mu \frac{de^\mu}{ds} \frac{de^\alpha}{dt}$$

Twist number: The twist of a ribbon measures how much it twists around its axis and is defined as the integral of the incremental twist around the ribbon. A formula for the twist is given by

$$\text{Tw}(K) = \frac{1}{2\pi} \int_K ds \varepsilon_{\mu\nu\alpha} \frac{dx^\mu}{ds} n^\nu \frac{dn^\alpha}{ds},$$

Link number is A [link invariant](#) defined for a two-component oriented [link](#) as the sum of $+1$ crossings and -1 crossing over all crossings between the two links divided by 2. For components α and β ,

$$\text{Lk}(\alpha, \beta) \equiv \frac{1}{2} \sum_{p \in \alpha \cap \beta} \epsilon(p),$$

Q13 When are 2 knots equivalent?

Definition 4 Let L, L' be knots (links). Then, $L \sim L'$ if the knot diagram of L' can be obtained from the knot diagram of L using a finite sequence of *Reidemeister moves*.

Q14. How does assembly work?

A14. Assembly means finding the shortest common superstring for all the reads. This is an NP complete problem. So use the greedy algorithm. This is done in two steps. First find best overlap between each 2 reads, next put the reads together. For first step, use convolution to find max overlap.

Q15. What is coverage in assembly?

A15. This is a measure of how many times a position is found in different reads. For example, 6x coverage means each position (base) is covered on 6 different reads or fragments.

Q16. what is a gene

A segment of a [DNA molecule](#) that contains all the information required for [synthesis](#) of a product ([polypeptide chain](#) or [RNA molecule](#)), including both coding and non-coding sequences. It is the [biological](#) unit of [heredity](#), self-reproducing, and transmitted from parent to [progeny](#). Each gene has a specific position ([locus](#)) on the [chromosome](#) map. From the standpoint of function, genes are conceived of as [structural](#), [operator](#), and [regulatory genes](#).

OR

A gene is a segment of a chromosome that encodes instructions that allow a cell to produce a specific product, typically a protein, that initiates one specific action.

Q17. what is an exon?

A segment of a eukaryotic gene that is transcribed as part of the primary transcript and is retained, after processing, with other exons to form a functional mRNA molecule. See DNA; intron; split gene; splicing.

Q18. what is an intron?

intron; intervening sequence A segment of DNA sequence of a eukaryotic gene, not represented in the mature (final) mRNA transcript, because it is spliced out of the primary transcript before it can be translated; a process known as intron splicing. Some genes of higher eukaryotes contain a large number of introns, which make up the bulk of the DNA sequence of the gene. Introns are also found in genes whose RNA transcripts are not translated, namely eukaryotic rRNA and tRNA genes. In these cases the intron sequence does not appear in the functional RNA molecule. *cf* exon.

DNA --- (transcript) ---> preRNA (intron+exons) --- (splice) --> RNA --- (translate) -> protein

Q19. what is HMM

A statistical method that describes a series of observations (in our case nucleotides) by a hidden stochastic process.

Another nice definition. It is like a profile. From some observations, we deduce hidden actions.

Q20. what is a suffix tree.

A given suffix tree can be used to search for a substring, $pat[1..m]$ in $O(m)$ time.

Q21: What repeats are there?

A. Simple and interspread repeats. Simple like ACACACAC, while interspread are LINE (Long Interspread repeats) and SINS (Short INTerspread repeats).

Q22. What is electrophoresis?

A. Sorting DNA pieces by length. DNA travel across a potential in a rate of travel $\sim 1/\text{Log}L$.

From help on clustALW <http://www-igbmc.u-strasbg.fr/BioInfo/ClustalW/help2.html>

Multiple alignments are carried out in 3 stages (automatically done from menu item 1 ...Do complete multiple alignments now):

1. all sequences are compared to each other (pairwise alignments);
2. a dendrogram (like a phylogenetic tree) is constructed, describing the approximate groupings of the sequences by similarity (stored in a file).
3. the final multiple alignment is carried out, using the dendrogram as a guide

this below also does multiple protein seq. alignment.

http://npsa-pbil.ibcp.fr/cgi-bin/npsa_automat.pl?page=npsa_clustalw.html

on divide-and-conquer, from

<http://www.cse.ucsc.edu/research/compbio/papers/samspace/node2.html>

The solution to this is to use a sequence alignment method that requires less space. In the case of finding the single best path, there is an elegant divide-and-conquer algorithm that requires only $O(n+m)$ space, where n is the sequence length and m is the model length [[Hirschberg, 1975](#)]. The approach of this algorithm is to find a midpoint of the best path without saving all $O(nm)$ dynamic programming entries, and then to solve two smaller problems, each of approximate size $nm/4$ using the same algorithm. This algorithm is well known in the computational biology community [[Myers & Miller, 1988](#)], and has, for example, been implemented in the HMMer package for sequence alignment to a trained HMM [[Eddy et al., 1995](#)].

More interesting and more complicated is the **number** of **unlabeled trees** on n vertices ;^)

Here you can get an asymptotic in R. Otter: The **number** of **trees** in the Annals of Mathematics, Vol 49, No 3 July 1948.

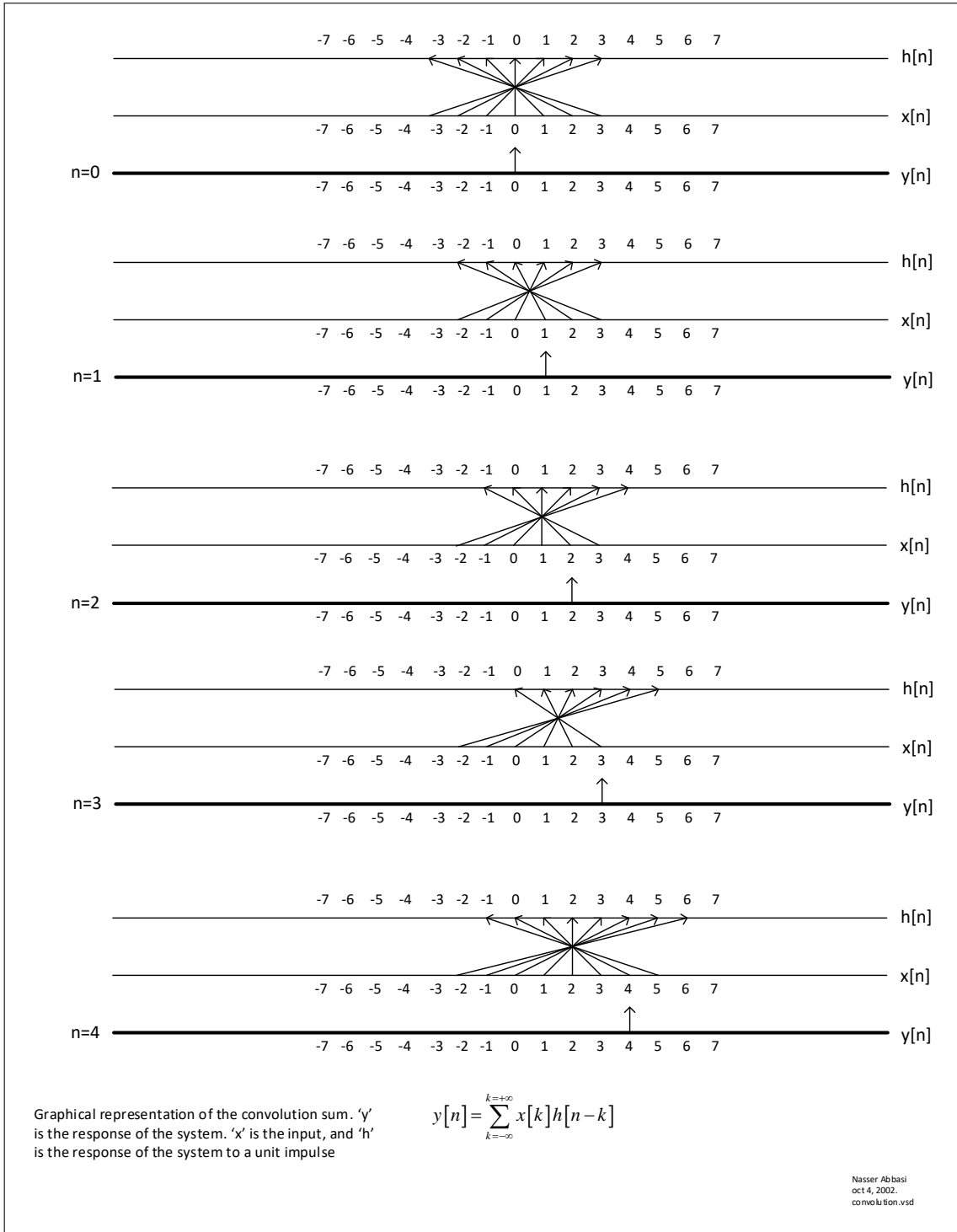


Figure 2.1: convolution diagram

2.2 FFT project

[up](#)

Notes by Nasser M. Abbasi

During math 127, UC Berkeley, 2002

This is a simple way to remember how to calculate the FFT of a vector.

If n is the number of coordinates (or data points) in the vector x , then let

$$X = \text{fft}(x)$$

X is a complex vector of the same number of coordinates (or data points) as x

Let $x = (a, b, c)$ be the vector (possibly complex) that we want to find the fft for.

We will do a dot product of the above vector with vectors whose coordinates are the roots of unity.

Recall that there are n roots ε such that $\varepsilon^n = 1$

But

$$1 = \cos(2\pi) + i \sin(2\pi) = e^{2\pi i}$$

so the n roots of unity are

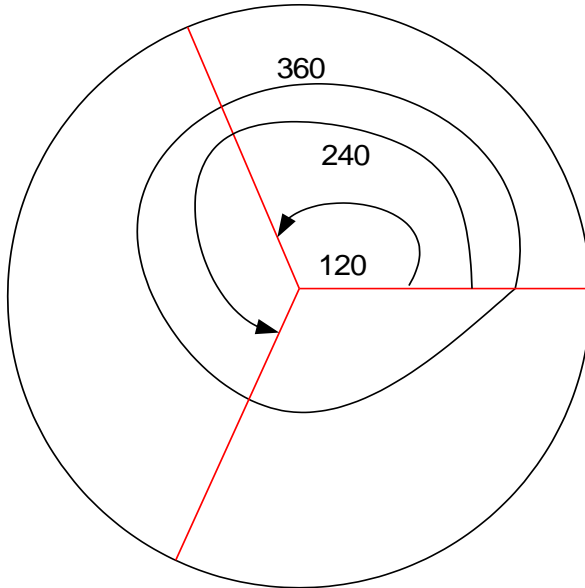
$$\varepsilon = 1^{\frac{1}{n}} = \cos\left(\frac{2\pi}{n}\right) + i \sin\left(\frac{2\pi}{n}\right) = e^{\frac{2\pi i}{n}}$$

So, we divide the angle 2π by the number of roots, and each root will have the same magnitude of 1, but it will be at an angle of $k\left(\frac{2\pi}{n}\right)$ multiples where $k = 0, 1, 2, \dots, n-1$.

This is because, with complex numbers, when we multiply one by the other, we add angles. Hence when we multiply a complex number by itself n times, we add n times the angle it had with the x -axis. Since we want to get 1 at the end (which has 360 angle), we divided 360 by n to get the above equation.

So, for $n=1$ there is one root, which is 1. for $n=2$ there are 2 roots, which are for $k=0, 1$, which are 1 and $e^{\pi i} = -1$ and so on.

To see this better, use the argand diagram. For example, this below are the 3 roots of unity. Since $n=3$, then we divide 360 degrees by the number of roots, and each unity root has an angle of $\frac{2\pi}{3}$ or 120 degrees away from the previous root.



3 roots of unity. Hence $360/3 = 120$ degrees.

What does the roots of unity have to do with FFT?

Let me show how they are used.

In the case of $n = 3$ (number of coordinates, or number of data points), we construct the 3 roots of unity.

Let $\omega = \exp^{\frac{2\pi i}{n}}$, then the roots of unity be written down as

$$\omega = (\omega^0, \omega^1, \dots, \omega^{n-1})$$

but $n = 3$, so we get

$$\omega = (\omega^0, \omega^1, \omega^2)$$

So, the exponent multipliers above, are the angle multipliers

Now, from this one set of roots of unity shown above, generate n sets by multiplying the exponents of ω inside the brackets by zero, then by one, then by two, then by three, etc... until $n-1$. When we multiply the exponent, this means we are rotating the root of unity vector around.

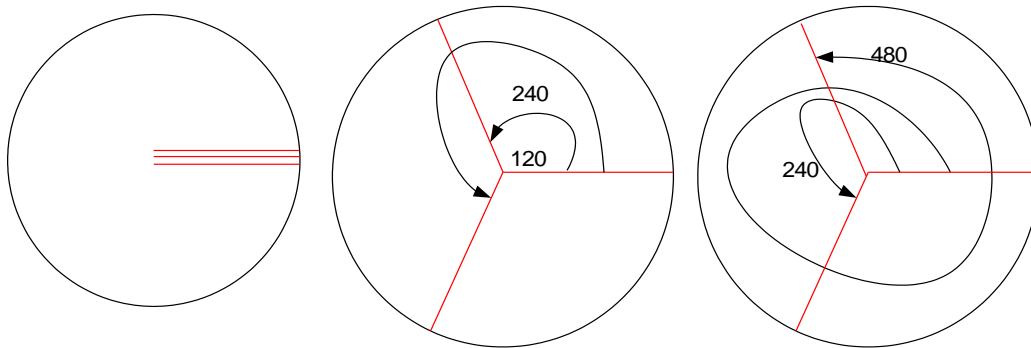
Since $n = 3$ here, we will get the 3 different sets of roots of unity, all generated from the original $(\omega^0, \omega^1, \omega^2)$:

$$(\omega^0, \omega^0, \omega^0)$$

$$(\omega^0, \omega^1, \omega^2)$$

$$(\omega^0, \omega^2, \omega^4)$$

This is a graphical representation of the above 3 sets



Notice that the roots are the same, we just change the angle of rotation to get to the root each time.

Now, align the x vector on top of these roots of unity vectors, we get

$$(a, b, c)$$

$$(\omega^0, \omega^0, \omega^0)$$

$$(\omega^0, \omega^1, \omega^2)$$

$$(\omega^0, \omega^2, \omega^4)$$

Now to get the coordinates of X , do the dot product of x with each of the vectors below it one at a time. Remember that the dot product of two vectors is just one number (possibly complex) and not a set of numbers (or a vector).

So, the first coordinate of X will be

$$(a, b, c) \bullet (\omega^0, \omega^0, \omega^0)$$

And the second coordinate of X will be the dot product of x with the second vector of the roots of unity, that is

$$(a, b, c) \bullet (\omega^0, \omega^1, \omega^2)$$

And the third and final coordinate will be

$$(a, b, c) \bullet (\omega^0, \omega^2, \omega^4)$$

and the n^{th} coordinate is

$$(a, b, c) \bullet (\omega^0, \omega^{1* n}, \omega^{2* n})$$

Example

Let me show this with an simple example. Let

$$x = (1, 4, 5, 6)$$

be the data we want to find its FFT. Here $n = 4$, hence

$$\omega = (\omega^0, \omega^1, \omega^2, \omega^3)$$

so we need 4 vectors of roots of unity generated from the above by multiplying the exponents by 0,1,2 and 3 at a time, we get

$$(\omega^0, \omega^0, \omega^0, \omega^0)$$

$$(\omega^0, \omega^1, \omega^2, \omega^3)$$

$$(\omega^0, \omega^2, \omega^4, \omega^6)$$

$$(\omega^0, \omega^3, \omega^6, \omega^9)$$

Now do the dot product of x with each one of these vectors one at a time. Each time we do a dot product, we get one data point in the FFT domain generated.

Notice that

$$\omega^0 = e^{0\left(\frac{2\pi i}{4}\right)} = 1$$

$$\omega^1 = e^{1\left(\frac{2\pi i}{4}\right)} = e^{\frac{i\pi}{2}}$$

$$\omega^2 = e^{2\left(\frac{2\pi i}{4}\right)} = e^{i\pi}$$

$$\omega^3 = e^{3\left(\frac{2\pi i}{4}\right)} = e^{\frac{3\pi i}{2}}$$

$$\omega^4 = e^{4\left(\frac{2\pi i}{4}\right)} = e^{2\pi i}$$

$$\omega^6 = e^{6\left(\frac{2\pi i}{4}\right)} = e^{3\pi i}$$

$$\omega^9 = e^{9\left(\frac{2\pi i}{4}\right)} = e^{\frac{9\pi i}{2}}$$

notice that

$$e^{i\theta} = \cos \theta + i \sin \theta$$

so we get

$$\omega^0 = 1$$

$$\omega^1 = i$$

$$\omega^2 = -1$$

$$\omega^3 = -i$$

$$\omega^4 = 1$$

$$\omega^6 = -1$$

$$\omega^9 = i$$

so, our 4 vectors of unity are now

$$(1, 1, 1, 1)$$

$$(1, i, -1, -i)$$

$$(1, -1, 1, -1)$$

$$(1, -i, -1, i)$$

Now do the dot product of x with each of the above vectors, and this will give us the FFT.

$$(1, 4, 5, 6) \bullet (1, 1, 1, 1) = 16$$

$$(1, 4, 5, 6) \bullet (1, i, -1, -i) = -4 - 2i$$

$$(1, 4, 5, 6) \bullet (1, -1, 1, -1) = -4$$

$$(1, 4, 5, 6) \bullet (1, -i, -1, i) = -4 + 2i$$

so,

$$FFT[x] = FFT \begin{bmatrix} 1 \\ 4 \\ 5 \\ 6 \end{bmatrix} = X = \begin{bmatrix} 16 \\ -4 - 2i \\ -4 \\ -4 + 2i \end{bmatrix}$$

2.3 Small note on terfoil_combinations

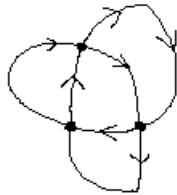
[up](#)

A small note on knot generation

Nasser M. Abbasi

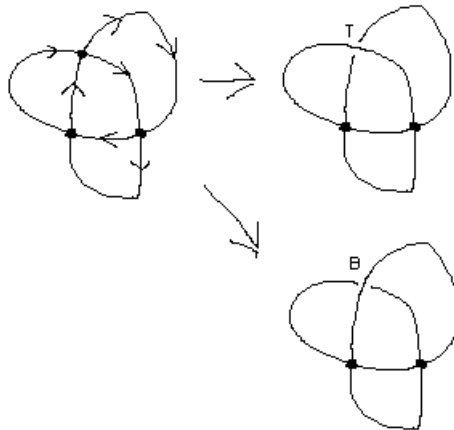
8/30/2002.

I want to generate all the knot diagrams from the planer graph generated from projecting a knot diagram into a plane.



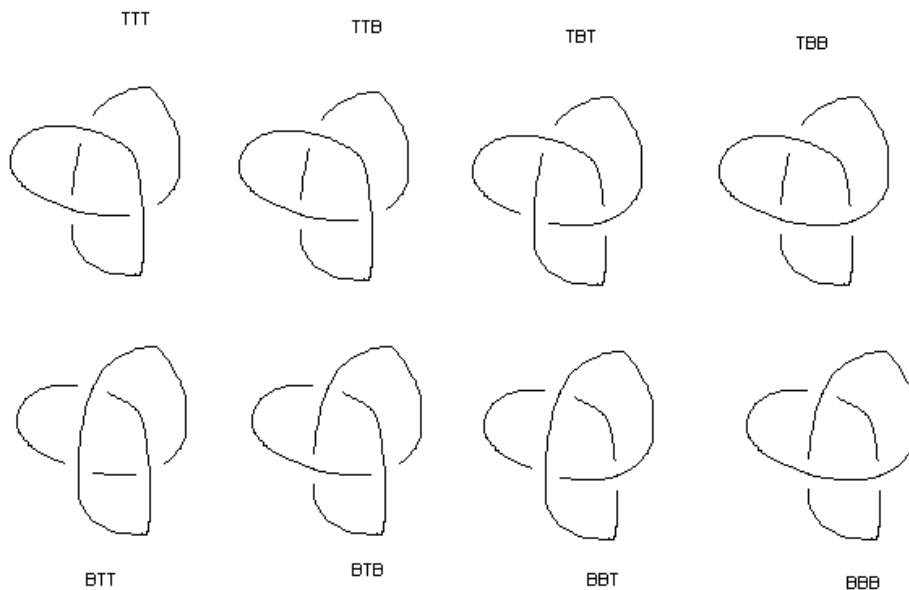
The above knot diagram has 3 vertices. So total knots that can be generated from it as 2^n or 8 diagram. This a simple way to generate those diagram. Assign a direction from any point on the curve and follow that all the way around until you get back to the starting point. At each crossing decide if the line will cross on TOP or BOTTOM of the other curve. Assign the letter T when going on TOP of the other curve, and assign the letter B when going below the other curve.

So, at each vertex, we have a choice of a T or a B :



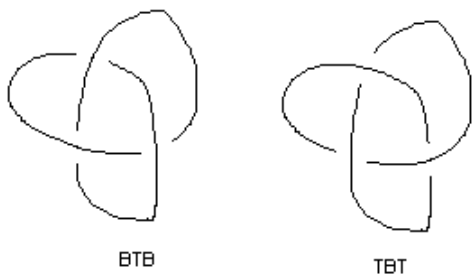
So, at the end we will have these permutations: TTT, TTB, TBT, TBB, BTT, BTB, BBT, BBB

draw all the above 8 combinations, we get:



The interesting thing, is that only TBT and BTB are knots. The rest are unknot. So, when we have the same letter following each others, we know right away that this is a unknot.

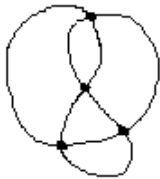
These are the only 2 unknots out of the 8 knots:



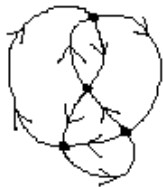
These are the left and right handed trifoil knots.

The question now is, can this be generalized? I.e., for any M vertices knot graphs, is it true that if we get a sequences of T's and B's with more than T or B following each other, then we have an unknot? Also, what is the ratio of the unknot to the knot being generated? In this case, we have 2/8 knots and 6/8 are unknot.

Let me try it with the figure 8 knot. This is the planer projection, it has 4 vertices. So total number of knot diagrams is $2^4 = 16$

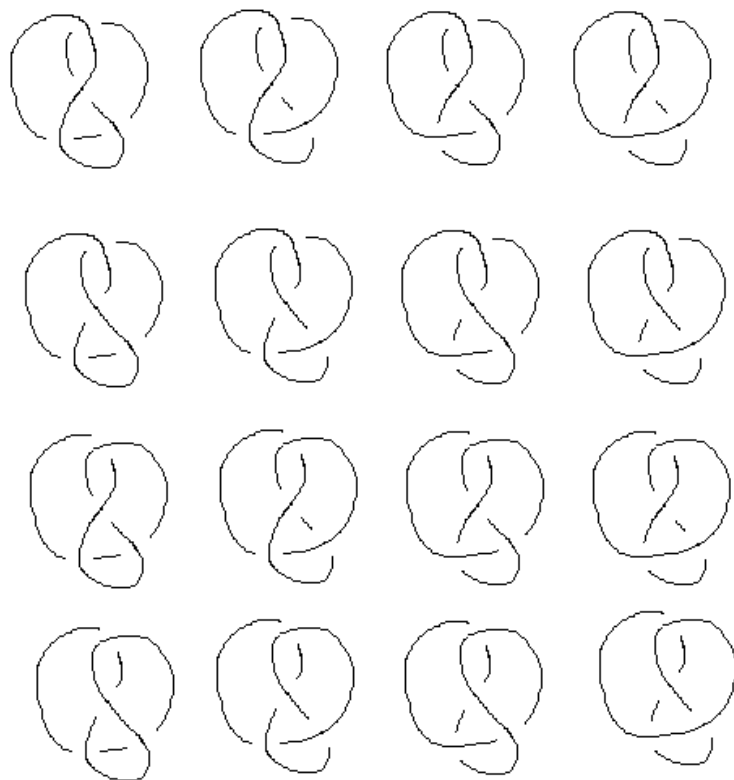


Assign a direction to follow:



Now, follow the arrow and assign a T or B according to the combination TTTT, TTTB, TTBT, TTBB, TBTT, TBTB, TBBT, TBBB, BTTT, BTTB, BTBT, BTBB, BBTT, BBTB, BBBT, BBBB

Here they draw, in the same order given above, from left to right, top to bottom:



Which are the unknot in the above? Looking at them I see numbers 1,2,3,4,5,8,9,12,13,14,15,16 are unknot.

These numbers refer to the diagram numbers above in the order left to right, top to bottom.

So, the following are the unknot and knot combinations:

TTTT, TTTB, TTBT, TTBB, TBTT (1,2,3,4,5) are unknot

TBTB, TBBT, (6,7) are knot

TBBB, BTBT, (8,9) are unknot

BTTB, BTBT, (10,11) are knot

BTBB, BBTT, BBTB, BBBT, BBBB (12,13,14,15,16) are unknot

The ratio for knot is $4/16$ and for unknot is $12/16$

Notice, this is double the ratio I saw above for $n=3$, where we had $2/8$ for knot and $6/8$ for unknot. Does this mean the ratio will double each time the number of vertices is increase by one?

