CSCI-GA.3520-001: Honors Analysis of Algorithms

Final Exam, Dec 21, 2021, 9:30am-1:30pm

- This is a four hour exam. There are six questions, worth 10 points each. Answer all questions and all their subparts.
- You can **only** use course notes. **No** access is allowed to textbooks, any other written or published materials, any online materials, and any other materials stored on devices.
- In the past years, to pass this exam, one needed to answer 3 or 4 problems well, instead of answering all the problems poorly.
- You must submit separate answers for separate questions.
- Read the questions carefully. Keep your answers legible, and brief but precise. Assume standard results and algorithms (i.e., those taught or referred to in class or homeworks).
- You must prove correctness of your algorithm and prove its time bound unless stated otherwise. The algorithm can be written in plain English (preferred) or as a pseudo-code.

Best of luck!

Problem 1

A gene is a string over an alphabet $\Sigma = \{A, C, G, T\}$. Let $X[1, \ldots, n]$ denote a new gene that has been discovered. You wish to determine how closely matched it is with another gene $Y[1, \ldots, m]$. A simple way to do it is by computing the cost of transforming X to Y. There are three operations:

- $\delta_{\mathsf{ins}} : \Sigma \to \mathbb{R}^+ \cup \{0\}$ which gives the cost of inserting a protein from Y into X.
- $\delta_{\mathsf{del}} : \Sigma \to \mathbb{R}^+ \cup \{0\}$ which gives the cost of deleting a protein from X.
- $\delta_{\mathsf{sub}} : \Sigma \times \Sigma \to \mathbb{R}^+ \cup \{0\}$ which gives the cost of substituting a protein in X with a protein from Y.

You goal is to find the smallest cost way to transform X into Y, by performing operations as above, starting from the end of X and Y. A convenient way to visualize such "matching" is by drawing an *alignment* between X and Y. For example, if $X = [A, T]$ and let $Y = [G, C]$, a possible alignment is:

A	$*$	T
$*$	G	C

This alignment indicates that to go from X to Y, you will substitute T in X with C in Y (indicated by the alignment of X's T with Y's C), then insert G from Y to X (indicated by the $*$ in X aligned with G), and finally delete A from X (indicated by A's alignment with $*$). The cost is given by $\delta_{\mathsf{del}}(A) + \delta_{\mathsf{ins}}(G) + \delta_{\mathsf{sub}}(T, C)$. As you can see, the top (resp. bottom) row of any alignment is the string X (resp. Y), with some $*$'s inserted, and each column corresponds to one of the three operations mentioned.

In all these parts, you will use a memoization matrix of dimension $m+1 \times n+1$ called c. where $c[i, j]$ will contain the minimum cost of alignment of substrings $X[1, \ldots, i]$ and $Y[1, \ldots, j]$, with cost function defined by δ.

(a) (3 points) Present a high-level specification based on dynamic programming to solve this problem. You do not need to present an algorithm to compute the algorithm, but instead merely present a recursive formula, with appropriate base cases of $c[0, j]$ and $c[i, 0]$ (after assuming that $c[0, 0] = 0$). An efficient implementation of your algorithm should run in time $O(mn)$.

One can also define the operation of *transposition*, which inherently is the sum of two consecutive substitutions, but with the order of characters in one of the string flipped. For example, if you have $X[1, \ldots, i]$ and $Y[1, \ldots, j]$, transposition will substitute $X[i]$ for $Y[j-1]$, and substitute $X[i-1]$ for $Y[j]$. No further operations can be performed on these indices after the transposition. The cost of such transposition is simply the cost of the two corresponding substitutions plus 1.

In regards to the previous example, this move would allow the following alignment, of cost $\delta_{\mathsf{sub}}(A, C) + \delta_{\mathsf{sub}}(T, G) + 1$.

A	T
C	G

(b) (3 points) Modify your recursive formula from the previous part to account for this transposition. An efficient implementation of your algorithm should run in time $O(mn)$. For simplicity, write your formula for $c[i, j]$ assuming $i, j \geqslant 2$ (otherwise, the same formula as in part (a) works, as no transposition is possible).

