BMI/CS 576 Fall 2015
Midterm Exam

Prof. Colin Dewey

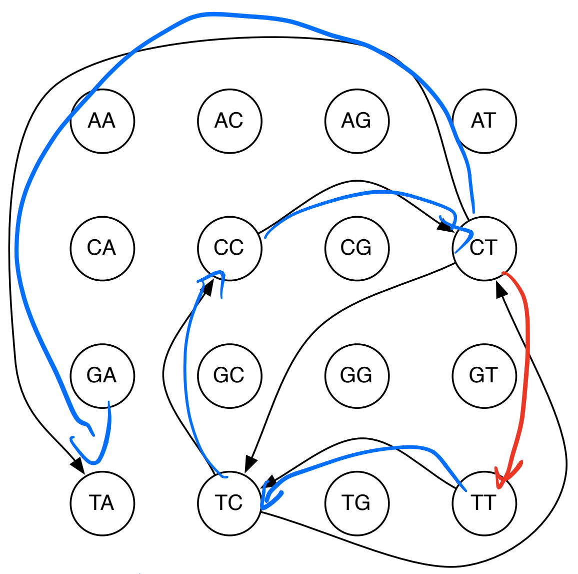
Tuesday, October 27th, 2015 11:00am-12:15pm

Name:_____________KEY_____________

Write your answers on these pages and show your work. You may use the back sides of pages as necessary. Before starting, make sure your exam has every page (numbered 1 through 9). You are allowed 2 (double-sided) pages of notes. Calculators are *not* allowed.

Problem	**Score**	**Max Score**
1	__________	25
2	__________	25
3	__________	25
Total	__________	75

1. (25 points) Suppose that you are given the following k-mer spectrum graph ($k = 3$) for the assembly of a short DNA sequence.



edge

(a) (5 points) Give one Eulerian path through the graph *and* the superstring to which it corresponds.

There are two possible Eulerian paths. The paths and their corresponding superstrings are:
$TT \rightarrow TC \rightarrow CC \rightarrow CT \rightarrow TC \rightarrow CT \rightarrow TA : TTCCTCTA$
$TT \rightarrow TC \rightarrow CT \rightarrow TC \rightarrow CC \rightarrow CT \rightarrow TA : TTCTCCTA$

(b) (5 points) A confused bioinformatician attempts to find a superstring by finding a Hamiltonian path through this graph. Give one Hamiltonian path through this graph, ignoring the vertices that have zero incident edges (i.e., have *indegree* = *outdegree* = 0).

There is one Hamiltonian path in this graph:
$TT \rightarrow TC \rightarrow CC \rightarrow CT \rightarrow TA$

(c) (5 points) Briefly explain why a Hamiltonian path in this graph (again ignoring vertices that have zero incident edges) does not solve the assembly problem.

In the k-mer spectrum approach to assembly, a valid superstring must contain exactly the set of k-mers represented by the edges of the graph. A Hamiltonian path through this graph does not traverse all edges of the graph, and thus the superstring corresponding to the Hamiltonian path does not contain all k-mers represented in the graph.

(d) (5 points) Suppose that we add the edge $CT \to TT$ to this graph. Does the graph still have an Eulerian path? Briefly justify your answer.