Does this mean, for $n=5$ we will get $8/32$ knot ratio and $24/32$ unknot ratio, and so forth for larger n 's ?

Write a program to find out.

Now, let me look at the pattern of the T's and B's.

For the unknot, I see either a 3 or more same consecutive letters (BBB or TTT), or a 2 same consecutive letters (TT or BB). I see in the knot pattern having a TT or BB next to each others as well, however, in the knot pattern these are flip flips, while for unknot pattern those where not.

For example, looking at unknot pattern TTBB, and the knot pattern TBBT. Both have the pattern 'BB'. But if we take 2 letters at a time from left to right, we never get the same two letters in the knot patter, while in the unknot pattern we do. How can I generalize this, so that given number of vertices, I can generate all the knot diagrams?

In the above, for $n=4$, we get
TBTB, TBBT, BTTB, BTBT

As the knots.

I can see the pattern here. If we assign X to TB, and Y to BT, then the pattern is

XX, XY, YX, YY

So, for $n=5$, will we get

XXX, XXY, XYX, XYY, YXX, YXY, YYX, YYY

Or

TBTBTB, TBTBBT, TBBTTB, TBBTBT, BTTBTB, BTTBBT, BTBTTB, BTBTBT

Let me try it and found out.

Chapter 3

HWs

Local contents

3.1	HW 1	78
3.2	HW 2	105
3.3	HW 3	138
3.4	HW 4	177
3.5	HW 5	203

3.1 HW 1

HW 1

Mathematics 127
Mathematical and Computational Methods in
Molecular Biology

Fall 2002
UC Berkeley, CA

Nasser M. Abbasi

Fall 2002 Compiled on August 2, 2022 at 8:06am [public]

Contents

1 Problems	3
2 Problem 1	6
3 Problem 2	12
4 Problem 3	17
5 Problem 4	19
6 Problem 5	21
7 Problem 6	25

1 Problems

Problem Set 1 (due Thursday September 12)
MATH 127: Mathematical and Computational Methods in Molecular
Biology

Please work on the starred problem alone.

Problem 1

Find a sequence of Reidemeister moves to untangle this unknot:

Can you do it without passing through a knot diagram with more than seven crossings?

Can you do it without using all three types of Reidemeister moves?

Problem 2

The goal of this exercise is to introduce you to the NCBI website which houses the GENBANK database (the public genome database). The website is at <http://www.ncbi.nlm.nih.gov/>. We will use this site extensively during the semester.

Go to the website and answer the following questions:

- Find the Ebola genome (Zaire Mayinga strain) and display it in FASTA format. How many occurrences of the string “GATTACA” are there in the genome?
- What is the current best estimate of the size of the mouse genome?
- How many amino acids are there in the human topoisomerase TOP IIIa gene?

Problem 3* [Brunnian links]

a) Construct a link of four components such that the removal of any component leaves a set of unlinked circles.

b) Construct a link of n components such that the removal of any component leaves a set of unlinked circles.

Problem 4

Consider the helix described by the vector equation

$$\mathbf{r}(t) = a\cos\omega t\mathbf{i} + a\sin\omega t\mathbf{j} + b\omega t\mathbf{k},$$

where ω is a positive constant. Prove that the tangent line makes a constant angle with the z -axis and that the cosine of this angle is

$$\frac{b}{\sqrt{a^2 + b^2}}.$$

Problem 5

Consider two oriented perpendicular lines in R^3 . The first, labeled C_1 consists of the set of points $(0, 0, z)$ where $-\infty < z < \infty$. The second, C_2 consists of the set of points $(a, y, 0)$ where a is a constant and $-\infty < y < \infty$. Let ϵ, δ be constants and consider the contribution to the linking number of the region where $-\epsilon < z < \epsilon$ and $-\delta < y < \delta$ and $a \rightarrow 0$ and show that it is $\frac{1}{2}$. In other words, show that the linking number integral reduces to

$$\lim_{a \rightarrow 0} \frac{1}{4\pi} \int_{-\epsilon}^{\epsilon} \int_{-\delta}^{\delta} \frac{a}{(a^2 + y^2 + z^2)^{\frac{3}{2}}} dy dz$$

and that it is equal to $\frac{1}{2}$.

Problem 6

Prove that the following process always will always produce an unknotted diagram: Start drawing. Whenever you encounter a previously drawn line, undercross it. Eventually return to the start.

2 Problem 1

HW1, MATH 127, UC Berkeley.
Due Thursday September 12, 2002.
By Nasser Abbasi (Alone).

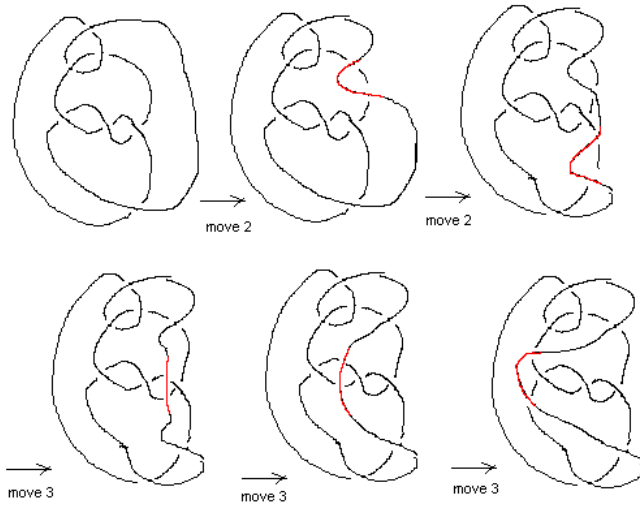
Problem 1

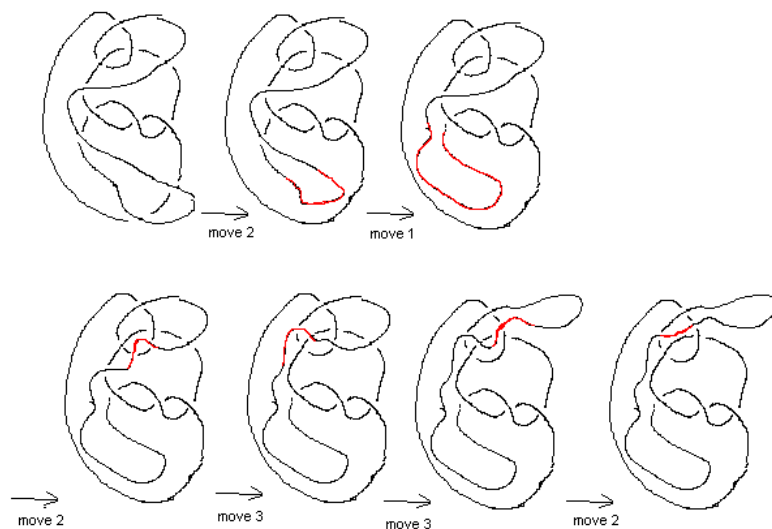
Q: Find a sequence of Reidemeister moves to untangle this unknot:

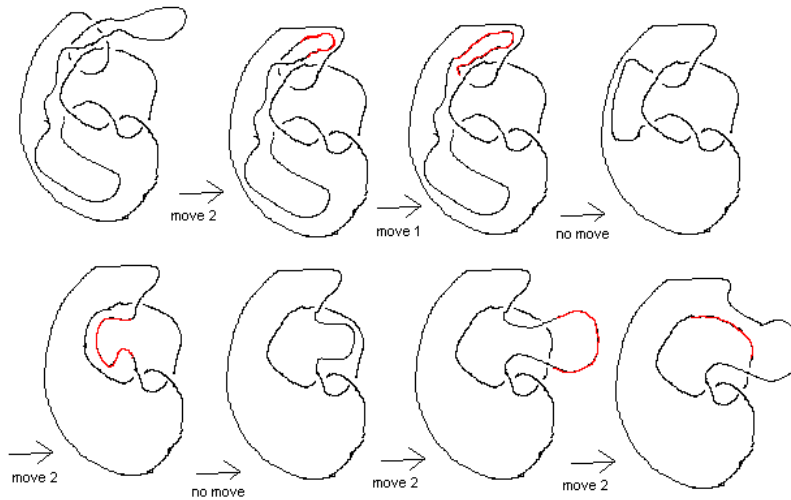


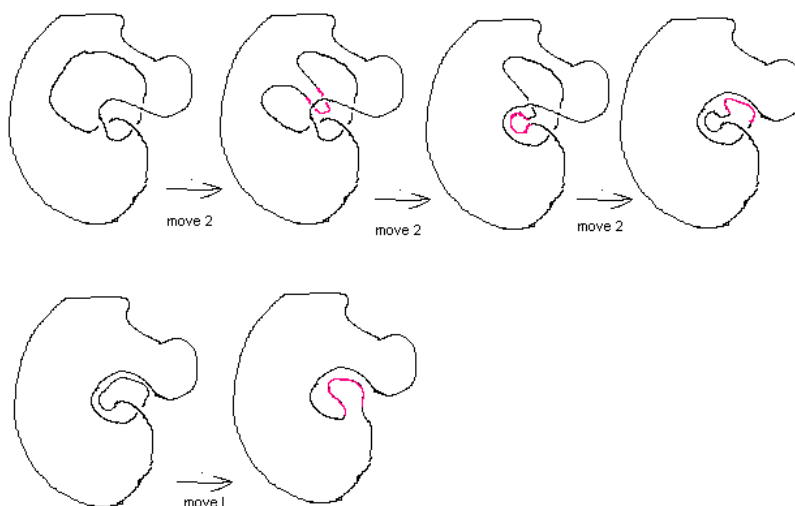
Answer:

The following diagrams show the Reidemeister moves. Diagrams go from left to right, top to bottom. Under each diagram I show the move number (1 which is a twist, 2 which is a slide, or 3 which is a slide over a cross). And I show in RED the part of the diagram that was affected by the Reidemeister move to make it easier to see the move.









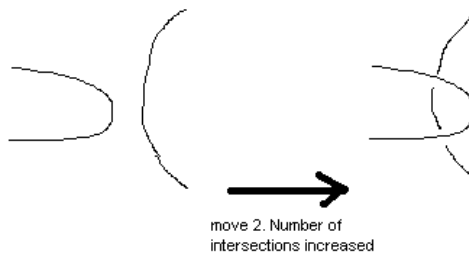
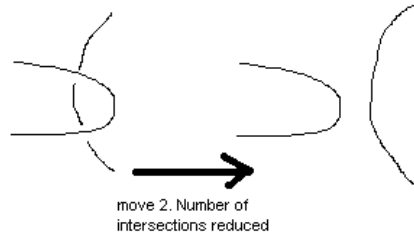
Q: Can you do it without passing through a knot diagram with more than 7 crossing?

Answer: NO. Since the initial diagram has 7 crossings, then only applying an untwist move, (move type I), initially will cause the number of crossing to decrease by one. Looking at the initial diagram there is no such initial move.

Hence we must start with a move type 2 or move type 3.

Type 3 move does not change the number of crossing, however in this diagram, there is no such initial move that can be made.

So, what is left is move type 2. This move can either decrease the number of intersections by 2 (call this move 2a type), or increase it by 2 (call this move 2b type) as seen in this diagram:



Looking at the original diagram, there is no possibility of starting with a move 2a type.

This means the only move left to start with is move 2b type, which increases the number of intersections by 2. Hence it is not possible to untangle the unknot without passing through a knot diagram with more than 7 crossings.

Q: Can you do it without using all three types of Reidemeister moves?

A: No. Move type 3 is needed to slide the right most edge to the left over the 3 crossings in the middle of the diagram. If we have started with the left most edge and slide that to the right instead, we still have to use move 3 to pass through the 3 intersections in the middle of the diagram.

3 Problem 2

HW1, problem 2.
MATH 127, UC Berkeley.
By Nasser Abbasi (worked on alone).

Q: Go to <http://www.ncbi.nlm.nih.gov> and answer the following questions

1. Find the Ebola genome (Zaire Mayinga strain) and display it in FASTA format.
How many occurrences of the string “GATTACA” are there in the genome?
2. What is the current best estimate of the size of the mouse genome?
3. How many amino acids are there in the human topoisomerase TOP IIIa gene?

Answer for part 1

!. I went to the above URL. Then selected “Nucleotide” in the search window.
Next, typed “Ebola” in the ‘for’ window. Then clicked GO.

Then a window appeared which listed all GENBANK records for Ebola. I Clicked on the second one in the list (locus AF499101), since that is the strain Zair Mayinga.

A new window appeared showing the GENBANK record for the this virus genome:

LOCUS	AF499101	18960 bp	RNA	linear	VRL 28-AUG-2002
DEFINITION	Zaire Ebola virus strain Mayinga, complete genome.				
ACCESSION	AF499101				
Etc...					

In the display option, I selected FASTA, Then clicked on the “TEXT” button. A new window came up showing the genome in FASAT format.

Showing below few lines of the sequence:

```
>gi|21702647|gb|AF499101.1| Zaire Ebola virus strain Mayinga, complete genome
CGGACACACAAAAAGAAAGAATTTTTAGGATCTTTGTGTGCGAATAACTATGAGGAAGATTAATAA
TTTTCTCTCATTGAAATTTATATCGGAATTTAAATTGAAATTGTTACTGTAATCACACCTGGTTTGTT
CAGAGCCACATCACAAAGATAGAGAACAACCTAGGTCTCCGAAGGGAGCAAGGGCATCAGTGTGCTCAGT
TGAAAAATCCCTTGTCACACCTAGGTCCTTATCATATCACAAGTTCACCTCAGACTCTGCAGGGTGATCC
.....
ATGATAATTAAGACATTGACCACGCTCATCAGAAGGCTCGCCAGAATAAACGTTGCAAAAAGGATTCCTG
GAAAAATGGTCGCACACAAAAATTTAAAAATAAATCTATTCTTTTGTGTGTCCTCA
```

To answer the question of how many occurrences of “GATTACA” in the sequence. I used BLAST 2.0 program. Went to <http://www.ncbi.nlm.nih.gov/gorf/bl2.html> web page I set the ‘EXPECT’ value to 1,000,000 and reduced the ‘WORD SIZE’ from 11 to 7. Typed in the query sequence “GATTACA” in the top search box. Then for the other

sequence (the Ebola sequence), entered the **ACCESSION** value AF499101 which I got from the above query. Next I clicked on the align button:

BLAST 2 SEQUENCES

This tool produces the alignment of two given sequences using [BLAST](#) engine for local alignment. The stand-alone executable for blasting two sequences (bl2seq) can be retrieved from [NCBI ftp site](#).
Reference: Tatiana A. Tatusova, Thomas L. Madden (1999), "Blast 2 sequences - a new tool for comparing protein and nucleotide sequences", FEMS Microbiol Lett. 174:247-250

Program: **blastn** | Matrix: Not Applicable

Parameters used in **BLASTN** program only:
Reward for a match: 1 | Penalty for a mismatch: -2

Open gap: 0 | and extension gap: 0 | penalties
gap_x_dropoff: 50 | expect: 0.000001 | word size: 7 | [Filter](#)

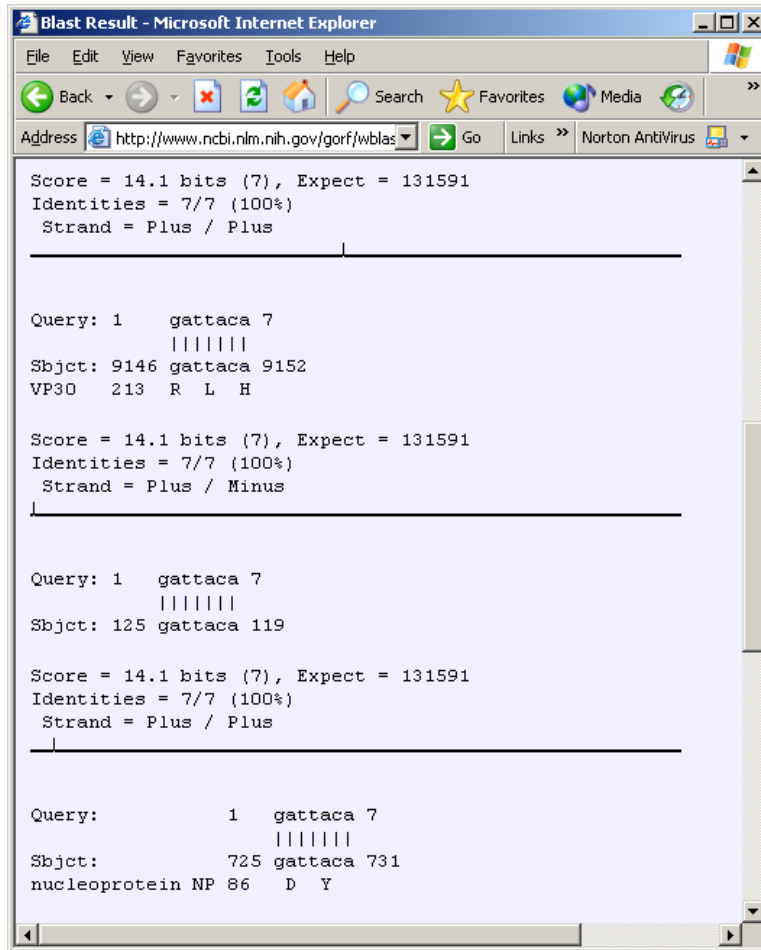
Sequence 1: Enter accession or GI: | or download from file:
or sequence in FASTA format from: | to: |
GATTACA

Sequence 2: Enter accession or GI: AF499101 | or download from file:
or sequence in FASTA format from: | to: |

Comments and suggestions to: blast-help@ncbi.nlm.nih.gov
Credits to: [Tatiana Tatusov](#) and [Tom Madden](#)

The result came back and showed **3 occurrences**.

From 9164..9152, 125..119, 725..731

**Answer for part 2 question:**

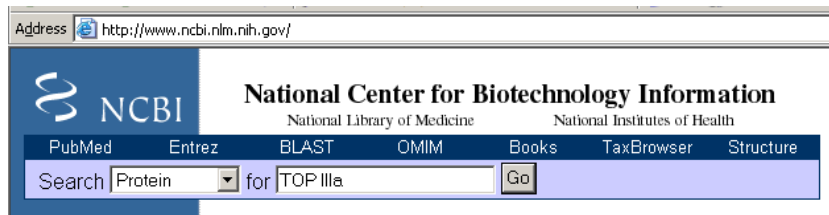
To find the best estimate for the size of the mouse genome, went to

<http://www.ncbi.nlm.nih.gov/genome/seq/MmProgress.shtml>

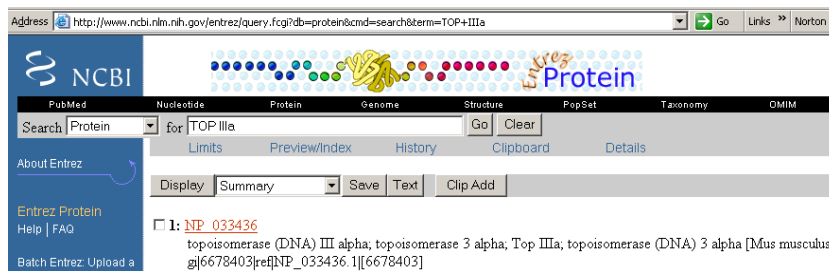
There are 21 chromosomes. The table shows the size of each chromosome in kilo bases (Kb). The total shown is 3,088,000 kb, this is little over 3 billion base pairs. (it is close to the human genome in size).

Answer for part 3 question

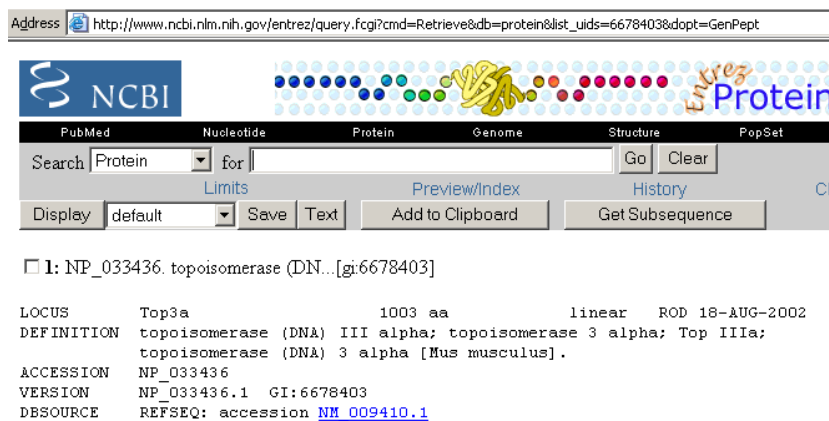
Went to the home page of NCBI again, and selected the Protein database, then typed in 'TOP IIIa' in the 'for' field as shown:



A new page came up with the result:



I clicked in the link and got this page:



The size of the sequence (number of AMINO acids since this is a protein sequence) is **1003**. This can be seen by looking the the features field 'source'

FEATURES	Location/Qualifiers
source	1..1003

4 Problem 3

HW1 problem 3

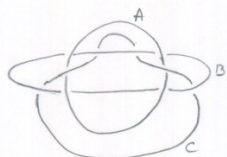
Nasser Abbasi

a) First make a 3 component trivial link. then add a link.

This is a 3 component trivial link:

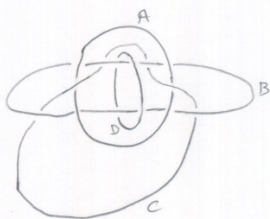


start with 2 links.



add a 3rd component 'c' so that the link remain trivial.

The above is trivial since we can slide 'c' down to cause A circle to slide up freeing all components.
now add a component to cause all 4 to become non-trivial link:



*

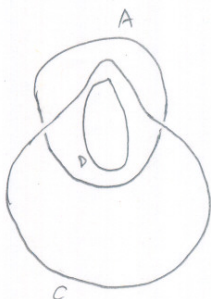
To show that the above is correct, I will remove, one at a time, A, B and C to show that The 3 component link resulting is trivial →

remove A :



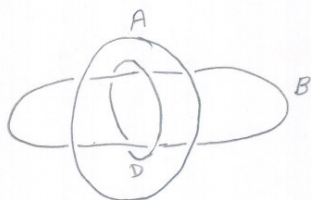
checks out ok.
'C' is separated out
Then 'D' slides out ok.

remove B :



This is clearly a trivial link also.

remove C :



This is also a trivial link.
since 'A' now is free. making
'D' free to slide out.

This completes answer A.

*Evermore
looked online.*

part (b): Unable to do. might be easier to do using Jones Polynomial.
might need to show a series progression somehow.

5 Problem 4

HW1 problem 4

Nasser Abbasi

MATH 127

To show that angle the tangent vector makes with the z axis, I find the equation of the vector and show that the z component is independent of t .

to $\sin \omega$ $\vec{r}(t) = (a \cos \omega t, a \sin \omega t, b \omega t)$

the $\vec{T}(t) = \frac{d\vec{r}(t)}{dt} = (-a\omega \sin \omega t, a\omega \cos \omega t, b\omega)$

this shows that the z component is independent of t .

To find the angle, use the rule

$$\cos \theta = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| |\vec{b}|} \quad \text{where } \vec{a}, \vec{b} \text{ are any two vectors}$$

and θ is angle between them.

in our case \vec{a} is the tangent vector \vec{T} , and \vec{b} is the unit vector along z . so

$$\cos \theta = \frac{(-a\omega \sin \omega t, a\omega \cos \omega t, b\omega) \cdot (0, 0, 1)}{|\vec{T}| \cdot 1}$$

To find $|\vec{T}|$:

$$\begin{aligned} & \sqrt{(-a\omega \sin \omega t)^2 + (a\omega \cos \omega t)^2 + (b\omega)^2} \\ &= \sqrt{a^2 \omega^2 \sin^2 \omega t + a^2 \omega^2 \cos^2 \omega t + b^2 \omega^2} \\ &= \omega \sqrt{a^2 (\cos^2 \omega t + \sin^2 \omega t) + b^2} \\ &= \omega \sqrt{a^2 + b^2} \quad \rightarrow \end{aligned}$$

$$\text{so } \cos \theta = \frac{(-a\omega \sin \omega t, a\omega \cos \omega t, b\omega) \cdot (0, 0, 1)}{\omega \sqrt{a^2 + b^2}}$$

$$\cos \theta = \frac{0 + 0 + b\omega}{\omega \sqrt{a^2 + b^2}} = \frac{b}{\sqrt{a^2 + b^2}}$$

and angle is constant (independent of t).

