McGill

December 2016
Final Examination

COMP 462 - COMPUTATIONAL BIOLOGY METHODS
COMP 561 - COMPUTATIONAL BIOLOGY METHODS AND RESEARCH

December 14th 2016, 18:00-21:00

Examiner: Mathieu Blanchette Assoc Examiner: Jérôme Waldispühl

Signature: _____ Signature: _____

Student Name:		McGill ID:									

INSTRUCTIONS:

- This is an OPEN BOOK examination
- Answer **DIRECTLY ON THE EXAM**
- This examination paper MUST BE RETURNED
- COMP 462 students: This examination is worth 40% of your final grade.
- COMP 561 students: This examination is worth 30% of your final grade.
- The total of all questions is 100 points. The value of each question is found in parentheses next to it.
- Two blank pages are added at the end in case you need extra space.
- Calculators are allowed
- No laptop computer, cell phones, etc. is allowed.
- This exam comprises 6 questions on 11 pages, including this cover page and two blank pages at the end.

Question 1. (28 points)

Answer each question with a short but precise explanation for each. *Credits will be given only if the justification is clear and correct.*

a) (4 points) True or False? *Justify.* If gene *G* has 10 mRNA copies present in a cell at time *t*, and if the expression of *G* does not change afterwards, then the protein encoded by *G* will never be present in the cell in more than 10 copies.

False. The number of copies of the protein encoded by gene G depends on a variety of factors, including the translational efficiency, the stability of the encoded protein, etc.

b) (4 points) Consider two samples S_1 and S_2 of cells where the gene expression profiles (number of mRNA copies of each gene) are absolutely identical, except for a single gene, called G_1, that is expressed 1,000,000 times more in sample 2 than in sample 1. You perform RNA-seq experiments on S_1 and S_2, and obtain expression levels for each gene, measured in FPKM (Fragment per Kilobase per Million reads). How will the FPKM expression of gene G_2 (a gene other than G_1) compare in the two samples, and why?

The FPKM value for G2 will be lower in sample 2 than in sample 1, because a larger fraction of reads will be mapped to G1. The key thing to remember is that RNA-seq measures the relative gene expression, not the absolute expression level.

c) (4 points) What is a transcription factor?

A transcription factor is a protein that regulates the expression of one or more genes by binding the DNA near)or not so near) the start of a gene and helping recruit the RNA polymerase.

d) (4 points) Consider a RNA sequence *S* containing 10 A's, 10 C's, 10 G's, and 10 U's, and sequence *T* also contains the same number of each nucleotide, but in a different order. *True or False? Explain.* The score of the RNA secondary structure produced by the Nussinov algorithm for *S* will always be equal to that for *T*.

False. This depends on the arrangement of the nucleotides. With AAA..CCC…GGG…UUU, we can form 19 pairs (10 A-U pairs, 9 C-G pairs, and 2 unpaired nucleotides for the loop).

e) (4 points) Consider two single nucleotide polymorphisms (SNPs) found the human population:

	Major allele (%)	**Minor allele (%)**
SNP1:	A (98%)	T (2%)
SNP2 :	C (55%)	G (45%)

Give two reasons why the minor alleles of the two SNPs have frequencies that are so different.
1) In SNP1, allele T may be deleterious (i.e. cause a reduction in fitness, e.g. a disease), which would lead to a lower allele frequency. Or allele A may cause an advantage.

2) In SNP1, allele T may be recent and may not have had time to spread to a large fraction of the population.

3) SNP2 may be under balancing selection (i.e. a case where it is beneficial to be heterozygous).

f) (4 points) Suppose you performed a RNA-seq experiment to measure gene expression (for m = 20,000 genes) in 100 diabetic patients and 100 non-diabetic patients. You then used a Student t-test to obtain a p-value for each gene. The result shows that the average expression level of gene G is 200 in diabetic patients and 100 in non-diabetic patients. However, the variances of the expression levels of G in the two groups are large, so the p-value obtained from the t-test is 0.15, i.e. the difference is not statistically significant. The expression level of G is then measured on 1000 patients from each group, and the average expression values and variances remain exactly the same as before. How will the p-value of the t-test for G change, if at all?

The t-value will remain the same, but the p-value will become smaller, because the number of degrees of freedom of the Student distribution increases with sample size.

g) (4 points) As seen in class, the Sankoff algorithm calculates the parsimony score of a given alignment column on a given phylogenetic tree. The algorithm is based on the following recurrence, which calculates a table S_u at node u based on the previously computed tables S_v and S_w, where v and w are the two children of u. For nucleotide a:

$$S_u[a] = \min(\, S_v[a],\, 1 + \min\{S_v[b] : b \neq a\}) + \min(\, S_w[a],\, 1 + \min\{S_w[b] : b \neq a\})$$

The algorithm can be modified to calculate the *weighted* parsimony score of an alignment column, where instead of trying to minimize the total number of substitutions performed along the branches of the tree, we assign each possible substitution from nucleotide x to nucleotide y a weight $w(x,y)$, and seek to minimize the total weight of the substitutions performed along the branches of the three. This is done by redefining $S'_u[a]$ as the *weighted* parsimony score of the subtree rooted at node u if u is labeled with a. Give the recurrence to calculate $S'_u[a]$.

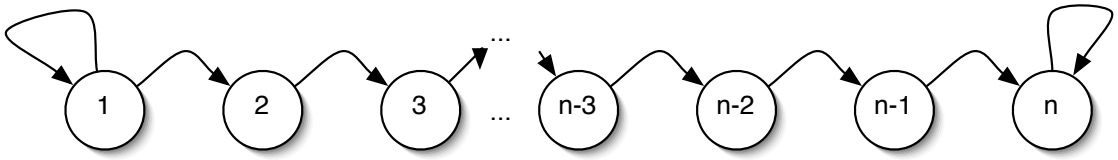
$$S'_u[a] = \min(\, S_v[a],\, \min\{S_v[b] + w(a,b) : b \neq a\}) + \min(\, S_w[a],\, \min\{S_w[b] + w(a,b) : b \neq a\})$$

Question 2. (25 points)

a) (3 points) Consider a hidden Markov model with *n* states. What is the worst-case running time of the Viterbi algorithm on a sequence of length *L* ? Use the Big-O notation. No justification is needed.

$O(L\ n^2)$

b) (10 points) Now, consider the following hidden Markov model with *n* states. Describe how to modify the Viterbi algorithm so that, on this particular type of linear HMM, the running time is asymptotically faster than in (a) ? Describe the modification required, and give the running time of the improved algorithm. Your answer could be in pseudocode, or simply refer to changes to the pseudocode given in class.



We first note that in this HMM, all but one of the states has a single possible previous state. This allows an optimization of the Viterbi algorithm:

*For state s=1: V(i,s) = V(i-1,s)*T(s,s)*E(s,i)*
*For state 1<s<n: V(i,s) = V(i-1,s-1)*T(s-1,s)*E(s,i)*
*For state s=n: V(i,s) = max{ V(i-1,s-1)*T(s-1,s)*E(s,i), V(i-1,s)*T(s,s)*E(s,i).*

*The execution of the dynamic programming algorithm then proceeds as in the normal Viterbi algorithm. Since the computation of each entry of the dynamic programming matrix is done in constant time, we get a running time of O(n*L).*

c) (8 points) Consider a 2-state HMM with states set S = {A, B}, alphabet $\Sigma = \{0, 1\}$, and with transition, emission, and initial state probabilities given below:

Transition probability matrix:

	A	B
A	0.5	0.5
B	0.5	0.5

Emission probability matrix:

	0	1
A	0.7	0.3
B	0.4	0.6

Initial state probability:

A	B
1	0

Consider the sequence of observations:

X =	1	0	1	1	1	1	0	1	1	0	0	0
P=	*A*	*A*	*B*	*B*	*B*	*B*	*A*	*B*	*B*	*A*	*A*	*A*

What is the path *P* that is mostly likely to have generated *X,* i.e. the path that maximizes Pr[$P \mid X$]? Write the path below the sequence *X*, each state aligned to the symbol it emitted.

Hint: You could obtain the answer by manually executing the Viterbi algorithm, but this would take a lot of time. Instead, take a good look at the probability tables; this should suggest a big shortcut.

d) (4 points) Explain briefly (2-3 lines) how you got the solution you gave in (c).

We note that the transition probabilities are all equal. This means that the only relevant variables are the emission probabilities and initial state probability. Since the initial state probability of A is 1, the path must start in that state. Afterward, the optimal path is the one that use A when emitting 0 and B when emitting 1.

Question 3. (15 points)

Consider the following alignment problem:
Given:

- A short DNA sequence *R*
- A long DNA sequence *G*
- A substitution cost matrix *M*
- A cost *c* for gaps (linear gap penalty)

Find: A substring *G'* of *G* that, when aligned to the entire sequence *R*, achieves the highest possible alignment score. The output should be in the form of the starting and ending positions of *G'* in *G*. The algorithm does not need to report the alignment itself.

a) (10 points) Describe a modification to one of the alignment algorithms seen in class to obtain an algorithm that is guaranteed to produce the optimal solution to the problem. The algorithm should run in time O($|R| * |G|$), where $|R|$ is the length *R* and $|G|$ is the length of *G*.

We can find the optimal alignment by a modification of the Needleman-Wunsch algorithm where the leading and ending gaps placed in sequence R have score zero.:

Let X(j,j) be the score of the optimal alignment of R(1…i) to G(1…j), where the leading gaps in placed in R have cost zero. X(i,j) is obtained using the standard NW algorithm, with two modifications:

1) *Initialization:*
   X(0,j) = 0 for all j
   *X(i,0) = c * i*

2) *Recurrence : same as in NW*
3) *Traceback: instead of performing traceback from X(|R|,|G|), we start from* $argmax_j \{X(|R|,j)\}$. *The traceback procedure ends when we reach i=0.*

b) (5 points) The Blast algorithm was designed to solve problems similar to the problem presented in part (a). However, this algorithm is not guaranteed to produce an optimal solution. Suppose that the sequence R that is given as input is actually an exact substring of G, i.e. R matches a portion of G exactly. True or False: Once the indexing of G is done, the running time of Blast on input R will be $O(|R|)$, i.e. it will be independent of $|G|$. Explain your answer.

