CpSc 8400: Design and Analysis of Algorithms

Instructor: Dr. Brian Dean — Spring 2014
Webpage: `http://www.cs.clemson.edu/~bcdean/` — TTh 11:00-12:15
Handout 18: Final Exam. 50 possible points. — Vickery 100

Unless otherwise stated, your algorithms should run in the RAM model of computation. When describing an algorithm, be both *precise* and *concise* — short answers are acceptable, as long as they address all of the key aspects of the problem at hand. Whenever you describe a new algorithm, you should always give its running time. For questions where a target running time is not specified, your score will depend largely on the speed of your algorithm. Randomization is always acceptable, unless explicitly forbidden. Good luck!

1. BST (5 points). You are given a set of n numbers, each an integer in the range $1 \ldots 10n$. Design a fast algorithm for building a binary search tree of height $O(\log n)$ on these numbers.

2. Carpool (5 points). You are given a directed graph G with n nodes and m edges, and you are told the cost $c_e \geq 0$ of traveling along each edge e. Dorian starts at node x and Ravi starts at node y. They would both like to travel to a common destination node z. If P_x represents edges along the $x \rightsquigarrow z$ path taken by Dorian and P_y the edges along the $y \rightsquigarrow z$ path taken by Ravi, the total cost of their travel is the cost of all the edges in $P_x \cup P_y$ (so it might be useful for them to arrange their paths so they overlap, allowing them to carpool and save money). Please describe a fast algorithm for computing paths P_x and P_y minimizing the cost of $P_x \cup P_y$.

3. Paranoid Max (5 points). Consider an array $A[1 \ldots n]$ of distinct elements that are randomly permuted (so each of the $n!$ orderings of these elements is equally likely). Suppose we scan sequentially through the array from position $i = 1$ up to position $i = n$ keeping a running maximum M. However, any time we encounter an element $A[i] > M$, we not only reset M to $A[i]$, but we also double-check all the elements in $A[1 \ldots i]$ (in $O(i)$ time) just to make absolutely certain that they are all no larger than M (and of course, this check will always succeed). What is the expected running time of this algorithm?

4. k-Partition (5 points). You are given an array $A[1 \ldots n]$ of integers and also an integer k. We say that A has a k-partition if its elements can be divided into k non-overlapping contiguous subarrays each of which has the same sum (and the subarrays must collectively contain all the elements from A). Please give a fast algorithm for testing whether or not A has a k-partition.

5. Security (5 points). The security log of the front door to a bank tells you the records of all people entering or exiting the building during the past day. These n records are of the form $(t_1, p_1, e_1) \ldots (t_n, p_n, e_n)$, where for each record i,

- t_i is an integer-valued time,
- p_i is the integer identifier for a person, and
- e_i is $+1$ if person p_i enters the bank at time t_i, or -1 if person p_i leaves the bank at time t_i.

All t_i's and p_i's are in the range $1 \ldots n^2$. The bank is initially empty at time $t = 0$. Please design a fast algorithm that determines if there are two times $t < t'$ (not necessarily integers) at which exactly the same people are in the bank, such that at least one entry or exit occurs between t and t'. For only partial credit, you can make the simplifying assumption that the p_i's or the t_i's (but not both) are at most $O(1)$.

6. Inequalities (5 points). You are given a set of m strict inequality constraints over the variables $x_1 \dots x_n$. For example, you might have $x_1 < x_2$, $x_2 < x_3$, and $x_3 < x_1$. As in this example, it might be impossible to assign values to the variables $x_1 \dots x_n$ so that all m inequalities are satisfied. Our goal is to assign values to these variables so as to satisfy a maximum number of the inequalities. Please design a 2-approximation algorithm for this problem, which will always satisfy at least $OPT/2$ inequalities, where OPT denotes the maximum number that can be satisfied.

7. Game (5 points). You are playing a game with your opponent in which you start with n numbers $A_1 \ldots A_n$ in a row. On your turn, you can select one of the two endpoints from this sequence and remove it. Then your opponent can select one of the two endpoints and remove it. The game proceeds in this fashion, until all the numbers have been removed; the winner is the player who removed the numbers with the highest total sum. Please design a fast algorithm that can tell if, moving first, you can guarantee winning.

8. Frequent Numbers (5 points). You are given an array $A[1 \dots n]$. Please design a fast *comparison-based* algorithm that will output every number occurring in at least 1% of all the entries in A. For example, if $n = 1000$, you should output any number occurring at least 10 times in A.

9. Delete Largest Half (5 points). You would like to design a data structure supporting two operations: *insert* (which inserts a new element), and *delete-largest-half*, which deletes the largest $n/2$ elements from the structure. Please show how to build this data structure so that both operations run in $O(1)$ amortized time. For full credit, use a potential function in your analysis.

10. Shortest Path (5 points). How can you quickly solve the single-source shortest path problem in a directed graph whose edge costs belong to $\{0, 1\}$?

This page may be used for scratch work.

This page may be used for scratch work.