Department of Computer Science

Final Examination

CS 362 Data Structures and Algorithms
Fall, 2004

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		
6	20		
Total	120		

1. **True or False**

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

(a) **True or False**: $4^{\log_2 n}$ is $o(n^2)$. *Solution: False.* $4^{\log_2 n} = n^2$.

(b) **True or False**: For some graph G, a BFS tree of G can be the same as a DFS tree of G. *Solution: True. Consider the case where* G *itself is a tree.*

(c) **True or False**: The number of possible subsets of n elements is $O(n^2)$. *Solution: False. It's* 2^n

(d) **True or False**: There is a greedy algorithm for $0-1$ knapsack which runs in $O(n \log n)$ time. *Solution: False. The greedy algorithm fails for* $0-1$ *knapsack on the example we went over in class. The greedy algorithm only works for fractional knapsack.*

(e) **True or False**: If there is an activity with finish time earlier than all other activities, then the greedy algorithm for activity selection will always choose this activity.*Solution: True.*

(f) **True or False**: Consider a graph $G = (V, E)$, with negative weight edges. The fastest time to solve the single source shortest paths problem on G is $O(|V|^3)$. *Solution: False. Bellman-Ford takes* $O(|V||E|)$ *time.*

(g) **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.*

(h) **True or False**: Consider a graph $G = (V, E)$, with negative weight edges. We can solve all pairs shortest paths on G in $O(|V|^3)$ time. *Solution: True. We can do this with Floyd-Warshall, taking* $O(|V|^3)$ *time.*

(i) **True or False**: If there is a polynomial time algorithm for some problem in NP, then all problems in NP can be solved in polynomial time *Solution: False. P is a subset of NP but this does not imply that all problems in NP can be solved in polynomial time.*

(j) **True or False**: We can always edge-color a graph G with maximum degree Δ using no more than $3\lceil \Delta/2 \rceil$ colors. *Solution: True*

2. **Short Answer (4 points each) Where appropriate, circle your final answer.**

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

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

(b) Solve the following recurrence using the master method: $T(n) = 2T(n/4) + 1$

Solution: $f(n) = 1$ *and* $af(n/b) = 2$, *so* $af(n/b) \geq cf(n)$ *for* $c > 1$. *Thus the leaf nodes dominate the recurrence. The total cost of the leaf nodes is* $2^{\log_4 n} = 2^{\log_2 n/\log_2 4} = \sqrt{n}$, *so* $T(n) = \Theta(\sqrt{n})$

(c) 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* $(\boldsymbol{L} - 2)(\boldsymbol{L} - 1)$. *This implies that the general solution is* $T(n) = c_1 + c_2 2^n$

(d) In your own words, define what it means for a problem A to be in the class NP.

Solution: A is in the class NP if it can be "verified" in polynomial time. In other words, if we were given a proof of the solution, we can verify this proof in time polynomial in the input size.

(e) In your own words, define what it means for a problem A to be NP-Hard.

Solution: If A can be solved in polynomial time, then all *problems in the class NP can be solved in polynomial time.*

3. **Asymptotic Notation**

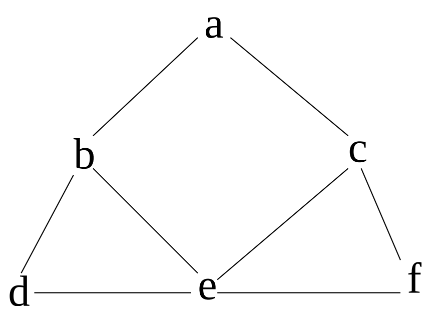
Prove that $n^2 = \Omega(n \log n)$. *Solution: Goal: Give positive constants c and n_0 such that $cn \log n \leq n^2$ for all $n \geq n_0$. The inequality we want then is:*

$$\begin{aligned} cn \log n &\leq n^2 \\ c \log n &\leq n \\ c &\leq \frac{n}{\log n} \end{aligned}$$

The right hand side of this inequality is increasing as n grows large. Thus if we choose $n_0 = 2$ and $c = 2$, it satisfies the inequality for all $n \geq n_0$. In other words, for $c = 2$ and $n_0 = 2$, it's the case that $cn \log n \leq n^2$ for all $n \geq n_0$

4. **Spanning Trees**

(a) Give BFS and DFS trees for the following graph. Assume that BFS and DFS are intially called with the vertex a and that the edges are stored in the adjacency lists in alphabetical order. Make sure that you label which tree is a BFS tree and which is a DFS tree.