QED

6 Problem 5

HW 1, problem 5

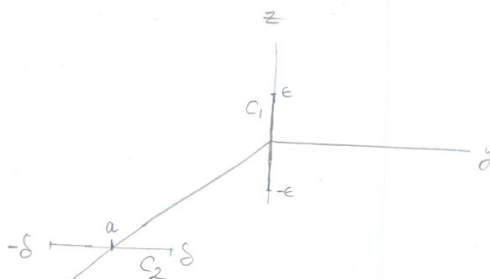
Nasser Abbasi

MATH 127

Consider two oriented, perpendicular lines in \mathbb{R}^3 . The first labeled C_1 consists of the set of points $(0, 0, z)$, where $-\epsilon < z < \epsilon$. The second C_2 consists of the set of points $(a, y, 0)$ where a is a constant and $-\delta < y < \delta$. Let ϵ, δ be constants and consider the ~~contribution~~ contribution to the linking number of the region where $-\epsilon < z < \epsilon$ and $-\delta < y < \delta$ and $a \rightarrow 0$ and show that it is $\frac{1}{2}$. In other words, show that the linking number integral reduces to

$$\lim_{a \rightarrow 0} \frac{1}{4\pi} \int_{-\epsilon}^{\epsilon} \int_{-\delta}^{\delta} \frac{a}{(a^2 + y^2 + z^2)^{\frac{3}{2}}} dy dz.$$

and that it is $= \frac{1}{2}$.

Answer

$$LK = \frac{1}{4\pi} \iint_{C_2} \int_{C_1} \frac{\bar{e} \cdot (\bar{t}_2 \times \bar{t}_1)}{r^2} ds_1 ds_2$$

where \bar{t}_2 is unit tangent vector on C_2
 and \bar{t}_1 is unit tangent vector on C_1
 and $\bar{e} = \frac{y_1 - y_2}{\|y_1 - y_2\|}$ where y_1 is eq. of point on C_1 and y_2 is eq. of point on C_2 .

$$\bar{t}_2 = \frac{d}{dy} (a, y, 0) = (0, 1, 0)$$

$$\bar{t}_1 = \frac{d}{dz} (0, 0, z) = (0, 0, 1)$$

$$\text{so } \bar{t}_2 \times \bar{t}_1 = \begin{vmatrix} i & j & k \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix} = (1, 0, 0)$$

$$\bar{e} = \frac{(a, y, 0) - (0, 0, z)}{\| (a, y, 0) - (0, 0, z) \|} = \frac{(a, y, -z)}{\sqrt{a^2 + y^2 + z^2}}$$

$$\text{so } LK = \frac{1}{4\pi} \int_{y=-\delta}^{\delta} \int_{z=-\epsilon}^{\epsilon} \frac{(a, y, -z)}{\sqrt{a^2 + y^2 + z^2}} \cdot (1, 0, 0) \frac{1}{r^2} dz dy.$$

$$\text{but } r^2 = a^2 + y^2 + z^2$$

$$\text{so } LK = \frac{1}{4\pi} \int_{-\delta}^{\delta} \int_{-\epsilon}^{\epsilon} \frac{(a, 0, 0)}{(a^2 + y^2 + z^2)^{3/2}} dz dy = \frac{1}{4\pi} \int_{-\delta}^{\delta} \int_{-\epsilon}^{\epsilon} \frac{a}{(a^2 + y^2 + z^2)^{3/2}} dz dy.$$

to have C_2 and C_1 have a link, 'a' must go to zero else C_2 and C_1 will have link of zero as is.

$$\text{so } LK = \lim_{a \rightarrow 0} \frac{1}{4\pi} \int_{-\epsilon}^{\epsilon} \int_{-\delta}^{\delta} \frac{a}{(a^2 + y^2 + z^2)^{3/2}} dy dz$$



To solve the integral. look at $\int_{-\delta}^{\delta} \frac{a}{(a^2 + y^2 + z^2)^{3/2}} dy$.

here, a and z are constants.

so above integral is $\int a (a^2 + y^2 + z^2)^{-3/2} dy$

$$= \frac{4ay}{(4a^2 + 4z^2) \sqrt{a^2 + y^2 + z^2}} \Big|_{y=-\delta}^{y=\delta} = \frac{2a\delta}{(a^2 + z^2) \sqrt{a^2 + \delta^2 + z^2}}$$

$$\text{so } \frac{1}{4\pi} \int_{-\epsilon}^{\epsilon} \frac{2a\delta}{(a^2 + z^2) \sqrt{a^2 + \delta^2 + z^2}} dz = \frac{1}{2\pi} \int_{-\epsilon}^{\epsilon} \frac{a\delta}{(a^2 + z^2) \sqrt{a^2 + \delta^2 + z^2}} dz$$

Dr, I used MAPLE to evaluate the above and take the $\lim a \rightarrow 0$, but I get ϕ as the final answer not $\frac{1}{2}$. please see attached.

That's okay.

Thanks!!

To compute by hand,

try subs. for polar coords.

$\frac{f(0)}{g(0)}$

```
> eq1 := (1/(4*Pi))*a*(a^2+y^2+z^2)^(-3/2);
>
```

$$eq1 := \frac{1}{4} \frac{a}{\pi (a^2 + y^2 + z^2)^{(3/2)}}$$

```
> eq2 := int(eq1, y=-delta..delta);
```

$$eq2 := \frac{1}{2} \frac{a \delta}{\pi (a^2 + z^2) \sqrt{a^2 + \delta^2 + z^2}}$$

```
> eq3 := int(eq2, z=-epsilon..epsilon);
```

$$eq3 := -\frac{1}{4} \delta \sqrt{-a^2} \left(-\ln \left(-2 \frac{\delta^2 + \epsilon \sqrt{-a^2 + a^2 + \sqrt{\delta^2} \sqrt{\epsilon^2 + a^2 + \delta^2}}{-\epsilon + \sqrt{-a^2}} \right) \right. \\ \left. + \ln \left(2 \frac{\delta^2 - \epsilon \sqrt{-a^2 + a^2 + \sqrt{\delta^2} \sqrt{\epsilon^2 + a^2 + \delta^2}}{\epsilon + \sqrt{-a^2}} \right) \right. \\ \left. + \ln \left(-2 \frac{\delta^2 - \epsilon \sqrt{-a^2 + a^2 + \sqrt{\delta^2} \sqrt{\epsilon^2 + a^2 + \delta^2}}{\epsilon + \sqrt{-a^2}} \right) \right. \\ \left. - \ln \left(2 \frac{\delta^2 + \epsilon \sqrt{-a^2 + a^2 + \sqrt{\delta^2} \sqrt{\epsilon^2 + a^2 + \delta^2}}{-\epsilon + \sqrt{-a^2}} \right) \right) / (a \pi \sqrt{\delta^2})$$

```
> limit(eq3, a=0);
```

0

Should I have taken $\lim_{\delta \rightarrow \infty}$ and $\lim_{\epsilon \rightarrow \pm \infty}$ also?

7 Problem 6

HW1 problem 6
Nasser Abbasi

To show this, I need to show that using Reidemeister moves only, the final knot can always be transformed to the unknot (i.e zero crossings).

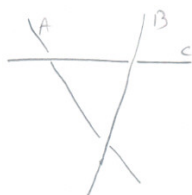
by definition, we can never get this shape.

HO
CO



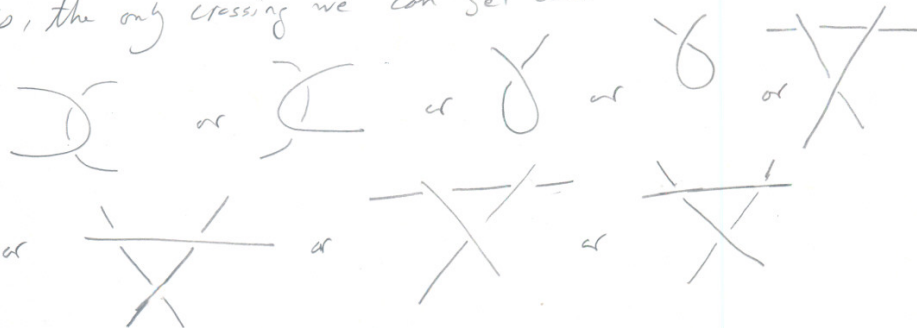
since c went over and under A which contradicts our method.

we also can never get this shape



since c has gone over A but under B which contradicts our method of building the knot.

So, the only crossing we can set are



Since our goal is to set number of crossing to zero, we use

Reim. moves to do this. when we see

$$\overbrace{) \)} \rightarrow) \) \quad \text{zero crossing}$$

$$\overbrace{) \)} \rightarrow) \) \quad \text{zero crossing}$$

$$\overbrace{) \)} \rightarrow \cup \quad \text{zero crossing}$$

$$\overbrace{) \)} \rightarrow \cup \quad \text{zero crossing.}$$

what is left is Reim. move 3. This move does not change the number of crossing.

now, I say that the only moves needed to transform this to the unknot are move I and II. (move II is used to slide over intersecting lines one at a time as in

$$\overbrace{) \)} \rightarrow \underbrace{\cup}_{\cup} \rightarrow \underbrace{\cup}_{\cup}$$

and move I to untwist. so no need to use move III.
it will not come up.

This means we can reduce the number of crossings to zero.

hence the unknot is produced.

3.2 HW 2

HW 2

Mathematics 127
Mathematical and Computational Methods in
Molecular Biology

Fall 2002
UC Berkeley, CA

Nasser M. Abbasi

Fall 2002 Compiled on August 2, 2022 at 8:08am [public]

Contents

1	Problems	3
2	Problem 1	4
3	Problem 2	10
4	Problem 3	13
5	Problem 4	14
6	Problem 5	16
7	Problem 6	17
8	Problem 7	18
	8.1 Problem 7 source code	26

1 Problems

Problem Set 2 (due Thursday September 26)

MATH 127: Mathematical and Computational Methods in Molecular Biology

Please work on the starred problem alone.

Problem 1

Estimate the number of nucleosomes used for supercoiling in human chromosome 17.

Problem 2*

Show (in full detail) that the directed writhe of a knot is invariant under Reidemeister moves of type 2 and 3.

Problem 3

Given two sequences of length n , and a scoring scheme $1, -1, -2$ (for match, mismatch gap), let the score of the optimal global alignment be G and the optimal local alignment be L .

- Prove that $L \geq G$, and find an example where strict equality holds.
- What is the maximum value of $L - G$?

Problem 4

If the Jones polynomial of a knot is $X(L)$ what is the Jones polynomial of the mirror image of the knot?

Problem 5

Compute the Jones polynomial of the trefoil knot. Then show that the trefoil knot and its mirror image are not equivalent (use problem 4).

Problem 6 (optional)

Given sequences of lengths n and m what is the maximum number of optimal alignments (with the same score) that can result from a scoring scheme of 1 for a match, a for a mismatch and b for a gap.

Problem 7 (optional)

- Implement the Needleman-Wunsch algorithm where the parameters are an input.
- Use the scoring scheme of 1 for a match, and 0 for a mismatch and gap to find the average length of the longest increasing subsequence in a permutation of length n by simulation.

2 Problem 1

Problem 1
 Math 127
 Nasser Abbasi

Link number is an invariant.

For relaxed DNA for chromosome 17 (97 million bases) From NCBI web page.

$L_k = \frac{97 \times 10^6}{10.5} + W_r$

but $W_r = 0$ for relaxed DNA (no supercoiling)

So $L_k = \frac{97 \times 10^6}{10.5}$ (round to an integer since L_k must be an integer)

when supercoiling, number of bases per turn is 10 instead of 10.5.

So $L_k = T_w + W_r$

so $\frac{97 \times 10^6}{10.5} = \frac{97 \times 10^6}{10} + W_r$

so $W_r = -461904$

so there are approx 461,904 nucleosomes

To confirm There are 200bp per nucleosomes (146 bp wrapped around 4 histons, and 56 linking DNA)

so $\frac{97 \times 10^6}{200} = 485,500$ (close to above calculations. This probably means not all the chromosome is constructed to have nucleosomes every where?)

total: ~~150~~
 $\frac{49}{50}$

problem 2

HW 2

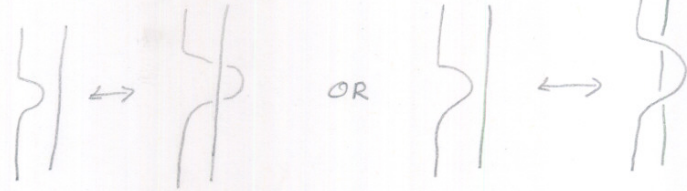
Nasser Abbasi

HO

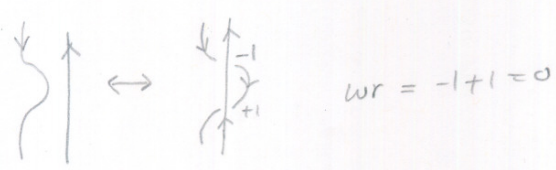
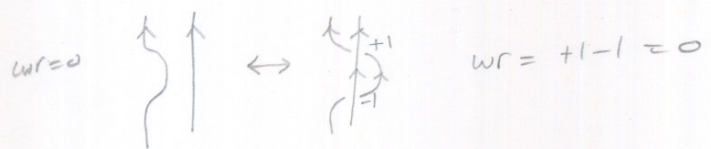
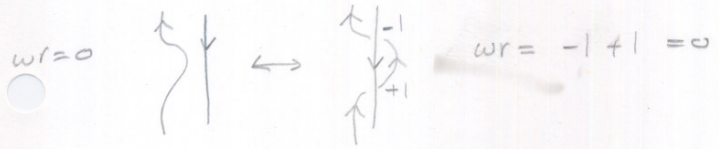
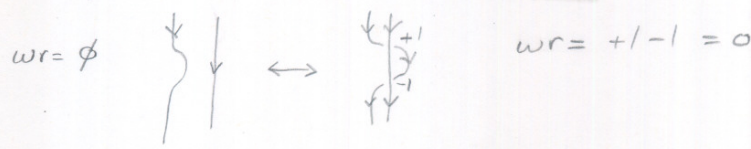
4 cases here

4 cases here.

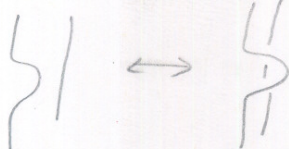

move type 2 is




to show directed writhe is invariant, I calculate wr before and after the move. I have to look at 4 cases per each subtype.



now look at the 4 cases for

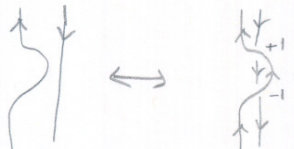



$wr = 0$



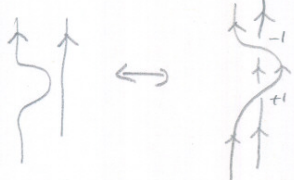
$wr = -1 + 1 = 0$

$wr = 0$



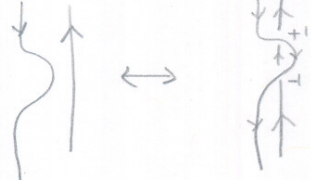
$wr = +1 - 1 = 0$

$wr = 0$



$wr = -1 + 1 = 0$

$wr = 0$



$wr = +1 - 1 = 0$

hence, under move type 2, directed writhe is invariant.

now I look at move type 3 \rightarrow

move type 3 is

①

or

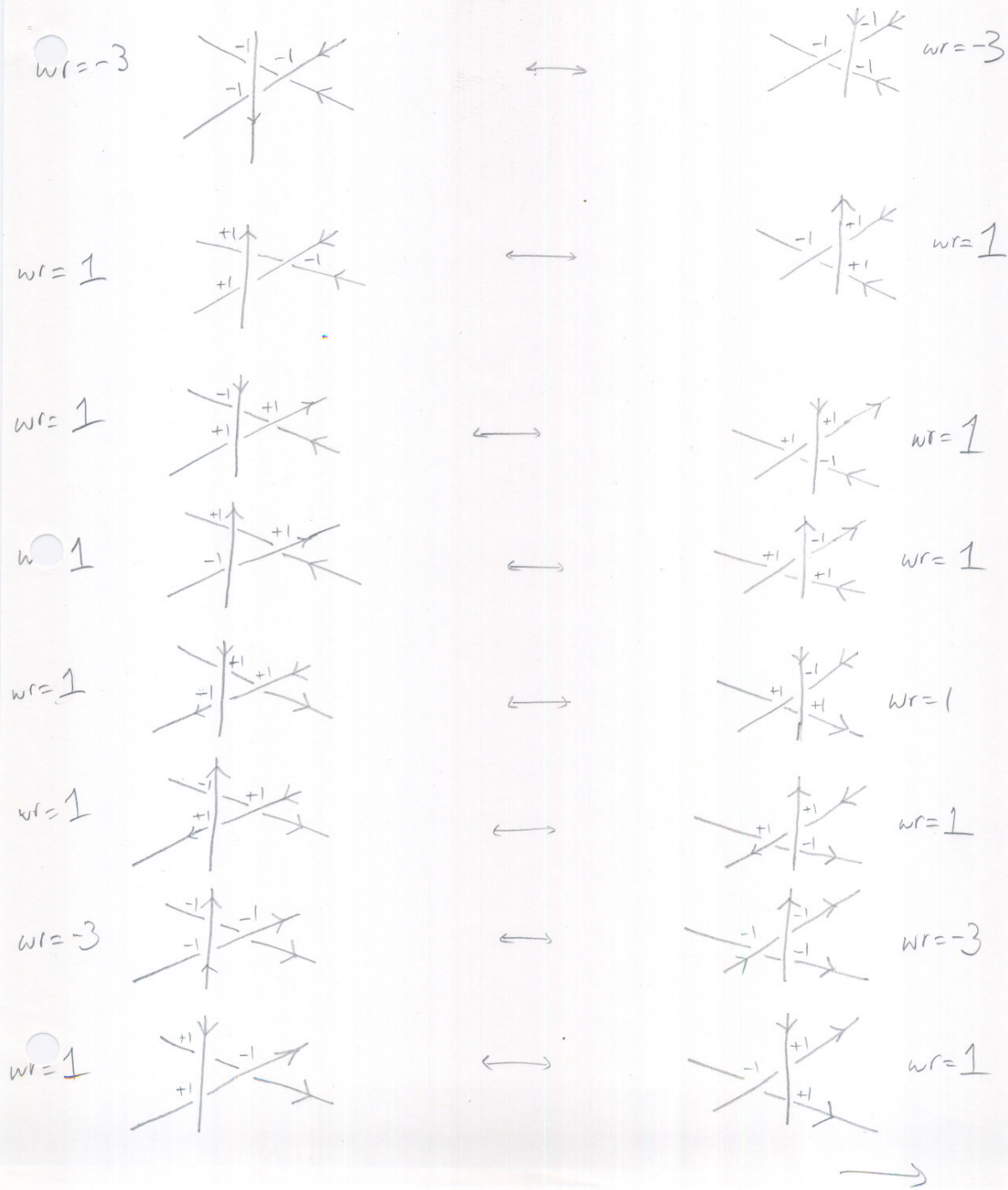
②

looking at ① now,
possible combinations are 8

$wr=1$		\leftrightarrow		$wr=1$
$wr=-3$		\leftrightarrow		$wr=-3$
$wr=1$		\leftrightarrow		$wr=1$
$wr=1$		\leftrightarrow		$wr=1$
$wr=1$		\leftrightarrow		$wr=1$
$wr=1$		\leftrightarrow		$wr=1$
$wr=-3$		\leftrightarrow		$wr=-3$

→

So more type 3 part ① is invariant. now look at part ②



So this shows that "i⁺ move" type 5, part \Leftarrow and

so I showed for type 2 and 3 that
directed write is invariant. QED

3 Problem 2

problem 3
HW 2
Nassim Abbas: ~~HW 2~~ x8

The score G is the sum of the scores at each cell in the matrix along the global alignment path.

The score L is the sum of the scores at each cell in the matrix over the local alignment path. there is one global alignment path in the matrix, but many local alignment paths.

Now, a local alignment path will stop when we hit on a cell in the matrix that have score of ϕ . so local alignment score must be ≥ 0 all the time to continue over that path.

let the sequences be a_1, a_2, \dots, a_n , b_1, b_2, \dots, b_n .

There are these extreme cases

Case 1 $a_i = b_i$ for all $i=1..n \Rightarrow L = n$ and $G = n$ (since a match score = 1)
so in this case $L = G$.

Case 2 $a_i \neq b_i$ for all $i=1..n \Rightarrow L = \phi$ and $G = -n$
this is because L score must be ≥ 0 by definition.

Case 3

$$\begin{array}{l} a_1 a_2 \dots a_n \text{ -----} \\ \text{-----} b_1 b_2 b_3 \dots b_n \end{array} \Rightarrow L = \phi \text{ and } G = 2n \times (-2) = -4n$$

since gap has -2 score.

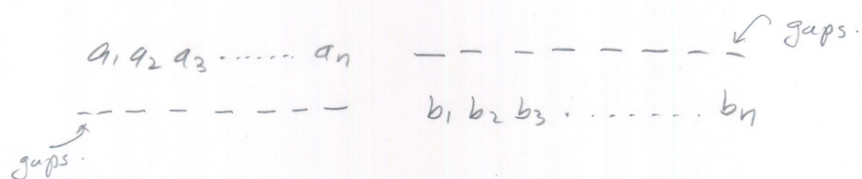
Other possible alignments must a G score greater than $-4n$ and less than n , and must have an L score greater than zero and less than n .

→

this means that $L \geq G$.

A case for strict equality is when $a_i = b_i \quad i=1, \dots, n$

(b) maximum value of $L - G$ is when L is max and G is minimum. this is $-4n$ for this case



or something as

$$\begin{array}{ccccccccc}
 a_1 & - & a_2 & - & a_3 & - & a_4 & - & a_n \\
 - & b_1 & - & b_2 & - & b_3 & - & b_4 & - & b_n
 \end{array}
 \Rightarrow
 \begin{array}{l}
 L = \phi \\
 G = -4n
 \end{array}$$

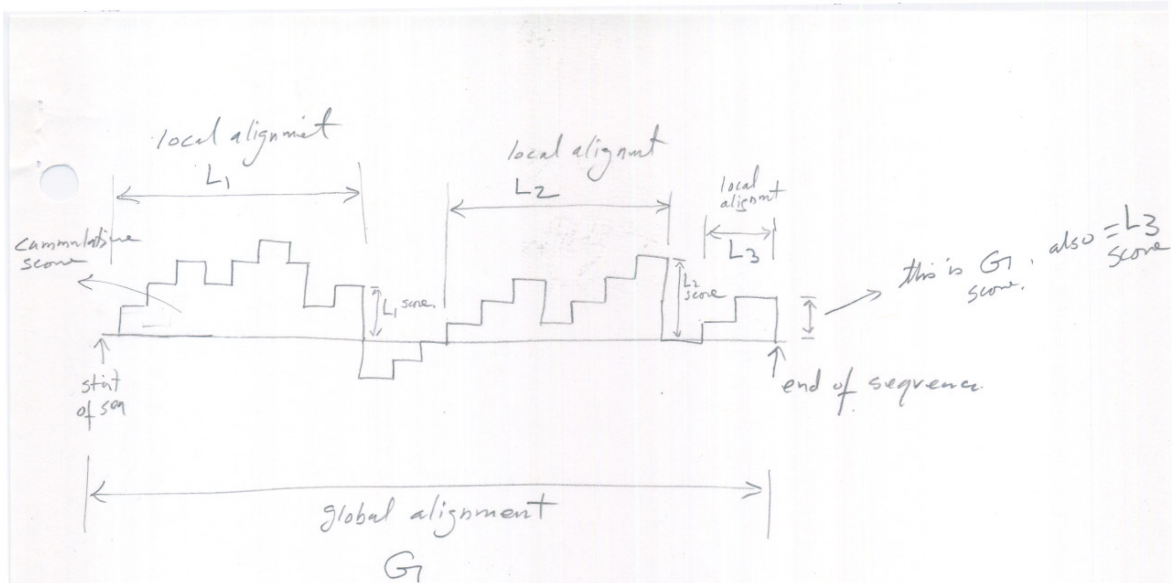
~~$L = \phi$~~ $\therefore L - G = \frac{8}{5}n$

Dr: while thinking about this, I came with this representation which I found usefull.

draw the cumulative score along the path as a stair steps. local alignment paths are the steps 'over' the zero level. the best local alignment is the continuous steps with the largest area over the zero level. the L score is the height of the last step.

The G score is the height of the last step that ends at the end of the sequence





From this diagram I see that $G_1 \leq L_{\max}$.

Since if $G_1 = L_3$, and L_3 happened to be the largest L of all L 's, then $G_1 = L$. otherwise L_1 or L_2 are larger than L_3 , and hence G_1 must be less than L_1 and L_2 as well.

This shows that $G_1 \leq L$ also.

4 Problem 3

Did not do.

5 Problem 4

Problem 4

HW 2

Nasser Abbasi MATH 127 ~~AD~~

- if Jones polynomial is $X(L)$, what is the Jones polynomial of the mirror image of the knot?

$$X(L) = (-A^3)^{-w(L)} \langle L \rangle$$

where $w(L)$ is the directed writhe of projection of knot.
to see the effect, let me look at trefoil knot and its mirror image

$$= -3$$

$$= 3$$

- So for mirror image we have directed writhe of different sign. what about $\langle L \rangle$ of image?
looking at bracket polynomial rules

rule 1 $\langle \bigcirc \rangle = 1 \rightarrow$ not affected.

rule 2 $\langle X \rangle = A \langle \curvearrowright \rangle + A^{-1} \langle \curvearrowleft \rangle$

For mirror image

$$\langle X \rangle = A \langle \curvearrowleft \rangle + A^{-1} \langle \curvearrowright \rangle$$

rule 3 $\langle LUO \rangle = C \langle L \rangle \rightarrow$ not affected.

