NAME: _____

CSE 421
Introduction to Algorithms
Practice Final Exam Autumn 2013

Anup Rao December 5

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	/60
2	/25
3	/25
4	/25
5	/25
Total	/160

1. (60 points, 5 each) For each of the following problems answer **True**, **False** or **Unknown** and BRIEFLY JUSTIFY you answer.

   (a) There is a polynomial time algorithm for deciding whether a boolean formula is satisfiable or not.

   (b) If the capacity of every edge in a flow graph (G, s, t) is increased by a multiplicative factor of 2 then the value of the maximum flow in G also is increased by a multiplicative factor of 2.

   (c) 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 ℓ edges going from A to B, then there are at least ℓ edge-disjoint paths from A to B.

(d) 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.

(e) 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.

(f) If 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 $2^{O(\log^2 n)}$.

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

(h) 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 $O(n^{\log_k r})$.

(i) There is an $O(n^2/\log n)$ time algorithm for multiplying two n-bit numbers.

(j) $NP \subseteq EXP$.

2. (25 points) Suppose you are choosing between the following three algorithms:

   - Algorithm A solves problems by dividing them into 8 subproblems of half the size, recursively solving each subproblem, and then combining the solutions in linear time.
   - Algorithm B solves problems of size n by recursively solving two subproblems of size $n - 3$ and then combining the solutions in constant time.
   - Algorithm C solves problems of size n by dividing them into 27 subproblems of size $n/3$, recursively solving them, and combining the solutions in $O(n^2)$ time.

   What are the running times of these algorithms?

3. (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.

4. (25 points) In a graph, an independent set is a set of vertices S, such that no edge is contained in the set. Give a polynomial time algorithm to find an independent set of largest size in a tree. HINT: Root the tree and use dynamic programming.

5. (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.