Solution: The DFS tree is the following edges $\{(a,b),(b,d),(d,e),(e,c),(c,f)\}$*. The BFS tree is the following edges:* $\{(a,b),(a,c),(b,d),(b,e),(c,f)\}$

(b) Assume you are given an undirected, connected graph G with n nodes and n edges. Give an algorithm to delete an edge in G which will not disconnect the graph.
Solution: Let T *be the BFS (or DFS) tree of* G*. This tree will have* $n-1$ *edges; we can safely delete the edge in* G *which is not in* T*.*

5. **Graph Theory**

Your first assignment in your first job after college is wiring up the office computers. With fond memories of CS362, you decide to model the office as a weighted, undirected, connected graph $G = (V, E)$. Each computer in the office is a vertex in V and the distance between any pair of computers x and y is given by the weight of the edge (x, y) in E. A *wiring* is a subset of the edges in E that connects all the computers. The *cost* of the wiring is just the sum of the weights of all the edges in the wiring.

Your boss is both cheap and self-centered. He demands that you find a wiring with the properties that: 1) it costs no more than any other wiring and 2) his own computer, represented by the special node b in V, is connected to the network by the shortest possible edge. In other words, one of the lightest-weight edges incident to b in G must be in the wiring.

(a) Give an algorithm which meets both demands of your boss. Show that your algorithm is correct.

Solution: All you need to do is call Prim's algorithm on G with the node b as the seed node. The first edge added will be one of the shortest edges incident to b. The algorithm finds a MST which will be a least-cost wiring.

(b) Now imagine that you have two bosses with computers represented by the nodes b_1 and b_2. They demand that you find a wiring with the properties that : 1) it costs no more than any other wiring and 2) each of them has their computer connected to the network with the shortest edge possible. Can you always meet both demands? If so, prove it. If not, give a weighted graph G and nodes b_1 and b_2 for which both demands can not be met.

Solution: You can always meet both demands. Let $e_1 = (b_1, x)$ *be a minimum weight edge incident to* b_1 *and let* $e_2 = (b_2, y)$ *be a minimum weight edge incident to* b_2*. Using the cut* $(b_1, V - \{b_1\})$ *and the safe-edge theorem, we can see that the set* $A = \{e_1\}$ *is a subset of some MST. Now there are two cases.*
Case 1: $x \neq b_2$*. In this case, the cut* $(b_2, V - \{b_2\})$ *respects* A*. Hence we can safely add* e_2 *(a light edge for this cut) to the set* A*. This implies that there is some MST containing both* e_1 *and* e_2*.*
Case 2: $x = b_2$*. In this case, we know that the cut* $(\{\{b_1, b_2\}, V - \{b_1, b_2\})$ *respects the set* A*. Hence we can safely add a light edge for this cut to* A*. Note that* $w(e_2) \leq w(e_1)$ *since* e_2 *is a minimum weight edge incident to* b_2*. Thus the weight of* e_2 *is no more than the weight of any edge incident to* b_1*. Thus,* e_2 *is a light edge for this cut. This implies that there is some MST containing both* e_1 *and* e_2*.*

6. **Approximation Algorithms**

Recall that a k-coloring of an undirected graph G is an assignment of one of k colors to each of the vertices so that no two adjacent vertices have the same color. In class we showed that the problem of determining whether a graph has a k-coloring is NP-Hard. In this problem, you will be designing an approximation algorithm for the optimization version of coloring.

Consider a graph G with maximum degree Δ. Give an algorithm to color G with $\Delta + 1$ colors. Hint: Use recursion (you can reduce to a smaller problem by removing a single vertex). Show that your algorithm is correct.

Solution: Color(G)*:*

(a) If G is just a single vertex, color that vertex with one color.

(b) Otherwise, pick any vertex v in G and let G' be G with the vertex v removed (note that G' has degree no more than Δ)

(c) Call Color(G') to color all vertices except for v with no more than $\Delta + 1$ colors

(d) Finally color v with a different color than its neighbors. v has at most Δ neighbors so there will be at least one free color to use to color v.