- So only rule 2 can change the $\langle L \rangle$ polynomial between
Knot and its mirror image.

I see that, from rule 2, $A \rightarrow A^{-1}$ in the mirror image. This means exponent changes sign \rightarrow

So, for bracket polynomial $\langle L \rangle$ in A , the mirror image will have each exponent of A having opposite sign.

For example, if $\langle L \rangle = A^{-5} + A^6$, then $\langle L \rangle' = A^5 + A^{-6}$ where $\langle L \rangle'$ is the bracket polynomial for the mirror image.

To summarise

$$\text{if } X\langle L \rangle = (-A^3)^{-w(L)} \langle L \rangle$$

$$\text{Then } X\langle L \rangle' = (-A^3)^{w(L)} \langle L \rangle'$$

where $w(L)$ is directed writhe of L projection, and $\langle L \rangle'$ is the same as $\langle L \rangle$ except exponents of A are of opposite sign.

6 Problem 5

problem 5
HW2
Nasser Abbasi

$$\begin{aligned}
 & \text{Diagram} = A \langle \text{Diagram} \rangle + B \langle \text{Diagram} \rangle \\
 & = A(A \langle \text{Diagram} \rangle + B \langle \text{Diagram} \rangle) + B(A \langle \text{Diagram} \rangle + B \langle \text{Diagram} \rangle) \\
 & = A(A(C \langle \text{Diagram} \rangle) + B(A \langle \text{Diagram} \rangle + B \langle \text{Diagram} \rangle)) \\
 & \quad + B(A(A \langle \text{Diagram} \rangle + B \langle \text{Diagram} \rangle) + B(A \langle \text{Diagram} \rangle + B \langle \text{Diagram} \rangle)) \\
 & = A(A(C(A \langle \text{Diagram} \rangle + B \langle \text{Diagram} \rangle)) + B(AC + B)) \\
 & \quad + B(A(AC + B) + B(A + BC)) \\
 & = A(A(C(AC + B)) + B(AC + B)) \\
 & \quad + B(A(AC + B) + B(A + BC)) \\
 & = A(A(AC^2 + BC) + BAC + B^2) \\
 & \quad + B(A^2C + AB + BA + B^2C) \\
 & = A(A^2C^2 + ABC + ABC + B^2) + BA^2C + AB^2 + B^2A + B^3C \\
 & = A^3C^2 + A^2BC + A^2BC + AB^2 + BA^2C + AB^2 + B^2A + B^3C
 \end{aligned}$$

7 Problem 6

HW #2
 Problem 6 (optional)
 Nasser Abbasi

For $n=1, m=1$

a_1, b_1 the possible alignments are

$$\begin{array}{c|c|c} a_1 - & - a_1 & a_1 \\ - b_1 & b_1 - & b_1 \\ \textcircled{1} & \textcircled{2} & \textcircled{3} \end{array}$$

i.e. 3 alignments

score of ① is $-2b$

score of ② is $-2b$

score of ③ is 1 or $-a$

so for $n=1, m=1$ max number of optimal alignments is $\boxed{1}$
 with same score

for $n=2, m=1$

$$\begin{array}{c|c|c|c|c} a_1 a_2 & a_1 a_2 & - a_1 a_2 & a_1 - a_2 & a_1 a_2 - \\ b_1 - & - b_1 & b_1 - - & - b_1 - & - - b_1 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ \text{score} = 1-2b & -2b-a & -2b-2b-2b & -2b-2b-2b & -2b-2b-2b \\ \text{or } -a-2b & \text{or } -2b+1 & = -6b & = -6b & = -6b \end{array}$$

so in this case we have 2 alignments with score $1-2b$, and two alignments with score $-(2b+a)$ and 3 alignments with score $-6b$.
 so here max is $\boxed{3}$

Need to generalize this recursively for n, m .
 sorry, no time to complete. hard problem!

8 Problem 7

Problem 7
HW2
MATH 127
By Nasser Abbasi
Sept 26, 2002.

Part (a):

Implemented needleman-Wunsch algorithm. Source code is below (also in floppy attached).

To test, I used the example given in the lecture to verify the output is correct. When running the program, it formats the output showing the sequences and the scoring matrix.

```
K> help nma_problem_7_part_a
```

```
function R=nma_problen7_part_a(s1,s2,match,mismatch,gap)
```

```
solves problem 7, HW 1 for MATH 127  
by Nasser Abbasi  
sept 25, 2002.
```

```
implements Needleman-Wunsch algorithm
```

```
INPUT
```

```
s1: is one sequence. example S1=['g' 'a' 't' 'c' 'g'];  
s2: is the second sequence.  
match: score for matching bases. such as 1  
mismatch: score for mismatch. such as -1  
gap: penatly factor for gaps. such as -2
```

```
NOTE: S1 is the row sequence at the top, and S2 is column sequence at left.
```

This is an example:

```

K» S1=['G' 'A' 'T' 'T'];
K» S2=['G' 'A' 'T' 'A' 'C' 'G' 'T'];
K» match=1;
K» gap=-2;
K» mismatch=-1;
K»
K» nma_problem_7_part_a(S1,S2,match,mismatch,gap)
    
```

		G	A	T	T
	0	-2	-4	-6	-8
G	-2	1	-1	-3	-5
A	-4	-1	2	0	-2
T	-6	-3	0	3	1
A	-8	-5	-2	1	2
C	-10	-7	-4	-1	0
G	-12	-9	-6	-3	-2
T	-14	-11	-8	-5	-2

K»

This is another example:

```

K» S1=['T' 'T' 'T' 'C' 'G' 'T' 'A' 'G' 'T' 'T'];
K» S2=['T' 'T' 'C' 'C' 'G' 'A' 'A' 'G' 'C' 'T'];
K» nma_problem_7_part_a(S1,S2,match,mismatch,gap)
    
```

		T	T	T	C	G	T	A	G	T	T
	0	-2	-4	-6	-8	-10	-12	-14	-16	-18	-20
T	-2	1	-1	-3	-5	-7	-9	-11	-13	-15	-17
T	-4	-1	2	0	-2	-4	-6	-8	-10	-12	-14
C	-6	-3	0	1	1	-1	-3	-5	-7	-9	-11
C	-8	-5	-2	-1	2	0	-2	-4	-6	-8	-10
G	-10	-7	-4	-3	0	3	1	-1	-3	-5	-7
A	-12	-9	-6	-5	-2	1	2	2	0	-2	-4
A	-14	-11	-8	-7	-4	-1	0	3	1	-1	-3
G	-16	-13	-10	-9	-6	-3	-2	1	4	2	0
C	-18	-15	-12	-11	-8	-5	-4	-1	2	3	1
T	-20	-17	-14	-11	-10	-7	-4	-3	0	3	4


```
function R=nma_problem7_part_a(s1,s2,match,mismatch,gap)
%function R=nma_problen7_part_a(s1,s2,match,mismatch,gap)
%
% solves problem 7, HW 1 for MATH 127
% by Nasser Abbasi
% sept 25, 2002.
%
% implements Needleman-Wunsch algorithm
%
% INPUT
% s1: is one sequence. example S1=['g' 'a' 't' 'c' 'g'];
% s2: is the second sequence.
% match: score for matching bases. such as 1
% mismatch: score for mismatch. such as -1
% gap: penatly factor for gaps. such as -2
%
% NOTE: S1 is the row sequence at the top, and S2 is column sequence at left.
%
```

```

% reserve space for the score matrix.
nRow=length(s2)+1;
nCol=length(s1)+1;

v=zeros(nRow,nCol);

%
% for needleman, set the boundary condition to gap penalites
%

for(i=2:size(v,2))
    v(1,i)=v(1,i-1)+gap;
end

for(i=2:size(v,1))
    v(i,1)=v(i-1,1)+gap;
end

for n=1:length(s2)
    nn= n+1;
    for m=1:length(s1)
        mm= m+1;
        if s2(n) == s1(m)
            diagonal= match;
        else
            diagonal = mismatch;
        end

        diagonal = diagonal + v(nn-1,mm-1);
        upScore   = v(nn-1,mm)+gap;
        leftScore  = v(nn,mm-1)+gap;

        v(nn,mm) = max([upScore, leftScore, diagonal]); % , 0]);
    end
end

% print the score matrix
fprintf('\t\t');
for(i=1:length(s1))
    fprintf('%c\t',s1(i));
end
fprintf('\n');

for(i=1:nRow)

    if(i==1)
        fprintf('\t');
    else
        fprintf('%c\t',s2(i-1));
    end

    for(j=1:nCol)
        fprintf('%d\t',v(i,j));
    end
    fprintf('\n');
end

```

Problem 7
 HW2
 MATH 127
 By Nasser Abbasi
 Sept 26, 2002.

Part (b)

In this part, the input is 'n' which is the length of the sequence. I'll use the MATLAB function 'perms' to generate all permutations of length n (which will be n! many). Then for each permutation, will use global alignment, then look at the score in the bottom right corner of the matrix. This gives me the length of the longest increasing subsequence for this one permutation. I add all these lengths and divide by n! to get the average.

I implemented this in the function nma_problem_7_part_b.m
 » help nma_problem_7_part_b

```
function R=nma_problen7_part_b(n,match,mismatch,gap)
```

```
solves problem 7 part b, HW 1 for MATH 127  

by Nasser Abbasi  

sept 25, 2002.
```

```
find the average length of the longest increasing subsequence  

in a permutation of length n.
```

INPUT

```
n : the length of the sequence.  

match: score for matching bases. such as 1  

mismatch: score for mismatch. such as -1  

gap: penatly factor for gaps. such as -2
```

Example runs:

```
» nma_problem_7_part_b(2,match,mismatch,gap)  

Average length of longest increasing subsequence in perms of length 2 is 1.500000  

» nma_problem_7_part_b(3,match,mismatch,gap)  

Average length of longest increasing subsequence in perms of length 3 is 2.000000  

» nma_problem_7_part_b(4,match,mismatch,gap)  

Average length of longest increasing subsequence in perms of length 4 is 2.416667  

» nma_problem_7_part_b(5,match,mismatch,gap)  

Average length of longest increasing subsequence in perms of length 5 is 2.791667
```

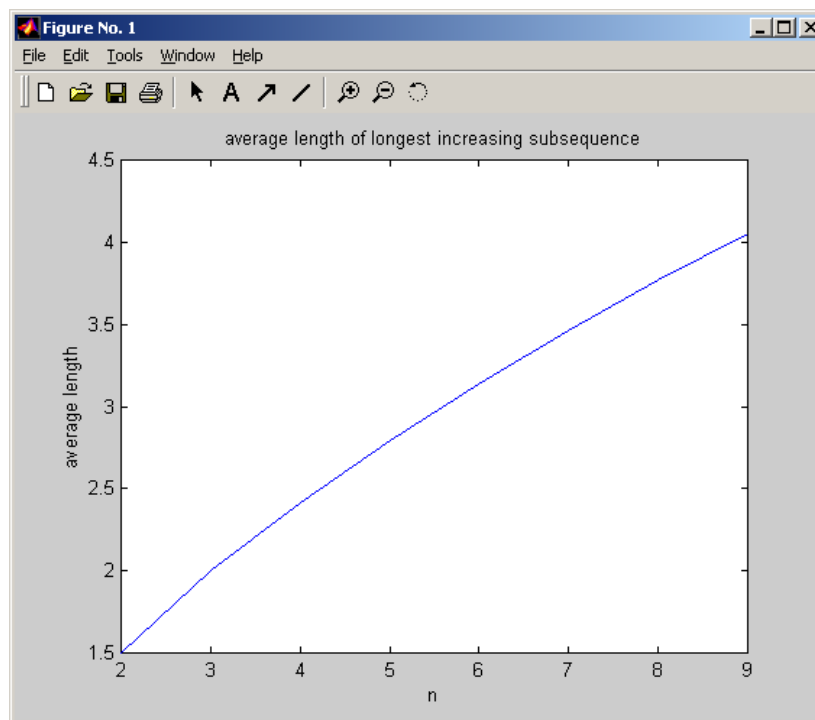
» nma_problem_7_part_b(6,match,mismatch,gap)
Average length of longest increasing subsequence in perms of length 6 is 3.140278

» nma_problem_7_part_b(7,match,mismatch,gap)
Average length of longest increasing subsequence in perms of length 7 is 3.465278

» nma_problem_7_part_b(8,match,mismatch,gap)
Average length of longest increasing subsequence in perms of length 8 is 3.770337

» nma_problem_7_part_b(9,match,mismatch,gap)
Average length of longest increasing subsequence in perms of length 9 is 4.059350

Plotting average length as function of n using MATLAB gives this:



```
function R=nma_problem7_part_b(n,match,mismatch,gap)
%function R=nma_problen7_part_b(n,match,mismatch,gap)
%
% solves problem 7 part b, HW 1 for MATH 127
% by Nasser Abbasi
% sept 25, 2002.
%
% find the average length of the longest increasing subsequence
% in a permutation of length n.
%
% INPUT
% n : the length of the sequence.
% match: score for matching bases. such as 1
% mismatch: score for mismatch. such as -1
% gap: penatly factor for gaps. such as -2
%
%

thePerms = perms(1:n);
totLen=0;

for(k=1:size(thePerms,1))
    totLen = totLen +
    getLengthOfLongestSubSequence(thePerms(k,:),match,mismatch,gap);
end

av=totLen/factorial(n);

fprintf('Average length of longest increasing subsequence in perms of
length %d is %f\n',...
    n,av);
```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function len=getLengthOfLongestSubSequence(seq,match,mismatch,gap)

s2=1:length(seq);
s1=seq;

%
% reserve space for the score matrix.
nRow=length(s2)+1;
nCol=length(s1)+1;

v=zeros(nRow,nCol);

%
% for needleman, set the boundary condition to gap penalites
%

for(i=2:size(v,2))
    v(1,i)=v(1,i-1)+gap;
end

for(i=2:size(v,1))
    v(i,1)=v(i-1,1)+gap;
end

for n=1:length(s2)
    nn= n+1;
    for m=1:length(s1)
        mm= m+1;
        if s2(n) == s1(m)
            diagonal= match;
        else
            diagonal = mismatch;
        end

        diagonal = diagonal + v(nn-1,mm-1);
        upScore = v(nn-1,mm)+gap;
        leftScore = v(nn,mm-1)+gap;

        v(nn,mm) = max([upScore, leftScore, diagonal]); % , 0]);
    end
end

len = v(end,end);

```

8.1 Problem 7 source code

```
function nma_alignment_main()

h0= nma_alignment_GUI;

nma_alignment_callbacks('init',h0);
```

```
function R=nma_problem7_part_a(s1,s2,match,mismatch,gap)
%function R=nma_problen7_part_a(s1,s2,match,mismatch,gap)
%
% solves problem 7, HW 1 for MATH 127
% by Nasser Abbasi
% sept 25, 2002.
%
% implements Needleman-Wunsch algorithm
%
% INPUT
% s1: is one sequence. example S1=['g' 'a' 't' 'c' 'g'];
% s2: is the second sequence.
% match: score for matching bases. such as 1
% mismatch: score for mismatch. such as -1
% gap: penatly factor for gaps. such as -2
%
% NOTE: S1 is the row sequence at the top, and S2 is column sequence at left.
%
%
% reserve space for the score matrix.
nRow=length(s2)+1;
nCol=length(s1)+1;

S=zeros(nRow,nCol); %setup space for scoring matrix.

%
% for needleman, set the boundary condition to gap penalites
%
for(i=2:size(S,2))
    S(1,i)=S(1,i-1)+gap;
end

for(i=2:size(S,1))
    S(i,1)=S(i-1,1)+gap;
end
```

```

for n=1:length(s2)
    nn= n+1;
    for m=1:length(s1)
        mm= m+1;
        if s2(n) == s1(m)
            diagonal= match;
        else
            diagonal = mismatch;
        end

        diagonal = diagonal + S(nn-1,mm-1);
        upScore = S(nn-1,mm)+gap;
        leftScore = S(nn,mm-1)+gap;

        S(nn,mm) = max([upScore, leftScore, diagonal]); %, 0]);
    end
end

% print the score matrix
fprintf('\t\t');
for(i=1:length(s1))
    fprintf('%c\t',s1(i));
end
fprintf('\n');

for(i=1:nRow)

    if(i==1)
        fprintf('\t');
    else
        fprintf('%c\t',s2(i-1));
    end

    for(j=1:nCol)
        fprintf('%d\t',S(i,j));
    end
    fprintf('\n');
end

%print the alignment
doAlignment(S,s1,s2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%5

```



```

%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function doAlignment(S,topSeq,leftSeq)

[nRow,nCol]=size(S);
i=nRow;
j=nCol;
A=zeros(2*(nRow+nCol-2),2); %create space for the alignment.
k=0;

top=topSeq(j-1);
btm=leftSeq(i-1);

while(1)
    k=k+1;
    %   if(i==1 | j==1)
    %       if(i==1)
    %           btm='-';
    %           while(j>=i)
    %               top=topSeq(j);
    %               A(1,k)=top;
    %               A(2,k)=btm;
    %               j=j-1;
    %               k=k+1;
    %           end
    %       else
    %           top='-';
    %           while(i>=1)
    %               btm=leftSeq(i);
    %               A(1,k)=top;
    %               A(2,k)=btm;
    %               k=k+1;
    %               i=i-1;
    %           end
    %       end
    %   end
    %   break;
%   end

[newi,newj]=maxParent(S,i,j);

if(newi==i) %same row
    top=topSeq(j-1);
    btm='-';
else
    if(newj==j) %same column
        top='-';
    end
end

```

```

        btm=leftSeq(i-1);
    else
        top=topSeq(j-1);
        btm=leftSeq(i-1);
    end
end
A(1,k)=top;
A(2,k)=btm;

if(newi==1 | newj==1)
    break;
end

i=newi;
j=newj;
end

for(i=k:-1:1)
    fprintf('%c',A(1,i));
end
fprintf('\n');
for(i=k:-1:1)
    if(isequal(A(1,i),A(2,i)))
        fprintf('|');
    else
        fprintf(' ');
    end
end
end
fprintf('\n');
for(i=k:-1:1)
    fprintf('%c',A(2,i));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [newi,newj]=maxParent(S,i,j)

[nRow,nCol]=size(S);
if(i==1 & j==1)
    newi=i;
    newj=j;
else

```

```

if(j==1)
    newi=i-1;
    newj=j;
else
    if(i==1)
        newj=j-1;
        newi=i;
    else
        if(S(i,j-1) > S(i-1,j-1))
            if(S(i,j-1)>S(i-1,j))
                newi=i;
                newj=j-1;
            else
                newi=i-1;
                newj=j;
            end
        else
            if(S(i-1,j-1)>=S(i-1,j))
                newi=i-1;
                newj=j-1;
            else
                newi=i-1;
                newj=j;
            end
        end
    end
end
end
end
end

```

```

function R=nma_problem7_part_b(n,match,mismatch,gap)
%function R=nma_problen7_part_b(n,match,mismatch,gap)
%
% solves problem 7 part b, HW 1 for MATH 127
% by Nasser Abbasi
% sept 25, 2002.
%
% find the average length of the longest increasing subsequence
% in a permutation of length n.
%
% INPUT
% n : the length of the sequence.
% match: score for matching bases. such as 1
% mismatch: score for mismatch. such as -1
% gap: penatly factor for gaps. such as -2
%
%

```

```

thePerms = perms(1:n);
totLen=0;

for(k=1:size(thePerms,1))
    totLen = totLen + getLengthOfLongestSubSequence(thePerms(k,:),match,mismatch,gap);
end

av=totLen/factorial(n);

fprintf('Average length of longest increasing subsequence in perms of length %d is %f\n',...
    n,av);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function len=getLengthOfLongestSubSequence(seq,match,mismatch,gap)

s2=1:length(seq);
s1=seq;

%
% reserve space for the score matrix.
nRow=length(s2)+1;
nCol=length(s1)+1;

v=zeros(nRow,nCol);

%
% for needleman, set the boundary condition to gap penalites
%

for(i=2:size(v,2))
    v(1,i)=v(1,i-1)+gap;
end

for(i=2:size(v,1))
    v(i,1)=v(i-1,1)+gap;
end

for n=1:length(s2)
    nn= n+1;
    for m=1:length(s1)
        mm= m+1;
        if s2(n) == s1(m)
            diagonal= match;

```

```
    else
        diagonal = mismatch;
    end

    diagonal = diagonal + v(nn-1,mm-1);
    upScore  = v(nn-1,mm)+gap;
    leftScore = v(nn,mm-1)+gap;