False. The running time will depend on the number of short perfect matches found between R and G. That number is proportional to $|G|$.

Question 4. (10 points)

Consider a set of 10 random DNA sequences, each of length L=1000 bp, where each nucleotide is selected randomly and independently with the following probabilities:
A: 10%, C: 30%, G: 40%, T: 20%.

a) (2 points) In the 10 sequences described above, which DNA word of length 6 has the largest expected number of occurrences? How many occurrences would you expect for that word?

GGGGGG is the word with the largest expected number of occurrences. That number is $10*(1000-6+1)*0.4^6$.

b) (3 points) Now, suppose the sequence ACTAGT is inserted once in each of the random sequences described in (a), at a randomly chosen position, by overwriting the 6-nucleotide sequence that was already there. What is the expected number of occurrences of ACTAGT in these 10 new sequences?

10 + expected number of matches occurring by chance at locations other than those where the motif was inserted. Since this motif is not self-overlapping, i.e. it is not possible to have two matches for the motif in overlapping portions of a sequence. We are left with 1000 – 11 – 5 positions where the motif could occur by chance. So the answer is:

$10 + 10*(1000-11-5)*0.1*0.3*0.2*0.1*0.4*0.2 = 10.47$

c) (2 points) In the sequences obtained in (b), what is the expected number of occurrences of the word found in (a)?

Word GGGGGG is also non-overlapping with ACTAGT, so the expected number of occurrences is:

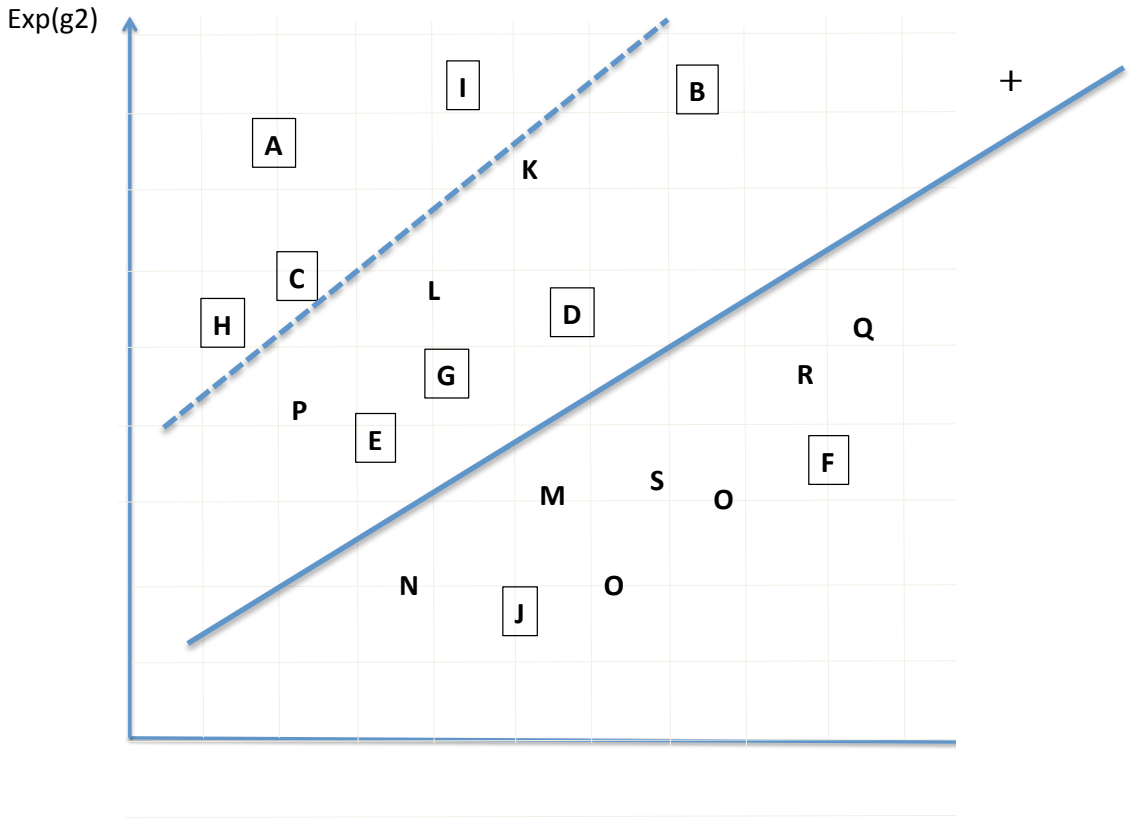
$10*(1000-11-5)*0.4^6 = 40.3$

d) (3 points) Your result in (b) and (c) should show that despite the fact that ACTAGT was inserted in each sequence, this word is not the one that is expected to be most common. However, if the random sequences were not of length 1000 bp but were instead of smaller length, at some point ACTAGT would become the word with the highest expected count. How short would the sequences to be in order for this to happen? More specifically: What is the largest value of L for which the expected count of ACTAGT exceeds the expected count of the word found in (a)?

We need to solve for L in the inequality:
$10 + 10*(L-11-5)*0.1*0.3*0.2*0.1*0.4*0.2 > 10*(L-11-5)*0.4^6$
This gives $L < (1 - 16*0.1*0.3*0.2*0.1*0.4*0.\ 2 + 16*0.4^6) / (0.4^6 - 0.1*0.3*0.2*0.1*0.4*0.2)$, *i.e.* $L<263.2$

Question 5. (12 points)

Consider the following result of an experiment where we measured the expression of two genes, g1 and g2. Assume that samples A, B, …, J (shown with boxes in the figure below) come from patients with a specific disease (positive examples), while samples K, L, …, S (shown without boxes) come from healthy patients (negative examples).



a) (2 points) Draw the linear classifier that obtains the smallest number of classification errors (False-positives + False-negatives) on this data set.

b) (2 points) What is the sensitivity of your classifier on this data?

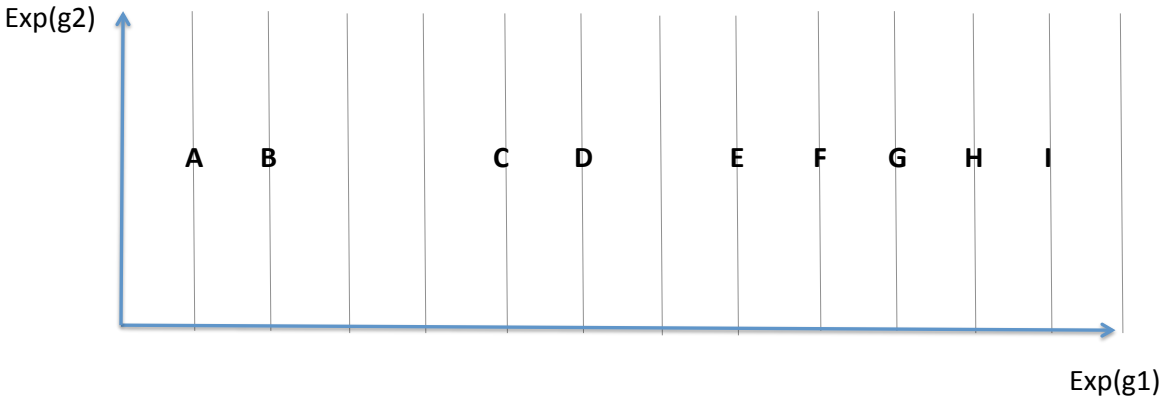
Sensitivity = 8/10

c) (2 points) What is the specificity of your classifier on this data?

Specificity = 7/10

d) (2 points) The positive predictive value of a predictor is defined as PPV = TP / (TP + FP). Draw using a dotted line the linear classifier that achieves the largest PPV. If there are multiple different classifiers that achieve the same optimal PPV, choose the one with the highest sensitivity.

e) (4 points) Now consider a different gene expression data set shown below.



Show what the result of both the single-linkage (left) and the complete-linkage (right) agglomerative clustering algorithm on this data, using the trees (dendrograms) showing the results of algorithm.

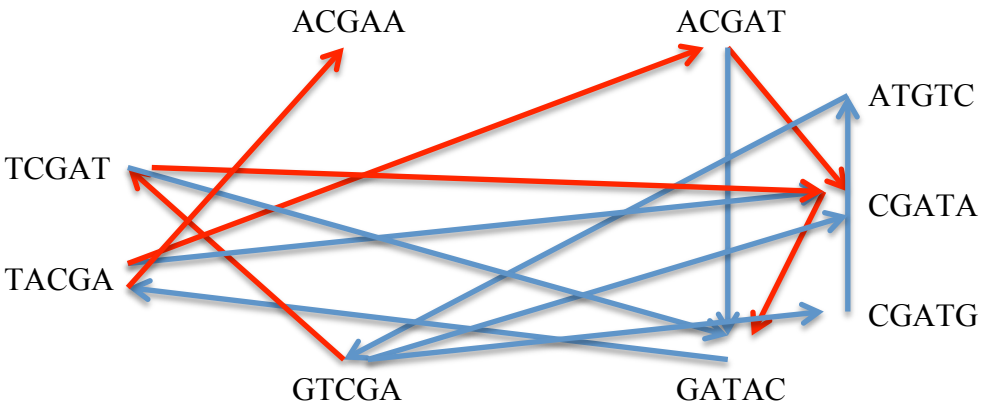
Single-linkage clustering dendrogram	Complete-linkage clustering dendogram
(too lazy to draw it… using the standards tree parenthesis notation instead)	
((A,B), ((C,D), ((E,F), ((G,H),I))))	*((A,B), ((C,D)), ((E,F), ((G,H),I)))*

Question 6. (10 Points)

Consider the following set of 8 short reads obtained from a (toy) genome sequencing experiment.

ACGAA, ACGAT, ATGTC, CGATA, CGATG, GATAC, GTCGA, TACGA, TCGAT

a) (5 points) Suppose we want to assemble the genome from which these reads were obtained. Draw the overlap graph of this set of reads, and specify the weight of the edges. Only consider overlaps of 3 bases or more.



In the graph, red edges have a weight of 1 and blue edges have a weight of 2.

b) (5 points) What is the sequence of the mini-genome that is the most likely to have generated this set of reads?

There was an error in the question, because the graph does not have a Hamiltonian path.