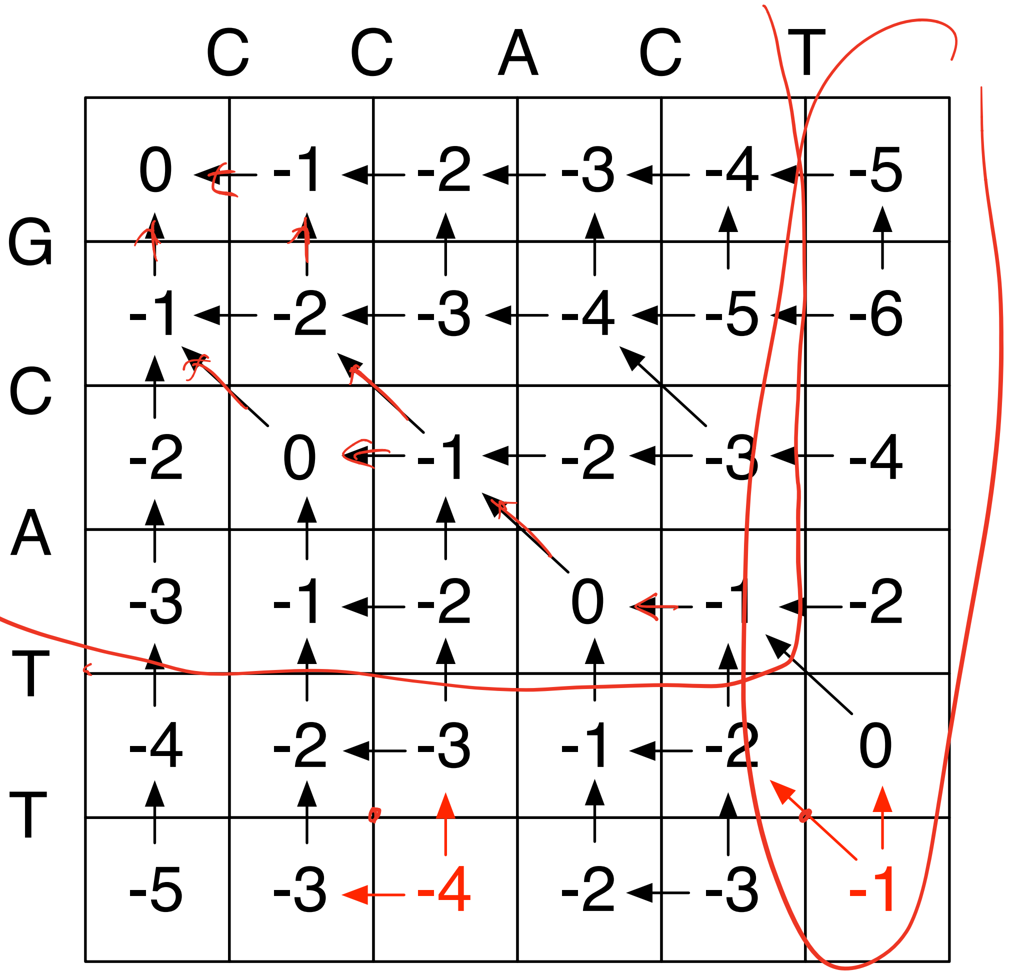
Yes. After the addition of the edge $CT \to TT$, all vertices are balanced except for CT, which has $indegree - outdegree = -1$, and TA, which has $indegree - outdegree = 1$. These properties guarantee the existence of an Eulerian path in this graph.

(e) (5 points) Suppose that we add the edge $TT \to TA$ to this graph. Does the graph still have an Eulerian path? Briefly justify your answer.

No. After the addition of the edge $TT \to TA$, the vertex TA has $indegree - outdegree = 2$, and the vertex TT has $indegree - outdegree = -2$ which precludes the existence of an Eulerian path in the graph.

2. (25 points) Consider the dynamic programming matrix for the global alignment of the sequences `GCATT` and `CCACT` with a linear gap penalty with parameters *match* = 1, *mismatch* = −3, and *space* = −1. Recall that the recurrence for this dynamic programming problem is

$$M[i,j] = \max \begin{cases} M[i-1,j-1] + S(x_i, y_j), & \overset{1,\,-3}{} \\ M[i-1,j] + space, & -1 \\ M[i,j-1] + space & -1 \end{cases}$$



		C	C	A	C	T
	0	-1	-2	-3	-4	-5
G	-1	-2	-3	-4	-5	-6
C	-2	0	-1	-2	-3	-4
A	-3	-1	-2	0	-1	-2
T	-4	-2	-3	-1	-2	0
T	-5	-3	-4	-2	-3	-1

```
- G C A -
C - C A C

G C - A -
- C C A C

G - C A -
- C C A C
```

(a) (5 points) Fill in the *values and traceback pointers* for the empty cells in the dynamic programming matrix.