    v(nn,mm) = max([upScore, leftScore, diagonal]); % , 0]);
end
end

len = v(end,end);
```

3.3 HW 3

HW 3

Mathematics 127
Mathematical and Computational Methods in
Molecular Biology

Fall 2002
UC Berkeley, CA

Nasser M. Abbasi

Fall 2002 Compiled on August 2, 2022 at 8:09am [public]

Contents

1	Problems	3
2	Problem 1	6
3	Problem 2	16
4	Problem 3	20
5	Problem 4	24
6	Problem 5	26
6.1	Problem 5 source code	31

1 Problems

Problem Set 3 (due Tuesday October 15)

MATH 127: Mathematical and Computational Methods in Molecular Biology

Please work on the starred problem alone.

Problem 1

Prove that a non-trivial connected graph is Eulerian if, and only if, every edge belongs to an odd number of cycles.

Problem 2*

The goal of this problem is to obtain some familiarity with running RepeatMasker:

You'll need to use the web server at

<http://ftp.genome.washington.edu/cgi-bin/RepeatMasker>

a) Get the sequence with accession AF548051 from Genbank. How long is the particular A tail of the repeat? Find the article in the literature (on which this Genbank entry is based), read it, and report on the maximum length of observed Alu A tails in the human genome. What is the apparent function of the A tail?

b) RepeatMask the sequence at the RepeatMasker web server. How does the answer depend on the type of organism selected? What SW score do you get? What is its meaning?

c) How dependent is RepeatMasker on the A tail? Try removing the A tail. Does RepeatMasker still find the repeat? Try extending it considerably (beyond the observed length of A tail in the genome)? Does RepeatMasker still find the repeat?

Problem 3

Consider the overlap detection method described in class for DNA: Compute the convolution vector 4 times, each time setting one of the bases to 1 and the rest to 0. Suppose you try to find the overlap between two random DNA sequences of length 500, each of which contains equal amounts of the four bases. What do you expect the maximum value of an element in the convolution vector to be? What kind of bound does this place on the minimum size overlap you can detect reliably with the method?

Problem 4

The paper by Pevzner, Tang and Waterman states that “the spectral alignment problem can be efficiently solved by dynamic programming in the

case where the number of mutations is small". Make this statement precise and describe the algorithm for solving it.

Optional Problem

Implement the overlap detection method discussed in class.

Note: this is easy using the MATLAB **FFT** and **IFFT** commands (type *help FFT* or *help IFFT* to learn more about them). To view the convolution vector you can use the **PLOT** command.

a) Write a function that takes as input two sequences f, g of the same length and finds the convolution $f * g$.

b) Test your program on the sequences [00001010110000010] and [01010110000010000]. What is the best shift? (Remember, you will have to reverse one of the sequences).

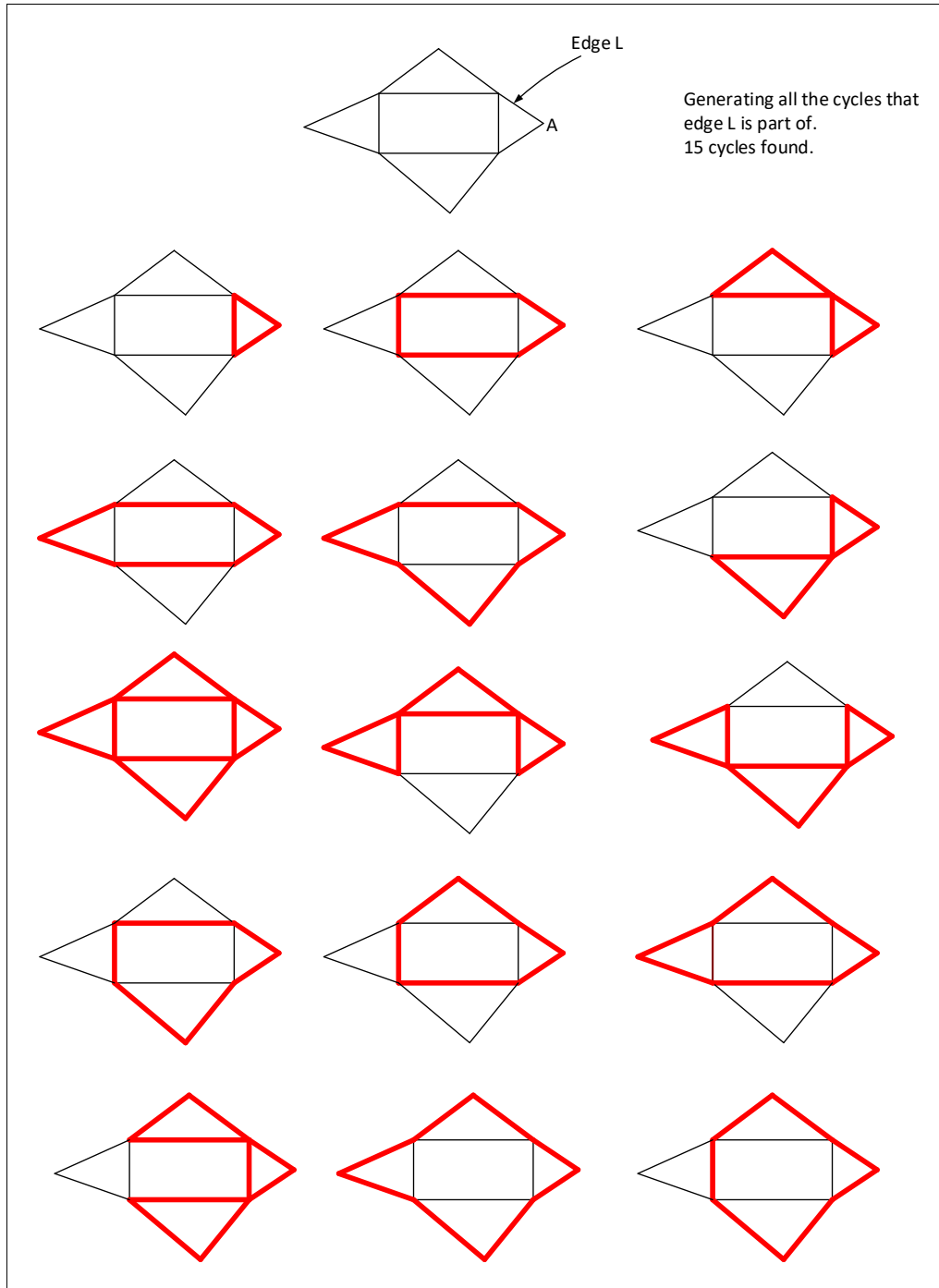


Figure 1: Example generating all cycles an edge is part of

2 Problem 1

Math 127, UC Berkeley.
 HW 3
 Problem 1
 By Nasser Abbasi

Question

Prove that a non-trivial connected graphs is Eulerian IFF every edge belongs to an odd number of cycles.

Answer

An euler graphs has an even degree on all of its vertices.

A cycle in the above, is meant to be a path v_1, v_2, \dots, v_n where $v_1 = v_n$

Since this is an IFF problem, There are two statements that needs to be proved.

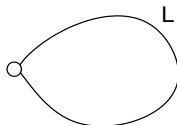
Statement one: IF a graph is Euler, then each edge belongs to an odd number of edges.

Statement two: IF each edge belongs to odd number of cycles, then the graph is Euler.

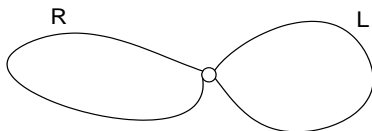
Proof for statement one

Proof by induction over the number of vertices v

For number of vertices = 1, the statement is true. Because given an Euler graph with one vertex, there are even number of edges, hence all edges must be of this form



Hence, each edge can only belong to 1 cycle. Since if we have more than one edge, as in



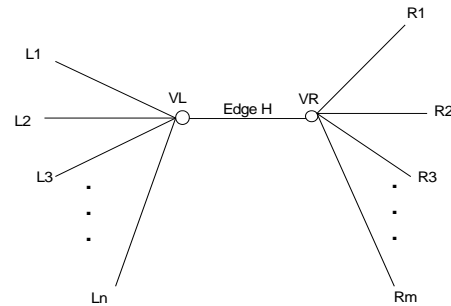
Then the cycle over L can not go over R, since this implies the vertex in the middle was visited twice. And the cycle over R can not go over L as well.

Hence the statement is true for all Euler graphs of one vertex.

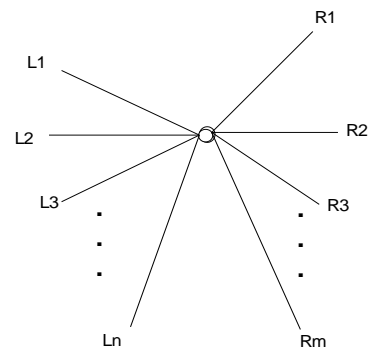
Now, Assume the statement is true for all connected Euler graphs with K vertices.

Now, need to show the statement is true for any Euler graph Z of K+1 vertices.

Looking at any one edge H in Z, such as (Figure 1 below)



Transform the graphs of K+1 vertices to K vertices, remove the edge H and collaps the two vertices to one vertex to get this (figure 2):



Since this is a K vertices Euler graph, then it will have all the edges in it part of odd number of cycles (by assumption). That is, number of cycles over each L edge is odd, and number of cycles of each R edge is odd.

Since the above also is a Euler graph, then the degree of the above one vertex is even. This means $n+m$ is an even number. Hence n and m can be both even, or can be both odd numbers. (will show below that only case possible is for n,m to be both odd numbers).

Now, looking back at figure 1 above, since this is a Euler graph, then the degree of VL is even, and the degree of VR is even.

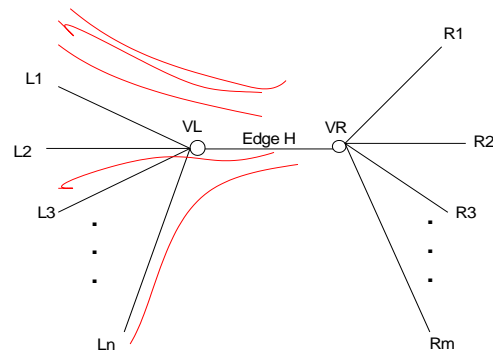
But edge H adds one degree to each vertex, that is $n+1$ is even, and $m+1$ is even.

hence n and m must be both odd numbers.

Now, I need to show that an edge such as H in the Euler graph of $K+1$ vertices (see figure 1) can only be part of an odd cycles.

There are two cases here.

Case one, all the cycles passing over the edges L_i (or edges R_j) also pass over edge H .



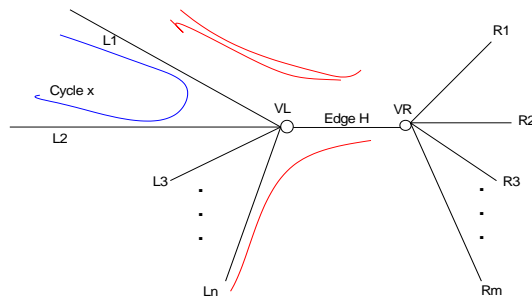
In this case, we have, since each edge in L has an odd number of cycles as shown above, the total number of cycles that edge H is part of is given by

$$q_1 + q_2 + \dots + q_n = Z$$

where N is an odd number, and each q is an odd number.

When we add odd number of times odd numbers, we get an odd number.
 Hence Z is an odd number.
 Hence number of cycles over H are odd.

Case two, in this case, not all cycles that pass over any or all edge L_i (or edges R_j) also pass over edge H as in the following figure example



Now, each time we get a cycle such as cycle x above, it will contribute 1 to the count of cycles to both edges it travels over (in this example edges $L1$ and $L2$).

This cycle also can not come back over another edge to cover H , since then it will visit vertex VL , and this is not possible by the definition of a cycle.

Hence for each such cycle as x above, I have to subtract 2 from the total possible number of cycles passing over edge H .

So in this case, total number of cycles passing over edge H is

$$(q_1 + q_2 + \dots + q_n) - (2 * \text{number of cycles not going over edge } H)$$

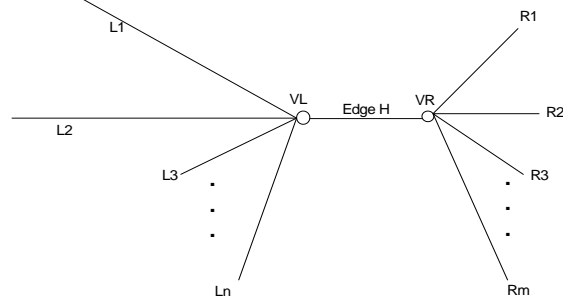
but since n is odd, (it is 1 subtracted from the degree of the vertex vL), and each q is odd, then the above is an odd number – even number. Which is an odd number.

Hence number of cycles over H are odd.

This completes the proof for statement one.

To prove statement two: *IF each edge belongs to odd number of cycles, then the graph is Euler.*

Looking at any one edge, such as H, in such a graph



I need to show, given that number of cycles over H are odd, then the degree of each vertex must be even (i.e. the graph is Euler).

I use proof by contradiction.

In a proof by contradiction, when we need to prove the following statement is true

IF $A \Rightarrow B$

We assume that given the following is true:

A and NOT B

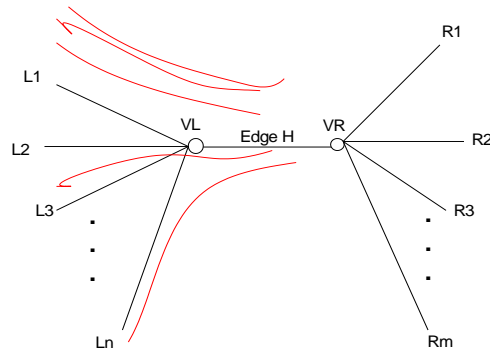
and we try to show this is not possible. Hence $A \Rightarrow B$.

So, in this example, A is 'each edge has odd number of cycles', B is 'Graph is Euler'.

So, Assume we have a graph with each edge has odd number of cycles, and it is NOT Euler.

Looking at the above figure, we have two cases.

Case one, all the cycles passing over edges L_i (or edges R_j) also pass over edge H.



Summing all cycles going over H, assume number of cycles over L_i is q_i ,
 $q_1 + q_2 + \dots + q_n = Z$

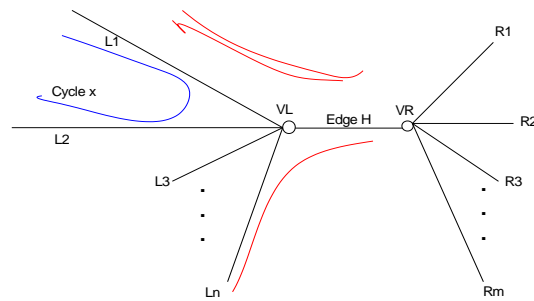
where Z is the cycles over H. But Z must be an odd number (given), and each q_i in the above sum is odd also (given).

The only way to add odd numbers N times and still get an odd number, is when N itself is odd.

Hence the number of L edges coming into VL in the above diagram is odd. Add the edge H itself, then the degree of vertex VL can only be even. Hence for this case, it is not possible to have odd number of cycles and have the vertex not have an even degree. Hence the graph must be Euler, which contradict the assumption.

Now to prove it for case two:

Case two: not all cycles that pass over the edges L_i (or edges R_j) also pass over edge H as in the example



Again, as argued earlier:

Now, each time we get a cycle such as cycle x above, it will contribute 1 to the count of cycles to both edges it travels over (in this example edges $L1$ and $L2$).

This cycle also can not come back over another edge (say $L3$) to cover H , since then it will visit vertex VL , and this is not possible by the definition of a cycle.

Hence for each such cycle as ' x ' above, we subtract 2 from the total number of cycles passing over edge H . So total number of cycles Z going over H is

$$(q_1 + q_2 + \dots + q_n) - (2 * \text{cycles not going over } H) = Z$$

or

$$(q_1 + q_2 + \dots + q_n) - \text{even number} = Z$$

Since Z is odd, then $(q_1 + q_2 + \dots + q_n)$ must be odd.

Hence n must be odd.

Hence the number of edges coming into VL in the above diagram is odd. Add the edge H itself, then the number degree of vertex VL can only be even. Hence for this case, it is not possible to have odd number of cycles and have the vertex not have an even degree. Hence the graph must be Euler, which contradict the assumption.

This completes the proof of the second statement.

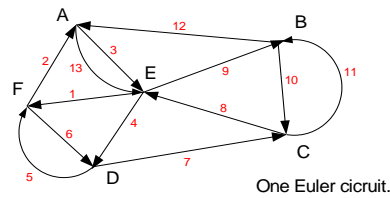
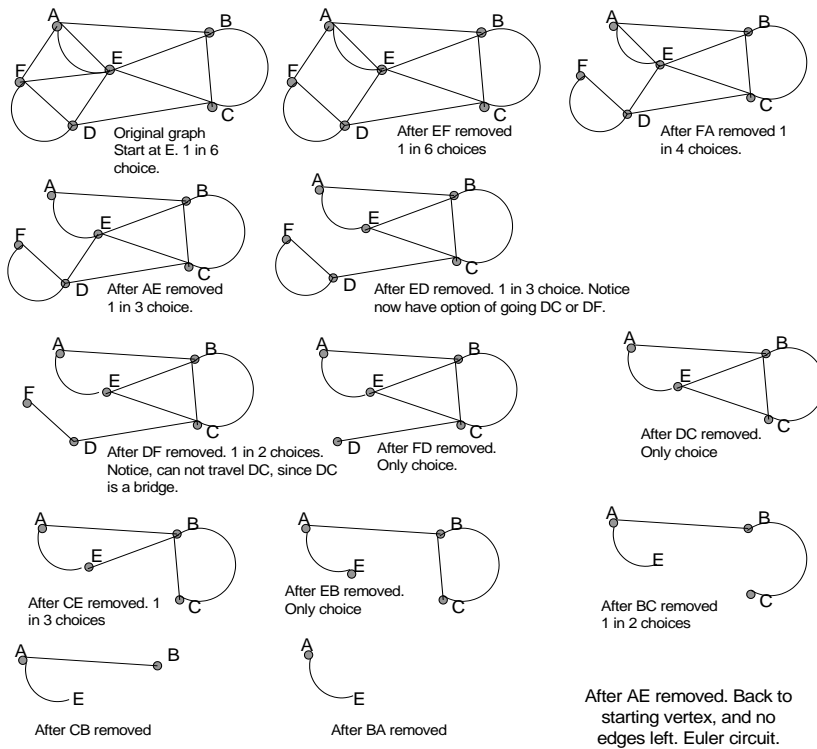
QED.

Problem 1 extra

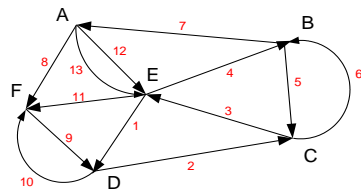
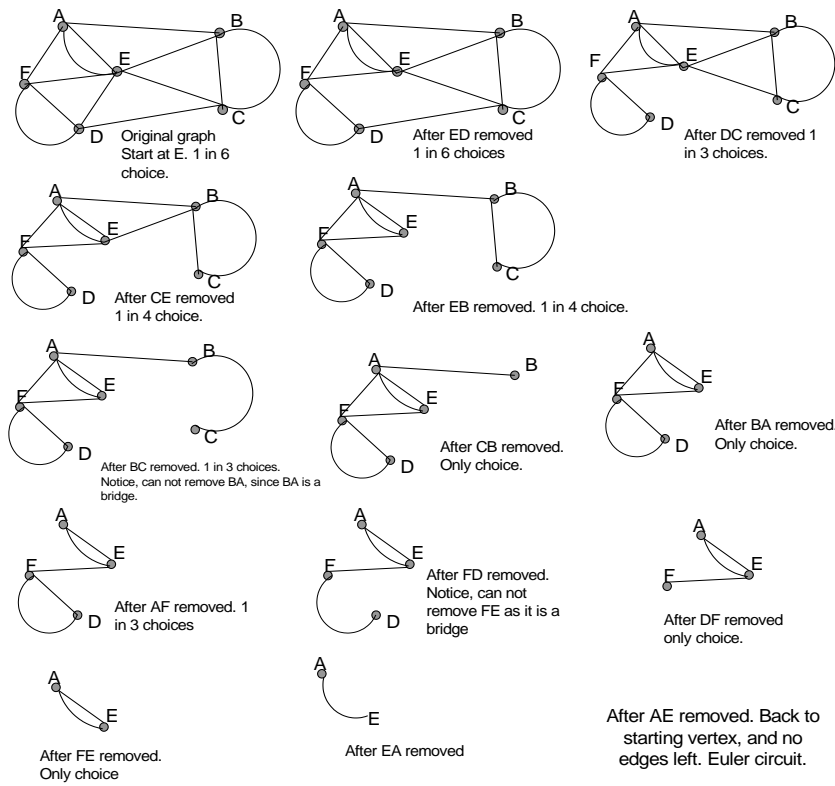
Extra on problem 1

Dr, as I was studying for this problem, I read about the Fleury algorithm to find all the cycles in an Euler graph. To learn how this algorithm works, I found 3 cycles in some graph, showing step by step how the algorithm works. I am including this below, (even though it is not required to solve this problem, but it helped me understand the subject a little more).

Only connected graphs with no vertices of odd degree can have an Euler circuit.
 So the above graphs must have at least one Euler circuit. To find, use Fleury algorithm:
 1. Start at any vertex.
 2. Pick an edge to travel. If one of the edges is a bridge (Going along it, there is no way to come back to the vertex other than on it, then do not select it, unless it is the one edge left from that vertex).
 3. Remove the edge traveled. If a vertex left with no edges leaving it, remove the vertex.
 4. repeat step 2 until no more edges left.
 5. The order of the edges traveled is the Euler circuit.

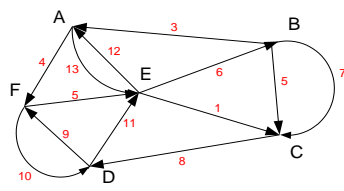
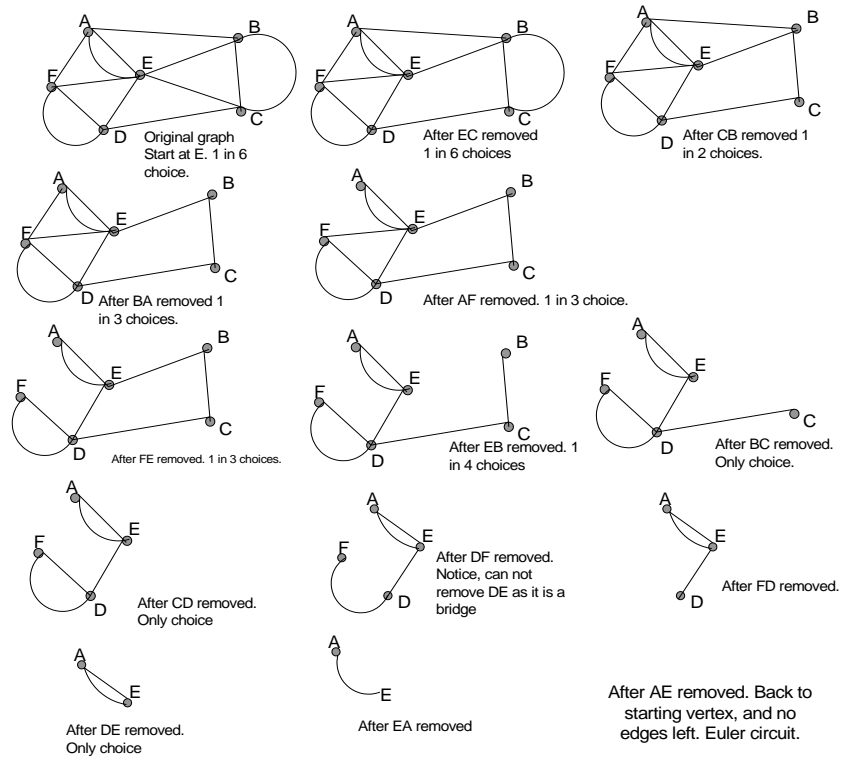


Showing Euler circuit. Visit each edge once, and visit all edges. Starting from E and back to E.



Showing Euler circuit. Visit each edge once, and visit all edges. Starting from E and back to E.

second Euler circuit starting from E



Showing Euler circuit. Visit each edge once, and visit all edges. Starting from E and back to E.

Third Euler circuit starting from E

3 Problem 2

MATH 127, UC Berkeley
HW 3
Problem 2
Nasser Abbasi

Part A

From NCBI web site <http://www.ncbi.nlm.nih.gov>, typed AF548051 in the 'for' window, with 'search' window set to Nucleotide, and hit 'GO'. The result is:

Search for AF548051 in the Nucleotide database. Results show: 1: AF548051 Homo sapiens clone AFAM1-Ya5NBC243.seq Alu Ya5a2 subfamily repeat region gi|23395425|gb|AF548051.1|[23395425]

Then select the 'FASTA' option in the display menu, and click on the link, I get

```
>gi|23395425|gb|AF548051.1| Homo sapiens clone AFAM1-Ya5NBC243.seq Alu Ya5a2 subfamily repeat region
GGCCGGGGCGCGGTGGCTCACGCCTGTAATCCACGACTTTGGGAGGCCGAGGCGGGCGGATCACGAGGTC
AAGAGATCGAGACCATCCCGGTAAACGGTGAAACCCCGTCTACTAAAAATACAAAAAAATTAGCCG
GGCGTAGTGGGGGGCGCCTGTAGTCCAGCTACTTGGGAGGCTGAGGCAGGAGAAATGGCGTGAACCCGGG
AGGCGGAGCTTGCAGTGAGCCGAGATCCCGCCACTGCCTCAGCCTGGGCGACAGAGCGAGACTCCGTC
TCAAAAAAAAAAAAAAAAAAAAAAAAAAGGAAA
```

The A tail of the above repeat is 'AAAAAAAAAAAAAAAAAAAAAAAAAGCAAA' which is **24 long**. (I stopped counting at just before 'GGAAA'. Since from the paper, it said to stop counting the A-tail when at least 2 consecutive non-adenosine bases show up.

From the article,

AUTHORS Roy-Engel,A.M., Salem,A.H., Oyeniran,O.O., Deininger,L.,
Hedges,D.J., Kilroy,G.E., Batzer,M.A. and Deininger,P.L.
TITLE Active alu element 'A-Tails': size does matter
JOURNAL Genome Res. 12 (9), 1333-1344 (2002)

From the article, it said the maximum A-length for Alu was 97 bases:

Some of the A-lengths in this group reach as high as 97 bases for Alu and almost 180 bases for L1.

The apparent function of the A-Tail, is that the length of the Alu A-Tail correlate to which Alu is active, that is, which Alu elements are able to retropose. So the A-Tail is a factor in determining the retropositional capability of the Alu sequence as it said in the citation.

Part B

cut/paste the above, and went to <http://ftp.genome.washington.edu/cgi-bin/RepeatMasker> and submitted the above sequence using default setting. Tried for all organisms. The result is shown below

Organism	Number of matching repeats found in DB	SW score	Matching repeats	family	Matching position In query
Primates	1	2870	AluYa5	SINE	1..311
RODENTs	2	528 772	PB1 FAM	SINE SINE	2..132 137..296
Other mammals	2	625 805	FLAM_A FAM	SINE SINE	2..133 137..300
other vertebrates	0				
Arabidopsis	1	219	(A)n	Simple_repeat	283..311
Grasses	0				
Drosophila	1	219	(A)n	Simple_repeat	283..311

By doing the search assuming the query sequence belongs to different organism (that is, search different repeats database), a different alignment score is obtained. The highest score was for Primates covering the whole query sequence. This means the primates have all of the query sequence (1..311) repeated in the genome in different locations. While for Rodents and other mammals, only parts of the sequence is repeated (2 parts, the first 'half' and the 'second' half). So, this repeat sequence is unique to Primates only.

In Arabidopsis and Drosophila, only the tail-A 'AAAAAAAAAAAAAAAAAAAAAAAAAGGAAA' was found to be a repeat.

Part C

Here, I removed the A-Tail. This is the subsequence AAAAAAAAAAAAAAAAAAAAAAAAAGGAAA from the original sequence, and re-run all the tests as above. This is the new table.

Organism	Number of matching repeats found in DB	SW Score	Matching repeats	family	Matching position In query
Primates	1	2637	AluYa5	SINE	1..282
RODENTs	2	528 649	PB1 FAM	SINE SINE	2..132 137..282
Other mammals	2	625 660	FLAM_A FAM	SINE SINE	2..133 137..282
other vertebrates	0				
Arabidopsis	0				
Grasses	0				
Drosophila	0				

So this shows that repateMasker was still able to find the repeats in the data base without the A-Tail.

Now, I extend the A-Tail, by adding 'A's to the end of the tail. I added 120 'A's to the A-tail, now the A-tail length is 149 (since the A-Tail original length was 29). I choose 120, since from the paper it said that the longest A-Tail was 97 in recent Alu insertions. So, this means an extension of 50 beyond the longest A-Tail

So, the new query sequence I used is this:

```
>gj23395425|gb|AF548051.1| Homo sapiens clone AFAM1-Ya5NBC243.seq Alu Ya5a2 subfamily repeat region
GGCCGGGCGCGGTGGCTCACGCCTGTAATCCAGCACTTTGGGAGGCCGAGGCGGGCGGATCACGAGGTC
AAGAGATCGAGACCATCCCGGCTAAACGGTGAAACCCCGTCTACTAAAAATCAAAAAAATTAGCCG
GGCGTAGTGGCGGGCCCTGTAGTCCAGCTACTTGGGAGGCTGAGGCAGGAGAATGGCGTGAACCCGG
AGCGGAGCTTGCAGTGAGCCGAGATCCCGCCACTGCACTCCAGCCTGGGCGACAGAGCGAGACTCCGTCTC
AAAAAAAAAAAAAAAAAAAAAAAAAGGAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
A
```

So, the new query sequence is now of length 311+120=431

I run the same tests as above, and the result is shown in this table

Organism	Number of matching repeats found in DB	SW Score	Matching repeats	family	Matching position In query
Primates	1	2880	AluYa5	SINE	1..431
RODENTs	2	528 773	PB1 FAM	SINE SINE	2..132 137..431
Other mammals	2	625 806	FLAM_A FAM	SINE SINE	2..133 137..431

other vertebrates	0				
Arabidopsis	1	1299	(A)n	Simple_repeat	283.431
Grasses	0				
Drosophila	1	1299	(A)n	Simple_repeat	283.431

The above shows extending the A-tail considerably did not affect repeatMasker ability to find the repeats.

Notice that in Arabidopsis and Drosophila, only the tail-A is the repeat. The more 'A's I added, the higher the score became for those organisms. This shows there are long simple_repeats of 'A's in those organisms.

4 Problem 3

MATH 127, UC Berkeley.
HW3
Problem 3
Nasser Abbasi

Answer

To help me understand this problem, I worked a simple example of how to use convolution to find similarity between 2 DNA sequences.

Original DNA sequences we want to find the where max alignment between them occur

$$\begin{array}{c} \text{ACGTT} \\ \text{TCGAA} \end{array}$$

Replace one letter at by 1, and all the others by zero. Do this for A,C,G,T at a time. So we get 4 sets of sequences. Do the convolution between each 2 sequences at a time. So, we get 4 convolution vectors a the end. Next, sum the 4 convolution vectors.

$$\begin{array}{r} \text{A=1} \quad \begin{array}{r} 10000 \\ 00011 \\ \hline \text{convolution} \quad 01100 \end{array} \quad \text{C=1} \quad \begin{array}{r} 01000 \\ 01000 \\ \hline 10000 \end{array} \quad \text{G=1} \quad \begin{array}{r} 00100 \\ 00100 \\ \hline 00100 \end{array} \quad \text{T=1} \quad \begin{array}{r} 00011 \\ 10000 \\ \hline 01100 \end{array} \end{array}$$

Now, sum the 4 convolution vectors

$$\begin{array}{r} 01100 \\ 10000 \\ 00100 \\ 01100 \\ \hline 12300 \\ \downarrow \\ \text{ACGTT} \\ \text{TCGAA} \end{array}$$

So, the max value is at position 3 (it has the value of 3). So, looking back the two DNA sequences, this tells us where the max similarity is.

Nasser Aslami
overlap_convolution.vad
oct 12, 2002.

The convolution is a circular convolution. Done using these steps.

1. Align the sequences on top of each others, multiply corresponding elements (i.e. element at position j from top row with element at position j from second row, and add the final result.

$$\begin{array}{r} \text{multiply} \downarrow \quad \begin{array}{r} 10000 \\ 00011 \\ \hline 00000 = 0 \end{array} \\ \text{sum} \rightarrow \end{array}$$

The above gives us the first element of the convolution vector, which is zero.

2. Now do a one position right shift of the lower sequence, and wrap around. So, a sequence 'abcde' when shifted one position to the right, will become 'eabcd'.

$$\begin{array}{r} 10000 \\ 10001 \\ \hline 10000 = 1 \end{array}$$

The above gives us the first element of the convolution vector, which is 1.

again, right shift the lower sequence again from above, and multiply and sum, we get

$$\begin{array}{r} 10000 \\ 11000 \\ \hline 10000 = 1 \end{array}$$

again, right shift the lower sequence again from above, and multiply and sum, we get

$$\begin{array}{r} 10000 \\ 01100 \\ \hline 00000 = 0 \end{array}$$

again, right shift the lower sequence again from above, and multiply and sum, we get

$$\begin{array}{r} 10000 \\ 00110 \\ \hline 00000 = 0 \end{array}$$

Since now the lower sequence has been shifted 4 times, and the size of the sequence is 5, we stop. The final convolution vector is then [01100].

To make sure the above makes sense and I did not make any mistakes, I worked a bigger example below to verify the method, to find the overlap between two larger DNA sequences.

Original DNA sequences we want to find the overlap between them, the red sections is where I would expect the sequences to overlap.

TATAGCCTCCTC
TCCTCATCCTGT

Resulting in TATAGCCTCCTCATCCTGT

	A=1	C=1	G=1	T=1
	010100000000 000001000000	000001101101 011010011000	000010000000 000000000010	101000010010 100100100101
convolution	010100000000	123123134122	000000000100	122122213112

Now, sum the 4 convolution vectors

010100000000	
123123134122	
000000000100	
122122213112	
255345347334	↓
TATAGCCTCCTC	
TCCTCATCCTGT	

So, the max value is at position 9 (it has the value of 7). So, looking back the two DNA sequences, this tells us where the overlap max position is.

Hassan Abbasi
overlap_convolution.vsd
oct 12, 2002.

The above shows the method does find the overlap region. (but it is not clear to me how does one go about determining the extent of the overlap if needed. May be the extent of the overlap is not needed, we just need a point to know where to fold the two sequences next to each other from) and this method gives us this.

Now, to answer the question. For the maximum value, it will occur when we have maximum overlap.

The max is when the two sequences are identical ofcourse. However, since these are random sequences, one would assume this is not likely to occur.

Since there are equal number of each base, there will be 125 of each type of base. These will be randomly distributed. So there is a chance of 1 out of 4 that an A from one sequence will be at the same position as an A in the other sequence, and the same for T,G and C.

Since the sequence length is 500, the chance of both sequences coming out the same is then $(1/4)^{500}$, which is almost impossible, but when this is the case, each convolution vector will a max value of 125, and the final convolution vector (the sum of the 4 convolution vectors will have a max value of **500**.

But for normal random distribution, the bases are uniformly distributed, and there is 25% chance that a base at one position in one sequence will be the same as the base in the same position in the second sequence. So, the convolution vector for say 'A' will have **a max value of 125/4 or about 32.25**. But we add 4 convolution vectors to get the final

convolution vector, and the chance that the vector for 'G' or 'C' or 'T' will have its maximum value at the same column as 'A' is $1/500$, so I can not just add 32.5 4 times to get 125. (I am assuming a fair random sequence generator, else I would have picked 125 as the maximum).

Instead, I think I should pick one maximum from one vector (say 'A') which is 32.5, and then assume the value at this column in the second, third and fourth row vectors from each of other 3 convolution vectors is the average value, which is 16 so, this gives a maximum of $32.5+16+16+16=80$.

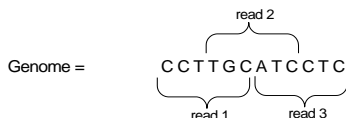
This 80 value gives a minimum size of 80 overlap that can be reliably detected (or about 20% of the size of the sequence).

5 Problem 4

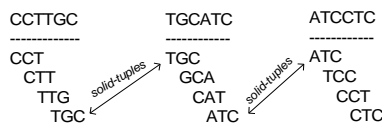
MATH 127
 HW 3
 Problem 4
 Nasser Abbasi

The Euler assembly algorithm is a determination of superpath in de-Bruijn graph. A read is an edge in such a graph. However, these reads should have little errors in them. To remove these errors early on, the reads are broken into 1-tuples and multiple sequence alignment is performed on these short strings and read errors detected and the original reads are then modified to remove these errors, and the new modified error-corrected reads (reduced errors) are then used in the de-Bruijn graph.

The spectral alignment problem is one solution to finding these read errors early on. To help me understand the concepts in the paper more, I made the following very simple example. Imagine a Genome G which is 'CCTTGCATCCTC', with 3 reads, each of length 6. let l , the length of the tuple be 3. This is what I get:



Assume we have 3 reads, each of length $n=6$. Assume $L=3$ (tuple size). Each read is broken into $n-L+1$, or 4 tuples as shown.



In the above, using $M=1$, there are those shown solid-tuples. Solid tuple are ones that belong to more than M reads. In the above simple example, there are 3 such solid tuples. The rest are called weak L -tuples. The approximated Genome (used by the error-correction method in the paper) will then be the set of all **solid-tuples**.

Now, a collection of L -tuples short strings is called a spectrum. A string 's' belongs this spectrum if each one of its L -tuples can be found in this spectrum. The spectral alignment problem is to find the minimum number of letter changes we have to make in the string 's' to cause all its tuples to be found in the given spectrum.

As an example, let string $s='CCTG'$, with $L=3$, it then has 'CCT' and 'CTG' as its two tuples. ($4-3+1=2$).

Given a spectrum $T=\{'ACT', 'TCT', 'CCA', 'CTG'\}$, we see that 's' as it stands does not belong to T, since one of 's' tuples (the first one) is missing from T.

To make 's' a T-string, we have to change one or more letters in 's', but when doing this change, we have to make sure we do not cause a removal of a tuple that was already found in T, else we just added a tuple in and removed one. In the above then, if we change the first letter in 's' from C to A, this will cause 's' now to be a T-string, since now the first tuple becomes 'ACT' instead of 'CCT' and this already in T. Hence this is the minimum change needed to make 's' be a T-string. A one letter change. **So, the reason why we need the number of mutations made to be small, so that we do not end up changing the original reads too much, causing the final approximated Genome to be much different from the actual one.** The approximated Genome (with error-correction) is the one used to construct the de-Bruijn graph.

The algorithm for spectral alignment works as follows.

Step 1. First construct the set of all solid L-tuples from all the initial reads from the sequencing project. (this is the set of all L-tuples that are found in more than M reads, where M is some threshold). This will be our initial spectrum, called T.

Step 2. Now, for each read which does not have all its L-tuples in T (i.e. it contains a number of weak L-tuples with respect to T) make the smallest letter changes (mutations) to cause one weak L-tuple to become a solid L-tuple (i.e. to show up in T). For example, in the above, I made one letter change in the weak L-tuple 'CCT' to cause to become a solid L-tuple.

Step 3. Now, generate the T spectrum again.

Repeat steps 2,3. Each time using the newly modified reads until no more reads with weak L-tuples is found (or stop at some threshold on number of mutations to make?).

This completes the algorithm, and generates new modified reads and an approximate Genome based on those error-corrected reads.

6 Problem 5

Math 127
HW3
Problem 5 (Optional)
Nasser Abbasi

Part (A)

I wrote a function that takes two DNA sequences, and returns the convolution vector.
The position of the maximum element in the convolution vector is where the overlap occurs.

This below are few examples showing how to use this function. (I highlight by red where the overlap actually is).

Example 1

```
>> clear all
>>% 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8
>> x='ACGTATTACCCCGGGCCC';
>> y='CCCGGGCCCTAGTATT';
>> nma_hw3_problem5(x,y)
ans =

    4    5    4    2    3    2    6   10   15    8    6    6    2    3    6    4    4    3

>> [v,I]=max(ans);
>> I

I =

    9
```

So, this says the overlap occurred at position 9, which is correct by looking at the above sequences.

Example 2

```
>>% 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
>> x='ACTGTTGATATATATATATA';
>> y='ATATATCGTCGTCGTCGTCG';
>> nma_hw3_problem5(x,y)

    4  4  3  5  4  6  2 10  3  7  4  9  1 10  3  7  5  7  2  6

>> [v,I]=max(ans);
>> I

I =

    8
```

Notice that there are another maximum location (of value 10) in the convolution vector, which is at position 14. This is the 'end' location. The `max()` function in matlab finds the first max in a vector. For overlap, I need the last one, the one near the edge.

Part B

To test the program against the given sequences, I use directly the circular convolution function I wrote (which is called by the function used in part A).

This is the output

```
>> help nma_cconv
```

```
function nma_cconv(A,B) implement direct circular convolution for A,B
vectors. Must be of equal length (for now).
returns V, the convolution vector
```

```
>>% 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7
>> x=[0 0 0 1 0 1 0 1 1 0 0 0 0 0 1 0];
>> y=[0 1 0 1 0 1 1 0 0 0 0 0 1 0 0 0 0];
```

```
>> nma_cconv(x,y)
```

```
ans =
```

```
1 2 1 5 1 2 1 1 1 2 1 1 1 1 2 1 1
```

```
>>
```

So, from the above, the maximum convolution is at position **4** and this gives the best shift.

```

function V=nma_hw3_problem5(x,y)
%function p=nma_hw3_problem5(x,y)
%
% This function takes 2 DNA sequences, and returns
% the convolution vector for the sequences
%
% calls the function nma_cconv.m to do the convolution

p=0;

if nargin ~=2
    error('x,y expected');
end

N=length(x);
if(N ~= length(y))
    error('x,y must be equal length');
end

DNA='ATCG';
V=zeros(N,1); % store final convolution vector here.

for(i=1:length(DNA))
    V=V+nma_cconv(normalize(x,(DNA(i))), ...
        normalize(y,(DNA(i))));
end

V=V';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% this function replaces each letter in S with 0
% except those that matches w, they are replaced with 1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function T=normalize(S,w)

S=upper(S);
T=zeros(length(S),1);

for(i=1:length(S))
    if(S(i)==w)
        T(i)=1;
    end
end
end

```

```
function v=nma_cconv(A,B)
%function nma_cconv(A,B) implement direct circular convolution for A,B
%vectors. Must be of equal length (for now).
%returns V, the convolution vector

%By Nasser Abbasi, Oct 12,2002.
%Written for HW 3, problem 5.
%Math 127, UC Berkeley.

N=length(A);

if(N ~= length(B))
    error('A,B must be of equal length');
end

Bi=1:N;
v=zeros(N,1);

for(i=0:N-1)
    v(i+1)=sum(A .* B(Bi));
    Bi=[N-i:N 1:N-(i+1) ];    %right sided circular shift
end

v=v';
```

6.1 Problem 5 source code

Matlab code

```
function V=nma_hw3_problem5(x,y)
%function p=nma_hw3_problem5(x,y)
%
% This function takes 2 DNA sequences, and returns
% the convolution vector for the sequences
%
% calls the function nma_cconv.m to do the convolution

p=0;

if(nargin ~=2)
    error('x,y expected');
end

N=length(x);
if(N ~= length(y))
    error('x,y must be equal length');
end

DNA='ATCG';
V=zeros(N,1); % store final convolution vector here.
V=V';

for(i=1:length(DNA))
    V=V+nma_cconv(normalize(x,(DNA(i))), ...
        normalize(y,(DNA(i))));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% this function replaces each letter in S with 0
% except those that matches w, they are replaced with 1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function T=normalize(S,w)

S=upper(S);
T=zeros(length(S),1);

for(i=1:length(S))
    if(S(i)==w)
        T(i)=1;
    end
end
end
```

```

function [y,zf]=nma_filter_v1(b,a,x,zi)
% function y=nma_filter_v1(b,a,x,zi)
%
% implement IIR filter to give the same output as matlab own
% filter.m function, and then convert the code to C++.
%
% The filter is a "Direct Form II Transposed"
% implementation of the standard difference equation:
%
%   a(1)*y(n) = b(1)*x(n) + b(2)*x(n-1) + ... + b(nb+1)*x(n-nb)
%               - a(2)*y(n-1) - ... - a(na+1)*y(n-na)
%
% Nasser Abbasi, sept 27,2002.
% done for the 5Prime project work.

b=b(:);
a=a(:);

if(size(b,2) ~= 1)
    error('b must be a vector');
end

if(size(a,2) ~= 1)
    error('a must be a vector');
end

nb = length(b);
na = length(a);

if( nb ~= na)
    error('length(a) must equal length(b) in this implementation');
end

if(a(1) ~= 1.0)
    a=a/a(1);
    b=b/a(1);
end

nChannels=size(x,2);
nScan=size(x,1);
y=zeros(nScan,nChannels);
zf=zeros(nb-1,nChannels);

if(nargin==4)
    if(~isempty(zi))
        nCol=size(zi,2);
    end
end

```

```

        if(nCol>nChannels)
            error('Zi number of columns can not be larger than number of X columns');
        end
        for(k=1:nCol)
            zf(:,k)=zi(:,k);
        end
    end
end

for(k=1:nChannels)
    for(n=1:nScan)
        nDelay=1;
        zf(:,k)=Z_(nDelay,x(:,k),y(:,k),zf(:,k),a,b,n-1);
        y(n,k)=b(1)*x(n,k)+zf(nDelay,k);
    end
    k
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% find the delay z at level at time t
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function zf=Z_(nDelay,x,y,zf,a,b,n)

nb=length(b);

if(n==0)
    return;
else
    if(nDelay==nb-1)
        zf(nDelay)=b(nDelay+1)*x(n)-a(nDelay+1)*y(n);
    else
        zf=Z_(nDelay+1,x,y,zf,a,b,n-1);
        zf(nDelay)=b(nDelay+1)*x(n)+zf(nDelay+1)-a(nDelay+1)*y(n);
    end
end
end

```

```

function c=nma_dconv(x,y)
%function c=nma_dconv(x,y)
% finds convolution using direct method.

if(length(x) ~= length(y))
    error('x and y must be same length');
end

n=length(x);

```

```

c=zeros(n,1);
yi=[1 n:-1:2];
yy=y(yi);

for(shift=0:n-1)

    t=0;
    for(i=1:n)
        t=t+x(i)*yy(i);
    end
    c(shift+1)=t;

    yi=[ mod((yi+1),n) ];
    I=find(yi==0);
    yi(I)=n;
    yy=y(yi);

end

```

```

function c = nma_cconv2(a,b,ctr)

if (exist('ctr') ~= 1)
    ctr = 0;
end

if (( size(a,1) >= size(b,1) ) & ( size(a,2) >= size(b,2) ))
    large = a; small = b;
elseif (( size(a,1) <= size(b,1) ) & ( size(a,2) <= size(b,2) ))
    large = b; small = a;
else
    error('one arg must be larger than the other in both dimensions!');
end

ly = size(large,1);
lx = size(large,2);
sy = size(small,1);
sx = size(small,2);

%% These values are the index of the small mtx that falls on the
%% border pixel of the large matrix when computing the first
%% convolution response sample:
sy2 = floor((sy+ctr+1)/2);
sx2 = floor((sx+ctr+1)/2);

% pad:

```



```

clarge = [ ...
    large(ly-sy+sy2+1:ly,lx-sx+sx2+1:lx), large(ly-sy+sy2+1:ly,:), ...
    large(ly-sy+sy2+1:ly,1:sx2-1); ...
    large(:,lx-sx+sx2+1:lx), large, large(:,1:sx2-1); ...
    large(1:sy2-1,lx-sx+sx2+1:lx), ...
    large(1:sy2-1,:), ...
    large(1:sy2-1,1:sx2-1) ];

c = conv2(clarge,small,'valid');

```

```

function v=nma_cconv(A,B)
%function nma_cconv(A,B) implement direct circular convolution for A,B
%vectors. Must be of equal length (for now).
%returns V, the convolution vector

%By Nasser Abbasi, Oct 12,2002.
%Written for HW 3, problem 5.
%Math 127, UC Berkeley.

N=length(A);

if(N ~= length(B))
    error('A,B must be of equal length');
end

Bi=1:N;
v=zeros(N,1);

for(i=0:N-1)
    v(i+1)=sum(A .* B(Bi));
    Bi=[N-i:N 1:N-(i+1) ]; %right sided circular shift
end

v=v';

```

Maple code

```

# by Nasser Abbasi, oct 16, 2002.
# how to read clipped data file using MAPLE.

# INPIT: filename, the clipped full path file name.
# OUTPUT: a Matrix that contains the clipped data. it has
#         as many rows as there are in the clipped file.

epg := proc(filename)
    local line,y,f,x,k;

```

```

try
  f := fopen(filename,READ);
catch:
  printf("Failed to open file %s\n",filename);
  return;
end try;

line := readline(f);
line := readline(f);

line := readline(f);
x := Matrix();
k:=1;
while( line <> 0 ) do
  y := sscanf(line,cat("%a" $ length(line)));
  y := y[3..nops(y)];
  y := convert(y,Vector[row]);
  if(k>1) then
    x := < x , y >;
  else
    x:= y;
  end if;
  k:= k+1;
  printf("read line %d\n",k);
  line := readline(f);
end do;

fclose(f);
return x;

end proc;

```

```

hw3:=module()
  export solve,nma_cconv;
  local normalize_it;

#####
#
#
#####
  solve := proc(x::string,y::string)

    local N,DNA,V,i,t1,c;

    N:= length(x);

```

```

if(N <> length(y)) then
  error("Length of sequences must match");
end if;

DNA:="ATCG";
V:=Vector(N);

for i from 1 to length(DNA) do
  c:=DNA[i];
  t1 := normalize_it(x,c);
  print(t1);
  #V:=V+nma_cconv(normalize_it(x,DNA[i]),normalize(y,DNA[i]));
end do

end proc;

#####
#
#####
#nma_cconv:=proc(x::string,y::string)

#end proc();

#####
#
#
#####
normalize_it := proc(s::string,c)

  local S,i,T;

  S:=StringTools[UpperCase](s);
  T:=Vector(length(S));

  print("s = "); print(s);
  print("S="); print(S);
  print("length S is"); print(length(S));
  print("c="); print(c);

  for i from 1 to length(S) do
    print("check on S[i] , c"); print(S[i]); print(c);

    if(S[i]=c) then
      print("set T[i] to 1"); print("i=",i);
      T[i]:=1;
      print("T now is "); print(T);
    else

```

