CS 314: Algorithms, Spring 2009

Midterm Exam — March 7, 2009

Name:	Email Address:

---

1. This is a closed-book and closed-notes exam. You are allowed one "cheat sheet" on standard 8.5 by 11 inch paper, handwritten on the front and back.
2. Print your full name and your email address in the boxes above.
3. Print your name at the top of every page.
4. Please write clearly and legibly. If I can't read your answer, I can't give you credit.
5. When asked to design an algorithm, you must also provide a runtime analysis for that algorithm. However, proofs of correctness are NOT required for algorithms on this exam. However, problems that explicitly ask you to prove something still require you to do proofs!
6. There are 4 problems on the exam. Your grade will be calculated based only on 3 out of the 4 problems. Please feel free to try all of them, though; I'll take the maximum of the 3 for your final grade.
7. As with any problem, you are welcome to write "I don't know" **and nothing else** in order to receive 25% of the total points. This only applies to entire problems, however; for example, you may not give an algorithm but not compute the runtime and still say "I don't know" for just the runtime portion of the problem.
8. Remember, these are NOT necessarily in order of difficulty. Please read all the problems first, and don't allow yourself to get stuck on a single problem.

#	1	2	3	4	Total
Max	20	20	20	20	60
Score					

1. Over spring break, you visit the exciting country of Vacationland. On your cab ride from the airport to the hotel, you examine the coins that you got at the currency exchange. The cab driver notices you examining the money, and proudly tells you that his country has the greatest monetary system ever. Fresh from your algorithms midterm, however, you realize that a greedy algorithm had better work on these coins in order for him to make a claim like that!

   So consider a set of coins which include the following values: 1 cent, 7 cents, 11 cents, and 23 cents. Does the greedy algorithm for making change on these denominations always produce the minimum number of coins possible?

   Please either prove that the greedy strategy is optimal or provide an example where it does not work optimally.

2. Recall that an *independent set* in a graph $G = (V, E)$ is a set of vertices S such that no two vertices in S have an edge going between them.

   Suppose you are given a tree $T = (V, E)$ along with a root vertex r in V and weights on all the vertices, written $w(v)$ for v in V. We will say that the weight of a set S of vertices is the sum of the weights of all the vertices in S: $w(S) = \sum_{v \in S} w(v)$.

   Give a algorithm to compute a maximum weight independent set in the tree T. (You do not need to include a proof of correctness; however, your algorithm does need to be correct!)

3. Give an algorithm to detect whether a given undirected graph contains a cycle. If the graph contains a cycle, then your algorithm should output one. (It should NOT output all cycles in the graph, just one of them.)

   For full credit, the running time of your algorithm should be $O(m + n)$ for a a graph with n vertices and m edges.

4. Suppose we are given an array A[1 .. n] with the special property that $A[1] \geq A[2]$ and $A[n-1] \leq A[n]$. We say that an element $A[k]$ is a local minimum if it is less than or equal to both its neighbors, or more formally, if $A[k?1] \geq A[k]$ and $A[k] \leq A[k+1]$. For example, there are five local minima in the following array:

9	**7**	7	2	**1**	3	7	5	**4**	7	3	**3**	4	8	**6**	9

We can obviously find a local minimum in $O(n)$ time by scanning through the array. Describe and analyze an algorithm that finds a local minimum in $O(\log n)$ time.

[Hint: With the given boundary conditions, the array must have at least one local minimum. Why?]

(scratch paper)

(scratch paper)