University of New Mexico
Department of Computer Science

Final Examination

CS 561 Data Structures and Algorithms
Fall, 2017

Name:	
Email:	

- This exam lasts 2 hours. It is closed book and closed notes with no electronic devices. However, you are allowed 2 pages of cheat sheets.
- *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. Briefly justify your answer where appropriate.

   (a) Solution to the recurrence $T(n) = T(n-1) + 1$

   (b) Solution to the recurrence $T(n) = 3T(n/3) + n$

   (c) Solution to the recurrence: $f(n) = 5f(n-1) - 5f(n-2) + 1$.

   (d) Suppose you throw n items uniformly at random into n bins. What is the expected number of pairs of items that will be in the same bin (asymptotically as a function of n)?

   (e) You toss n sensors onto a k-regular graph with n nodes. A pair of sensors is said to *collide* if they fall on the same node or on two nodes that are neighbors. What is the expected number of sensors that collide (asymptotically as a function of n and k)?

(f) Runtime of fastest algorithm determine if an undirected graph with n nodes and m edges is bipartite. Recall that a bipartite graph is one where the nodes can be partitioned into two sets such that all edges have endpoints in both sets.

(g) Compute the longest common subsequence of a pair of strings each of length n.

(h) There is a function on a data structure with amortized cost $O(1)$, but worst case cost $O(n)$. If you call this function exactly $\sqrt{n}$ times on a newly initialized data structure, what is the worst case cost of the sum of all of these calls?

(i) Time to find a *maximum* weight spanning tree on a graph with n nodes and m edges.

(j) Time to solving the single source shortest paths problem on a graph with n vertices and m edges, where there may be negative edge weights, but no negative cycles .

2. **Reductions and Induction**

The first two questions deal with the following problem. **Set Cover:** In the Set Cover problem, you are given a ground set $U = \{x_1, \ldots x_n\}$ and a collection of m subsets of $S_i \subseteq U$, and an integer k. The question is, can you select a collection C of at most k of these subsets such that each element of U is in at least one subset in the collection?

(a) (3 points) Show that Set Cover is in NP by showing that if the answer to the question is yes, there exists an efficiently verifiable proof of that fact.

(b) (7 points) Show that Set Cover is NP-Hard by a reduction from one of the following: 3-SAT, VERTEX-COVER, INDEPENDENT-SET, 3-COLORABLE, HAMILTONIAN-CYCLE, or CLIQUE.

(c) (10 points) Prove by induction that any binary tree of height h has at most $2^{h+1} - 1$ nodes. Don't forget to include the base case, inductive hypothesis and inductive step. Hint: Do induction on h. For the inductive step, what do you get if you remove the root?

3. **Expensive Paths**

Let $G = (V, E)$ be a directed graph with the cost of each edge e given by $c(e)$. Let $k > 0$ be an integer, and let s and t be vertices of V.

(a) (8 points) Give an algorithm to find the cost of the most expensive path from s to t that uses exactly k edges. If a path of k edges does not exist from s to t, you should return $-\infty$.

(b) (2 points) What is the runtime of your algorithm?

(c) (8 points) Now give an algorithm to find the cost of the most expensive path from s to t that uses *at most* k edges. If a path of k or fewer edges does not exist from s to t, you should return $-\infty$.

(d) (2 points) What is the runtime of your new algorithm?

4. **Knapsack**

Recall that in the $0-1$ Knapsack problem, there are n different items. Each item i has a weight w_i and a value v_i, and the knapsack can hold total weight W. Your goal is to find a subset of items with maximum value that can fit in the knapsack. $0-1$ Knapsack is NP-Hard, so in this problem, you will design an approximation algorithm for it based on rounding of a linear programming.

(a) (10 points) Write an integer program that maximizes the total value of items in the knapsack.

(b) (5 points) Now describe how you would change this to a linear program, and round the solution to this linear program to get a subset of items to put in the knapsack. Your subset should have expected weight at most W and an expected value of OPT, where OPT is the best value for a knapsack holding W weight. (Your answer should be $3-4$ sentences.)

(c) (5 points) Your boss now wants a solution that with high probability has weight less than W. Now the value of your knapsack just needs to be $OPT/2$, where OPT is the best value for a knapsack holding weight $W/2$, and you know that $OPT = \frac{3}{4}V$ where $V = \sum_{i=1}^{n} v_i$. Briefly describe a randomized algorithm to achieve your bosses demands with probability of error that is polynomially small in n, and sketch why your algorithm works. HINT: Use Markov's inequality, a union bound, and a rounding based on a variant of your linear program. When you use Markov's to bound the value, use it on the items *not* put in the knapsack.

5. **Saia Trucking**

There are n different trucking routes from Albuquerque to Los Angeles, and you must select one route each day. For each day t, $1 \leq t \leq T$, there is probability vector p_t, such that $p_t[i]$ is the probability of having an accident on route i on day t. This probability vector is revealed only *after* you choose your route for day t.

Let $OPT = \min_{1 \leq i \leq n} \sum_{t=1}^{T} p_t[i]$, i.e. the minimum over each route of the sum of probabilities of failure on that route over all days. Note that OPT gives the minimum expected number of accidents over all routes (by linearity of expectation).

(a) (5 points) You want to create an online gradient descent algorithm to achieve an expected number of accidents close to OPT. On day t, let y_t be a vector of probabilities over the routes selected by your algorithm, and let $f_t(y_t)$ be the convex function to minimize in online gradient descent. What is $f_t(y_t)$? Hint: It'll be a function of both y_t and the vector p_t.

(b) (3 points) Argue that your function f_t is always convex (1 sentence).

(c) (4 points) Let ALG be the expected number of accidents from your online gradient descent algorithm over all T days. Let D be the diameter of your convex search space, and let G be the maximum norm of a gradient vector. Give an upper bound for ALG as a function of G, D, T and OPT.

(d) (4 points) Let D be the diameter of your convex search space for this problem. What is the value of D?

(e) (4 points) Let G be the maximum norm of the gradient vector for any function f_t. What is the value of G? Hint: Recall that the entries of p_t are values between 0 and 1