See the red values and traceback pointers in the matrix above.

(b) (5 points) Show how to reuse this dynamic programming matrix to find an optimal alignment of the sequences `GCA` and `CCAC`. Give both an *optimal alignment* of the sequences and the *optimal score*.

`GCA` is a prefix of `GCATT` and `CCAC` is a prefix of `CCACT`, therefore the matrix consisting of the first four rows and first five columns of the dynamic programming matrix for the original sequences solves the problem for these prefixes.
The optimal score is -1, which is obtained by one of three alignments:

```
GC-A-
-CCAC
```
✓

```
-GCA-
C-CAC
```
✓

```
G-CA-
-CCAC
```
✓

(c) (5 points) Explain how to reuse this dynamic programming matrix to find an optimal alignment of the sequences `GCATTC` and `CCACTG`. You do not need to give an optimal alignment or its score.

`GCATT` is a prefix of `GCATTC` and `CCACT` is a prefix of `CCACTG`, therefore the dynamic programming matrix for the longer sequences is the same as that of the shorter sequences, but with an additional column to the right and an addition row at the bottom. All that is needed to do is compute the entires of the last column and last row of this larger matrix. All other entries will have the same values and traceback pointers as their corresponding entry in the smaller matrix.

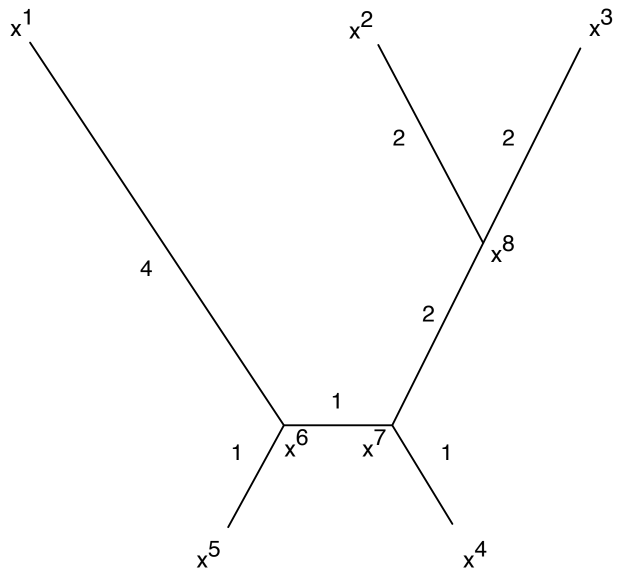
(d) (5 points) Suppose that we modify the global alignment algorithm such that we find the maximum value in the last column of the dynamic programming matrix (not necessarily the lower-right entry) and traceback from that value to the upper-left entry. Briefly describe what task this algorithm solves *in general* (e.g., not specifically for the sequences in this problem).

This algorithm solves the problem of finding the highest scoring global alignment of *some prefix* of the first sequence (which indexes the rows) with the entirety of the second sequence (which indexes the columns). Equivalently, it solves the problem of finding the highest scoring global alignment of the entirety of both sequences with no penalty for insertions in the first sequence at the end of the alignment.

(e) (5 points) Suppose that we are interested in finding the maximum number of matches (aligned characters that are identical) that could be obtained by an alignment of the two sequences. Briefly describe how to modify the values of the scoring parameters such that the dynamic programming algorithm computes this maximum possible number of matches for you.

If we change the scoring parameters to $match = 1$, $mismatch = 0$, and $space = 0$, then the score of an alignment will simply be the number of matches within it. With this scoring function, the dynamic programming algorithm will then find the maximum number of matches possible between two sequences.

3. (25 points) Suppose that the true unrooted phylogenetic tree relating five biological sequences, x^1, x^2, x^3, x^4, and x^5, is the one given below. The sequences x^6, x^7, and x^8 represent sequences of ancestral nodes in this tree.