(c) (4 points) In part (b), we looked at a recursive formula which supported transposition of two characters which can also be considered as a reversal of two adjacent characters. Generalize the above part to support reversals of arbitrary length, i.e., $X[i], \ldots, X[i+k]$ is now substituted with $Y[j+k], \ldots, Y[j]$ respectively for any valid i, j, k. As before, after invoking the reverse operation no further operations can be performed on any of these indices, and the cost is the sum of $(k+1)$ corresponding substitutions plus k. (Thus, $k = 1$ is the case of transposition, and $k = 0$ is the case of simple substitution.) Notice, the value of k is allowed to change as you proceed with your matching, and there is no upper bound on k.

Also state and justify the new running time of an efficient implementation of the new recursive formula. For simplicity of exposition, do not state the base case, and assume that $c[i, j] = \infty$ when $i < 0$ or $j < 0$.

Problem 2

Let $G = (V, E)$ be a directed, weighted graph with non-negative edge weights. In addition to a weight, an edge is also assigned a color. The color is defined by the function $c : E \to \{red, green, blue\}$. Finally, you are also given a start vertex s.

(a) (3 points) For each node v, determine the shortest path that starts from s and ends at v such that the path begins with a *red* edge and ends with a *blue* edge. You will do this by giving a reduction to the single source, shortest path problem and then run Dijkstra's exactly once. State and justify the running time of your algorithm.

(b) (3 points) For each node v, determine the shortest path that starts from s and ends at v such that the colors on the edges of the path alternate as *red*, *blue*, *green* in that order, and beginning with *red* and ending with *green*. We will call such a path as RBG-path. You will do this by giving a reduction to the single source, shortest path problem then run Dijkstra's exactly once. State and justify the running time of your algorithm.

(c) (4 points) Modify your solution in part (b) to now computed the shortest relaxed-RBG path. A relaxed-RBG path is one which requires you to alternate colors between *red*, *blue*, *green* in that order, but you may choose to begin the alternation at any color, not just at the *red* edge. Similarly, you may end having traversed any color, not just *green*. Solve this problem by giving a reduction to the single source, shortest path problem then run Dijkstra's exactly once. State and justify the running time of your algorithm.

Problem 3

An undirected graph $G = (V, E)$ with $|V| = n$ vertices, where n is divisible by 5, is said to be *5-balanced-colorable* if there is a partition of the set of vertices into five disjoint sets $V_1, \ldots, V_5$ such that each V_i is of cardinality $\frac{n}{5}$ and forms an independent set, i.e., there are no edges between vertices of V_i.

1. (2 points) Show an algorithm that given a graph $G = (V, E)$, decides if it is 5-balanced-colorable in time $O(5^n \text{poly}(n))$, where $\text{poly}(n)$ denotes any polynomial in n. Justify the correctness of the algorithm and analyze its running time.

2. (2 points) For an integer $k \geqslant 0$, define $\text{Ham}(k)$ as the number of $1's$ in the binary (base-2) representation of k. For example, $\text{Ham}(5) = 2$ and $\text{Ham}(15) = 4$. Prove that for all integers $k, \ell \geqslant 0$, it holds that $\text{Ham}(k + \ell) \leqslant \text{Ham}(k) + \text{Ham}(\ell)$.

3. (2 points) For a set S of positive integers, define $f(S)$ as the integer whose binary representation has $1's$ in the positions indicated by S. For example, $f(\{1, 4, 5\}) = 1+8+16 = 25$. Show that if $S_1, \ldots, S_5 \subseteq \{1, \ldots, n\}$ are all of cardinality $\frac{n}{5}$, then their union equals $\{1, \ldots, n\}$ if and only if $f(S_1) + \cdots + f(S_5) = 2^n - 1$. (You can assume the result in the previous item).

4. (4 points) Show an algorithm that given a graph $G = (V, E)$, decides if it has a 5-balanced-colorable in time $O(2^n \text{poly}(n))$, where $\text{poly}(n)$ denotes any polynomial in n. Justify the correctness of the algorithm and analyze its running time. (You can assume the result in the previous item).

   Hint: Identify V with $\{1, \ldots, n\}$. First enumerate all integers $f(S)$ such that $S \subseteq \{1, \ldots, n\} = V$ is an independent set.

