University of New Mexico
Department of Computer Science

Final Examination

CS 561 Data Structures and Algorithms
Fall, 2010

Name:	
Email:	

- "Nothing is true. All is permitted" - Friedrich Nietzsche. Well, not exactly. *You are not permitted to discuss this exam with any other person.* If you do so, you will surely be smitten. You may consult any *other* sources including books, papers, web pages, computational devices, animal entrails, seraphim, cherubim, etc. in your quest for truth and solutions. Please acknowledge your sources.
- *Show your work!* You will not get full credit if we cannot figure out how you arrived at your answer. A numerical solution obtained via a computer program is unlikely to get much credit, if any, without a correct mathematical derivation.
- Write your solution in the space provided for the corresponding problem.
- If any question is unclear, ask for clarification.

Question	Points	Score	Grader
1	25		
2	25		
3	20		
4	25		
5	25		
Total	120		

1. **Spanning Trees and Amortizations**

   (a) (7 points) Prove that in a graph where all edge weights are the same, that any shortest path tree will also be a minimum spanning tree.

(b) (7 points) Professor Plum postulates that if every edge on an undirected graph has a unique positive weight, then the shortest path tree rooted at v on that graph is always the same as the minimum spanning tree found by Prim's algorithm when seeded initially with the vertex v. Is this correct? If so, prove it. If not, give a counter example.

Consider a data structure over an initially empty list that supports the following two operations. APPEND-NUMBER(x): Adds the number x to the beginning of the list; and MIN-MAX: Traverses the list, computing the minimum and maximum element in the list, and then creates a new list that contains only two numbers, the minimum and maximum of the old list.

i. (4 points) Assume an arbitrary sequence of n operations are performed on this data structure. What is the worst case run time of any particular operation?

ii. (7 points) Show that the amortized cost of an operation is $O(1)$ using the potential method. Make sure to prove your potential function is valid.

2. **Shortest Paths, Commodities and the Moral Torpidity of American Enterprise**

"A million dollars isn't cool. You know what's cool? A billion dollars" - Sean Parker in The Social Network

What if we change the all-pairs, shortest paths problem so there is some cost associated with each intermediate vertex that is traversed in a path? In particular, we are given a graph G that has weights, w, on all the edges and also weights, w', on all the vertices. The weight of a path is the sum of the weights of all edges in that path and the weights of all vertices in the path except for the start and end vertex. For example, the weight of the path v_1, v_2, v_3, v_4 is $w(v_1 \to v_2) + w(v_2 \to v_3) + w(v_3 \to v_4) + w'(2) + w'(3)$. The weights of both edges and vertices may be negative. Your first goal is to design an algorithm that either computes the weight of the shortest path between every pair of vertices if there is no negative weight cycle in the graph, or else if there is a negative weight cycle, correctly returns that this is the case.

You will show how to modify the Floyd Warshall algorithm to solve this new variant of all pairs shortest paths. As in lecture, let $dist(u, v, r)$ be the shortest path from u to v that traverses vertices numbered r or less. Also, assume that the vertices are uniquely labelled from 1 to n

(a) (8 points) Write down a recurrence relation for $dist(u, v, r)$ for this new variant of the shortest paths problem.

(b) (7 points) Write down an algorithm, based on Floyd Warshall that solves this variant of the shortest path problem. Remember your algorithm should return that there is a negative weight cycle if one exists.

(c) (10 points) Now imagine that you're trying to make money in the following variant of the arbitrage problem. There are n commodities $g_1, g_2, \ldots, g_n$ and an n by n table R of exchange rates, such that one one unit of commodity g_i buys $R[i, j]$ units of commodity j. Moreover, there are *taxes* on each commodity given by $t_1, t_2, ..., t_n$ such that if i is an intermediate commodity in a sequence of trades, you are taxed at rate t_i when you convert to commodity i. For example, when you convert to the intermediate commodity "pork bellies", you are taxed at a rate of .05. A *valid* sequence of trades, starts and ends with the same commodity. The *revenue* from a sequence of trades is the amount of the first commodity you are left with if you start with one unit of the first commodity initially and perform the trades in the sequence. For example, if there is a sequence of commodities $g_{i_1}, g_{i_2}, \ldots, g_{i_k}, g_{i_1}$, then the revenue for that sequence is
$R[i_1, i_2] \cdot (1 - t_2) \cdot R[i_2, i_3] \cdot (1 - t_3) \cdots R[i_{k-1}, i_k] \cdot (1 - t_k) \cdot R[i_k, i_1]$

(Note that you are never taxed for the start/end commodity in the valid sequence, and that the revenue may be less than 1.)

Design an algorithm that either 1) if there is a sequence of trades with revenue greater than 1, outputs some revenue greater than 1 that is obtainable; or 2) if there is no valid sequence with revenue greater than 1, outputs the maximum revenue possible from a valid sequence of trades.