Actually, the more meaningful translation of the genome assembly problem uses not the shortest Hamiltonian path (which may not exist), but the path corresponding to the Chinese Postman Problem, which is the shortest path that visits all the vertices <u>at least</u> once.

Page left blank intentionally. Use it if you need extra space.

Page left blank intentionally. Use it if you need extra space.

NOTE TO COMP 462/561 – Fall 2016 students: These are questions taken from the 2014 final exam. It does not include all questions on that exam, because some of those questions covered material we did not cover this year. Consequently, you should expect that your exam will be a bit longer than what you have here.

Question 1. (16 points, 4 points each)

Indicate whether the following statements are true or false. **Give a <u>two</u> line explanation for each**. *Credits will be given only if the justification is clear and correct.*

a) True or False? *Justify.* Indels that occur in the coding region of a gene will always cause a frame shift.

False. Indels that are of size that is a multiple of 3 will not cause a frame shift.

b) True or False? *Justify.* An RNA-seq experiment can be used to measure the abundance of each of the 20,000 human proteins in a given sample.

False. An RNA-seq experiments measures the abundance of mRNA transcripts, not of the proteins they encode. The amount of protein copies present at time t in a cell is a function of the transcriptional level of the gene in the past, the splicing and translational efficiency, and the protein's and mRNA's degradation (among others).

c) Give *two* reasons why the secondary structure inferred for a given RNA sequence using the Nussinov algorithm may not always reflect the true structure this sequence adopts in cells.

1) The energy model used by the algorithm is very coarse; it does not capture all the fine details of the structure, e.g. pseudoknots, non-canonical interactions, etc.
2) The structure of an RNA sequences is dynamic. At a given time t, it may not be folded in its most stable structure.

d) (4 points) Gene expression can be assessed using a RNA-seq experiment. In that case, the expression level of gene *g* is obtained using the FPKM measure (Fragment Per KiloBase Per Million reads).

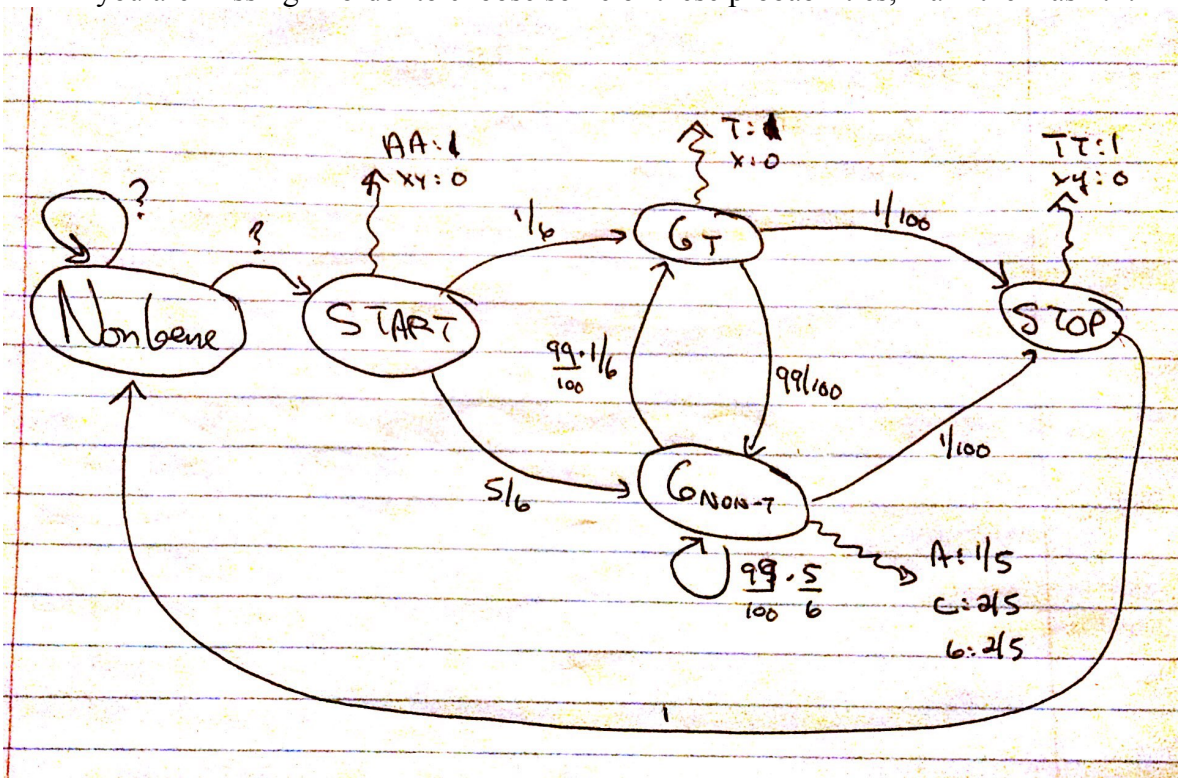
Explain the two types of normalization this entails, and why they are necessary.
"per KiloBase":
Normalizes for the length of the gene. Because the number of reads observed for a given gene is proportional to the length of gene, one needs to factor this out in order to be able to compare genes of different lengths.
"per Million Reads":
Normalizes for the amount of sequencing done. Allows comparing two datasets for which the coverage were different.

Question 2. (16 points)

Suppose that the Philae mission to comet 67P was equipped with a DNA sequencer. It finds DNA on the comet and researchers determine that on that comet, genes have a highly simplified structure:

- Proteins are made of only 4 types of amino acids. Each amino acid is encoded by a single nucleotide.
- Genes always start with ‘ A A ’
- Genes always end with ‘ T T ’
- The body of genes is made of any number of ‘A’, ‘C’, or ‘G’, ‘T’, but never contain two consecutive T’s, as those would be interpreted as a STOP signal. In genes, C’s and G’s are each twice as frequent as A’s and T’s.
- Genes contain no introns
- Genes are on average 100 bp long.

a) (9 points) Draw an HMM that could be used to make gene predictions in this type of DNA. Include all the transition and emission probabilities. If you think that there’s some information you are missing in order to choose some of these probabilities, mark them as “?”.



b) (4 points) Indicate what information you would need to be able to choose the value of probabilities you’ve marked as “?”.

We would need to know the average length of intergenic regions.

c) (3 points) If no one is able to give you the information you specified in (b), name the algorithm that you could use to estimate them?

Baum-Welch algorithm, or Viterbi training.

Question 3. (15 points)

When using Sanger sequencing to read a short piece of DNA, the probability of sequencing error, i.e. calling nucleotide x whereas the correct nucleotide is y, increases as a function of the position within the read. Suppose that a read has length 1000 and that Pr[error at position p] = p / 1000. Assume that insertions (i.e. the inclusion in a read of a nucleotide that was not present in the DNA sequence) never happen. Deletions (the omission of a nucleotide) are possible but very rare, so we will assume that a read contains at most one deletion of one nucleotide.

a) (10 points) Give a modified version of the Needleman-Wunsch algorithm that could be used to align two reads, under the assumption given below. Pay particular attention to the substitution and indel scoring schemes.

Suppose we are aligning reads S and T, and we are given a substitution matrix M and gap penalty c.

We get the following recurrence:

$$X(i,j) = \max \begin{cases} X(i-1,j-1) + M'(S[i],i,T[j],j) \\ X(i-1,j) + c \\ X(i,j-1) + c \end{cases}$$

where M'(S[i],i, T[j],j) is the expected score of the match between the unknown nucleotide at position i in S and that at position j in T. That is:

$$M'(a,i,b,j) = M(a,b)*(1-i/1000)*(1-j/1000) + \sum_{b' \neq b} M(a,b')*(1-i/1000)*(j/1000) + \sum_{a' \neq a} M(a',b)*(i/1000)*(1-j/1000) + \sum_{a' \neq a} \sum_{b' \neq b} M(a',b')*(i/1000)*(j/1000)$$

We initialize X(0,0) = 0, X(1,0)=c, X(0,1)=c. We then calculate only entries of the form X(i-1,i), X(i,i), and X(i+1,i), because only those correspond to scenarios with at most one deletion per read.

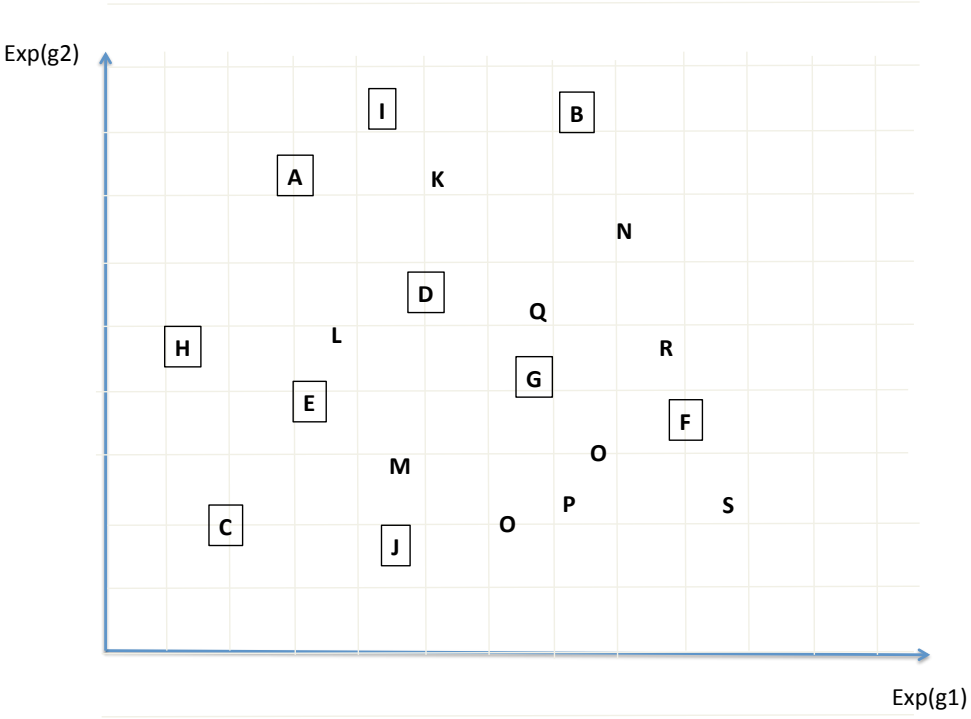
Thus the algorithm runs in time O(n), where n is the length of the reads.

b) (5 points) Suppose that you are interested in reading as accurately as possible the sequence of a portion of a genome and that you have multiple reads originate from that region. Describe informally in 4-5 lines how could you combine the information contained in the reads in order to obtain the most accurate prediction about the exact sequence of that genomic region?

