Winter 2024 **Algorithm Design and Analysis: Endsem (Sec-A)** **Max: 80 Marks**
May 4, 2024 **180 minutes**

Roll No: _____ **Name:** _____

Question	1	2	3	4	5	6	7	8	9	Total
Marks										

Instructions: *Please write solutions independent of each other. This is a closed book test. So, you can not use books or lecture notes. Please note that your solution must fit in the space provided. The extra sheet will be provided* only *for roughwork. So try to be precise and brief. Meaningless blabber fetches negative credit.*

All The Best!

---

1. **(8 Marks)** For each of the statements, just write whether the statements are True or False.

   (a) Let A be an array of n numbers such that every element in A are from the set $\{1, 2, \ldots, 3n\}$. Then, there does not exist any algorithm without using hashmap/hashtable that can find if there are two distinct indices i and j such that $A[i] = A[j]$ in $O(n)$-time.

   (b) Let L_1 be a problem that is in NP and L_2 be a problem such that there is a valid polynomial-time reduction that transforms an instance of L_1 into an instance of L_2. If L_2 can be solved in polynomial-time, then any problem that is in NP can be solved in polynomial-time.

   (c) Given a directed acyclic graph with positive edge weights and two vertices s and t. If a longest path (in terms of total edge weights) P from s to t passes through u, then the subpath of P that starts from u and ends at t is also a longest path betwen u and t.

   (d) Let G be an undirected graph with n vertices and m edges. Then, there is an algorithm that can check whether there exists a cycle in G in $O(n)$-time and the running time is independent of m.

2. **(7 Marks)** Given a graph G, a vertex subset $S \subseteq V(G)$ is called *independent set* if for every $u, v \in S$, $(u, v) \notin E(G)$. Consider the problem where the input is a path graph $G = (v_1, \ldots, v_n)$ with each vertex given positive weights. The edges are (v_i, v_{i+1}) for every $1 \leq i \leq n-1$. The objective is to find the maximum possible weight of an independent set in G.

   Here is a partial algorithm using divide and conquer approach.

   Function $\mathsf{MAXIND}(G, n)$.

   (i) If $n \leq 5$, then use brute-force approach (looking at all possible subsets) and return the total weight of a maximum weight independent set.

   (ii) $r \leftarrow \lfloor n/2 \rfloor$.

   (ii) $G_A \leftarrow$ the path with the first $r-1$ vertices, i.e. $(v_1, \ldots, v_{r-1})$.

   (iii) $G_B \leftarrow$ the path $(v_{r+2}, \ldots, v_n)$.

   (iv) $G_C \leftarrow$ the path $(v_{r+1}, \ldots, v_n)$.

   (v) $G_D \leftarrow$ the path $(v_1, \ldots, v_r)$.

   (vi) $I_A \leftarrow \mathsf{MAXIND}(G_A, r-1)$.

   (vii) $I_B \leftarrow \mathsf{MAXIND}(G_B, n-r-1)$.

   (viii) $I_C \leftarrow \mathsf{MAXIND}(G_C, n-r)$.

   (ix) (fill up here)

   (a) What is the above algorithm's recurrence and time complexity? Give a proper explanation of your answer.

3. **(10 Marks)** Given an undirected graph $G = (V, E)$ with n vertices, m edges, and two specified vertices s and t. A shortest path between s and t is a path with minimum number of edges. Design an $O(n + m)$-time algorithm that computes the number of shortest paths between s and t. Explain the running time of your algorithm. Your algorithm does not have to output all the shortest paths between s and t, just the number of shortest paths between s and t suffices.

   (if your algorithm description goes past this page, then also it is okay, but please make sure to properly mention where you write the time complexity of your algorithm)

   - A brief description of your algorithm.

- Explanation of running time of your algorithm.

4. **(15 Marks)** Let G be a directed acyclic graph (DAG) whose vertices have some labels from some fixed alphabet $\Sigma = \{a, b\}$, and let $S[1, \ldots, r]$ be a string of length r over Σ. The number of vertices of G is n and the number of edges is m. Design a poly$(n + m + r)$-time algorithm to find the length of a longest path whose label is a subsequence of S. Explain the time complexity of your algorithm as well.

   (**Comment:** You can assume that you are given an array Label$[1, \ldots, n]$ for every vertex. If your algorithm description goes past this page, then also it is okay, but please make sure to properly mention where you write the time complexity of your algorithm)

   - A brief description of your algorithm (if your algorithm uses dynamic programming, then write down all the steps, i.e., subproblem definition, recurrence of subproblem, the specific subproblem solving the actual problem, algorithm description, and running time explanation).

