University of New Mexico
Department of Computer Science

Final Examination

CS 362 Data Structures and Algorithms
Spring, 2006

Name:
Email:

- Print your name and email, *neatly* in the space provided above; print your name at the upper right corner of *every* page. Please print legibly.
- This is an *closed book* exam. You are permitted to use *only* two pages of "cheat sheets" that you have brought to the exam and a calculator. *Nothing else is permitted.*
- Do all the problems in this booklet. *Show your work!* You will not get partial credit if we cannot figure out how you arrived at your answer.
- Write your answers in the space provided for the corresponding problem. Let us know if you need more paper.
- Don't spend too much time on any single problem. The questions are weighted equally. If you get stuck, move on to something else and come back later.
- If any question is unclear, ask us for clarification.

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

1. **True or False**

True or False: (circle one, 2 points each)

(a) **True or False**: The greedy algorithm for fractional knapsack only takes a fractional amount of at most one item. *Solution: True*

(b) **True or False**: If there is an activity with start time later than the finish time of all other activities, then the greedy algorithm for activity selection will always choose this activity.*Solution: True - this activity conflicts with no other activities so will always be chosen.*

(c) **True or False**: Let T be the shortest path tree rooted at x for some graph G. Then for any vertex y, the shortest path from x to y in G is the same as the path from x to y in T.*Solution: True*

(d) **True or False**: Consider a graph $G = (V, E)$, with negative weight edges. We can solve the single source shortest paths problem on G in $O(|V||E|)$ time. *Solution: True. Bellman-Ford takes $O(|V||E|)$ time.*

(e) **True or False**: Consider a graph $G = (V, E)$, with negative weight edges. The fastest time to determine if there is a negative cycle in G is $O(|V||E|)$. *Solution: True. We can do this with Bellman-Ford, taking $O(|V||E|)$ time.*

(f) **True or False**: For a connected graph G, the BFS, DFS, MST and shortest path trees will all have the same number of edges. *Solution: True. If G has n vertices, all spanning trees of G will have $n - 1$ edges.*

(g) **True or False**: Consider a graph $G = (V, E)$, with negative weight edges. We can determine if G contains a negative cycle in $O(|V|^3)$ time. *Solution: True. We can do this with Bellman-Ford, taking $O(|V|^3)$ time.*

(h) **True or False**: Consider an operation "foo" that has an amortized cost of 1. Then over n calls to "foo", the worst case run time of a single call can be $O(n)$. *Solution: True*

(i) **True or False**: We know of a problem that is both in the class NP and in the class P. *Solution: True. Any problem in NP is also in P.*

(j) **True or False**: The TSP approximation algorithm discussed in class finds a tour that has a cost within 2 of optimal on all weighted graphs. *Solution: False. It only guarantees a two approximation if the weights on the graph obey the triangle inequality.*

2. **Short Answer (5 points each) Where appropriate, circle your final answer.**

(a) Solve the following recurrence using the master method: $T(n) = 2T(n/3) + n$

Solution: $f(n) = n$ *and* $af(n/b) = (2/3)n$ *so* $f(n) \geq c * af(n/b)$ *for* $c > 1$. *Thus the root node dominates. Thus* $T(n) = \Theta(n)$.

(b) Solve the following recurrence using annihilators: $T(n) = 3T(n-1) + 2T(n-2)$. Give the solution in general form i.e. do not solve for the constants

Solution: The annihilator is $\boldsymbol{L}^2 - 3\boldsymbol{L} - 2$ *which factors to* $(\boldsymbol{L} - (3 + \sqrt{17})/2)(\boldsymbol{L} - (3 - \sqrt{17})/2)$. *This implies that the general solution is* $T(n) = c_1((3 + \sqrt{17})/2)^n + c_2((3 - \sqrt{17})/2)^n$

(c) Consider the problem CLIQUE discussed in class. If someone proves that $P = NP$, what can be said about this problem? If someone proves that $P \neq NP$, what can be said about this problem?

Solution: If $P = NP$, CLIQUE can be solved in polynomial time. If $P \neq NP$, there is no polynomial time algorithm that can solve CLIQUE.

(d) Prove that $100 * \sqrt{n} = O(n)$.