Suppose you have n reads covering a particular position of the genome we want to sequence (assuming homozygosity). The simplest approach would be to infer the majority base (i.e. the base observed the most often in the reads aligned to this position) as the most likely base. However variable error rates should also be factored in. One way would be to give more weight to bases that are close to the beginning of their read. A better way would be to use Bayes theorem to calculate Pr[true nucleotide | R1, R2, .. Rn], factoring in position-dependent error rates.

Question 5. (16 points)

Consider the following result of a microarray experiment, where in each case we measured the expression of only two genes, g1 and g2. Assume that samples A, B, …, J (shown with boxes in the figure below) come from patients with a specific disease, while samples K, L, …, S (shown without boxes) come from healthy patients.



a) (5 points) Draw the linear classifier that obtains the smallest number of classification errors (False-positives + False-negatives) on this data set.

I can't easily draw it now, but the line separates points ***I, B, A****, K****, D, H****, L,* ***E, C*** *from points N, Q, R,* ***G, F****, M, O, P, O', S,* ***J.***

Note: The sample O is present in the figure twice by mistake. Assume it is O and O'.

b) (3 points) What is the sensitivity of your classifier on this training data?

7/10.

c) (3 points) What is the specificity of your classifier on this training data?

8/10

d) (5 points) If one wants a sensitivity of at least 90%, where should the boundary of the linear classifier be moved in order to maintain a specificity that remains as high as possible? Draw you answer on the figure above, using a dashed line.

The classifier would separate R, O, O', P, F, S from the rest.

Question 7. (15 Points)

You are given two RNA sequences, $S = s_1 \ldots s_m$ and $T = t_1 \ldots t_n$, that you want to align. Sequence S has a known nested secondary structure (without pseudoknots), given to you in the form of a list of pairs of positions in S that form base pairs:
Struct = { $(l_1,r_1), (l_2,r_2), \ldots, (l_k,r_k)$ }, where $1 \leq l_i < r_i \leq m$ for all $i \in \{1 \ldots k\}$.
The secondary structure of sequence T is not known but is believed to be related to that of S.

Give an alignment algorithm to align sequences S and T using a given substitution matrix M and linear gap penalty c, subject to the following constraint:
If nucleotides s_i and s_j form a base pair in S (i.e. $(i, j) \in$ Struct), and t_a is aligned to s_i and t_b is aligned to s_j, then t_a and t_b must be complementary nucleotides (i.e. A-U, C-G, G-C, or U-A pairs).

This question turned out to be harder than expected. Any solid attempt at it was marked generously. Here is my solution:

Here, we have to use a combination of Needleman-Wunsch and Nussinov algorithms.
Let X(i,j,k,l) be the score of the optimal structure-preserving alignment for si…sj against tk…tl.
We get

$$X(i,j,k,l) = \max \{ X(i+1,j-1,k+1,l-1) + M(Si,Tk) + M(Sj,Tl) \text{ if: (1) } (i,j) \text{ not in Struct}$$

Or

(2) Tk and Tl are complemenary

$$X(i+1,j,k,l) + c$$
$$X(i,j,k+1,l) + c$$
$$X(i,j-1,k,l) + c$$
$$X(i,j,k,l-1) + c$$

$$Max_{i < i' < j} Max_{k < k' < l} \{ X(i,i',k,k') + X(i'+1,j, k'+1,k) \}$$
$$\}$$

Page left blank intentionally. Use it if you need extra space.

Page left blank intentionally. Use it if you need extra space.

COMP 462/561 – Computational Biology Methods
Midterm examination – October 16th 2019, 4:05 – 5:25

NAME: _____ MCGILL ID #: _____

Instructions:

- The exam has 4 questions
- Answer directly on the exam
- This is an open book exam.
- No calculators, computers, or any electronics allowed.

1) Short answer questions (44 points)

a) (6 points) In bacteria, the average length of proteins in the is 320 amino acids. What is the average length of a the protein-coding portion of a gene (in nucleotides)?

3 * 320 = 960

b) (8 points) In the Blast algorithm, an ungapped extension phase is performed before doing the gapped extension phase. Why not skip the ungapped extension phase and go directly to the gapped extension phase?

This would make the algorithm very slow, as the ungapped extension is a process that allows the algorithm to decide not to perform the csotly gapped extension phase when the result of the ungapped extension is not good enough. However, skipping the ungapped extension would not have any negative impact on the accuracy of the results.

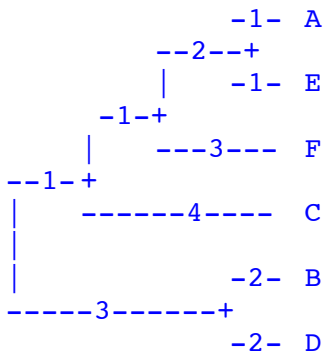
c) (8 points) List at least three possible consequences of a nucleotide substitution within the coding portion of a gene on the amino acid sequence of the protein it encodes?

- No change to the protein sequence (Synonymous mutation)
- Change of a single amino acid (Non-synonymous mutation)
- Creation of a premature STOP codon (nonsense mutation)
- Alteration of START codon, results in a shorter protein and (possibly) a frameshift
- (in eukaryotes) The mutation may alter a splice site, making the improperly spliced.

d) (12 points) Consider an evolutionary process that follows exactly the molecular clock assumption, i.e. mutations occur at a constant rate, and suppose that we are given 6 very long sequences so that we can estimate with perfect accuracy the number of years of divergence between any two given species. The resulting distance matrix is:

	A	**B**	**C**	**D**	**E**	**F**
A	-	10	8	10	2	6
B	10	-	10	4	10	10
C	8	10	-	10	8	8
D	10	4	10	-	10	10
E	2	10	8	10	-	6
F	6	10	8	10	6	-

Draw the rooted phylogenetic tree and branch lengths that best match this distance matrix.
Hint: Although you could apply the UPGMA algorithm and get the correct answer, you can probably guess the correct tree.



f) (12 points) Consider the problem discussed in class to introduce Hidden Markov Models, where a person walks randomly in the city of Montreal, across different neighrborhoods (S = {F, E, C}), hearing different greetings (alphabet Sigma = {b, h, n, a}) every minute. Assume that the Emission and Transition probability matrices are the following:

Emission =

	b	**h**	**n**	**a**
F	1	0	0	0
E	0	1	0	0
C	0.3	0.2	0.25	0.25

Transition:

	F	**E**	**C**
F	0.9	0	0.1
E	0	0.9	0.1
C	0.5	0.5	0

Initial State Probability: 0.333 for all states

Consider the following sequence of observations:

X =	b	b	b	b	b	h	h	h	h	h	h
Optimal path P =	F	F	F	F	C	E	E	E	E	E	E

What is the path *P* for which Pr[*P,X*] is maximized? Write your answer above, and explain your answer below.

Hint: Although you could use the Viterbi algorithm to answer the question, you should instead take a careful look at the emission and transition probability tables, which should suggest an easier approach.

Explanation (max 5 lines):

Observing b is much more likely from state F than from E or C. Observing h is much more likely from E than from F or C. So fom the perspective of emission probabilities, we'd like to be in state F for the first 5 steps, and then transition to F for the last 6 steps. However the trasition probability from F to E is zero. So we have to go through the C state. The only question is then: when should the transition F->C->E happen? It should happen as shown above, because Pr[b|C] > Pr[h|C].

2) Short-ish answer questions (14 points)

The global pairwise alignment problem with linear gap penalty is meaningful and interesting when the score c of gap is neither too large nor too small in comparison to the score of a match and mismatches. In this question, we will explore what happens to the solutions to the optimal alignment problem when we go beyond those values. We will work under the following assumptions:

- We are using a substitution cost matrix M where $M(a,a) = +1$ and $M(a,b) = -1$ for $a \neq b$.
- We are using a linear gap penalty: score(L) = $c*L$, where L is the length of the gap and c is the gap cost.
- IMPORTANT: The two sequences being aligned have *exactly* the same length n.

a) (7 points) When c becomes too large negatively, the solution to the alignment problem beomes trivial, because the optimal alignment is always an alignment with no gaps at all, irrespective of the content of the two sequences. Determine at what values of c this guaranteed to happen, and explain your reasoning. The value of c may depend on n.

We first note that if the two sequences have exactly the same length, then the number of gaps inserted in each sequence must be equal, i.e. the total number of gaps in the alignment must be either 0, 2, 4… The smallest number of gaps in a non-gap-free alignment is thus 2.

The highest score for a gap-containing alignment is achieved on an example like this one:

```
S = ACGTACGT…ACGTACGT
T = CGTACGT…ACGTACGTA
```

If c is not too large, the optimal alignment is

```
S = ACGTACGT…ACGTACGT-
T = -CGTACGT…ACGTACGTA
```

which has score (n-1) + 2 c

When c becomes too large, we get an alignment with no gaps at all, with score –n.
So we solve for c in the inequality –n > (n-1) + 2 c:
➔ c < -n + 1/2

b) (7 points) When c becomes too small (i.e. close to zero), the solution to the alignment problem also becomes less interesting (but not trivial), because the optimal alignment never contains any mismatches at all, irrespective of the content of the two sequences. Determine at what value of c this happens, and explain your reasoning.

This happens when the score of a mismatch (-1) is worse than the score of 2 gaps (2c), because when that's the case we're always better off replacing any mismatches with two gaps, e.g.

```
A    ➔   A-
G        -G
```

So the answer is 2c>-1, i.e. c>-1/2

3) Prefix-suffix alignment (20 points)

Consider the following variant of the pairwise sequence alignment problem.
Given: - Sequences $S = s_1 s_2 \ldots s_m$, and $T = t_1 t_2 \ldots t_n$,
- Scoring scheme using an linear gap penalty c and a substitution matrix M

Find: The prefix $s_1 \ldots s_i$ of S and the suffix $t_j \ldots t_n$, of T, that, when optimally aligned, yield the highest possible alignment score.

