CSE 551 Foundations of Algorithms
Makeup Final, Spring 2019
Closed Books, Closed Notes
Time: 1 hour 40 minutes
Answer any five questions
Each question carries 20 pts.

Problem 1: A magnetic tape contains n programs of length $l_1, ..., l_n$. We know how often each program is used: a fraction p_i of requests to load a program concern program i, $(1 \leq i \leq n)$. (This of course implies $\sum_{i=1}^{n} p_i = 1$) Information is recorded along the tape at constant density, and the speed of the tape is also constant. Each time a program has been loaded, the tape is rewound to the begining.

If the programs are held in the order $i_1, ..., i_n$, the average time required to load a program is

$$T = c \sum_{j=1}^{n} \left[p_{i_j} \sum_{k=1}^{j} l_{i_k} \right],$$

where the constant c depends on the recording density and the speed of the drive. Our objective is to minimize T.

i) If the programs are stored in order of increasing values of l_i, will it minimize T ? If your answer is *yes*, then prove it. If your answer is *no*, then provide an example where the strategy fails to minimize T.

ii) If the programs are stored in order of decreasing values of p_i, will it minimize T ? If your answer is *yes*, then prove it. If your answer is *no*, then provide an example where the strategy fails to minimize T.

iii) If the programs are stored in order of decreasing values of p_i/l_i, will it minimize T ? If your answer is *yes*, then prove it. If your answer is *no*, then provide an example where the strategy fails to minimize T.

Problem 2 Consider a job scheduling problem with n jobs $J_i, 1 \leq i \leq n$. An execution time $t_i, 1 \leq i \leq n$ and a deadline $d_i, 1 \leq i \leq n$ is associated with each job. A schedule is said to be *best-possible*, if there exists no other schedule where fewer number of the jobs violate their respective deadlines.

(i) Develop an algorithm to find the *best-possible* schedule.

(ii) Analyze your algorithm to establish its correctness (i.e., your algorithm indeed finds a solution where *fewest* number of jobs violate their deadlines).

(iii) Analyze your algorithm to find its computational complexity.

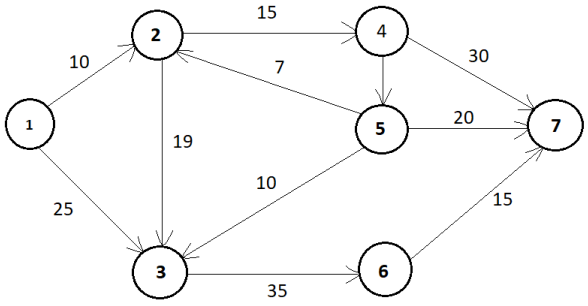

Figure 1: Flow Network

Problem 3: Suppose that the capacity of a cut $C(S : S')$ in a network, is defined in the following two ways:

$$(i)\ C(S : S') = \sum_{e \in (S:S')} C(e) \quad \texttt{and} \quad (ii)\ C(S : S') = \sum_{e \in (S:S')} C(e) \ - \sum_{e \in (S':S)} C(e)$$

where C(e) represents the capacity of the edge e. Will maximum flow from a source node s to a destination node t be equal to capacity of the minimum cut in case definition (i) is used? Will it be true if definition (ii) is used? If your answer "yes" then prove the assertion. If your answer "no" then provide a counterexample. Using Ford-Fulkerson algorithm compute the maximum flow from node 1 to 7 in the network shown in Fig 1. Find the minimum cut (using the standard definition) in this network. Show all your work.

Problem 4: Define (i) Hamiltonian Cycle Problem and (ii) Traveling Salesman Problem. If Hamiltonian Cycle Problem is NP-complete, can you transform it to prove that the Traveling Salesman Problem is also NP-complete? If your answer is "yes", then show how it can be done. Otherwise, explain why it cannot be done.

Suppose the following algorithnm is used to the Traveling Salesman Problem. Starting from city 1 do the following: Visit the nearest city if it hasn't been visited before. Is this algorithm going to find the optimal Traveling Salesman tour? If your answer is "yes", then prove the claim. Otherwise, provide an example to show that the algorithm will fail to find the optimal tour.

Problem 5: Find the smallest and the second smallest of n elements with $n + \lceil log\ n \rceil - 2$ comparisons. Explain your algorithm and then analyze it's complexity to show that it indeed finds the smallest and the second smallest of n elements with $n + \lceil log\ n \rceil - 2$ comparisons.

Problem 6: In class we discussed the maximum number of compatible activity selection problem, where each activity had a specific start time and a specific finish time. Two acitities were considered *compatible* if their corresponding intervals (between start and finsish times) did not overlap. The goal of the problem was to select the largest set of mutually compatible activities. Consider a version of that problem, where each acitivity, in addition to start and a finish times, has a *profit* associate with it. The profit can only be made if the corresponding activity is selected. In this new version of the problem we need to find the set of mutually compatible activies that *maximizes* the profit. Design an algorithm for the problem and analyze it's complexity. Show all your work.