Assignment 4
COSC 3P03: Design and Analysis of Algorithms
Winter 2016

Due: April 1, Friday, 5:00 PM.

1. (10) For the chained matrix multiplication problem where we are supposed to find an optimal order by which to calculate $M = M_1 \times M_2 \times \cdots \times M_n$, we know that we can use the technique of dynamic programming to solve the problem. For each of the following suggested greedy ideas, provide a counter example where the greedy solution does not work:
(a) First multiply the matrices M_i and M_{i+1} whose common dimension r_i is smallest, and continue in the same way (note that M_i is a r_{i-1} by r_i matrix);
(b) First multiply the matrices M_i and M_{i+1} whose common dimension r_i is largest, and continue in the same way;
(c) First multiply the matrices M_i and M_{i+1} that minimize the product $r_{i-1}r_ir_{i+1}$, and continue in the same way;
(d) First multiply the matrices M_i and M_{i+1} that maximize the product $r_{i-1}r_ir_{i+1}$, and continue in the same way.

Note that when applying a greedy strategy, the idea is to be carried out through the entire process until an ordering of the multiplications has been found (in other words, you don't just apply it once).

2. (20) We have unlimited number of bins each of capacity 1, and n objects of sizes s_1, s_2, ..., s_n, where $0 < s_i \leq 1$. Our job is to pack these objects into bins using the fewest bins possible (optimization). The decision version can be stated as follows: given n objects and k, is there a packing using no more than k bins. Show how to solve one version if you know how to solve the other version, i.e., show how to solve the decision version by solving the optimization version and show how to solve the optimization version by solving the decision version.

3. (20) (a) Let P_1, P_2, ..., P_n be a set of n programs that are to be stored on a memory chip of capacity L. Program P_i requires a_i amount of space. Note that L and a_i's are all integers. Clearly, if $a_1 + a_2 + ... + a_n \leq L$, then all the programs can be stored on the chip. So assume that $a_1 + a_2 + ... + a_n > L$. The problem is to select a maximum subset Q of the programs for storage on the chip. A maximum subset is one with the maximum number of programs in it. Give a greedy algorithm that always finds a maximum subset Q such that

$$\sum_{P_i \in Q} a_i \leq L.$$

You also have to prove that your algorithm always finds an optimal solution. Note that a program is either stored on the chip or not stored at all (in other words, you do not store part of a program).
(b) Suppose that the objective now is to determine a subset of programs that maximizes the amount of space used, that is, minimizes the amount of unused space. One greedy approach for this case is as follows: As long as there is space left, always pick the largest remaining program that can fit into the space. Prove or disprove that this greedy strategy yields an optimal solution.

4. (20) Suppose that in a 0-1 knapsack problem, the order of the items when sorted by increasing weight is the same as their order when sorted by decreasing value. Give an efficient algorithm to find an optimal solution to this variant of the knapsack problem, and argue (i.e., prove) that your

algorithm is correct. What is the running time of your algorithm?

5. (10) Consider the machine scheduling problem studied in class where we have n jobs with start and finish times for job i being s_i and f_i, $s_i < f_i$ and the goal is to schedule them using fewest machines possible. Show that the greedy idea that always selects the shortest job first does not always give an optimal solution.

An easy way to represent a job with a starting time s and finish time f is to use horizontal line segment of length $f - s$.

6. (20) The greedy algorithm to solve the fractional knapsack problem optimally selects objects in decreasing value to weight ratios (discussed in class). For the 0-1 knapsack problem:
(a) show that the greedy algorithm does not work;
(b) show that the greedy method not only does not work, it yields arbitrarily bad solutions, that is, given any $0 < \epsilon < 1$, construct an example where the ratio of the greedy solution to the optimal solution is $< \epsilon$. Note that if you are able to (b), then your answer will also serve as an answer to (a). In this case, there is no need to do (a). On the other hand, if you are unable to do (b), you should try to do (a).