CSE521: Design and Analysis of Algorithms December 9, 2022

Final

Anup Rao Due: December 18, 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. (100 points, 5 each) For each of the following problems answer **True** or **False** and BRIEFLY JUSTIFY you answer.

   (a) Let (S,T) be a minimum st-cut in a flow graph (G,s,t). If the capacity of every edge in G is increased by 2 then the value of the maximum flow in G is increased by 2 times the number of edges from S to T in G.

   (b) There is an exponential time $(2^{n^{O(1)}})$ algorithm for Vertex Cover.

   (c) There is a linear time algorithm for finding the $n/10$'th smallest number in a list of n numbers.

   (d) If $T(n) = 3T(n/2) + n$ for $n \geq 2$ then $T(n)$ is $\Theta(n^{\log_3 2})$.

   (e) If $T(n) = 12T(n/4) + n^2$ for $n \geq 4$ then $T(n)$ is $\Theta(n^2)$.

   (f) If $3SAT$ has a linear time algorithm, then so does every problem in NP.

   (g) Every decision problem has an exponential time algorithm.

   (h) The average degree of a vertex in a tree with n vertices is always $2 - 2/n$.

   (i) There is a polynomial time algorithm that takes an input to Program-SAT and outputs a linear program, such that the linear program has a feasible solution if and only if the output of Program-SAT is 1.

   (j) Linear programs can be solved in polynomial time.

   (k) If e is an edge whose weight is lower than the weight of all other edges in an undirected graph, then every minimum spanning tree must contain e.

   (l) If there is an optimal solution to a linear program, then there is a vertex of the polytope that achieves this optimal value.

   (m) Suppose you are given three flow networks G_1, G_2, G_3 which have the same vertices and edges. For each edge e, suppose the capacity of the edge in G_1 is $c_1(e)$, in G_2 it is $c_2(e)$ and in G_3 it is $c_1(e) + c_2(e)$. Then if v_1, v_2, v_3 are the corresponding values of the maximum flows in each of the networks, we must have $v_3 = v_1 + v_2$.

   (n) There is a polynomial time algorithm for the Vertex Cover that finds the optimal vertex cover up to a factor of 2.

[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`.

(o) If G is an undirected graph and s, t are distinct vertices such that for every partition of the vertices A, B with $s \in A, t \in B$, there are at least ℓ edges going from A to B, then there are at least ℓ edge-disjoint paths from s to t.

(p) Say that an integer x is a sum of squares if it satisfies $x = y^2 + z^2$ for some integers y, z. Define $f(x)$ to be 1 if x is the sum of squares, and 0 otherwise. Then if there is a polynomial time algorithm for 3SAT, there is also a polynomial time algorithm for deciding whether x is a sum of squares.

(q) There is an $O(n^3)$ time algorithm to find a cycle of negative weight (if one exists) in a directed graph with possibly negative edge weights.

(r) If $3SAT$ has an algorithm that runs in time $2^{O(\log^2 n)}$, then we can find an optimal vertex cover in a graph in time $2^{O(\log^2 n)}$.

(s) There is an $O(m \log n)$ time algorithm for finding the minimum spanning tree in a graph.

(t) Suppose we discover a way to express the product of two $k \times k$ matrices using at most r multiplication operations and k^2 addition operations, where k, r are constants such that $3 > \log_k r > 2$. This will lead to an algorithm for multiplying two $n \times n$ matrices in time $O(n^{\log_k r})$.

2. (20 points) **1-2-3 Minimum Spanning Tree** Let $G = (V, E)$ be an undirected graph. Suppose that each edge e has a cost $c(e)$, with $c(e) \in \{1, 2, 3\}$. Give an $O(n + m)$ time algorithm to compute a minimum spanning tree for G. HINT: First find a fast algorithm for the case where all edges have the same weight.

3. (15 points) In class we saw how to multiply two degree n polynomials $P(X)$ and $Q(X)$ in time $O(n \log n)$ using the fast fourier transform. The algorithm we designed broke up the problem into two subproblems of equal size, giving the recurrence $T(n) \leq 2T(n/2) + O(n)$. Say you want to run such an algorithm on a server farm where the number of servers is a power of 3. For this reason, it is more efficient to break up the problem into 3 subproblems of equal size, giving the recurrence $T(n) \leq 3T(n/3) + O(n)$. Explain how you would modify the algorithm we saw in class to accomplish this.

4. (25 points) In class we saw an algorithm for computing the k'th smallest number in a given list of n numbers in time $O(n)$. The algorithm worked by breaking the numbers into $n/5$ sets of size 5, and the then considering the medians of these sets. We set w to be the median of these medians, and used it to split all the numbers and recurse. What bound would you obtain if you analyzed the algorithm that uses sets of size 4 in the above process? (The median of 4 numbers $a_1 \leq a_2 \leq a_3 \leq a_4$ is $\frac{a_2+a_3}{2}$).