University of New Mexico
Department of Computer Science

Final Examination

CS 561 Data Structures and Algorithms
Fall, 2012

Name:	
Email:	

- This exam lasts 2 hours. It is open book and open notes but no electronic devices are permitted.
- *Show your work!* You will not get full credit if we cannot figure out how you arrived at your answer.
- Write your solution in the space provided for the corresponding problem.
- If any question is unclear, ask for clarification.

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

1. **Short Answer**

   Answer the following questions using *simplest possible* θ notation. Draw a box around your final answer. No need to justify answers for problems on this page.

   (a) If a data structure supports an operation "foo", such that n calls to "foo" take $\theta(n \log n)$ time, what is the amortized cost of "foo"?

   (b) How many bits are needed to write the number $n!$ in binary?

   (c) $\sum_{i=1}^{n} \frac{i}{n}$

   (d) Fastest time to find the *longest* path in a directed graph, G, with no cycles, and where all edge weights are positive. Assume there are n nodes in G and m edges and give your answer in terms of n and m.

   (e) Suppose you have a hash function that hashes n items into an array of length m. What is the expected number of colliding pairs of elements (asymptotically, as a function of n and m)?

(f) Assume you have a graph with n nodes and m edge. Assume further that you have a set-union data structure which somehow ensures all operations are $O(1)$ amortized time. What is the new runtime of Kruskal's algorithm?

(g) Assume you have a graph with n nodes and m edge. Assume further that you have a heap data structure which somehow ensures all operations are $O(1)$ amortized time. What is the new runtime of Prim's algorithm?

(h) Assume you have a graph with n nodes and m edge. What is the fastest time to determine if the graph has a cycle of odd length?

(i) Imagine you have two skip lists, each containing n distinct items. What is the expected time to merge them into a new skip list over all $2n$ of the distinct items? The new skip list should be able to support all skip-list operations with the same time costs as for any skip lists over $2n$ items.

(j) Solution to the following recurrence relation: $f(n) = 2f(n-1) - f(n-2) + 1$.

2. **Short Answer**

   (a) Prof. Goofus conjectures that if there is a graph G such that 1) G contains all possible edges (i.e. it is a clique) and 2) each edge in G has a unique weight, then the minimum spanning tree for G will always contain the $n - 1$ lightest edges of G (where n is the number of nodes in G). Show Goofus is wrong by drawing a weighted graph and giving a MST for that graph that disproves the conjecture.

(b) Consider the following problem about sending a message in a wireless network that is under attack. Assume there are n times steps, and Alice sends in $c\sqrt{n}$ of these steps selected uniformly at random, and Bob listens in $c\sqrt{n}$ of these steps selected uniformly at random. Further there is an adversary that jams $n/2$ of the time steps selected uniformly at random. A step is said to be *good* if 1) the step is not jammed; 2) Alice sends in the step; and 3) Bob listens in the step. What is the expected number of good time steps? How large should c be to ensure that the expected number of good steps is least one?

Hint: Linearity of Expectation.

3. **Graph Coloring** A 3-regular undirected graph is a graph, $G = (V, E)$ where every node has exactly 3 neighbors. Recall that to properly color a graph, you must assign a color to each vertex of the graph so that every edge in E connects vertices with two different colors.

   Prove by induction that any 3-regular graph can be colored with 4 colors. Prove this by induction on n, the number of vertices in G. Don't forget to include the Base Case (BC), Inductive Hypothesis (IH) and Inductive Step (IS). Hint: Remember in the IS to create a **smaller** subproblem and to use the IH to solve it.

4. **Debs and Poodles**

There are n debutants (debs) and m poodles. Each deb i has a quota n_i of poodles that she can adopt. Each poodle j has a set S_j of debs it prefers. Your goal is to assign poodles to debs in such a way that no deb exceeds her quota of poodles, and each poodle j is matched with exactly one deb in its set S_j.

(a) Show how you can use Max-flow to find an assignment if possible. What is the worst case runtime of your algorithm (you can use $F(v, e)$ to denote the runtime of max flow on a graph with v nodes and e edges).

(b) You run the algorithm from part a) and unfortunately do not find a assignment that meets the criteria. The debs believe your algorithm is correct but they believe your implementation of it has a bug. Based on your understanding of max-flow, how can you prove to the debs that your program is correct and that no assignment is possible? Your proof strategy must work for every instance where there is no assignment.

5. **Sensor Networks**

Your boss presents you with the following problem. You are given a graph $G = (V, E)$ that represents a sensor network. The nodes of V represent sensors, and each node $v \in V$ has a positive weight $w(v)$ that indicates the importance of the data collected by that sensor. There is an edge $(u, v) \in E$ if sensors u and v are neighbors in the network, and in such a case, u and v can not both be turned on because of interference issues. You want to turn on a subset of the nodes with maximum total weight that avoids any interferences.

Your goal is thus to find a subset $S \subseteq V$ such 1) no two nodes in S are neighbors; and 2) the sum of the weights of all nodes in S is maximized among all such sets.

Your boss thinks this problem is pretty easy, but decides to be generous and give you the entire weekend to work on it.

(a) Show that this problem is NP-Hard by a reduction from one of the following NP-Hard problems: 3-SAT, CLIQUE, INDEPENDENT SET, VERTEX COVER, 3-COLORABLE or HAMILTONIAN CYCLE.

(b) Now your boss decides to create a new easier problem by assuming that G is a path (see Figure 1). He claims that the following greedy algorithm will find an optimal set S for this new problem. GREEDY initially starts with an empty set S. Then it repeats the following until the set V is empty: 1) find a vertex $v \in V$ of maximum weight; 2) add v to S; and 3) remove all neighbors of v from V.

Show your boss is wrong by giving a problem instance (a graph G that is a path, and weights for the vertices of G), such that GREEDY does not return the optimal solution for that problem.

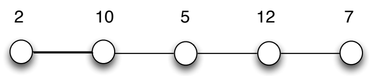

Figure 1

(c) Now describe how to solve this new problem (where G is a path) using dynamic programming. Let the vertices in the path have labels $v_1, \ldots, v_n$. Let $m(i)$ be the max weight of the set S that is obtainable by considering only vertices v_1 through v_i. First give the recurrence relation (and base case(s)) for $m(i)$ below.

(d) Now describe how to create a dynamic program to find S. What is the run time of your algorithm as a function of n?