For example, for S = A T A C A G A T A A G C
T = G A A G T A T G A T G A T
with c = -1 and $M(a,b)$ = +1 if a=b and -1 otherwise, the optimal alignment has score +3, which is obtained with i = 8 and j = 6, as shown in bold below:

A T - A C A G A T A A G C
G A A G T **A T G A T - G A T**

Task: Describe an algorithm to solve the prefix-suffix exactly. Your algorithm must run in time O(m n). The output should be (1) the score of the optimal alignment, and (2) the values of i (position where the prefix of S ends) and j (position where the suffix of T starts). You don't need to output the optimal alignment itself.

We achieve this with a modified version of the Needleman-Wunsch algorithm, where the gaps inserted in S before the first nucleotide are free, and the gaps inserted in T after the last nucleoide are also free. This requires a modification to the initialization of the dyn. Prog. Algo, and another for starting the traceback.

Let X be the dyn. Prog. Table.

X(0,j) = 0 ← First modification
X(i,0) = c*i

X(i,j) = same as in NW algo

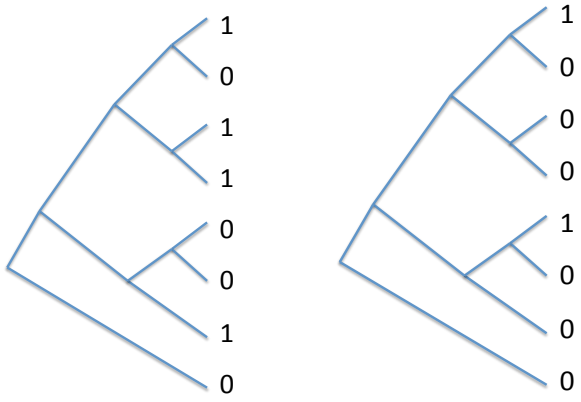
Initiate traceback from argmax{X(i,|T|): i =1…|S|} ← Second modification
End trace back when we reach i=0.

4) Maximum Parsimony algorithms (20 points)

Consider the following problem, which is a variant of the Maximum Parsimony problem.

We have a set of *n* species placed at the leaves of a known phylogenetic tree *T*. Each species *s* is known to have or not have a particular binary propery P (for example, the ability to perform photosynthesis). This is denoted by *P(s)* = 1 or *P(s)* = 0. We assume that the property *must have arisen only once* during evolution (either before the root of *T*, or at some point along a single branch of *T*). However, the property may have been *lost multiple times* independently along different branches of the tree.

Task: Describe a polynomial time algorithm that computes the minimum number of losses (edges where P(parent)=1 and P(child)=0) needed to explain the data at the leaves of the tree, subject to the constraint that *there can be at most one gain (edges where P(parent)=0 and P(child)=1)*. See examples to the righ.



Score = 2 losses Score = 4 losses

There are many ways to solve this question. Here's one.

First, we find the most recent common ancestor of all the leaves where P=1. In the two examples above, this is the left child of the root. This node is where the property first appeared. [It cannot have appeared later than this, because it would then require two gains to explain the data at the leave. If it appeared earlier than this, it would be sub-optimal as it would require some additional losses. Call this node X. All internal nodes located outside of the subtree rooted at X will be assigned value 0. To label the internal nodes in the subtree rooted at X, we used a modified version of the Sankoff algorithm, in which gains are disallowed:

Su[1] = max(Sv[0]+1, Sv[1]) + max(Sw[0]+1, Sw[1])
Su[0] = Sv[0] + Sw[0]

The rest of the algorithm is the same as before.

Note: An alternate approach to label the nodes of the subtree rooted at X is to use the following rule. To label node u, choose 0 is all the leaves of the subtree rooted at u have value 0. If that's not the case, set u to 1.

This page is left blank intentionally. Use it if you need extra space.

This page is left blank intentionally. Use it if you need extra space.

COMP 462/561 – Computational Biology Methods
Midterm examination – October 18th 2012

NAME: _____

1) Short answer questions (30 points; 5 points each)

a) What is the main difference between the structure of genes in Prokaryotes (e.g. bacteria) and in Eukaryotes (e.g. human) ?

Eukaryotic genes generally have introns, whereas prokariotic genes do not.

b) Name and describe briefly the three main biochemical steps that lead to the production of a protein from a eukaryotic gene.

1)Transcription, which transcribes the DNA of a gene into pre-messenger RNA

2)Splicing, which removes introns, to obtained the mature messenger RNA.

3)Translation, which converts the mRNA to a protein sequence.

c) In protein-coding regions of genomes, why are insertions and deletions of 3 consecutive nucleotides more frequently observed than those of 1 or 2 nucleotides?

Because indels of size that is not a multiple of 3 result in frame shifts, so that all codons downstream are affected. This means that the amino acid encoded by the sequence downstream of the indel are likely to be completely changed. Indels of length 3 affect only one or two amino acids.

d) If only 16 amino acids existed (instead of 20), what would be the minimal length of codons? Why? Watch out: that's a trick question!

3. You'd need 16 codons to encode the 16 amino acids, plus one for the stop codon. So you'd need ceiling ($\log_4$ (16+1)) = 3 nucleotides per codon.

e) Give one advantage and one disadvantage of using a Blast seed of size w=11 versus a blast seed of size w=12.

Advantage of w=11: Better sensitity.
Disadvantage of w=11: Higher running time.

f) What is the expected length of an open reading frame (ORF) in a random sequence where the nucleotides are chosen independently with probability $p_A = p_T = 1/3$, $p_C = p_G = 1/6$? Recall that the three stop codons are TAA, TAG, and TGA.

*Probability of a stop codon = 1/3*1/3*1/3 + 1/3*1/3*1/6 + 1/3*1/6*1/3 = 2/27. So the expected length of an ORF is 1 / (2/27) = 27/2 = 13.5 codons = 40.5 nucleotides.*

2) Pairwise sequence alignment (20 points)

We have studied the Needleman-Wunsch dynamic programming algorithm that finds the optimal alignment between two sequences, given a substitution cost matrix M and a gap penalty scheme. We have studied two types of gaps penalties:
(i) linear gap penalty where the cost of an insertion or deletion of size n is cost(n) = a * n,
(i) affine gap penalty where the cost of an insertion or deletion of size n is cost(n) = a * n + b.

Write a polynomial-time dynamic programming algorithm that is able to find the optimal alignment for any given *arbitrary* convex indel cost function (i.e. a gap cost function cost(n) such that cost(n+ n') <= cost(n) + cost(n') for all n,n' >= 0; an example would be cost(n) = log(n)).
Hint: The running time of the algorithm should be $O(n^3)$.

We use a variant of the NW algorithm. Consider sequences S=s1…sm, T = t1…tn. Let X(i,j) be the score of the optimal alignment between prefixes of length i and j of S and T resepectively. Then,

X(i,0) = cost(i) for all i =1…m
X(0,j) = cost(j) for all j = 1…n

For i=1…m,
For j= 1…n
X(i,j) = max {
X(I,j) + M(si,sj),
$max_{g \text{ in } \{1 \ldots i\}}$ *{ X(i – g , j) + cost(g)}*
$max_{g \text{ in } \{1 \ldots j\}}$ *{ X(i , j - g) + cost(g)}*
}

Then the algorithm proceeds as for NW…

3) Multiple sequence alignment (20 points)
The algorithm seen in class for progressive multiple alignments aims at optimizing the sum-of-pairs score. Suppose that we are instead in the following problem:

Maximum Parsimony Multiple Alignment Problem
Given: A set of sequence S1,…,Sn, and a tree T with one leaf per sequence
Find: The alignment of S1,…,Sn such that the parsimony score of the alignment on tree T is minimized.

Here, for the purpose of computing the parsimony score of a given alignment column, we consider a gap as a fifth character, other than A, C, G, and T.

Explain how to modify the progressive alignment algorithm seen in class to adapt it to the Maximum Parsimony Multiple Alignment Problem.

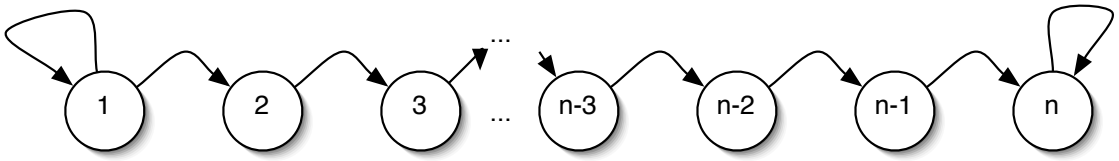
The only change that is that when computing the alignment for node u in the tree, we score the matching of one set of aligned nucleotides (coming from the left subtree of u) to another set of aligned nucleotides (coming from the right subtree of u), one use calculates the parsimony score rather than the sum-of-pairs score.

4) Hidden Markov Models (20 points)

a) (5 points) Consider a general hidden Markov model with n states. What is the worst-case running time of the Viterbi algorithm on a sequence of length L ?

$O(L\ n^2)$

b) (15 points) Now, consider the following hidden Markov model with n states. Is it possible to modify the Viterbi algorithm so that, on this particular type of linear HMM, the running time is asymptotically faster than in (a) ? Describe the modification required, and give the running time of the improved algorithm.



Use the following recurrence for the Viertbi algorithm:

*When k=1, V(k, i) = V(k,i-1) * T(1,1) * E(k, xi)*
*When k= 2… n-1, V(k, i) = V(k-1,i-1) * T(k-1,k) * E(k, xi)*
*When k= n, V(k, i) = max { V(k-1,i-1) * T(k-1,k) * E(k, xi), V(k, i-1) * T(k,k) * E(k, xi) }*

BONUS question (5 points)

Prove that the neighbor-joining algorithm will always produce the correct tree topology if given an ultrametric distance matrix on four species.

