Midterm Examination

CS 4104 (Spring 2016)

Assigned: Tuesday, March 15, 2016.
Due: on Canvas before the beginning of class on Thursday, March 24, 2016.

Name: _____

PID: _____

Instructions

1. **You must work on your own for this examination, i.e., you are not allowed to have a partner.**

2. For every algorithm you describe, prove its correctness, and state and prove the running time of the algorithm. I am looking for clear descriptions of algorithms and for the most efficient algorithms and analysis that you can come up with. I am not specifying the desired running time for each algorithm. I will give partial credit to non-optimal algorithms, as long as they are correct.

3. You may consult the textbook, your notes, or the course web site to solve the problems in the examination. Of course, the TA and I are available to answer your questions. You **may not** work on the exam with anyone else, ask anyone questions, or consult other textbooks or sites on the Web for answers. **Do not use** concepts from Chapters 6 and later in the textbook.

4. You must prepare your solutions digitally, i.e., do not hand-write your solutions.

5. I prefer that you use LaTeX to prepare your solutions. However, I will not penalise you if you use a different system. To use LaTeX, you may find it convenient to download the LaTeX source file for this document from the link on the course web site. At the end of each problem are three commented lines that look like this:

   ```
   % \solution{
   %
   % }
   ```

   You can uncomment these lines and type in your solution within the curly braces.

Good luck!

Problem 1 (15 points) Let us start with some quickies. For each statement below, say whether it is true or false. You do not have to provide a proof or counter-example for your answer.

1. Every tree is a bipartite graph.
2. $\sum_{i=1}^{n} i \log i = \Theta(n^2 \log n)$.
3. You solve a problem on an input of size n by dividing it into three subsets of size $n/3$, solving each sub-problem recursively, and combining the solutions in $O(n)$ time. The running time of your algorithm is $O(n \log n)$.
4. In a directed graph, $G = (V, E)$, each edge has a weight between 0 and 1. To compute the length of the *longest* path that starts at u and ends at v, we can change the weight of each edge to be the reciprocal of its weight and then apply Dijkstra's algorithm.
5. Suppose π is the shortest s-t path in a directed graph. Then, there can be two nodes u and v in π such that length of the portion of π connecting u to v is larger than the length of the shortest u-v path in the graph.

Problem 2 (10 points) Consider the problem of minimising lateness that we discussed in class. We are given n jobs. For each job $i, 1 \leq i \leq n$, we are given a time $t(i)$ and a deadline $d(i)$. Let us assume that all the deadlines are distinct. We want to schedule all jobs on one resource. Our goal is to assign a starting time $s(i)$ to each job such that each job is delayed as little as possible. A job i is *delayed* if $f(i) > d(i)$; the *lateness of the job* is $\max(0, f(i) - d(i))$.

Define

1. the *lateness of a schedule* as $\max_i \big(\max \big(0, f(i) - d(i)\big)\big)$ and
2. the *delay of a schedule* as $\sum_{i=1}^{n} \big(\max \big(0, f(i) - d(i)\big)\big)$.

Note that although the words "lateness" and "delay" are synonyms, for the purpose of this problem we are defining them to mean different quantities: the lateness of a schedule is the *maximum* of the latenesses of the individual jobs, while the delay of a schedule is the *sum* of the latenesses of the individual jobs.

Consider the algorithm that we discussed in class for computing a schedule with the smallest lateness: we sorted all the jobs in increasing order of deadline and scheduled them in this order. We proved that the earliest-deadline-first algorithm correctly solves the problem of minimising lateness. If we were to use the *same proof* to try to demonstrate that the algorithm correctly solves the problem of minimising delay, where does the proof break down?

Problem 3 (30 points) Inspired by the Flint Water Study, you start thinking about using multiple sensors embedded in different parts of the city to automatically test water quality. Sensors can communicate with each other. However, each sensor has limited resources (e.g., it can contact only "nearby" sensors) and these communications can be unreliable and asymmetric. The goal of each sensor is to relay the information it has collected back to a "master" sensor. After deploying the sensors, you can measure which pairs can communicate and how reliable each link is. Your goal is to determine the maximum number of sensors that can communicate with the master through a reliable series of links.

Fortunately, you have taken CS 4104! You model the problem as follows. You have a directed graph $G = (V, E)$. Each node in V corresponds to a sensor. The master sensor is a specific node s in V. Every edge $e = (u, v)$ directed from node u to node v has a reliability $r(e)$ that lies between 0 and 1. If an edge has reliability of 0, it is equivalent to the edge not existing in G. Since communications are asymmetric, it is not necessary that $r(u, v) = r(v, u)$. The reliability of a path is the product of the reliabilities of the edges in that path. Given such a graph G and a parameter $q > 0$, describe an algorithm to compute the largest set S of nodes in V such that for every node u in S there is a directed path from u to s with reliability at least q.

Note: I have used the problem of placing sensors to inspire this question. I suggest you develop your solution using the language of graphs. In other words, you should be able to present your solution even if you read only the second paragraph of this question.

Problem 4 (20 points) You are given a list of n real numbers (the list can contain both positive and negative numbers). Give an efficient algorithm to determine the contiguous sub-list with the largest sum. More formally, suppose the numbers are $l_1, l_2, \ldots, l_{n-1}, l_n$. Your mission, should you choose to accept it, is to compute two indices $1 \leq i < j \leq n$ such that

$$s(i,j) = \sum_{k=i}^{j} l_k$$

is the largest over all possible choices of i and j. Note that to compute $s(i,j)$ we sum up all the numbers between indices i and j inclusive.

Problem 5 (25 points) After graduating from Virginia Tech, you start working at a company that makes games for phones. You are assigned to work on a game called PoodleJump. The goal here is for a player to move a character (a Poodle, to be precise) so that it jumps from one ledge to another. The position of a ledge is specified by its x-coordinate and its z-coordinate. The Poodle can only jump to the left. A jump is *good* if the poodle jumps to the left and if the ledge l that the poodle lands upon has the following property: every ledge to the right of ledge l, i.e., with x-coordinate larger than that of l, is below l, i.e., has a smaller z-coordinate than that of l; Otherwise, the jump is bad. See Figure 1. The poodle starts at the rightmost ledge, i.e., the one with the largest x-coordinate. The game ends as soon as the Poodle makes a jump that is bad. Given a set of n ledges (and their positions), devise an efficient algorithm to compute the largest number of good jumps a player can make and the sequence of ledges that comprise these jumps.

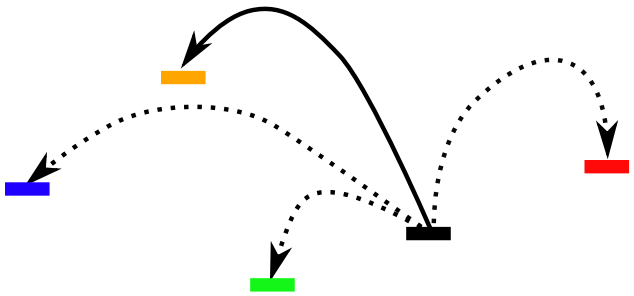

Figure 1: An illustration of good and bad jumps. The jumps from the black ledge to the red, green, and blue ledges are bad: red because the poodle jumps to the right; green because the black ledge is to the right of and above the green ledge; and blue because the orange ledge is to the right of and above the blue ledge. The only good jump from is from the black ledge to the orange ledge.