- Explanation of the running time of your algorithm.

5. **(8 Marks)** A graph H is said to be a clique if every pair of vertices of H are adjacent to each other. The Cluster Vertex Deletion problem is defined as follows.

   - **Input:** An undirected graph $G = (V, E)$ and an integer k.
   - **Question:** Is there $S \subseteq V(G)$ such that $|S| \leq k$ and every connected component of $G - S$ is a clique?

   Prove that Cluster Vertex Deletion is in NP.

   - Certificate or Proposed Solution of the Problem:

   - A text descrition of the algorithm:

6. **(7 Marks)** Two problems 3-Colorability and 5-Colorability are defined as follows.

3-Colorability

- **Input:** An undirected graph $G = (V, E)$.
- Is there a coloring function $\lambda : V(G) \to \{1, 2, 3\}$ such that for every edge $(u, v) \in E(G)$, $\lambda(u) \neq \lambda(v)$?

5-Colorability

- **Input:** An undirected graph $G = (V, E)$.
- Is there a coloring function $\lambda : V(G) \to \{1, 2, 3, 4, 5\}$ such that for every edge $(u, v) \in E(G)$, $\lambda(u) \neq \lambda(v)$?

Consider the following algorithm to suggest a **polynomial-time reduction** from 3-Colorability to 5-Colorability.

Reduction: The main steps are as follows.
(i) Let G be an instance of 3-Colorability and $V(G) = \{u_1, \ldots, u_n\}$.

(ii) For every $i \in \{1, 2, \ldots, n\}$, create a vertex v_i.

(iii) For every pair of vertices (u_i, u_j), if $(u_i, u_j) \in E(G)$, then make $(v_i, v_j) \in E(G)$.

(iv) For every $i \in \{1, \ldots, n-1\}$, add edges (u_i, v_i) and (u_i, v_{i+1}).

(v) Then, add edges (u_n, v_n) and (u_n, v_{n+1}).

(vi) The output graph is H.

Either prove that this algorithm is a polynomial-time reduction from 3-Colorability to 5-Colorability, or give a counter example to disprove that this is not a valid reduction.

(if you give a counter example, please clearly explain in the example why this reduction is not a valid reduction in that example)

7. **(8 Marks)** If there is a set S of n distinct elements, then given an integer k, design a polynomial-time algorithm that counts the number of possible tuples (A, x) such that $A \subseteq S$, $x \in A$, and $|A| = k$.
(If your algorithm uses subroutines that computes $r!$ for any r, then your solution will be awarded zero marks)

   - A brief description of your algorithm:

- Explanation of the running time of your algorithm:

8. **(10 Marks)** The CS department of a university has a flexible curriculum with a complicated set of graduation requirements. The department offers n courses and there are m requirements. A student has to satisfy all of the m requirements to graudate. Each requirement specifies a subset A of courses and the number of courses that must be taken from the subset. The subsets for different requirements may overlap but each course can be used to satisfy at most one requirement.
   Design a poly$(n+m)$-time algorithm that determines whether a student can graduate.
   Example: There are 5 courses x_1, x_2, x_3, x_4, x_5 and $m = 2$. The requirements are **(i)** at least two courses must be taken from $\{x_1, x_2, x_3\}$, and **(ii)** at least two courses must be taken from $\{x_3, x_4, x_5\}$. A student can graudate with courses $\{x_1, x_2, x_3, x_4\}$ but cannot graduate with courses $\{x_2, x_3, x_4\}$.

(answer of Q8 continued)

9. **(7 Marks)** Consider the following flow network. The numbers denote the capacities of the edges. Illustrate the execution of Ford-Fulkerson's Algorithm to compute a maximum flow from s to t in this network. Your illustration must show step by step execution of the Ford-Fulkerson's Algorithm, particularly the changes in the residual graph, and the flow value in every edge at each step.

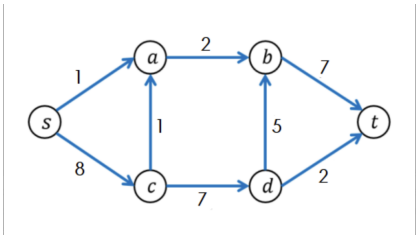

Figure 1: Flow network