Problem 4

1. (4 points) Show an algorithm that given as input a sequence of integers $(a_1, \ldots, a_n)$, outputs a list $(r_1, \ldots, r_n)$ where r_i is the number of indices $j \in \{i + 1, \ldots, n\}$ satisfying that $a_j > a_i$. Prove your algorithm's correctness and analyze its running time. For full credit, your algorithm should run in time $O(n \log n)$.

2. (6 points) Given a sequence of integers $(a_1, \ldots, a_n)$, a *increasing triple* is a triple (i, j, k) such that $1 \leqslant i < j < k \leqslant n$ and $a_i < a_j < a_k$. Show an algorithm that counts the number of increasing triple in an input sequence. Prove your algorithm's correctness and analyze its running time. For full credit, your algorithm should run in time $O(n \log n)$.

Problem 5

Let $G(U, V, E)$ be a random bipartite graph constructed as follows. Let U, V be the sets of left and right side vertices respectively with $|U| = |V| = n$. For every pair $(u, v), u \in U, v \in V$, we include the pair (u, v) in the edge set E with probability p, independently for all pairs (u, v). Here $0 < p < 1$ is a parameter to be determined (it depends on n).

For any subset $X \subseteq U$, $1 \leqslant |X| \leqslant n - 1$, let $\Gamma(X) \subseteq V$ denote the set of its neighbors, i.e.

$$\Gamma(X) = \{v \mid v \in V \text{ such that there is some } u \in X \text{ such that } (u, v) \in E\}.$$

This problem seeks to determine the value of p, as small as possible, so that the graph $G(U, V, E)$ has the following property with positive probability: for **every** subset $X \subseteq U,\ 1 \leqslant |X| \leqslant n - 1$, it holds that $|\Gamma(X)| > |X|$.

1. For fixed sets $X \subseteq U,\ Y \subseteq V,\ |X| = |Y| = r,\ 1 \leqslant r \leqslant n - 1$, let $\mathcal{E}_{X,Y}$ denote the event that $\Gamma(X) \subseteq Y$. Determine (just write the answer)

$$\Pr[\mathcal{E}_{X,Y}].$$

2. Let $\mathcal{E}$ be the event that for **some** $X \subseteq U,\ Y \subseteq V,\ |X| = |Y| = r,\ 1 \leqslant r \leqslant n - 1$, we have $\Gamma(X) \subseteq Y$. Express the event $\mathcal{E}$ in terms of the events $\mathcal{E}_{X,Y}$. Just write the answer.

3. Give an upper bound on $\Pr[\mathcal{E}]$. It is OK to leave the answer in the form of a mathematical expression.

4. What is the smallest value of p (that you can think of) such that $\Pr[\mathcal{E}] \leqslant \frac{1}{10}$? It may be a function of n. For full credit, your answer should be correct up to a constant factor. Give a justification, but no need to formally prove an upper or lower bound on the value of p.

   Hint: Consider the case $r = 1$. Also, are the cases r and $n - r$ symmetric? Why? If so, one may assume that $r \leqslant \frac{n}{2}$?

5. Assume now that the event $\overline{\mathcal{E}}$ holds, i.e. that the bipartite graph $G(U, V, E)$ has the property that for **every** $X \subseteq U,\ 1 \leqslant |X| \leqslant n - 1$, $|\Gamma(X)| > |X|$. Do you recall a theorem that holds under this condition and guarantees existence of a certain structure in the graph?

Problem 6

Recall that an instance φ of 3SAT consists of n Boolean variables $x_1, \ldots, x_n$ and m clauses $C_1, \ldots, C_m$ where every clause is of the type $\ell_i \vee \ell_j \vee \ell_k$, i, j, k are distinct, and ℓ_i is a literal, either x_i or $\overline{x}_i$. A Boolean assignment to the variables is called satisfying if it satisfies every clause.

An instance φ of 3SAT is called tri-partite if the set of variables V is partitioned into three disjoint sets X, Y, Z (so that $V = X \cup Y \cup Z$) and every clause contains one literal from X, one literal from Y, and one literal from Z. Let

$$\text{TRI PARTITE 3SAT} = \{\ \varphi \mid \varphi \text{ is a tri-partite instance that has a satisfying assignment}\}.$$

Show that TRI PARTITE 3SAT is NP-complete.