NAME: _____

CSE 421
Design and Analysis of Algorithms
Final Exam

Anup Rao December 13

DIRECTIONS:

- Answer the problems on the exam paper.
- You are allowed a single cheat sheet.
- Justify all answers with proofs, unless the facts you need have been proved in class or in the book.
- If you need extra space use the back of a page
- You have 1 hour and 50 minutes to complete the exam.
- Please do not turn the exam over until you are instructed to do so.
- Good Luck!

1	/50
2	/25
3	/25
4	/25
Total	/135

1. (50 points, 5 each) For each of the following problems answer **True** or **False** and BRIEFLY JUSTIFY your answer.

   (a) If a linear program is written in standard form, then its optimum value can never exceed the value of any valid solution to its dual.

   (b) 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 A to B.

   (c) There is an $O(n^2)$ time algorithm for finding the minimum spanning tree in a graph.

   (d) If an edge of a graph has larger weight than every other edge, it cannot be part of *any* MST.

(e) Say that an integer x is a sum of squares if it satisfies $x^2 = y^2 + z^2$ 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 computing whether x is a sum of squares.

(f) We have seen an $O(n^2)$ time algorithm to find a cycle of negative weight (if one exists) in a directed graph with possibly negative edge weights.

(g) If $3 - SAT$ 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 $2^{O(\log^2 n)}$.

(h) If a linear program has no solution, meaning that there is no way to satisfy its inequalities, then its dual must also have no solution.

(i) 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 $O(n^{\log_r k})$.

(j) Suppose someone came up with an algorithm that on input A, c, b can solve the following optimization problem in polynomial time.

$$\begin{aligned} &\text{maximize } c^\intercal x \\ &\text{subject to} \\ &\qquad Ax \leq b \\ &\qquad x \in \{0,1\}^n \end{aligned}$$

Namely, it can find the best 0/1 vector satisfying the linear constraints. Such an algorithm implies that $P = NP$.

2. (25 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. (25 points) You are going on a long road trip. You start the trip at milepost 0. Along the way, there are n hotels, at mileposts $a_1 < a_2 < \ldots < a_n$, where a_n is the location of your destination. You want to pick a subset of these hotels to stop at during your trip. You must stop at the final hotel. You would ideally like to travel 200 miles a day, but this may not be possible. If you travel x miles in between two hotels, the penalty for that section is $(200 - x)^2$. Give a polynomial time algorithm that determines the optimal set of hotels for you to stop at to minimize the total penalty.

4. (25 points) Given an undirected graph with a vertex s and a subset of vertices T, we would like to find a small set of edges that disconnects s from T. For each choice of set of edges S, let $q(S)$ denote the number of vertices of T that are not connected to s after deleting the edges of S from the graph. We would like to find an algorithm that finds a small set of edges that disconnects a lot of vertices. To this end, give a polynomial time algorithm that computes a set of edges S maximizing the quantity $q(S) - |S|$. HINT: Add a vertex t, and edges from every vertex of T to t. Compute the minimum $s - t$ cut in the graph.