3. **Linear Programming**

   You are taking a penalty shot in a soccer game and you must decide whether to kick the ball left or right. At the same time, your opponent, the goalie, must decide whether to dive left or to dive right. Through careful study of your skills and the goalie's skills, you have determined the probabilities of a goal in all cases: $p_{\ell,\ell}$ when you both go left, $p_{r,r}$ when you both go right, $p_{\ell,r}$ when you kick left and your opponent dives right, and $p_{r,\ell}$ when you kick right and your opponent dives left. Note that these probabilities are not necessarily symmetric because of right/left handed/footed-ness.

   Your goal is to to determine a probability distribution over your 2 choices that will maximizes your probability of a goal, *given that your opponent knows your strategy and plays optimally for that strategy*! In particular, you want to compute optimal probabilities q_ℓ of kicking left and $1 - q_\ell$ of kicking right.

   (a) (10 points) Write down a linear program to find your optimal strategy and the probability of getting a goal from that strategy if your opponent plays optimally. Hint 1: If your opponent knows your strategy, her own strategy will either be to dive left with probability 1, or dive right with probability 1. Hint 2: Use a trick similar to the one that we used in designing the linear program for the shortest paths problem.

Now consider a generalization of the problem where the goal is divided into n regions. You must decide on one of the n possible regions to kick the ball and the goalie must decide on one of n possible regions to dive to. You now have for all $1 \leq i, j \leq n$, probability $p_{i,j}$ which is the probability of scoring a goal if you kick to region i and the goalie dives to region j.

(b) (10 points) Write a linear program to determine a probability distribution over your n choices that will maximizes your probability of a goal, given that your opponent knows your strategy and plays optimally for that strategy. Hint: let q_i be your probability of kicking to region i.

4. **Network Flow and Applications**

   A computer s is initially infected with a virus in a directed network and there is a critical computer t that you want to ensure will not become infected. You are able to shut down any computers except for s and t, and you want to prevent infection by shutting down the minimum number of computers such that there is no longer any path from s to t.

   (a) (10 points) Given a directed network G, and special nodes s and t, describe an algorithm to find the minimum number of computers that must be shut down to prevent the spread of infection from s to t.

(b) (15 points) Assume you are given a directed graph G where every vertex has out-degree k and in-degree k for some particular value k. Describe an algorithm to output a collection of cycles such that every vertex in the graph is in exactly one cycle in the collection. Show that your algorithm is correct. Hint: Bipartite Matching.

5. **Randomized Algorithms**

Consider a situation where we have n servers and n clients. The servers all know a message m and the clients want to learn that message. Our goal is to design an algorithm that ensures that all clients learn m, while sending the smallest number of messages possible. We have access to a global random number generator R that generates a number uniformly at random between 1 and $\sqrt{n}$, and which all the servers can read.

Consider the following algorithm:

(a) Each client chooses a subset S of $\sqrt{n}$ of the n servers, uniformly at random from all such subsets. The client then generates $\sqrt{n}$ requests by choosing independently for each $s \in S$, a tag t that is an integer distributed uniformly at random between 1 and $\sqrt{n}$. The client sends each such request (s, t) to the server s

(b) A random number r is generated by the global random number generator and all servers read that number

(c) Every server s considers the requests they have received of the form (s, r). If there are less than $k\sqrt{n}$ such requests, then s sends m to each client that it received such a request from. k is a parameter to be determined later.

Unfortunately, some of the clients are *bad* in that they may disregard the first line of the algorithm, sending out more than $\sqrt{n}$ requests, which may not necessarily be generated randomly. Note that the number of messages sent by each good client and server is always only $O(\sqrt{n})$. In this problem, you will show that even with the bad clients around, and even with this bound on communication costs, the protocol still has a good chance of ensuring all the good clients will learn m.

Call a sever *overloaded* if in the last step of the algorithm, it receives greater than or equal to $k\sqrt{n}$ requests. Note that you can assume that each server receives at most n requests, since if they receive two requests from the same client, they can assume that client is bad and just throw out its requests.

(a) (5 points) Derive an upper bound on the probability that a fixed server is overloaded.

(b) (5 points) Give an upperbound on the expected number of servers that are overloaded. Note: The events that two different servers are overloaded are NOT independent.

(c) (5 points) Now use Markov's inequality to bound the probability that the number of overloaded servers is greater than or equal to $n/6$. If everything is going well, you should be able to show this probability is no more than $6/k$.

(d) (5 points) Now assuming that at most $n/6$ servers are overloaded, calculate the probability that a given client fails to send a request to any server that is not overloaded. Note: The servers that a single client sends requests to are NOT chosen independently (since a given server can not be chosen more than once). You may find the following bound from the book helpful:

$$(x/y)^y \leq \binom{x}{y} \leq (xe/y)^y$$

(e) (5 points) Finally, use union bounds and the above results to bound the probability that *any* good client does not receive the message. For n large, what is a good approximation to the probability of failure?