CSE548/AMS542 Spring 2008 Analysis of Algorithms

Midterm

- This is an open book exam, but no computer is allowed.
- There are 7 problems and 35 points total.
- You can refer to the algorithms we have covered in class without referring to the details. If you give a greedy algorithm but do not give the proof of its correctness, you may lose all (if your algorithm is wrong) or partial points (if your algorithm is correct).
- In class on Thursday, March 27th, 2008. Exam starts at 9:50am and ends at 11:10am sharp.
- Be brief. Write down the key steps in your algorithm and you may not have time to elaborate in details.
- If you run out of space, you can write your solution on extra pages. Remember to write down your name and student ID on every extra page you submit.

By signing below I declare that I follow the rule of academic integrity and finish the exam on my own, without the help of others.

Name _____

ID _____

Signature _____

Score	Grade
1	
2	
3	
4	
5	
6	
7	
Total	

1. **Short questions** (5pts) Answer the following questions. Simply give the **final** answer. You **do not** need to give proofs or counter-examples.

   (a) True or False: If a directed graph has k distinct cycles, then for any depth-first search (DFS) of this directed graph, there are at least k back edges.

   (b) Consider a positively weighted undirected graph G. Suppose you replace the weight of every edge with its negation (e.g. $w(u, v)$ becomes $-w(u, v)$), and compute the minimum spanning tree of the resulting graph using Kruskal's algorithm. True or False: The resulting tree is a maximum cost spanning tree for the original graph.

   (c) Consider a positively weighted directed graph G and source vertex s. Suppose you replace the weight of every edge with its negation and compute the shortest paths using Dijkstra's algorithm. True or False: The resulting paths are the longest (i.e., highest cost) simple paths from s to every vertex in the original digraph.

   (d) What is the solution to the recurrence $T(n) = 3T(n/4) + \sqrt{n}$?

   (e) What is the solution to the recurrence $T(n) = T(n/3) + T(n/2) + n \log n$?

2. **Longest increasing subsequence** (5pts) Given a list of numbers, find the longest increasing subsequence. The elements in this subsequence need not be adjacent. For example, for the input $1, 5, 3, 2, 4$, the longest increasing subsequence is $1, 2, 4$ (or $1, 3, 4$). Your algorithm should run in polynomial time. If you get $O(n \log n)$ you can get extra 5 points.

3. **Sum to zero** (5pts) Given an array of n real numbers (not necessarily sorted) and a value v. Find out whether there are two numbers that sum to v, in $O(n \log n)$ time.

4. **Sorting with flips** (5pts) Suppose you have a list of numbers, and you want to sort them. You can only do the *flip* operation, i.e., take an interval of numbers and reverse them completely. For example, with an input 3712645, you can flip the interval 645 and get a new sequence 3712546.

   (a) Describe an algorithm to sort a list of numbers with flip operations. (1pts)

   (b) How many flip operations does your algorithm use? Use $\Theta()$ notation. That is, you need to give an upper bound (big-O) for the running time and also give one input instance to show that for some input the algorithm does take the same asymptotic time as shown in your upper bound. (4pt)

5. **Minimum spanning tree** (5pts) Given a weighted graph $G = (V, E)$, and its minimum spanning tree T. Now suppose that the weight of two edges e_1, e_2 are swapped, update the minimum spanning tree T in time $O(m + n)$.

6. **Minimum perimeter triangle** (5pts) Let P be a set of points in the plane. Find in time $O(n \log n)$ the triangle $\triangle p_1 p_2 p_3$ with smallest perimeter, among all triangles whose vertices are points of P. The perimeter is the sum of lengthes of edges of the triangles.

7. **Smallest cover** (5pts) Let X be a set of n intervals on the real line. We call a subset of intervals $Y \subseteq X$ to be a *cover* of X if any value contained in some interval of X is contained in some interval of Y. Give an efficient algorithm to find a cover of minimum number of intervals.