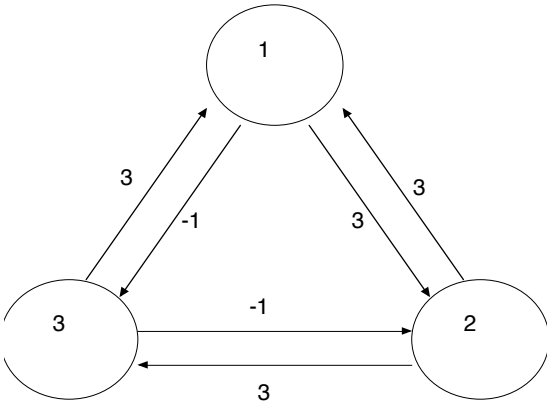
Solution: Goal: Give positive constants c and n_0 such that $100 * \sqrt{n} \leq cn$ for all $n \geq n_0$. The inequality we want then is:

$$\begin{aligned} 100\sqrt{n} &\leq cn \\ 100/\sqrt{n} &\leq c \end{aligned}$$

The left hand side of this inequality is decreasing as n grows large. Thus if we choose $n_0 = 1$ and $c = 20$, it satisfies the inequality for all $n \geq n_0$. In other words, for $c = 20$ and $n_0 = 1$, it's the case that $100\sqrt{n} \leq cn$ for all $n \geq n_0$.

3. **Graph Theory**

Recall that in the Floyd-Warshall algorithm $dist(u, v, r)$ is defined to be the shortest path from u to v where all intermediate vertices (if any) are numbered r or less. For the following graph, fill in the distance arrays computed by Floyd-Warshall for all values of r. *In the distance arrays, let the row be the vertex the path starts at and let the column be the vertex the path ends at.*



$r = 0$

	1	2	3
1			
2			
3			

$r = 1$

	1	2	3
1			
2			
3			

$r = 2$

	1	2	3
1			
2			
3			

$r = 3$

	1	2	3
1			
2			
3			

$r = 0$

	1	*2*	*3*
1	*0*	*3*	*-1*
2	*3*	*0*	*3*
3	*3*	*-1*	*0*

$r = 1$

	1	*2*	*3*
1	*0*	*3*	*-1*
2	*3*	*0*	*2*
3	*3*	*-1*	*0*

Solution:

$r = 2$

	1	*2*	*3*
1	*0*	*3*	*-1*
2	*3*	*0*	*2*
3	*2*	*-1*	*0*

$r = 3$

	1	*2*	*3*
1	*0*	*-2*	*-1*
2	*3*	*0*	*2*
3	*2*	*-1*	*0*

4. **Spanning Trees**

(a) Assume you are given a connected graph G with n nodes and at least n edges. Give an algorithm to remove an edge from G without disconnecting the graph. *Solution: You just need to find a spanning tree for G, using say DFS, and then remove an edge in G that is not in this spanning tree.*

(b) Consider the following traveling repairperson problem. There is a weighted, undirected graph G with all positive edge weights. The nodes of G represent cities and the weights on the edges represent distances between cities. There is a special vertex s in the graph. A traveling repairperson starts at the city s and wants to visit every node in the graph. However, the repairperson needs to travel back to the node s after visiting each city (to get new parts). Give an algorithm to find the shortest total route for the repairperson. Note that the repairperson is allowed to visit each city multiple times.

Solution: The algorithm first finds a shortest path tree, T, rooted at s. The for each node v in the graph, the repair person first travels from s to v along the path given by T and the repair person then travels from v back to s along the same path. The total cost of this route is minimized since we are always taking shortest paths from s to each city, v.

5. **NP-Hardness**

In your first job after taking CS362, you are presented with the following problem. There is a set of k machines at your company. There are n programs that need to be run on these machines where $n > k$. There is a list of conflicts: this is just a list of pairs of jobs, where each pair in the list represents two jobs that can not be run on the same machine. Your goal is to write a program that will assign each job to a single machine such that no two jobs that conflict are assigned to the same machine.

(a) Which problem that we discussed in class is this problem equivalent to? Describe how it is equivalent.

Solution: This problem is equivalent to the graph coloring problem. Each machine is a job. If two jobs conflict, there is an edge between the corresponding nodes. Assigning each job to one of the k machines is equivalent to finding a k-coloring for the graph.

(b) What does this equivalence tell you about the likelihood of being able to successfully write an efficient program to solve this problem?

Solution: Graph coloring is NP-Hard so unless $P = NP$, you will not be able to solve this problem in polynomial time.