```
        print("set T[i] to 0");
        T[i]:=0;
    end if;
end do;

print("T=",T);
return T;

end proc;
end module;
```

3.4 HW 4

HW 4

Mathematics 127
Mathematical and Computational Methods in
Molecular Biology

Fall 2002
UC Berkeley, CA

Nasser M. Abbasi

Fall 2002 Compiled on August 2, 2022 at 8:10am [public]

Contents

1 Problems	3
2 Problem 1	6
3 Problem 2	10
4 Problem 3	21
5 Problem 4	23

1 Problems



Problem Set 4 (due Thursday October 31)

MATH 127: Mathematical and Computational Methods in Molecular Biology

Problem 1

Consider the cube – tetrahedron hidden Markov model discussed in class. Suppose that the output probabilities in the cube state are all $\frac{1}{6}$ and in the tetrahedron state they are all $\frac{1}{4}$. Suppose in addition that each transition probability is $\frac{1}{2}$, and that the probability of starting in each state is $\frac{1}{2}$. What is the probability, under the model, of the observed sequence {3, 4, 6, 2}? What is the most likely sequence of states to have produced this observed output?

→ Viterbi

→ Forward-backward algorithm.

Problem 2

GENSCAN is a freely available program for finding genes in DNA sequences. It is based on hidden Markov models.

<http://genes.mit.edu/GENSCAN.html>

a) Submit the sequence U73304 to GENSCAN. The organism you will use is vertebrate. The easiest way to submit the sequence is to select the FASTA format for the sequence on the NCBI website, and then to copy and paste it into the GENSCAN window. You will see that GENSCAN finds the single exon in the DNA exactly. GENSCAN also annotates the polyA signal. What is this signal? Does GENSCAN get it correct?

b) Submit the sequence AF276990 to GENSCAN. This is a much longer (213343 bp) very recently sequenced part of the canis familiaris (dog) genome. Copy and paste is again the best way to put the sequence into GENSCAN. BLAST each of the GENSCAN predicted peptides (these are the proteins that the predicted genes would code for) against the nr database using blastp. Which of the predictions do you believe? For each gene, either cite evidence for it being a true prediction, or explain why you think the prediction is false. You may also want to use tblastn, which translates the DNA sequences in the database and compares them to your protein query.

c) You will see that the 10th prediction is in fact the dog version of the RAD50 human gene. Do you think all the predicted exons are exactly right? If yes explain why, and if not describe the false exons and explain how the prediction could be corrected.

✓ **Problem 3***

Consider a state with a self-transition probability of p in a hidden Markov model. Clearly the probability of outputting d symbols consecutively from the state and then leave the state is $f(d) = p^{d-1}(1-p)$. What is the expected length of output from the state (i.e. calculate $\sum_d df(d)$).

Problem 4

Construct a **constant** space Viterbi algorithm for the four state gene finding HMM (analogous to Hirschberg's algorithm for alignment). What is its running time complexity?

Optional Problems

1. Draw the state space diagram for a gene finding HMM that will not allow stop codons to span introns.
2. Derive an algorithm for **sampling** a state path through an HMM with probability proportional to the probability (weight) of the path. What is the complexity of the algorithm.
3. (Possible final project for the class). Download the latest set of RefSeq genes from genome.ucsc.edu and BLAST the genes to construct a non-redundant parameter training set for a human or mouse HMM gene finder.

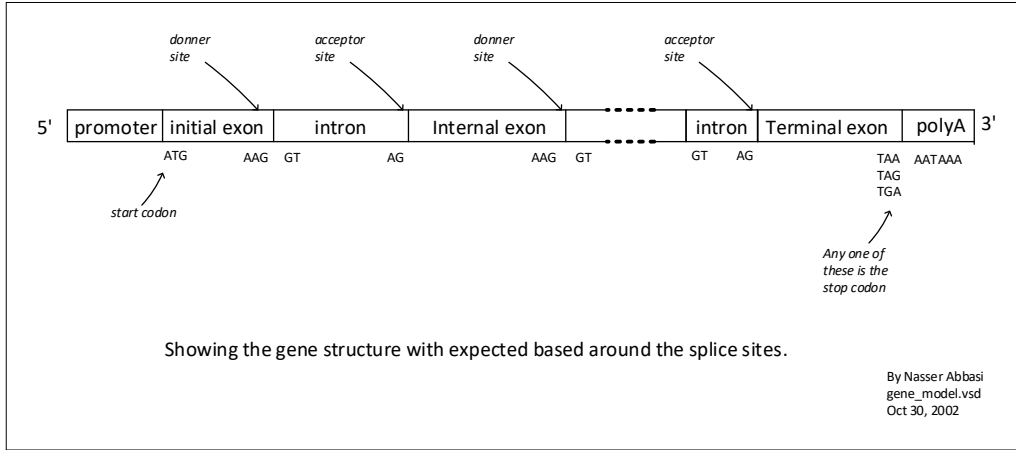


Figure 1: Gene model

2 Problem 1

MATH 127
HW 4
Problem 1

Nasser Abbasi

$\frac{35}{40}$

Problem 1.

let S be state when in cube. let T be state when in tetrahedron.

$\pi(S) = \frac{1}{2}$; $\pi(T) = \frac{1}{2}$

$P_{SS} = \frac{1}{2}$, $P_{TT} = \frac{1}{2}$, $P_{ST} = \frac{1}{2}$, $P_{TS} = \frac{1}{2}$

$b_S(k) = \frac{1}{6}$ which is prob. to emit symbol 'k' when in state S

$b_T(k) = \frac{1}{4}$ " " " " " "

T

x7

To find the prob. of emitting $\{3, 4, 6, 2\}$ use the Forward algorithm $\alpha(i, t)$.

$P(3, 4, 6, 2) = \alpha(S, t=4) + \alpha(T, t=4)$

to calculate α for state S at $t=4$ start at $t=1$ and go forward

$\alpha(S, 1) = \pi_S b_S(3)$

$\alpha(S, 2) = [\alpha(S, 1) P_{SS} + \alpha(T, 1) P_{TS}] b_S(4)$

$\alpha(S, 3) = [\alpha(S, 2) P_{SS} + \alpha(T, 2) P_{TS}] b_S(6)$

$\alpha(S, 4) = [\alpha(S, 3) P_{SS} + \alpha(T, 3) P_{TS}] b_S(2)$

$\alpha(i, t)$ means the prob. of being in state 'i' at time 't'

$b_S('x')$ means the emission prob. of emitting 'x' when in state 'S'

→

To calculate α for state T up to $t=4$

$$\alpha(T,1) = \pi_T b_T('3')$$

$$\alpha(T,2) = [\alpha(T,1) P_{TT} + \alpha(S,1) P_{ST}] b_T('4')$$

$$\alpha(T,3) = [\alpha(T,2) P_{TT} + \alpha(S,2) P_{ST}] b_T('6')$$

$$\alpha(T,4) = [\alpha(T,3) P_{TT} + \alpha(S,3) P_{ST}] b_T('2')$$

now, $\alpha(T,1) = \frac{1}{2} \cdot \frac{1}{4} = \frac{1}{8}$ (0.1250)

$$\alpha(S,1) = \frac{1}{2} \cdot \frac{1}{6} = \frac{1}{12}$$
 (0.0833)

so $\alpha(S,2) = \left[\frac{1}{12} \cdot \frac{1}{2} + \frac{1}{8} \cdot \frac{1}{2} \right] \frac{1}{6} = \left(\frac{1}{24} + \frac{1}{16} \right) \frac{1}{6} = \frac{5}{288}$ (0.0171)

$$\alpha(T,2) = \left[\frac{1}{8} \cdot \frac{1}{2} + \frac{1}{12} \cdot \frac{1}{2} \right] \frac{1}{4} = \left(\frac{1}{16} + \frac{1}{24} \right) \frac{1}{4} = \frac{5}{192}$$
 (0.0260)

$$\alpha(S,3) = \left[\frac{5}{288} \cdot \frac{1}{2} + \frac{5}{192} \cdot \frac{1}{2} \right] \frac{1}{6} = \frac{25}{6912}$$
 (0.0036)

$$\alpha(T,3) = \left[\frac{5}{192} \cdot \frac{1}{2} + \frac{5}{288} \cdot \frac{1}{2} \right] \frac{1}{4} = \frac{25}{4608} \approx 0$$

$$\alpha(S,4) = \left[\frac{25}{6912} \cdot \frac{1}{2} + \frac{25}{4608} \cdot \frac{1}{2} \right] \frac{1}{6} = \frac{125}{165888} = \frac{25}{576 \cdot 144}$$
 Ans

$$\alpha(T,4) = \left[\frac{25}{4608} \cdot \frac{1}{2} + \frac{25}{6912} \cdot \frac{1}{2} \right] \frac{1}{4} = \frac{125}{110592} \approx \frac{25}{576 \cdot 96}$$
 - 3

so final Prob = $\frac{125}{165888} + \frac{125}{110592}$

$$= \frac{625}{331776}$$

$$= \boxed{0.001883801119}$$

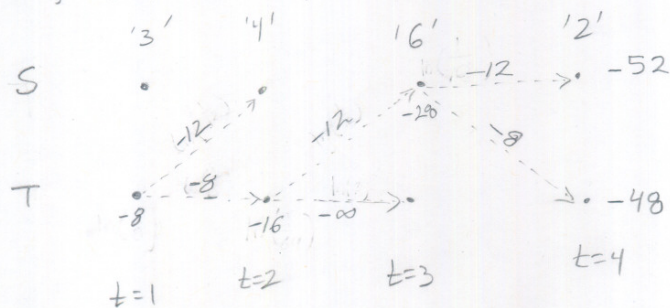
$\approx 7.5 \times 10^{-4}$

Dr, I did not use \log_2 here, but will in the next problem.

To find most likely sequence of states to have produced the sequence $\{3, 4, 6, 2\}$, I use the Viterbi algorithm. (γ)
 in the algorithm, find the weight on the edge from one state to another, and take the maximum weight at time t .
 at time $t+1$, use the max. weight from t in addition to weight to transition.

weight on an edge is $\ln(P_{ij} b_j(x_t))$
 so, at $t=1$, $\gamma_1(S) = \ln(\pi_S b_S('3')) = \ln(\frac{1}{2} \cdot \frac{1}{6}) = \ln(\frac{1}{12}) = -12$
 and $\gamma_1(T) = \ln(\pi_T b_T('3')) = \ln(\frac{1}{2} \cdot \frac{1}{4}) = \ln(\frac{1}{8}) = -8$

So first state is T since that is the larger one.



at $t=2$, weight of edge from $T_{t=1}$ to $T_{t=2}$ is $\ln(P_{TT} b_T('4')) =$

$\ln(\frac{1}{2} \cdot \frac{1}{4}) = \ln(\frac{1}{8}) = -8$, while weight of edge from $T_{t=1}$ to $S_{t=2}$ is

$\ln(P_{TS} b_S('4')) = \ln(\frac{1}{2} \cdot \frac{1}{6}) = \ln(\frac{1}{12}) = -12$

So at $t=2$, take max $\begin{cases} -8 & -8 \\ -8 & -12 \end{cases} = \begin{cases} -8 \\ -16 \end{cases} = -8$

So at $t=2$, state is 'T' since this gives max. \rightarrow

now at $t=3$, look at edge $T_{t=2} \rightarrow T_{t=3}$ and edge

$$T_{t=2} \rightarrow S_{t=3}$$

$$\text{weight on } T_{t=2} \rightarrow T_{t=3} = \ln(P_{TT} b_T('6')) = \ln\left(\frac{1}{2} \cdot \phi\right) = \ln(\phi) \approx -\infty$$

$$\text{weight on } T_{t=2} \rightarrow S_{t=3} = \ln(P_{TS} b_S('6')) = \ln\left(\frac{1}{2} \cdot \frac{1}{6}\right) = \ln\left(\frac{1}{12}\right) = -12$$

$$\text{so max } \begin{cases} -28 & -\infty \\ -28 & -12 \end{cases} = \max \begin{cases} -\infty \\ -40 \end{cases} = -40$$

so state at $t=3$ is 'S' since this is the max.

at $t=4$, look at edges $S_{t=3} \rightarrow S_{t=4}$ and $S_{t=3} \rightarrow T_{t=4}$.

$$\text{for weight on } S_{t=3} \rightarrow S_{t=4} = \ln(P_{SS} b_S('2')) = \ln\left(\frac{1}{2} \cdot \frac{1}{6}\right) = \ln\left(\frac{1}{12}\right) = -12$$

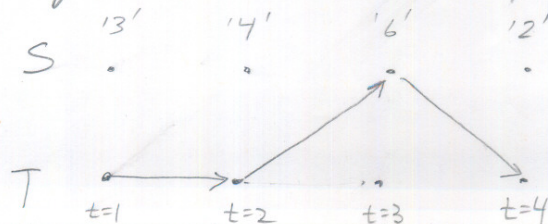
$$\text{for weight on } S_{t=3} \rightarrow T_{t=4} = \ln(P_{ST} b_T('2')) = \ln\left(\frac{1}{2} \cdot \frac{1}{4}\right) = \ln\left(\frac{1}{8}\right) = -8$$

$$\text{so max } \begin{cases} -40 & -12 \\ -40 & -8 \end{cases} = \max \begin{cases} -52 \\ -48 \end{cases} = -48$$

so state at $t=4$ is 'T'

so state sequence is $\{T, T, S, T\}$ OK the

most likely state transitions to have produced $\{3, 4, 6, 2\}$



3 Problem 2

HW 4
 Problem 2
 MATH 127, UC Berkeley
 By Nasser Abbasi

Part A

The sequence **U73304** submitted to genscan. This is the output:

```

GENSCANW output for sequence 23:11:05

GENSCAN 1.0 Date run: 28-Oct-102   Time: 23:11:10
Sequence gi : 5665 bp : 40.65% C+G ; Isochore 1 ( 0 - 43 C+G%)
Parameter matrix: HumanIso.smat

Predicted genes/exons:

Gn.Ex Type S .Begin ...End .Len Fr Ph I/Ac Do/T CodRg P.... Tscr..
-----
1.01 Sngl +   122  1540 1419  1  0  49  52  1310 0.987 118.25
1.02 PlyA +  2132  2137    6
  
```

Looking at the sequence in DNA format, I see that position 122 for exon start to be ATG (shown in red): **tatga**agtcg

The last 3 nucleotides up to position 1540 are TGA (shown in red) ggctctg**ga**

the PolyA sequence according to GENSCAN is from 2132 to 2137. Below I show the sequence from 2121 up to 2140 showing in red where GENSCAN predicted the polyA signal

```

1 2 3 4 5 6 7 8 9 0  1 2 3 4 5 6 7 8 9 0
ATAACTTTAG  AAATAAACCT
  
```

GENSCAN did correctly find the polyA (polyadenylic acid) site. This special consensus signal (AATAAA, which becomes AAUAAA in mRNA) is a special site that is recognized in the pre mRNA during *the splicing process* as to where to cleavage the pre mRNA at to produce the final mRNA.

So this signal is used to know where to cut (cleavage) the pre mRNA at during the splicing process.

Also, during the splicing process, a particular polymerase will recognize this signal and then add about 60-200 Adenylic Acid (A nueclitieds) which as called the A-tail, to the end of this site during a process called polyadenulation. From the reference paper we were given to read (Active Alu Element A-tails: size does matter), it says: "the length of the Alu A-tail is one of the principle factors in determining the retropositional of an Alu element"

peptide_1|150_aa

Part b

The sequence **AF276990** submitted to GENSCAN. The result shows that 10 Genes found.

The I went to <http://www.ncbi.nlm.nih.gov/BLAST/> and clicked on the link to blastp. Then in the new screen, made sure the data base is set to 'nr' (non-repeats). And copied/paste each of the GENSCAN predicted peptides to the blastp window and run BLASTb. Then when the result is obtained, I clicked on the 'FORMAT' button. Then a new screen comes up which shows all the protein sequences that were matched to the query sequences ordered by decreasing blast score. In the table below I show the score for the top sequence hit from each run.

This is the description of blastp from the NCBI web page (the one I used is the standard protein-protein blast):

Protein BLAST allows one to input protein sequences and compare these against other protein sequences.
Standard protein-protein BLAST - Takes protein sequences in FASTA format, GenBank Accession numbers or GI numbers and compares them against the NCBI [protein databases](#).

The database used to search against is 'nr' which is defined as:

nr
All non-redundant GenBank CDS translations+PDB+SwissProt+PIR+PRF

Matrix used for similarity by blastp is BLOSUM62.

This is the final result shown in the table below.

BLASTP result

GENSCAN peptide number	Blastp highest score	E value	LOCUS of the highest matching protein sequence	Authors	Definition of the highest matching protein sequence
Peptide_1 150_aa	34	0.43	LOC224881	NCBI Annotation Project	similar to Retrovirus-related POL polyprotein (Endonuclease) [Mus musculus].
Peptide_2 442_aa	660	0.0	KIAA0202	NCBI Annotation Project.	similar to Septin-like protein KIAA0202 [Homo sapiens].
Peptide_3 405_aa	175	9e-43	Ccni	Jensen	cyclin I [Mus musculus].
peptide_4 968_aa	708	0.0	LOC192762	NCBI Annotation Project.	similar to KINESIN-LIKE PROTEIN KIF3A (MICROTUBULE PLUS END-DIRECTED KINESIN MOTOR 3A) [Mus musculus].
peptide_5 131_aa	244	2e-64	AF244915_1	Yang,S.	interleukin-13 [Canis familiaris].
peptide_6 303_aa	N/A	N/A	N/A	N/A	No significant similarity found
peptide_7 64_aa	N/A	N/A	N/A	N/A	No significant similarity found
peptide_8 271_aa	213	1e-54	CAA99729	Offenberg,H .H.	RAD50 homologue hsRAD50 [Homo sapiens].
peptide_9 408_aa	615	e-175	BAA90817	Kitamura	glyceraldehyde-3-phosphate dehydrogenase [Canis familiaris].
peptide_10 847_aa	941	0.0	RAD50	Dolganov	RAD50 homolog isoform 1 [Homo sapiens].

Next, I used tblastn against each peptide to search for all 6 reading frames. From NCBI web page, this is the definition of tblastn:

Protein query - Translated db [tblastn] - Takes a protein query sequence and compares it against an NCBI [nucleotide database](#) which has been translated in all six reading frames.

tblastn result

GENSCAN peptide number	tBlastn highest score	E value	LOCUS of the matching Nucleotide sequence	Authors	Definition of the highest matching Nucleotide sequence
Peptide_1 150_aa	77	3e-13	AC004775	Kimmerly	Homo sapiens chromosome 5, P1 clone 1308e5 (LBNL H13), complete sequence.
Peptide_2 442_aa	665	0.0	AK057797	Nishi,T.,	Homo sapiens cDNA FLJ25068 fis, clone CBL05137, highly similar to Mus musculus Sep2 mRNA.
Peptide_3 405_aa	191	2e-46	AC004775	Kimmerly	Homo sapiens chromosome 5, P1 clone 1308e5 (LBNL H13), complete sequence.
peptide_4 968_aa	723	0.0	BC032599	Strausberg, R.	Homo sapiens. Similar to kinesin family member 3A, clone IMAGE:5333541, mRNA.
peptide_5 131_aa	259	3e-68	AF244915	Yang,S.,	Canis familiaris interleukin-13 mRNA, complete cds.
peptide_6 303_aa	49	6e-04	AY079157S1	Zangerl	Canis familiaris glucocorticoid receptor DNA binding factor 1 (GRLF1) gene
peptide_7 64_aa	36	0.78	AL845433	Whitehead, S.	Human DNA sequence from clone RP11-674N8 on chromosome X, complete sequence.
peptide_8 271_aa	231	9e-59	HSRAD50	Offenberg, H.H.	H.sapiens mRNA for RAD50.
peptide_9 408_aa	605	e-173	RABGLY3PHO	Applequist	Oryctolagus cuniculus glyceraldehyde-3-phosphate dehydrogenase mRNA, complete cds.
peptide_10 847_aa	1196	0.0	RAD50	Dolganov	Homo sapiens RAD50 homolog (S.cerevisiae) (RAD50), transcript variant 2, mRNA.

To better compare blastp result with tblastn, I show the result in this table.

Blastp and tblastn score comparison

GENSCAN peptide number	blastp highest score	blastP E value	tblastn highest score	tblastn E value
Peptide_1 150_aa	34	0.43	77	3e-13
Peptide_2 442_aa	660	0.0	665	0.0
Peptide_3 405_aa	175	9e-43	191	2e-46
peptide_4 968_aa	708	0.0	723	0.0
peptide_5 131_aa	244	2e-64	259	3e-68
peptide_6 303_aa	N/A	N/A	49	6e-4
peptide_7 64_aa	N/A	N/A	36	0.78
peptide_8 271_aa	213	1e-54	231	9e-59
peptide_9 408_aa	615	e-175	605	e-173
peptide_10 847_aa	941	0.0	1196	0.0

Conclusion. To answer the question on which prediction I should believe, I use the blast score and the expect value E as the main criteria. The expect value is defined in NCBI web page. Here is the text

Q: What is the Expect (E) value?

The Expect value (E) is a parameter that describes the number of hits one can "expect" to see just by chance when searching a database of a particular size. It decreases exponentially with the Score (S) that is assigned to a match between two sequences. Essentially, the E value describes the random background noise that exists for matches between sequences. For example, an E value of 1 assigned to a hit can be interpreted as meaning that in a database of the current size one might expect to see 1 match with a similar score simply by chance. **This means that the lower the E-value, or the closer it is to "0" the more "significant" the match is.** However, keep in mind that searches with short sequences, can be virtually identical and have relatively high EValue. This is because the calculation of the E-value also takes into account the length of the Query sequence. This is because shorter sequences have a high probability of occurring in the database purely by chance.

The higher the scores and the lower the E values, the more belivable the prediction will be. Since this means GENSCAN did produce a sequence which actually exist in the database and documented to high similarity score.

This table below is my final result of the prediction by GENSCAN.

GENSCAN peptide number	True or false prediction?	WHY?
Peptide_1 150_aa	FALSE	Low score and E values from both blastp and tblastn
Peptide_2 442_aa	TRUE	High score and E value from both blastp and tblastn. Documented protein sequence.
Peptide_3 405_aa	TRUE	As above
peptide_4 968_aa	TRUE	As above
peptide_5 131_aa	TRUE	As above
peptide_6 303_aa	FALSE	Blastp failed to find a significant match, tblastn low score.
peptide_7 64_aa	FALSE	As above
peptide_8 271_aa	TRUE	High score and E value from both blastp and tblastn. Documented protein sequence.
peptide_9 408_aa	TRUE	As above.
peptide_10 847_aa	TRUE	As above.

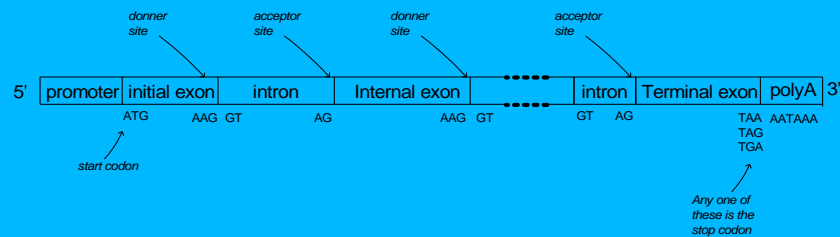
Part C

10.17	PlyA	-	165036	165031	6						1.05	
10.16	Term	-	176151	176008	144	0	0	97	52	134	0.997	7.63
10.15	Intr	-	178556	178443	114	2	0	50	100	132	0.999	10.32
10.14	Intr	-	178723	178631	93	1	0	68	86	85	0.978	5.54
10.13	Intr	-	179279	179169	111	2	0	75	69	65	0.885	2.96
10.12	Intr	-	181859	181666	194	0	2	50	115	210	0.999	18.19
10.11	Intr	-	182421	182295	127	0	1	36	90	106	0.999	4.93
10.10	Intr	-	182850	182661	190	2	1	38	111	149	0.935	10.87
10.09	Intr	-	189215	188978	238	0	1	34	94	120	0.491	2.95
10.08	Intr	-	189936	189761	176	2	2	63	107	174	0.999	15.46
10.07	Intr	-	193421	193264	158	2	2	85	100	50	0.998	3.79
10.06	Intr	-	195824	195618	207	2	0	72	50	201	0.999	13.05
10.05	Intr	-	197297	197104	194	0	2	31	54	257	0.971	14.89
10.04	Intr	-	199077	198912	166	0	1	94	53	145	0.994	10.21
10.03	Intr	-	199440	199312	129	0	0	35	76	134	0.900	6.87
10.02	Intr	-	201315	201221	95	1	2	91	18	110	0.880	3.06
10.01	Init	-	210056	209849	208	2	1	81	88	137	0.977	12.03

For 10th prediction. To help me solve this I needed to find what happens around the splice sites. This is the exon/intron boundaries. Looking at a gene from 5' to 3', the following is typically found

1. Initial exon starts with ATG
2. Exon ends with AAG. (Donor site)
3. Intron starts with GT.
4. Intron ends with AG (Acceptor site).
5. Terminal exon (last exon in a gene) ends with TAA or TAG or TGA.
6. PolyA is AATAAA.

I drew a diagram to help illustrate the above:



Showing the gene structure with expected based around the splice sites.

By Nasser Abbasi
gene_model.vsd
Oct 30, 2002

So, for each exon/intron boundaries as predicted by GENSCAN, I verified if the above is correct or not. I put the result in this table. In this table, I show for each exon the 2 bases at the end of the codon before, the codon at the start and end of the exon, and the 2 bases at the start of the next intron. If those value meet the diagram above, then I call the prediction correct. Note that GENSCAN found this 10th gene on the reverse strand, so in this table below I show the on both strands, then in the final table I show it from 5' to 3' sense to make it easier to compare with the above diagram.

exon	Exon position	2 bases at end of previous intron	Codon at start of this exon	Codon at end of this exon	2 bases at start of next codon
Init	210056: 209849	N/A	5' CAT 3' 3' GTA 5'	5' CTC 3' 3' GAG 5'	5' AC 3' 3' TG 5'
Internal	201315: 201221	5' CT 3' 3' GA 5'	5' AAT 3' 3' TTA 5'	5' CTG 3' 3' GAC 5'	5' CT 3' 3' GA 5'
Internal	199440: 199312	5' CT 3' 3' GA 5'	5' ATT 3' 3' TAA 5'	5' CTT 3' 3' GAA 5'	5' AC 3' 3' TG 5'
Internal	199077: 198912	5' CT 3' 3' GA 5'	5' AAC 3' 3' TTC 5'	5' CCT 3' 3' GGA 5'	5' AC 3' 3' TG 5'
internal	197297: 197104	5' CT 3' 5' GA 5'	5' GAC 3' 3' CTG 5'	5' CAT 3' 3' GTA 5'	5' AC 3' 3' TG 5'
Internal	195824: 195618	5' CT 3' 3' GA 5'	5' ATT 3' 3' GAA 5'	5' AGC 3' 3' TCG 5'	5' AC 3' 3' TG 5'
Internal	193421: 193264	5' CT 3' 3' GA 5'	5' AGC 3' 3' TCG 3'	5' TTC 3' 3' AAG 5'	5' AC 3' 3' TG 5'
Internal	189936: 189761	5' CT 3' 3' GA 5'	5' TTG 3' 3' AAC 5'	5' CTC 3' 3' GAG 5'	5' AC 3' 3' TG 5'
Internal	189215: 188978	5' GA 3' 3' CT 5'	5' TAA 3' 3' GTT 5'	5' CTC 3' 3' GAG 3'	5' AC 3' 3' TG 5'
Internal	182850: 182661	5' CT 3' 3' GA 5'	5' TGC 3' 3' ACG 5'	5' CTG 3' 3' GAG 5'	5' AC 3' 3' TG 5'
Internal	182421: 182295	5' TC 3' 3' AG 5'	5' CAT 3' 3' GTA 3'	5' ACC 3' 3' TGG 5'	5' TT 3' 3' AA 5'
Internal	181859: 181666	5' CT 3' 3' GA 5'	5' AAA 3' 3' TTT 5'	5' CTT 3' 3' GAA 5'	5' AC 3' 3' TG 5'
Internal	179279: 179169	5' CT 3' 3' GA 5'	5' ATC 3' 3' TAG 5'	5' TTT 3' 3' AAA 5'	5' AC 3' 3' TG 5'
Internal	178723: 178631	5' CT 3' 3' GA 5'	5' CAT 3' 3' GTA 5'	5' CTT 3' 3' GAA 5'	5' AC 3' 3' TG 5'
Internal	178556: 178443	5' CT 3' 3' GA 5'	5' TTG 3' 3' AAC 5'	5' CTT 3' 3' GAA 5'	5' AC 3' 3' TG 5'
Terminal	176151: 176008	5' CT 3' 3' GA 5'	5' AAT 3' 3' TTA 5'	5' TCA 3' 3' AGT 5'	5' GC 3' 3' CG 5'
PolyA	165036: 165031	5' AG 3' 3' TC 5'	5' TTTATT 3' 3' AAATAA 5'	N/A	N/A

Now I show the above table, but list everything from 5' to 3' sense. I.e. when looking at 3' ATG 5', I list it now as 5' GTA 3'

exon	2 bases at end of previous intron	Codon at start of this exon	Codon at end of this exon	2 bases at start of next codon	GENSCAN probability	Correct prediction?
Init	N/A	ATG	GAG	GT	12.03	YES
Internal	AG	ATT	CAG	AG (error)	3.06	NO
Internal	AG	AAT	AAG	GT	6.87	YES
Internal	AG	CTT	AGG	GT	10.21	YES
Internal	AG	GTC	ATG	GT	14.89	YES
Internal	AG	AAG	GCT	GT	13.05	YES
Internal	AG	GCT	GAA	GT	3.79	YES
Internal	AG	CAA	GAG	GT	15.46	YES
Internal	TC (error)	TTG	GAG	GT	2.95	NO
Internal	AG	GCA	GAG	GT	10.87	YES
Internal	GA(error)	ATG	GGT	AA (error)	4.93	NO
Internal	AG	TTT	AAG	GT	18.19	YES
Internal	AG	GAT	AAA	GT	2.96	YES
Internal	AG	ATG	AAG	GT	5.54	YES
Internal	AG	CAA	AAG	GT	10.32	YES
Terminal	AG	ATT	TGA	GC	7.63	YES
PolyA	N/A	AATAAA	N/A	N/A	1.05	YES

Some observation: From what I understood, the codon at the end of each internal exon should be AAG. However from the table above, this does not show to be the case, so since I am not sure if this rule is correct now, I will not use it.

I will only use the rules that says that the start of each intron after an exon should be GT and the end of each intron just before an exon start should be AG.

Based on these two rules, I see that GENSCAN did not correctly predict 3 exons. These are the ones where I wrote an 'error' next to them in the above table. Also notice that the ones that GENSCAN did not predict correctly has LOW probability of less than 5.

4 Problem 3

problem 3
HW 4
Math 127

Nasser Abbasi

Expected length means the average length when the average is taken for $d \rightarrow \infty$.

$$\text{so } E = \lim_{d \rightarrow \infty} \underbrace{\frac{1}{d} \sum_{i=1}^d i f(i)}_{\text{average}}$$

Note: since probability to output d symbols is $f(d)$, then number of symbols outputted is $d f(d)$.

$$E = \lim_{d \rightarrow \infty} \frac{1}{d} \sum_{i=1}^d i P^{i-1} (1-P)$$

$$= \lim_{d \rightarrow \infty} \left(\frac{1}{d} d(1-P) \sum_{i=1}^d i P^{i-1} \right)$$

where I have taken $(1-P)$ out of the sum and replaced by $d(1-P)$.

$$E = (1-P) \lim_{d \rightarrow \infty} \sum_{i=1}^d i P^{i-1}$$

now I need to find closed form sum for $S = \sum_{i=1}^d i P^{i-1}$

$$S = 1 + 2P + 3P^2 + 4P^3 + \dots + (d-1)P^{d-2} + dP^{d-1}$$

$$PS = P + 2P^2 + 3P^3 + 4P^4 + \dots + (d-1)P^{d-1} + dP^d$$

subtract PS from S gives

$$(S - PS) = (1-P)S = 1 + P + P^2 + P^3 + \dots + P^{d-1} + dP^d \rightarrow$$

So

$$(1-P)S = 1 + P + P^2 + \dots + dP^d$$

Now, since $0 \leq P \leq 1$, probability, then the sum on the right is convergent and equals $\frac{1}{1-P}$ in the limit $d \rightarrow \infty$.

$$\text{So } (1-P)S = \frac{1}{1-P}$$

$$S = \frac{1}{(1-P)^2}$$

$$\text{So } E = (1-P) \frac{1}{(1-P)^2} = \boxed{\frac{1}{1-P}} \quad \checkmark$$

So expected length of output from state is $\frac{1}{1-P}$.

$$\text{For example, for } P = .1 \Rightarrow E = 1.1111$$

$$P = .2 \Rightarrow E = 1.25$$

$$P = .3 \Rightarrow E = 1.42857$$

$$P = .4 \Rightarrow E = 1.66$$

$$P = .5 \Rightarrow E = 2$$

$$P = .6 \Rightarrow E = 2.5$$

$$P = .7 \Rightarrow E = 3.33$$

$$P = .8 \Rightarrow E = 5$$

$$P = .9 \Rightarrow E = 10$$

$$P = .99 \Rightarrow E = 100$$

This makes sense, since if says the higher the prob to be in the state, the more symbols one will expect to output.

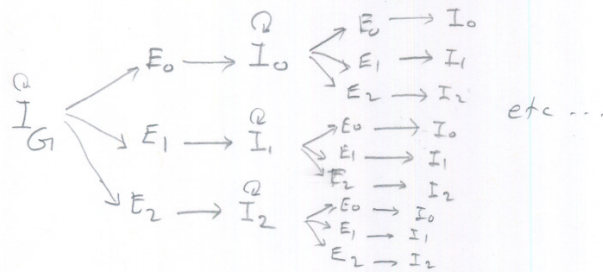
5 Problem 4

HW # 4
 Problem # 4
 MATH 127 x8

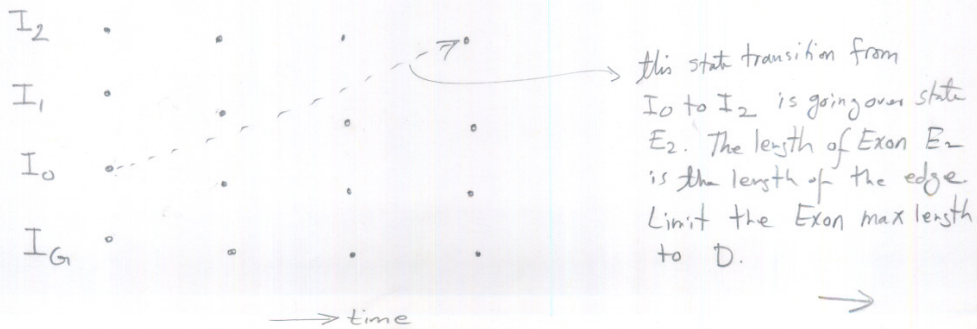
Nasser Abbas:

It is possible to construct a 4 state HMM for finding genes since the path from a specific Intron state to a specific Intron state must go through one unique Exon state. so we can 'hide' the go through one unique Exon state. so we can 'hide' the exon state by implicitly represent them on the edge from one I state to another.

one way to represent this:



so we see that to go from I_0 to I_2 for example, we must go over state E_2 only. so no need to have a separate 'E' state. In terms of HMM, this can be drawn as



at each intron state emit a sequence of bases, depending on the length of the edge.

since we set max length of exon to be D , we need to only look D time steps (or bases) back when trying to find the max vertex weight in order to add to it the transition weight for the Viterbi algorithm.

recall that for the Viterbi algorithm.

example for i, k states

$$\delta(t, i) = \max \begin{cases} \delta(t-1, i) + P_{ii} b_i(\text{symbol at time } t) \\ \delta(t-1, j) + P_{ji} b_i(\text{symbol at time } t) \\ \vdots \\ \delta(t-1, k) + P_{ki} b_i(\text{symbol at time } t) \end{cases}$$

time state

i.e. find the max at time t , by looking at the max at time $(t-1)$ plus the transition weight and emission weight to current state.

since now we want to account for Exon of max length D , need to look D long ago. so Viterbi becomes

we look upto D time steps back

example for 2 states i, j only

$$\delta(t, i) = \max \begin{cases} \delta(t-1, i) + P_{ii} b_i(\text{symbol at time } t) \\ \delta(t-1, j) + P_{ji} b_i(\text{symbol at time } t) \\ \delta(t-2, i) + P_{ii} b_i(\text{symbol at time } t) \\ \delta(t-2, j) + P_{ji} b_i(\text{symbol at time } t) \\ \delta(t-3, i) + P_{ii} b_i(\text{symbol at time } t) \\ \delta(t-3, j) + P_{ji} b_i(\text{symbol at time } t) \\ \vdots \\ \delta(t-D, i) + P_{ii} b_i(\text{symbol at time } t) \\ \delta(t-D, j) + P_{ji} b_i(\text{symbol at time } t) \end{cases}$$

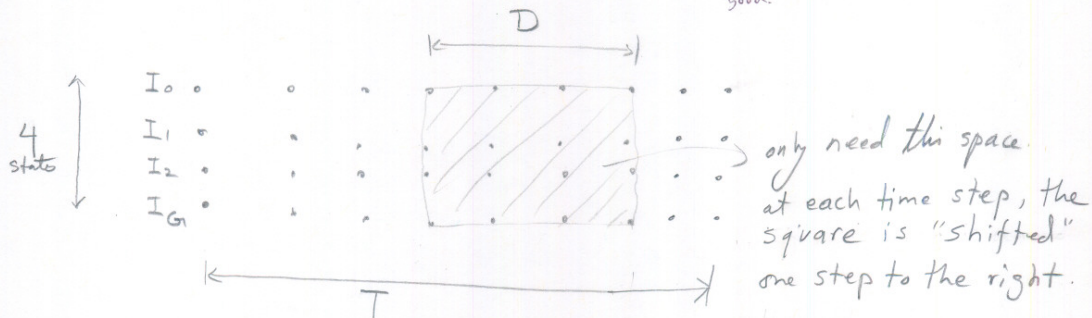
since here we have 4 states only, then for T long symbols this method will take time complexity $O(D 4^2 T) = O(16DT)$

16 DT because at each time step we have to look at $(4 \times D)$ states per each state. but we have 4 states, so we get $16 \times D$. and since there are T positions or time steps, we get $16DT \Rightarrow O(DT)$

Now, I will show how to construct a constant space.

Viterbi algorithm for this HMM. using the idea from Hirschberg space improvement where one column in the matrix is only needed and reused over and over, here I will have a space to store only $\underline{D \times 4}$ states.

Looking at this diagram



So for a given D (say 500 as max Exon length allowed) space complexity is 2000 or constant.

i.e I only need to keep storage to remember $\delta(t, i)$ for D steps back and for 4 states. so constant space.

The running time complexity remains as shown before $O(DT)$ where T is the total length of the sequence. use divide and conquer strategy PES

3.5 HW 5

HW 5

Mathematics 127
Mathematical and Computational Methods in
Molecular Biology

Fall 2002
UC Berkeley, CA

Nasser M. Abbasi

Fall 2002 Compiled on August 2, 2022 at 8:11am [public]

Contents

1 Problems	3
2 Problem 2	5
3 Problem 4	9

1 Problems

Problem Set 5 (due Tuesday November 26)

MATH 127: Mathematical and Computational Methods in Molecular Biology

Problem 1

a) Consider a tree with n leaves and internal nodes all of which have degree k . How many edges are there in the tree?

b) How many **unlabelled** trees are there on 2,3,4 vertices? Conjecture a formula for the number of unlabelled trees on n vertices.

Problem 2

Go to <http://www.ch.embnet.org/software/ClustalW.html>. Enter the sequences:

