Illinois Institute of Technology
Department of Computer Science

Third Examination

CS 430 Introduction to Algorithms
Fall, 2010

10:30am–12:30pm, Monday, December 6, 2010
121 Life Sciences Building

Print your name and student ID, *neatly* in the space provided below; print your name at the upper right corner of *every* page. Please print legibly.

Name:
Student ID:

This is an *open book* exam. You are permitted to use the textbook, any class handouts, anything posted on the web page, any of your own assignments, and anything in your own handwriting. Foreign students may use a dictionary. *Nothing else is permitted*: No calculators, laptops, cell phones, Ipods, Ipads, etc.!

Do all five problems in this booklet. *All problems are equally weighted, so do not spend too much time on any one question.*

Show your work! You will not get partial credit if the grader cannot figure out how you arrived at your answer.

Question	Points	Score	Grader
1	20		
2	20		
3	20		
4	20		
5	20		
Total	100		

1. **Disjoint Sets**

   Consider the disjoint set data structure of section 21.3 in CLRS. Suppose that we do k LINK and FIND operations (CLRS, page 571), and all the LINK operations precede all the FIND operations. (Assume that n MAKESET operations have been done, at no cost, before the LINK and FIND operations.) Use the accounting method of amortized analysis to prove that the cost of the k LINK and FIND operations is $O(k)$, independent of n.

   (*Hint*: Charge \$2 for each LINK and \$1 for each FIND; use \$1 of the LINK charge to pay for the operation and attach the other \$1 to the root of the tree that becomes child during the LINK.)

2. **BFS on Undirected Graphs**

An undirected graph $G = (V, E)$ is *bipartite* if the vertex set v can be partitioned into sets X and Y in such a way that every edge in E has one vertex in X and the other in Y. This means that the vertices of G can be colored red and blue so that each edge connects a red vertex to a blue vertex.

(a) Prove that an undirected graph is bipartite if and only if it does not contain an odd-length cycle.

(b) Use BFS (CLRS page 595) to obtain an $O(|V| + |E|)$-time algorithm that determines whether a graph is bipartite. If the graph is bipartite, your algorithm should produce an appropriate red/blue coloring of the vertices; if not, your algorithm should produce an odd-length cycle.

2. **BFS on Undirected Graphs, continued.**

3. **Minimum Spanning Trees**

Consider the following claim:

> Let G be an arbitrary connected, undirected graph with a distinct, non-negative cost $c(e)$ for every edge of G. Let e^* be the cheapest edge in the graph; that is $c(e^*) < c(e)$ for every edge $e \neq e^*$. Then there is a minimum spanning tree of G that contains the edge e^*.

Is this statement true or false? If false, show a counterexample; if true, prove it.

4. **NP-Completeness**

The *subgraph-isomorphism problem* takes two undirected graphs G_1 and G_2 and asks whether G_1 is isomorphic (see page 1171 of CLRS) to a subgraph of G_2. Prove that the subgraph-isomorphism problem is NP-complete.

(*Hint*: Remember that to prove NP-completeness you must prove two things—that verification is possible in polynomial time and that the problem in NP-hard.)

5. **Approximation Algorithms**

A *matching* M of a graph $G = (V, E)$ is a subset of the edges of G with the property that no two edges of M share the same node. A matching is *complete* when it contains exactly $\lfloor |V|/2 \rfloor$ edges.

Consider the following algorithm to approximate the TSP for a set of cities V and the complete graph G on vertices V, whose distances satisfy the triangle inequality:

(i) Find a minimum spanning tree T of the cities.

(ii) Find the set S of cities of T of *odd degree* and find the shortest complete matching M of S in G.

(iii) Return the cities in order of an Eulerian tour of the cities that uses only edges from $T \cup M$ (see CLRS, page 623).

Prove that

$$\frac{\text{Cost of Approximate Tour}}{\text{Cost OptimalTour}} \leq \frac{3}{2}.$$

(*Hint*: Any TSP tour, including the optimal tour, contains within it two matchings—alternate edges; consider the matching of lower cost, together with the minimum spanning tree, and use the method of section 35.2.1 in CLRS.)

5. **Approximation Algorithms, continued.**