(a) (5 points) Suppose that you are given the true distances between each pair of sequences. Which pair of leaves would the UPGMA algorithm choose to join first? Justify your answer.

The leaves for x^4 and x^5 would be joined first because they have distance = 3, which is the minimum over all pairwise distances between leaves.

(b) (5 points) Suppose that you are given the true distances between each pair of sequences. Recall that the neighbor-joining algorithm uses "corrected distances" $D_{ij} = d_{ij} - (r_i + r_j)$ where $r_i = \frac{1}{|L|-2} \sum_{m \in L} d_{im}$ and L is the set of nodes remaining to join. Which pair of leaves would the neighbor-joining algorithm choose to join first? Justify your answer. *Hint: to simplify your arithmetic, note that the pair with the minimal value of D_{ij} also has the minimal value of $(|L| - 2) \times D_{ij}$.*

Because we are given the true pairwise distances, which are derived from the true tree, the distances are additive and thus neighbor-joining will reconstruct the true tree. In reconstructing the true tree, starting with the leaf nodes, the first join must either be between leaves x^1 and x^5 or between x^2 and x^3. Thus, we must compare D_{15} with D_{23}, or, equivalently, $(|L|-2)D_{15}$ with $(|L|-2)D_{23}$.

$$\begin{aligned}(|L|-2)D_{15} &= (|L|-2)\times d_{15} - (d_{12}+d_{13}+d_{14}+d_{15}+d_{15}+d_{25}+d_{35}+d_{45})\\ &= 3\times 5-(9+9+6+5+5+6+6+3)\\ &= 15-49\\ &= -34\end{aligned}$$

$$\begin{aligned}(|L|-2)D_{23} &= (|L|-2)\times d_{23} - (d_{12}+d_{23}+d_{24}+d_{25}+d_{13}+d_{23}+d_{34}+d_{35})\\ &= 3\times 4-(9+4+5+6+9+4+5+6)\\ &= 12-48\\ &= -36\end{aligned}$$

Since $(|L|-2)D_{23} < (|L|-2)D_{15}$, then $D_{23} < D_{15}$ and neighbor-joining would select leaves x^2 and x^3 to join first.

(c) (5 points) Do the pairwise distances between the leaves of the true tree obey the properties of an *ultrametric*? Briefly explain why or why not.

No. There are a couple of ways to see this. First, note that if the distances were an ultrametric, then UPGMA would give the correct tree, but we saw in part (a) that UPGMA would join x^4 with x^5 first, which is not compatible with the true tree. Thus, by contradiction, the distances cannot be an ultrametric. Alternatively, we can examine the distances between all triplets of leaves and find that for leaves x^1, x^4, and x^5, the distances are $d_{14}=6$, $d_{15}=5$, and $d_{45}=3$, which are all different, violating the definition of an ultrametric.

(d) (5 points) Suppose that the sequences at the leaves are $x^1 = A$, $x^2 = C$, $x^3 = C$, $x^4 = A$, and $x^5 = G$. Give the minimum unweighted parsimony cost of this tree and an assignment of sequences to the ancestral nodes that achieves this cost.

The minimum unweighted parsimony cost is 2 and an assignment to the internal nodes that achieves this cost is $x^6 = A$, $x^7 = A$, $x^8 = C$.
One can verify this via Fitch's algorithm. For example, suppose we place a root node x^9 on the edge in between x^7 and x^8. Then we have that

$$\begin{aligned} R_6 &= \{A, G\} \\ R_7 &= \{A\} \\ R_8 &= \{C\} \\ R_9 &= \{A, C\} \end{aligned}$$

and regardless of the assignment to the root node, the other internal nodes must be assigned as $x^6 = A$, $x^7 = A$, $x^8 = C$.

(e) (5 points) Suppose that x^2 is the *outgroup*. Where is the root of this tree located?

The root is located somewhere along the edge between nodes x^2 and x^8.