```
GATTACA
AGAGACGATGA
GAGAAGGGAAGGAATTACA
GATATATGCA
GAGAGTG
```

Align the sequences and look at the output (clustalw aln format). How sensitive is the multiple alignment to the extension and separation gap penalties?

Problem 3*

Consider the dynamic programming method for aligning k sequences of length n (generalization of Needleman-Wunsch). The divide-and-conquer Needleman-Wunsch algorithm can be generalized to the k sequences problem. What is its running time and space requirement?

Problem 4

a) Find the accession AC129884 at NCBI. What organism is this sequence from? How many pieces is it in?

b) Go to <http://pipeline.lbl.gov/cgi-bin/GenomeVista>. Put in the GENBANK accession above, and find it on the **Mouse Genome**

c) What genes does the sequence contain? (it may help to click on TextBrowser and then look at the Vista picture)

d) Go to

http://www.nisc.nih.gov/open_page.html?/projects/zooseq/pubmap/PubZooSeq_Target

Which target region contains the gene of AC129884? Click on that target. How many organisms have sequence available in GENBANK?

Optional Problem Prove the formula conjectured in 1b for the number of unlabelled trees on n vertices.

2 Problem 2

Problem 2
HW 5.
Math 127, UC Berkeley, Fall 2002.
Nasser Abbasi

Went to

<http://www.ch.embnet.org/software/ClustalW.html>

and used the UI to enter the 5 sequences.

for help on ClustalW, I found this site is useful:

http://www.swbic.org/origin/proc_man/Clustal/search/help.html

Before I show the results, first I needed to better understand the parameters and what they actually mean. From the above help, this the summary:

Open gap penalty: Increasing the gap opening penalty *will make gaps less frequent*.

Extend gap penalty: Increasing the gap extension penalty *will make gaps shorter*.

Separation gap penalty: This is the same as gap distance. From the net, I found this definition:

“GAP SEPARATION DISTANCE tries to decrease the chances of gaps being too close to each other. Gaps that are less than this distance apart are penalised more than other gaps. *This does not prevent close gaps; it makes them less frequent*, promoting a block-like appearance of the alignment”

So, the above tells me that if I increase the gap separation penalty, I should see gaps more far apart.

ClustalW

Valid format for input is: FASTA(Pearson)

max number of sequences = 30

max total length of sequences = 10000

[Help page](#)

Scoring matrix :	<input type="text" value="Blosum"/>		
Opening gap penalty :	<input type="text" value="10"/>	Extending gap penalty :	<input type="text" value="0.05"/>
End gap penalty :	<input type="text" value="10"/>	Separation gap penalty :	<input type="text" value="0.05"/>
Output format :	<input type="text" value="Clustal"/>	Output order :	<input type="text" value="Input"/>

Input sequences: (see above for valid formats)	<pre>>seq1 GATTACA >seq2 AGAGACGATGA >seq3 GAGAAGGGAAGGAATTACA</pre>
---	---

I started by fixing the value of the extend gap penalty and changing the separation gap penalty. Then fixed the separation gap penalty and changed the extend gap penalty. Then changed both at the same time. These are the result of these trials:

Tries done to see the effect of changing the gap separation penalty:

Extend Gap	Separation Gap	Multiple alignment	observation
0.05	0.0	seq1 -----GATTACA seq2 AGAGACG-----ATGA-- seq3 -GAGAAGGGAAGGAATTACA seq4 -GATAT-----ATGCA- seq5 -GAGAG-----TG---	Original default setting
0.05	0.01	same as above	No change seen
0.05	0.07	same as above	No change seen
0.05	0.08	same as above	No change seen
0.05	0.09	same as above	No change seen
0.05	0.1	same as above	No changes seen
0.05	1.0	same as above	No changes seen
0.05	2.0	same as above	No changes seen
0.05	3.0	same as above	No changes seen
0.05	5.0	same as above	No changes seen
0.05	20	same as above	No changes seen
0.05	200	same as above	No changes seen

Conclusion: In the above sequences, the gap separation penalty have no effect. This shows ClusalW is not sensitive to this penalty, at least in this example.

Tries done to see the effect of changing the gap extension penalty:

Extend Gap	Separation Gap	Multiple alignment	observation
0.05	0.06	seq1 -----GATTACA seq2 AGAGACG-----ATGA-- seq3 -GAGAAGGAAGGAATTACA seq4 -GATAT-----ATGCA- seq5 -GAGAG-----TG--- *	Original default setting
0.1	0.06	same as above	No changes seen
0.2	0.06	same as above	No changes seen
0.3	0.06	same as above	No changes seen
0.4	0.06	same as above	No changes seen
0.5	0.06	same as above	No changes seen
0.51	0.06	same as above	No changes seen
0.52	0.06	same as above	No changes seen
0.53	0.06	Same as above	No changes seen
0.530000001	0.06	Same as above	No changes seen
0.530000002	0.06	seq1 -----GATTACA seq2 -----AGAGACGATGA-- seq3 GAGAAGGAAGGAATTACA seq4 -----GATAT-ATGCA- seq5 -----GAGAG--TG--- *	A tiny change in the extend gap penalty now shows large effect for first time. First gape on seq3 is gone, and gaps inside seq 2,4,5 are gone. GAPS HAVE BECOME SHORTER AS EXPECTED.
0.54	0.6	Same as above	No changes seen
1.0	0.6	Same as above	No changes seen
2.0	0.6	Same as above	No changes seen
8.99999952	0.6	Same as above	No changes seen
8.99999953	0.6	seq1 -----GATTACA seq2 -----AGAGACGATGA-- seq3 GAGAAGGAAGGAATTACA seq4 -----GATATATGCA-- seq5 -----GAGAGTG-----	A tiny change now shows another change. Now all internal gaps are gone. GAPS HAVE BECOME SHORTER AS EXPECTED.
100	0.6	Same as above	No changes seen

Conclusion: ClustalW is more sensitive to gap extension penalty. The larger this penalty, the less gaps are seen inside the sequences as expected. It is very sensitive in that a change from 0.530000001 to 0.530000002 (a change on only 0.000000001) causes such a large effect in the alignment as shown above. As the penalty is increased all the way to 8.99999952 no more change is seen. But a change from 8.99999952 to 8.99999953 caused the final gap inside the last 2 sequences to close.

3 Problem 4

Problem 4
HW 5
Math 127, UC Berkeley, Fall 2002.
Nasser Abbasi

Part a). from the NCBI web page, AC129884 sequence is from organism *Ornithorhynchus anatinus*. *Genbank common name*: platypus (to be honest, I do not know what this organism is supposed to be, I just got the name from the NCBI default display for this locus). The length of the sequence is 121,483 bp

For the number of pieces this sequences is made of, I looked down the description, and in the comment section it says that this sequence is a working draft, **and it is made of 7 contigs**.

- * 1 4203: contig of 4,203 bp in length
- * 4304 15192: contig of 10,889 bp in length
- * 15293 24847: contig of 9,555 bp in length
- * 24948 34501: contig of 9,554 bp in length
- * 34602 51540: contig of 16,939 bp in length
- * 51641 76311: contig of 24,671 bp in length
- * 76412 121483: contig of 45,072 bp in length.

Part b)

Went to <http://pipeline.lbl.gov/cgi-bin/GenomeVista> .

First I needed to understand what GenomeVista does. This is below the description from the above web page:

GenomeVista allows users to perform comparative analysis of their own data sets using the Berkeley Genome Pipeline (Godzilla). The **draft** or **finished** sequences are aligned with the base genome of your choice, and conserved region analysis is performed. The resulting alignments can be browsed via the Vista Genome Browser or the Godzilla Text Browser.

So, GenomeVista locates a sequence on either the human or the mouse genome. The question asks to find this sequence on the mouse genome. So, I set the 'Base Genome' choice to 'Mouse feb 2002' and entered the above accession number. This is the result:

Chromosome 6
 Total Groups: 1 (sorted by alignment size)
 6:17409146-17533827 (2 alignments, 121.8Kbp) [Text Browser](#) [Vista Genome Browser](#)

Godzilla Text Browser

Select Genome Pair:
 Position in the Base Genome:
 (Format: chr11:113030619-113173035)

The above result is a little confusing to me. At the top it says that 2 alignments found on Chromosome 6 of the mouse genome. But in the lower part under the text browser, it lists a position in Chromosome 11. I assume this is just to show how the format looks like. I.e. it is an example. (but it should actually say so).

So, I clicked on the 'Text Brower' to see where on Chr 6 these sequences found. And this is the result.

Hits on chr6:17409146-17533827
[RefSeq in this region](#) [View in Vista Browser](#) [View at UCSC](#) [Get conserved regions](#)

user query Contig info	Location on mouse	matches number of matches
AC129884-7 (user scontig) Contig Sequence length = 45072bp aligned: between 3401-43318 (39918bp)	chr6:17409146-17454346 Sequence (softmasked) RefSeq Conserved Regions length=45201bp	12337
AC129884-6 (user scontig) Contig Sequence length = 24671bp aligned: between 51566-73595 (22030bp) on the reverse complement	chr6:17517206-17533827 Sequence (softmasked) RefSeq Conserved Regions length=16622bp	6218

From the above, these are the locations of the sequence on mouse genome:

chr6:17409146-17454346 length=45,201bp
chr6:17517206-17533827 length=16,622bp

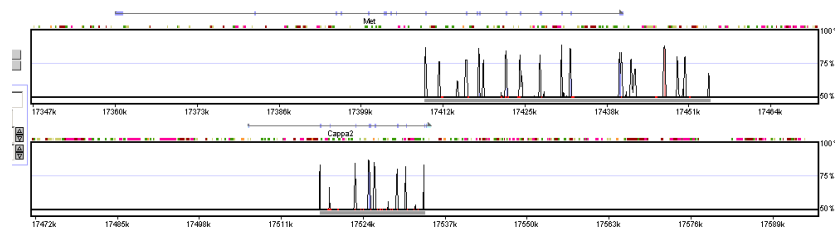
Looking at the NCBI output, I see that the above alignment seem to have been made on contigs 5 and 7 of the sequence, because those are the lengths closes to result from GenomeVista. (The original sequence had a length of 121,483, but the locations found have smaller lengths to them, so that is why I assume that the alignments was made on the whole sequence but only pieces 5 and 7 were found on the mouse genome).

Part c)

To find what genes in this sequence, I searched in both the mouse genome and the human genome.

For the mouse genome:

Clicked on the TextBrowser, then for each alignment (there are 2 of them as shown above), I click on the 'Vista' link to the right of the screen. To see both alignments, I zoomed out. This is the result



In the above, the first alignment (the first window above) is contig **chr6:17409146-17454346** length=45,201bp While the second alignment (in the lower window) is contig **chr6:17517206-17533827** length=16,622bp

The question asks to find the genes contained in the sequence.

Clicked on the 'VISTA' link, this shows this result (the vista plot shows the gene name on top of the diagram on the arrow line).

Contig	Gene
chr6:17409146-17454346	MEL
chr6:17517206-17533827	CAPPA2

For the human genome, similarly, the following genes found (there were 3 contigs found when search human genome june 2002).

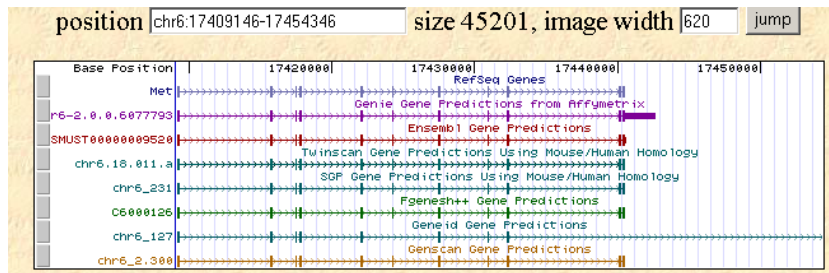
Contig	Gene
chr7:114880956-114924418	MET
chr7:114927608-114928484	NO GENE FOUND
chr7:115006025-115034040	CAPZA2

So, the answer to part C is: Mei, Met, Cappa2, and Capza2. 4 genes, 2 in mouse genome and 2 in human genome.

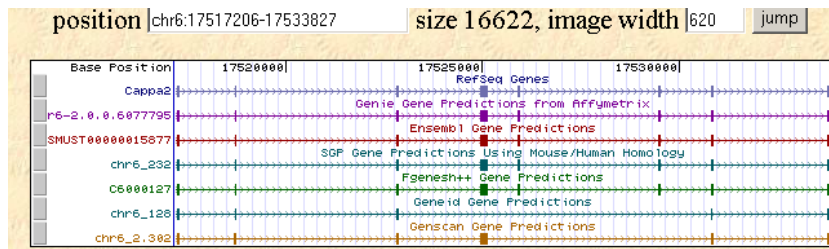
(note: It was hard to read the names of the genes on top of the diagrams in the vista window, but I zoomed in to verify that the names are correct as above).

While working on this part, I show the result of gene prediction using the UCSD software. This below is the output when I hide everything except the gene predictions from a number of applications.

I clicked on the UCSD browser and in the 'position' window, I typed in **chr6:17409146-17454346** (which is the first contig) which contains gene Mei:



This is for the second contig which contains gene Cappa2:



part d)
went to

http://www.nisc.nih.gov/open_page.html?/projects/zooseq/pubmap/PubZooSeq_Targets

For MET gene, it is contained in target 1

For MEL gene, it is contained in target 1

For CAPPA2, it is contained in target 1

For CAPZA2, it is contained in target 1.

So the answer to part d is target 1.

Target 1 is about 1.5 Mbases. Organisms shown are : Chimp, Orangutan, Baboon, Macaque, Vervet, Lemur, Pig, Horse, Cow, Cat, Dog, Aibat, Cibat, Rabbit, Hedgehog, Mouse, Rat, Opossum, Dunnart, Platypus, Chicken, Zebrafish, Fugu, Tetraodon.

24 organisms.