Sorry, ran out of time to write this one…

Solution to HW4 – 2019

Q1.a.
All suffixes:

$	14
C$	13
GC$	12
AGC$	11
TAGC$	10
ATAGC$	9
GATAGC$	8
AGATAGC$	7
CAGATAGC$	6
ACAGATAGC$	5
GACAGATAGC$	4
AGACAGATAGC$	3
GAGACAGATAGC$	2
AGAGACAGATAGC$	1

Sorted in lexicographical order gives:

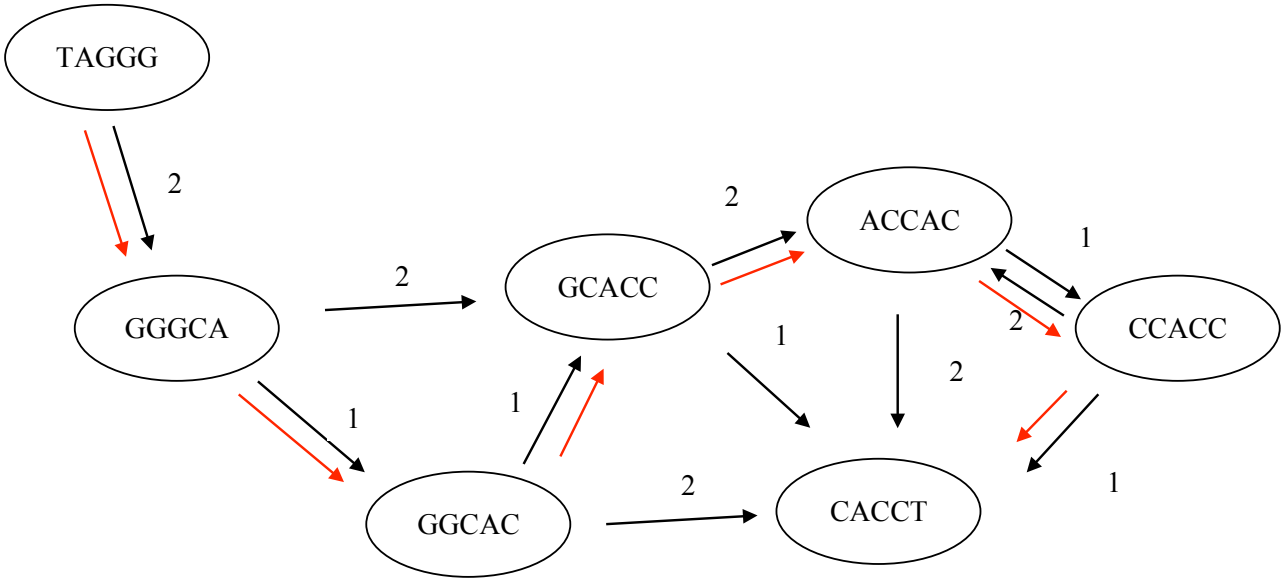
$	14
ACAGATAGC$	5
AGACAGATAGC$	3
AGAGACAGATAGC$	1
AGATAGC$	7
AGC$	11
ATAGC$	9
C$	13
CAGATAGC$	6
GACAGATAGC$	4
GAGACAGATAGC$	2
GATAGC$	8
GC$	12
TAGC$	10

b. Use binary search to locate the substring s in the suffix array S

```
Initialize l = 0, r = len(S)
while l < r:
        mid = (l + r) / 2
        if s > S[mid]:
                l = mid + 1
        elif s<S[mid]
                r = mid – 1
        else    return  mid
return None
```

Searching for AGAT, we will have:
l=0, r=13, mid=6

l=0, r=5, mid=2
l=3, r=5, mid = 4 ➔ Found AGAT
Q2.a.



b. TAGGGCACCACCT
The shortest Hamiltonian path is denoted with red arrows.

Question 3:

a) (5 points) In the genome, some genes have sequences that are very very similar to others (e.g. there might be two genes whose sequences are 99% identical). What difficulties would that cause to the analysis of RNA-seq data? (2-3 lines for each)

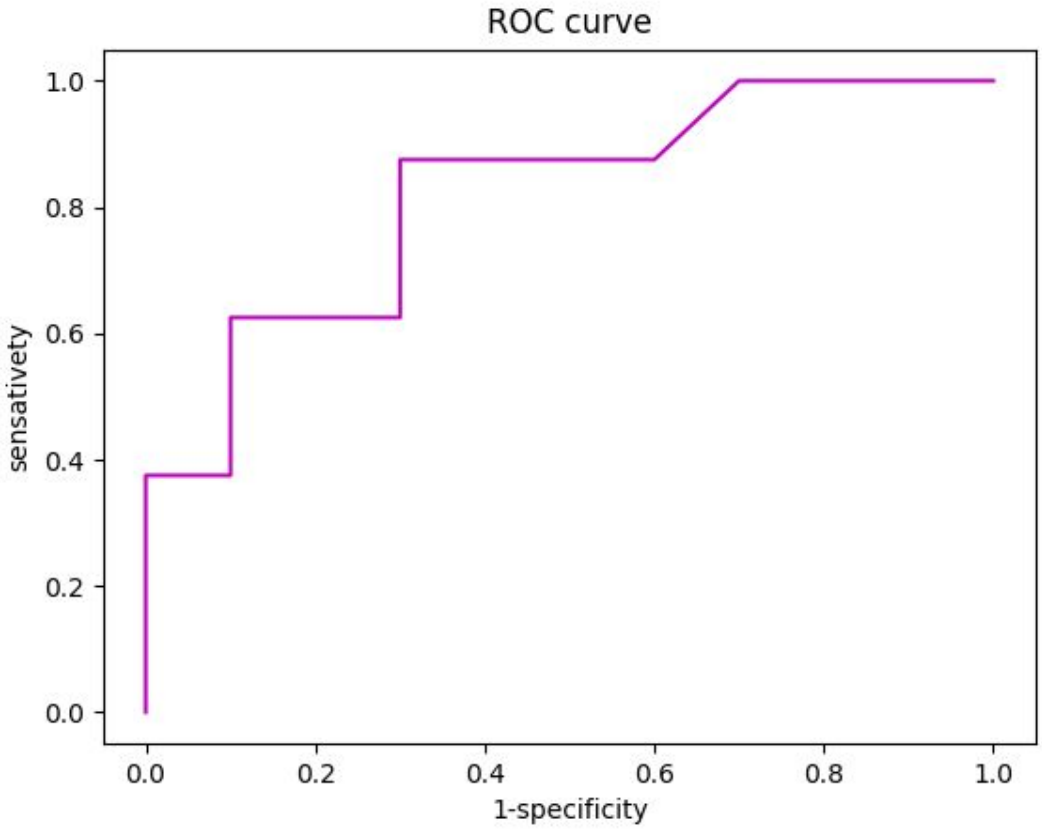
The main issue is that this will make the mapping of reads uncertain. Many reads will map equally well to both genes (these reads are called multi-mapping reads). We then have the choice to exclude multi-mapping reads (which would result in underestimating the expression of one or both of the genes), or counting them for both genes (which would result in over-estimating their expression), or assigning a count of 0.5 for each gene. The best approach would probably be to exclude multi-mapping reads, but then, when normalizing the gene expression using the FPKM approach, use the "mappable" gene length rather than the actual gene length, where "mappable" gene length corresponds to the length of the portion of the gene for which there is no multi-mapping issue. Still, if a gene have an exact copy elsewhere in the genome, we would not be able to know which copy is being expressed.

b) (5 points) Give two reasons why the expression measurements obtained from RNA-seq experiments may not necessarily reflecting the abundance of proteins in the sample?

See solution to past final exam.

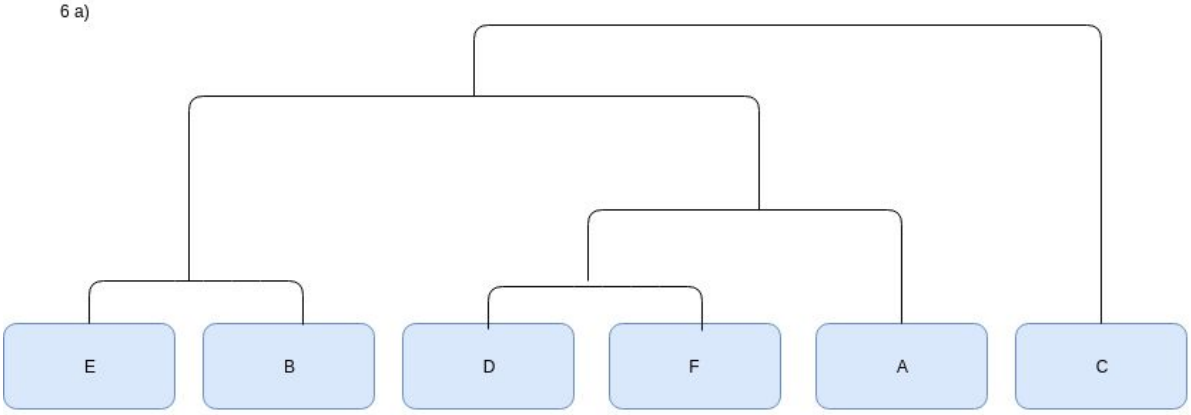
4 a) t-stat : 2.69, p-value : 0.0163
b) Consult the answer from the last year's assignment. The p-value of 0.017 with 100000 iterations using my code.
c) Consult the answer from the last year's assignment.

