CS601 Algorithms and Complexity **2019-20 Sem I**

Assignment 1

Total Marks: 60 *Deadline: October 14, 2 PM*

Note: You are supposed to solve and write these questions on your own.

Lecture 14 on Moodle has been updated with a simpler proof for the correctness of a greedy strategy. It is similar to what was done in the class, but cuts out unnecessary details.

Que 1 [4+8+4+4 marks]. Suppose you are given a set of n course assignments today, each of which has its own deadline. Let the i-th assignment have deadline d_i and suppose to finish the i-th assignment it takes ℓ_i time. Given that there are so many assignments, it might not be possible to finish them on time. If you finish an assignment at time t_i which is more than its deadline d_i, then the difference $t_i - d_i$ is called the lateness of this assignment. Since you want to maintain a balance among courses, you want that the maximum lateness over all assignments is as small as possible. In other words, minimize $\max_i L_i$, where L_i is defined to be $t_i - d_i$ if $t_i > d_i$, and 0 otherwise.

Can you show that a greedy algorithm will give you an optimal solution?

Greedy Strategy 1: Do the assignments in increasing order of their lengths (ℓ_i).

Greedy Strategy 2: Do that assignment first whose deadline is the closest.

Greedy Strategy 3: Do that assignment first for which $d_i - \ell_i$ is the smallest.

Two of these strategies don't work. Give examples to show that they don't work. One of the strategies actually work. Prove that it works by following three steps (or write a proof in your own way):

(i) Argue that there is an optimal solution which includes the first step of the greedy algorithm.

(ii) Define a subproblem after you have taken the first step of the greedy. Show that you can combine an optimal solution of the subproblem with the first step of greedy to get a valid solution for the original problem.

(iii) Show that this valid solution for the original problem is optimal (you will need to use part (i) for this). Here, you basically need to argue that if it's not optimal then the subproblem solution itself was not optimal, which gives you a contradiction.

Que 2 [10 marks]. Suppose there are 8 tourist guides, of which, 2 can speak both German and French, 2 can speak only German, 2 can speak only French, and the rest 2 can speak neither of the two languages. Every guide has their own charges. We want to select 3 guides that minimize the total charges. But, we have an additional constraint that we need to take at least 1 French-speaking guide and at least 1 German-speaking guide (the two can be the same person).

The constraints can be put in other words as: the number of non-French-speaking guides (there are 4 such guides) selected should be at most 2 and the number of non-German-speaking guides selected should also be at most 2. A simple greedy strategy could be the following. We can start selecting the guides in increasing order of their charges. We should also keep counters for non-French-speaking and non-German-speaking guides. Whenever these numbers hit the threshold we stop considering that category. Give an example to show that the greedy strategy fails to give an optimal solution.

Que 3 [4+16 marks]. This question is based on what we will discuss on Oct 9.

There are n items with their prices $\{v_1, v_2, \ldots, v_n\}$ and weights $\{w_1, w_2, \ldots, w_n\}$. You won a lottery where you can take any set of items whose total weight is at most W. You want to maximize the total price value of the items you take.

Show by examples that the following greedy strategies will not give an optimal solution.

1. Pick the item with largest v_i first.
2. Pick the item with largest v_i/w_i (price per kg) first.

Give an efficient algorithm to compute the optimal solution.

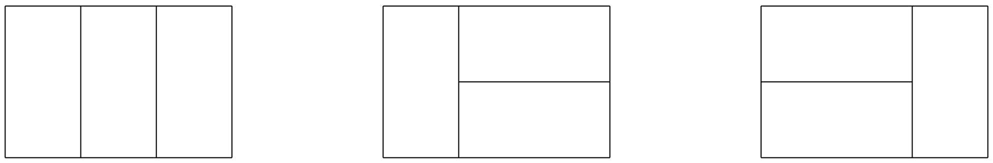

Figure 1: Three arrangements for 2×3 rectangle.

Que 4 [10 marks]. You have 2-dimnensional bricks of size 2×1. You have to completely fill an $n \times 3$ rectangle with these bricks. Bricks can be either put horizontally or vertically. You need to find the number of possible arrangements of bricks modulo 23. Can you do this in only $O(\log n)$ bit operations.

The figure 1 shows that there are 3 arrangements for $n = 2$.

Note that we need a bound on the number of bit operations. That means, if you are adding two ℓ-bit numbers, its cost will be $O(\ell)$; it will not be unit cost. You might be concerned that the number of arrangements is too large to express in $O(\log n)$ bits, but remember that you only need to output its value modulo 23.