Assignment 3

Algorithms: Correctness and Complexity

Due Dec 2 at 5:00PM, 2016

Q0 [10+5] Scheduling meetings

(a) [10] Design a greedy algorithm for the following problem. We are given a set of meetings M each of which has a start time $s(x) \in \mathbb{N}$ and an end time $f(x) \in \mathbb{N}$. We need to schedule the as many of meetings as possible in $r \in \mathbb{N}$ rooms, where $r > 1$. Each meeting can be in any room.

Bonus: (b) [5] Prove the algorithm correct.

Q1 [10] We want to solve the all sources / all sinks reachability problem: Given a simple directed graph $G = (V, E)$, where $V = \{0, ..n\}$. Compute a matrix $C : V \times V \to \mathbb{B}$ so that, for all s and t, $C(s,t)$ is *true* if there is a path from s to t and otherwise *false*. Note that we only need to consider paths of length 0 up to n.

(a) [5] Design an algorithm that works by increasing the path maximum length by 1. Initialize C for all paths of length 0. Then extend it to paths of length 0 or 1, then to paths of length up to (and including) 2, then for paths of length up to (and including 3) and so on considering paths of length one longer in each iteration.

(b) [5] Design an algorithm that works by doubling the maximum path length in each iteration. Initialize C for paths of length 0 or 1, then compute C for paths of length 0, 1, or 2, then for paths of length 0, 1, 2, 3, or 4, then for paths, and so on, doubling the length of paths considered in each iteration.

Q2 [20+5] Consider the following scheduling problem. Input: a set of tasks T (without duplicates). Each task x is associated with a positive integer duration $d(x)$, a natural deadline $dl(x)$, and a real profit $p(x)$. Each task is to be assigned a start time $s(x) \in \mathbb{N} \cup \{\infty\}$. If $s(x)$ is a natural number then $s(x) + d(x)$ must be less than $dl(x)$; if $s(x) = \infty$ then the task is never started. The following objective function should be maximized $\sum x \in T \mid s(x)+d(x) < dl(x) \cdot p(x)$. At most one task may be done at a time.

(a) [5] Demonstrate by example that none of the following greedy approaches will maximize profit:

- Sort the tasks in order of duration (shortest first). Examine the tasks in this order committing to those that can be done before their deadline and rejecting those that can't.
- Similarly, but sorting by deadline (earliest first).
- Similarly, but sorting by profit (highest first).
- Similarly, but sorting by wage, where wage is profit over duration (highest first).

(b) [5] Develop a brute-force, recursive algorithm to find the maximum profit. Be sure to state pre- and postconditions. [Hint: Sort T by deadline: earliest first. Then write a function that takes a number k and a time and computes the maximum profit that can be made within the given time limit using only the first k tasks of t.]

(c) [5] Develop a top-down (memoizing) dynamic programming algorithm to find the maximum profit.

(d) [5] Develop a bottom-up dynamic programming algorithm to find the maximum profit. What is the complexity of your algorithm in terms of the number of tasks n and the last deadline dlf.

Bonus (e) [5] Find a way to extract from the table, a set of tasks that can be done to maximize profit. Once we have this set, scheduling is easy.