5 a) threshold 1.4, 2.1, 2.2 gives 4 prediction errors.
b) The ROC curve

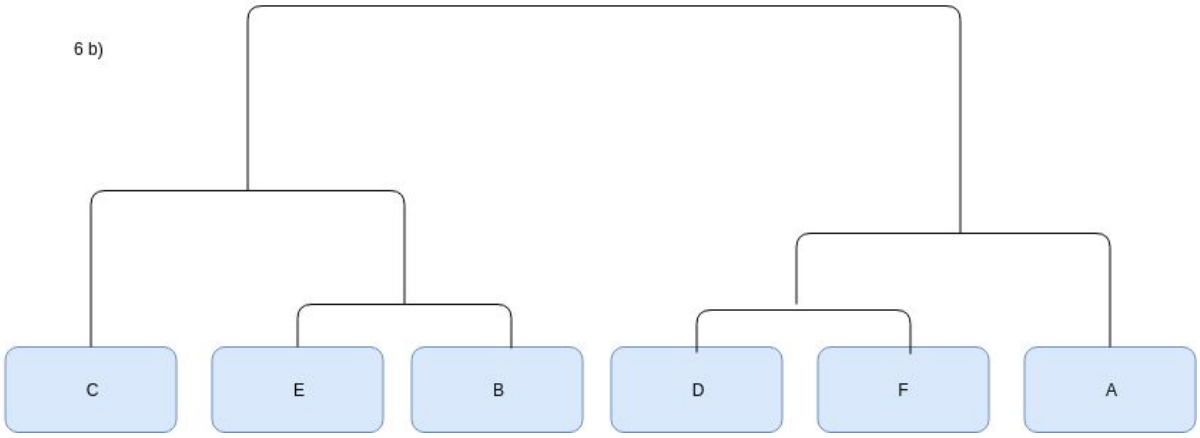


6)

6 a)



6 b)

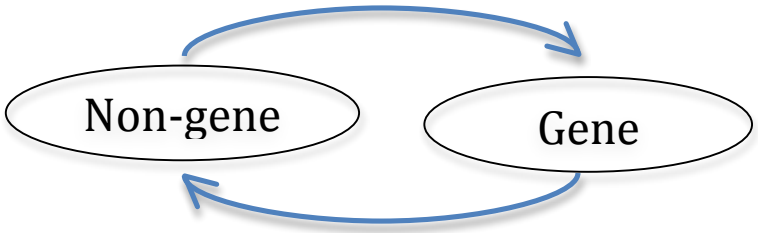


COMP 462 / 561 - Homework #3
Due on November 11 2019, 23:59 on MyCourses

IMPORTANT: Question 1 of this assignment requires a substantial amount of programming, and portions (c), (d), and (e) depend on you having a working program. **Do not leave this to the last minute!** Get started with the programming *early*, and get our help if you need.

Question 1 (80 points).
See code attached separately.

Question 2. (20 points)
If we think of an HMM as a machine to generate a random sequence of observations, the number of consecutive steps the path will remain at a given state follows a geometric distribution (https://en.wikipedia.org/wiki/Geometric_distribution). This means, for example, that, in our gene-finding HMM, the length distribution of exons, introns, and intergenic regions will be assumed to be geometric. However, in reality, these regions have length distributions that can be far from geometric. Consider the very simple two-state gene finding HMM shown below. Assume that we have a desired gene length distribution, provided in the form of a discrete probability distribution for lengths ranging from 1 to 1000: Pr[length = k] = p_k.
Describe (in at most half a page) how you could modify the HMM below to produce a second HMM where the distribution over the duration of stay in the set of gene states is exactly the target length distribution. Note that the Gene state will probably need to be subdivided in several Gene sub-states. Describe not only the states of your HMM but also the transition probabilities.

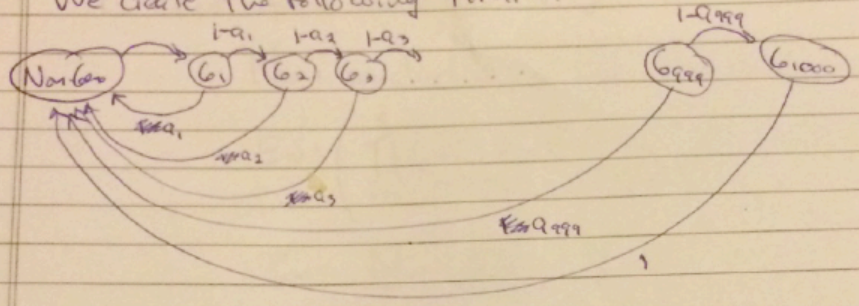


Good luck!

3. There are several solutions. I think the best is:

Assume $p_k = Pr[\text{length} = k]$ is given to us a target length distribution for genes

We create the following HMM:



How to choose $a_1 \ldots a_{999}$ so that the probability a gene of length k is p_k?

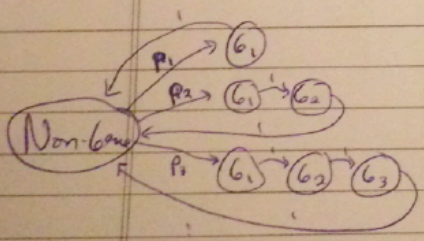
We want $a_1 = p_1$

$$(1-a_1)\cdot a_2 = p_2 \iff a_2 = p_2/(1-a_1) = p_2/(1-p_1)$$

$$(1-a_1)(1-a_2)\cdot a_3 = p_3 \iff a_3 = p_3/((1-a_1)(1-a_2))$$

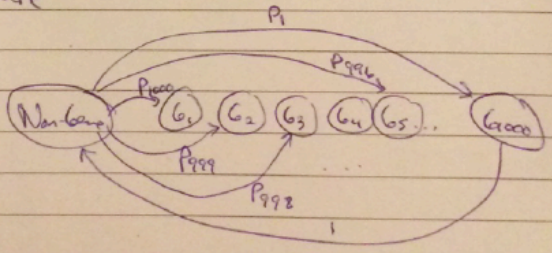
$$\vdots$$

Two alternate solutions are



(Less desirable because HMM gets very large)

or



(Less desirable because we can't assign position-specific emission probabilities)

COMP 462 / 561 – Fall 2019
Homework #2

Question 1. (30 points)
To answer this question, you will need to use some online tools implementing some of the algorithms seen in class. These might include:
Blast (http://www.ncbi.nlm.nih.gov/BLAST)
and
the LIRMM Phylogenetic inference package
http://phylogeny.lirmm.fr/phylo_cgi/index.cgi

If you have never used Blast before, you may find this tutorial useful: http://www.ncbi.nlm.nih.gov/books/NBK1734/ . In particular, learn what an E-value is, and what the different Blast versions do (Blastn vs MegaBlast vs Blastp vs TBlastN, etc.)

Context: You are doctor and you have a patient suffering from a mysterious infection. You've extracted DNA from the infected area, sequenced it, and obtained the DNA sequence at http://www.cs.mcgill.ca/~blanchem/561/mystery.fa .

a) (15 points) What do you think is the cause of the infection? What is the name of the disease?
Hint: Default parameters for blastn may not result in the identification of very useful hits. Consider changing some of these parameters in order to try to identify hits with good E-values.

Using the default Blastn parameters, we get no hit with a good E-value. To increase the sensitivity of the search, we can reduce the word size from 11 to 7, we get hits for the Zika virus, with a very good E-value (E=9e-26).

b) (15 points) Suppose there exist treatments for various strains of the pathogen. Five known strains exist, with sequences given at:
http://www.cs.mcgill.ca/~blanchem/561/strains.fa
Which treatment (i.e. that for which strain) is the most likely to be appropriate for your patient? Use a phylogenetic inference tool such as http://phylogeny.lirmm.fr/phylo_cgi/index.cgi to figure it out. Explain your answer.

Paste your sequences in the web interface, get the tree, se what species is most closely related to mystery.

Question 2. (30 points)
The first algorithm to calculate the parsimony score of a given set of nucleotides at the leaves of a given rooted binary tree was invented by Fitch and Wagner in 1971. The algorithm works as follows. For each node u of the tree T, define a set X_u as follows:

If u is a leaf then
$X_u = \{x\}$, where x is the nucleotide at leaf u.
Score(u) = 0

If u is an internal node with children v and w, then

If ($X_v \cap X_w = \emptyset$) then
$X_u = X_v \cup X_w$
Score(u) = Score(v) + Score(w) + 1
Else
$X_u = X_v \cap X_w$
Score(u) = Score(v) + Score(w)

After all the X sets have been computed, starting from the leaves back to the root, Score(root) is the desired parsimony score.

Question: Prove that the algorithm always yields the correct (minimal) parsimony score. Perhaps the easiest (but not the only) way to do this is to show that the Fitch algorithm will always produce the same answer as Sankoff's algorithm, which we can assume is a correct algorithm. Try to be as formal as you can in your proof, but if you don't have experience writing mathematical/algorithmic proofs, just explain your reasoning as clearly as possible.

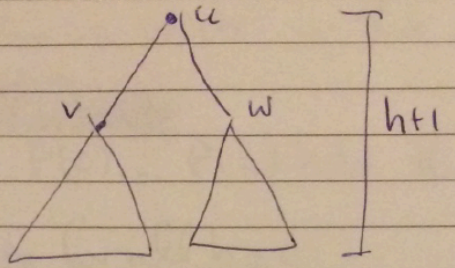
2. We prove the correctness of the algorithm by induction on the height of the tree

Let $P(h)$ be the proposition: In the Fitch algorithm, if node u is the root of a tree of height h, then $score(u) = Parsimony\ Score(u)$ and X_u = set of nucleotides that can be used to label node u and obtain an optimal score.

① Base case: If $h=0$, then the tree consists of a single node u. The algorithm yields $score(u)=0$, and $X_u = \{x\}$, where x is the nucleotide at node u. Both are correct. Thus $P(0)$ holds.

Induction hypothesis: $P(a)$ holds for $0 \le a \le h$

Consider a tree rooted at node u, where the tree has height $h+1$.

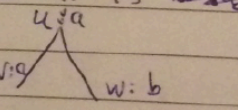


Then the subtrees rooted at nodes v and w both have height $\le h$, so the induction hypothesis applies to them.

If $X_v \cap X_w \neq \emptyset$, then the optimal labeling is obtained by labeling node u with any nucleotide $a \in X_v \cap X_w$, and labeling v and w with the same nucleotide.
This yields a parsimony score of score(v) + score(w)
Thus, the algorithm is correct in that case

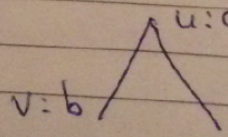
If $X_v \cap X_w = \emptyset$, then the optimal solution is either

① {label u with $a \in X_v$:
label v with a
label w with $b \neq a \in X_w$
→ This yields a solution with score: score(v)+score(w)+1



OR

② {label u with $a \in X_w$
label v with $b \neq a \in X_v$
label w with a
→ This yields a solution with score: score(v)+1+score(w)



Thus, node u can be labeled with any nucleotide from $X_v \cup X_w$, and the score is score(v)+score(w)+1

Thus the algorithm is correct in this case.

