INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR
Final-Autumn Semester 2018-19

Date of Examination: 20 November 2018
Session (FN/AN): AN
Duration: 3 hours
Full Marks: 100
Subject No: CS60007
Subject: ALGORITHM DESIGN AND ANALYSIS
Department/Center/School: COMPUTER SCIENCE AND ENGINEERING
Specific charts, graph paper, log book, calculator, etc., required: NO
Special instruction (if any): NA

Answer question 8 and any four of the first seven questions. Prove correctness and derive runtime of all your algorithms unless specified otherwise.

1. An independent set of an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a subset $\mathcal{W} \subseteq \mathcal{V}$ such that there does not exist any edge $e = \{u, v\} \in \mathcal{E}$ with both end points in $\mathcal{W}$ (that is, $|e \cap \mathcal{W}| \leqslant 1$). An undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is called an interval graph if every vertex $v \in \mathcal{V}$ can be associated with some interval $\mathcal{I}_v = [a, b)$ in $\mathbb{R}$ such that there is an edge e between u and v if and only if $\mathcal{I}_u \cap \mathcal{I}_v \neq \emptyset$. Design an algorithm which, given an interval graph $\mathcal{G}$ with corresponding interval I_v for every vertex $v \in \mathcal{V}$, computes the size of a maximum independent set of $\mathcal{G}$. Prove correctness of your algorithm and derive its runtime.

[20 Marks]

2. Suppose we have a stick of length n for some positive integer n. Let $p_\ell, \ell \in \{1, 2, ..., n\}$ denote the utility of a stick of length ℓ. Design a dynamic programming based algorithm which, given n and $p_1, \ldots, p_n$ as inputs, finds a way to break a stick of length n into pieces such that the total utility (the sum of utilities of the pieces) is maximized.

[20 Marks]

3. Describe an $\mathcal{O}(n)$-time algorithm that, given a set $\mathcal{S}$ of n distinct numbers and a positive integer $k \leqslant n$, determines the k numbers in $\mathcal{S}$ that are closest to the median of $\mathcal{S}$. You may assume that there exists an $\mathcal{O}(n)$-time algorithm which, given a set of n numbers and an integer $i \in \{1, 2, ..., n\}$, finds the i-th smallest number.

[20 Marks]

4. Let $\mathcal{G}$ be a connected weighted undirected graph with distinct edge weights. Prove that the following algorithm finds an MST of $\mathcal{G}$. "Initially all the edges are unmarked. Let e be an unmarked edge with highest weight. If removing e does not disconnect the graph, remove it; otherwise mark it. Repeat this process until the graph consists of marked edges only. Then output the current graph as an MST of the original graph."

[20 Marks]

5. (i) Construct directed graphs on n vertices with non-negative capacities on the edges and special vertices s and t which have:

(a) Exponentially (as function of n) many minimum s-t cuts and a unique maximum s-t flow.

[4 Marks]

(b) More than one maximum s-t flows and a unique minimum s-t cut.

[3 Marks]

(c) More than one maximum s-t flows and more than one minimum s-t cuts.

[3 Marks]

(ii) König's Theorem states that the size of a maximum matching in a bipartite graph is the same as the size of a minimum vertex cover. Prove König's Theorem using flow arguments.

[10 Marks]

6. In the PARTITION problem, the input is a set $\mathcal{S}$ of positive integers and we need to compute if there exists a subset $\mathcal{A} \subseteq \mathcal{S}$ such that $\sum_{x\in\mathcal{A}} x = \sum_{y\in\mathcal{S}\setminus\mathcal{A}} y$. In the KNAPSACK problem, the input is a set $\mathcal{O} = \{o_i : i \in \{1, 2, ..., n\}\}$ of n objects; the weight and value of the object o_i are w_i and v_i respectively (w_i and v_i both are positive integers). The input also consists of two integers $\mathcal{W}$ and $\mathcal{V}$. We need to compute if there exists a subset $J \subseteq \{1, 2, ..., n\}$ such that $\sum_{j\in J} w_j \leqslant \mathcal{W}$ and $\sum_{j\in J} v_j \geqslant \mathcal{V}$. Prove that the PARTITION problem reduces to the KNAPSACK problem in polynomial time.

[20 Marks]

7. In the MAX-CUT problem, the input is an unweighted and undirected graph and we need to find a cut of maximum size. The size of a cut is the number of edges crossing the cut. Design a $\frac{1}{2}$ factor deterministic polynomial time approximation algorithm for the MAX-CUT problem.

[20 Marks]

8. Let $\mathcal{G}$ be an undirected, weighted graph, and s and t be any two vertices. Suppose all the edge weights are positive real numbers except one edge $(u, v) \in \mathcal{G}$; the weight of the edge (u, v) is some negative real number. Design a $\mathcal{O}(m + n \log n)$ time algorithm to find the distance of t from s where n and m denote the number of vertices and edges in $\mathcal{G}$ respectively. Feel free to use any algorithm covered in the class to design your algorithm. For this question, you do not need to prove correctness and running time of any algorithm covered in the class.

[20 Marks]