CSE 102 — Spring 2021 – Advanced Homework 4

Greedy Algorithms
(upto 10 points)

If the problem requires a proof, state clearly whether the (i) proof uses "Greedy Stays Ahead" method of *Activity Selection Problem*, or (ii) proof uses "Exchange Argument" method of *Fractional Knapsack Problem*, or (iii) a proof method different than these two mentioned above. In any case, a rigorous proof, with the level of detail used in the lecture for the two problems stated above, is expected. In all cases where greedy algorithm works, present one or two concrete examples to illustrate how the algorithm works.

Attempt **ONE** of the following problems:

1. **Job Scheduling with Deadlines:** (5 points) We have a set of n jobs to execute, each of which takes unit time. each job has a deadline d_i. At any time $T = 1, 2, ...$, we can execute exactly one job. Job i earns a profit $g_i > 0$ iff it is executed no later than time d_i. The goal is to find the subset of jobs that maximizes profit.

   Describe a greedy algorithm. Present an implementation plan or pseudo-code with time complexity $O(n^2)$ and explain clearly why the complexity is $O(n^2)$.

2. **Disk Storage Problem:** (7 points) Let $P_1, , ...P_n$ be n programs to be stored on a disk. Program P_i required s_i megabytes of storage and the capacity of the disk is D megabytes, where $D < \sum_{i=1}^{n} s_i$.

   (a) We want to maximize the number of programs stored on the disk. Prove or give a counter-example that one can use a greedy algorithm that selects the programs in order of non-decreasing s_i.

   (b) We want to use as much of the capacity of the disk as possible. Prove of give a counter-example that one can use a greedy algorithm that selects programs in order of non-increasing s_i.

3. **Scheduling to Minimize Average Completion Time;** (7 points) Problem 16.2 (Page 447) of CLRS, 3rd edition.

4. **Off-line Caching:** (7 points) Problem 16.5 on Page 449 of CLRS, 3rd Edition.

5. **Magnetic Tape Access Problem:** (10 points) Enclosed on the next page.

6. **Greedy Approximation to 0-1 Knapsack Problem:** (up to 10 points: simple proof without external resources) Consider the 0-1 knapsack problem: Given n items with weights $w_1, ..., w_n$ and value $v_1, ...v_n$ and a total weight of W, where each w_i, v_i and B are positive integers, find a subset S of items that a thief would like to steal so that the total weight is smaller than W and the total value is maximum.

   The thief just took CSE 102, bared passed the class but knows that a greedy solution does not work. He/she needs your help.

   Design a greedy algorithm which does not necessarily provide the best solution but provides a solution which is better than at least half of the best solution. In other words, if the best solution allows stealing the items of total value V, your algorithm should guarantee that the thief will be able to steal goods worth at least of total value $\frac{V}{2}$. [Hint: This algorithm is known as 2-approximation algorithm. You

choose the best of 2 greedy algorithms. The first greedy algorithm is same as in fractional knapsack problem. The second greedy algorithm is ....]

Prove that your algorithm works so that the thief is willing to share at least half of the loot with you.

Problem 6.22. Let $P_1, P_2, \ldots, P_n$ be n programs to be stored on a tape. Program P_i requires s_i kilobytes of storage; the tape is long enough to hold all the programs. We know how often each program is used: a fraction π_i of requests concern program i (and so $\sum_{i=1}^{n} \pi_i = 1$). Information is recorded along the tape at constant density, and the speed of the tape drive is also constant. After a program is loaded, the tape is rewound to the beginning. If the programs are held in the order $i_1, i_2, \ldots, i_n$ the average time required to load a program is therefore

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

where the constant c depends on the recording density and the speed of the drive. We want to minimize $\bar{T}$ using a greedy algorithm. Prove or give a counter-example for each of the following: we can select the programs (a) in order of nondecreasing s_i; (b) in order of nonincreasing π_i; (c) in order of nonincreasing π_i / s_i.