Conclusion:
If P(0)... P(h) hold, then P(h+1) holds

$\Rightarrow$ P(h) holds for all $h \geq 0$

Question 3. (40 points)
Phylogenetic trees can be built from non-genetic data, and in fact this was the only type of information available prior to the 1970's, when DNA sequence became possible. Here, you will design and implement an algorithm similar to Sankoff's algorithm, but that will work on quantitative traits (things about a species that can be measured with integers) rather than genetic data. Suppose that you have information about *k* traits for each species. These traits are measured as non-negative integers. For example, for mammals, trait1 might be the length of the forearm (in cm), trait2 might the volume of the skull (in cm^3), trait3 might be the lifespan (in years), etc. As species evolve, their traits change but those changes are generally slow (although there are exceptions). Thus, a parsimony-based phylogenetic inference approach makes sense. The problem we want to solve is the following:

Input:

- A set of *n* species, with, for each species *i*, a vector of *k* integers $D_i = (D_{i,1}, D_{i,2}, \ldots, D_{i,k})$ representing the measurements made for the *k* traits
- A phylogenetic tree *T* with leaves labeled with the *n* species.

Goal:
Assign a vector D_u to each internal node $u = n+1, \ldots, 2n-1$ such that $\sum_{(u,v)\in E(T)} |D_u - D_v|_1$ is minimized, where $|D_u - D_v|_1 = \sum_{i=1}^{k} |D_u(i) - D_v(i)|$.

a) (25 points) Write the pseudocode of algorithm to solve this problem. You can assume that no trait value will ever exceed a given maximum value *M*.
b) (5 points) What is the running time of your algorithm (using big-O notation)?
c) (10 points) The problem formulation above is not ideal in cases where the range of values of different traits differs significantly (e.g. the lifespan ranges between 0 and 50 years, whereas the volume of the brain ranges from 0 to 1500 cm^3). **Explain why and propose a modification to the problem formulation that would make it more biologically relevant. You do not need to propose a revised algorithm to go with your revised problem formulation.**

(3) The objective is to minimize

$$Obj = \sum_{(u,v)\in T} \|D_u - D_v\|_1 = \sum_{(u,v)\in T} \sum_{k=1}^{K} |D_u(i) - D_v(i)|$$

Note that $Obj = \sum_{i=1}^{K} \sum_{(u,v)\in T} |D_u(i) - D_v(i)|$

$= \sum_{i=1}^{K} \text{ParsimonyScore}(i)$

Thus we only need to figure out how calculate the ParsimonyScore of a ~~given position~~ single position of D at a time.

To calculate the parsimony score for position i Mimicking the Sankoff algorithm, let

$S_u[n]$ be the parsimony score of the subtree rooted at u, if u is labeled with integer $n \leq M$

Then

$$S_u[n] = \begin{cases} 0 & \text{if } u \text{ is a leaf and } D_u[i] = n \\ +\infty & \text{if } u \text{ is a leaf and } D_u[i] \neq n \\ \min_{n' \leq M} \{S_v[n'] + |n' - n|\} + \min_{n' \leq M} \{S_w[n'] + |n' - n|\} & \text{if } u \text{ is not a leaf} \end{cases}$$

The rest of the algorithm proceeds exactly like the Sankoff algorithm. The score of the optimal solution is $\min_{n \le M} \{X_{root}[n]\}$

and the optimal labeling is obtained by tracing back the solution from there.

b) The running time for a tree with n leaves is

$O(K \cdot M^2 \cdot n)$

c) The formulation of the problem is not ideal because different traits have different ranges.
For example two species with life span of 5 and 50 years respectively are very different from each other, whereas two species with brain volume 1005 and 1050 cm^3 are actually quite similar. The current formulation will give too much weight to traits with large values.

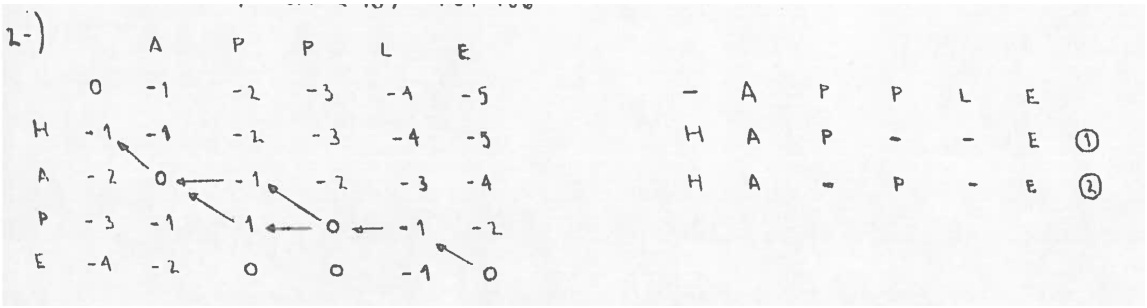
One solution would be to assign different weights to each trait, to optimize

$$Obj = \sum_{i=1}^{n} \sum_{(u,v) \in T} |D_u(i) - D_v(i)| \cdot w(i)$$

where $w(i)$ would be inversely proportional to the variance of trait i.

Good luck!

Question 1:



Question 2:

```
S= CCCC
T = CACACAC

Best alignment with linear gap penalty:
C – C – C – C - C
C A C A C A C A C
```
Score with linear gap penalty = +1
Score with affine gap penalty = -5

Best alignment with affine gap penalty:
```
C C C C C - - - -
C A C A C A C A C
```
Score with linear gap penalty = -3
Score with affine gap penalty = -3

Question 3.

a) The two sequences cannot have lengths that are too different, as this would force us to use two consecutive gaps somewhere. In fact, a solution is only achievable if |m-n| <= min(m,n)+1. That's because |m-n| is the number of gaps that will need to be inserted in the shorter sequence. Those gaps have to be interleaved with nucleotides. Starting and ending with a gap, we get min(m,n)+1.

Example for m=7, n=3
```
AAAAAAA
-A-A-A-
```

b) We first observe that the no-multi-gap alignment is a special case of the pairwise alignment problem with affine gap penalty, when the gap extension penalty is infinite: score(L) = -d - e*(L-1), where e = +inf. (Note: this is slightly different from the affine penalty scoring scheme presented in class, which was cost(L) = a + b*L, but it doesn't make a big difference).

Following the algorithm presented in the Durbin et al. book (Equation 2.16), we introduce three dynamic programming tables: M, Ix, and Iy. M is computed with no change. Since e=+inf, Ix(i,j) and Iy(i,j) reduce to just M(i-1,j)-d and M(i,j-1)-d (respectively). Initialization is done as follows (not detailed in the book):
M(0,0)= 0
M(i,0) = M(j,0) = +inf, for i,j>0

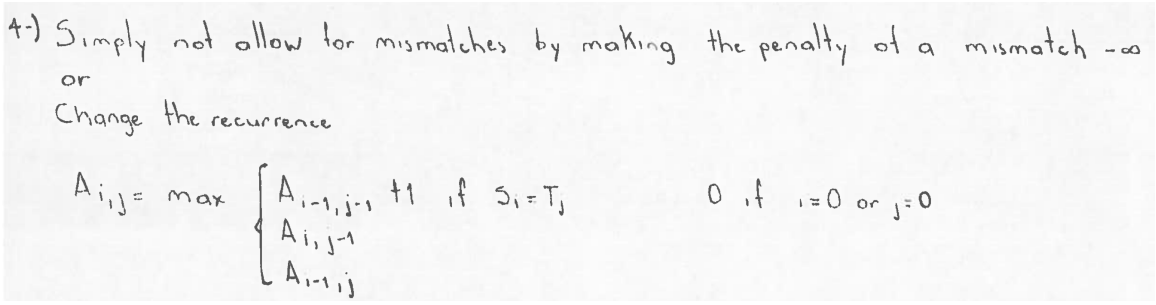
Ix(0,0)=0
Ix(1,0) = -d
Ix(i,0) = -inf for i>1

Iy(0,0)=0
Iy(0,1) = -d
Iy(0,j) = -inf for j>1

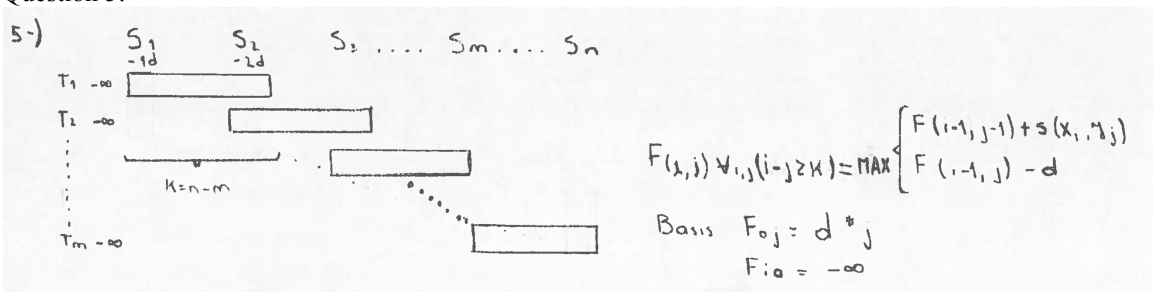
The rest of the algorithm proceeds like described in the book. Trace-back is performed from max(M(m,n), Ix(m,n), Iy(m,n)).

c) I'm too old for this…

Question 4:



Question 5:



Question 6:

#2) S_1: TGC

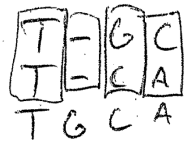
S_2: TCA

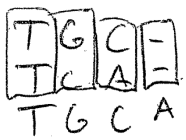
S_3: TGCA

First align S_1 and S_2

the best alignment is
```
TGC   score: -1
TCA
```

then align  with TGCA

the best alignment is  score = 2-4+0+0 = -2

or  score = 2+0+0-4 = -2

and the final score of the alignment is:
```
T-GC
T-CA   score: 3-4-1-1 = -3
TGCA
```

But there exist a better alignment.
```
TGC-
T-CA   score 3-3+3-3 = 0
TGCA
```

Question 7:

```
S = A A A A A A A A A A A A A A A A A A A A
T = A A A A C A A A A C A A A A C A A A A C
```

These two sequences are 80% identical but contain no exact match of size 5.

Bonus Question:

This can be done using the Hirschberg algorithm, which is described here.
https://en.wikipedia.org/wiki/Hirschberg%27s_algorithm