CSE521: Design and Analysis of Algorithms November 11, 2022

Midterm

Anup Rao Due: November 21, 2022

Read the fine print[1]. An algorithm is said to run in polynomial time if it runs in time $O(n^d)$ for some constant d on inputs of size n.

1. (45 points, 5 each) For each of the following problems answer **True** or **False** and BRIEFLY JUSTIFY you answer.

   (a) $n^{3.05} = O(n^3 \log n)$.

   (b) $2^n = \Omega(2^{2n})$.

   (c) There is a polynomial time algorithm for deciding whether a graph has an odd length cycle or not.

   (d) If the running time of an algorithm satisfies the recurrence $T(n) \leq 3T(n/2) + n^5$, then $T(n) = O(n^3)$.

   (e) Suppose an input to the MST problem does *not* have distinct edge weights. Suppose there is a cycle in the graph where two of the edges have the heaviest weight. Is it true that every MST must exclude *both* of these heavy edges?

   (f) In an input to the MST problem, if e is the unique lightest edge touching a vertex v, then e must be part of every MST.

   (g) There is an $O(n^{1.1})$ time algorithm for multiplying two n-bit numbers.

   (h) If an undirected graph contains a matching of size k, then it is possible that it also has a vertex cover of size $k - 1$

   (i) You are given an input the set cover problem, with m sets over an n element universe. You are promised that $m > n$, and either at most $\sqrt{n}$ of the sets can cover the whole universe, or at least n sets are required. There is a polynomial time algorithm for deciding which case is satisfied by the input.

[1]No collaboration on this one. The problems have been carefully chosen for their pedagogical value, and hence might be similar to those given in past offerings of this course at UW, or similar to other courses at other schools. Using any pre-existing solutions from these sources, for from the web, constitutes a violation of the academic integrity you are expected to exemplify, and is strictly prohibited. Most of the problems only require one or two key ideas for their solution. It will help you a lot to spell out these main ideas so that you can get most of the credit for a problem even if you err on the finer details. Please justify all answers. Some other guidelines for writing good solutions are here: `http://www.cs.washington.edu/education/courses/cse421/08wi/guidelines.pdf`.

2. (20 points) You are given an $n \times n$ checkerboard and a checker. You are allowed to move the checker from any square x to a square y on the board as long as square y is directly above x, or y is directly to the left of the square directly above x, or y is directly to the right of the square directly above x. Every time you make a move from square x to square y, you earn $p(x, y)$ points, where $p(x, y)$ is an integer (possibly negative) that is part of the input. Give an algorithm that on input the values $p(x, y)$, outputs the maximum score that can be achieved by placing the checker on some square on the bottom row and moving the checker all the way to some square of the top row. The algorithm should run in polynomial time.

3. (15 points) Suppose you are choosing between the following three algorithms:

   - Algorithm A solves the problem by dividing it into five subproblems of half the size, recursively solves each subproblem, and then combines the solution in linear time.
   - Algorithm B solves problems of size n by recursively solving two subproblems of size $n - 1$, and then combines the solution in constant time.
   - Algorithm C solves the problem by dividing it into nine subproblems of one third the size, recursively solves each subproblem, and then combines the solutions in quadratic time.

   What are the running times of each of these algorithms?

4. (20 points) Given two strings $x_1, \ldots, x_m$ and $y_1, \ldots, y_n$, we want to calculate the length of the longest common substring, namely the largest k for which there are i, j such that $x_i x_{i+1} \ldots x_{i+k-1} = y_j y_{j+1} \ldots y_{j+k-1}$. Show how to do this in time $O(mn)$.