University of New Mexico
Department of Computer Science

Final Examination

CS 561 Data Structures and Algorithms
Fall, 2016

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 $T(n) = T(n-1) + n$ *Solution:* $\theta(n^2)$

   (b) Solution to the recurrence $T(n) = 2T(n/2) + \log n$ *Solution:* $\theta(n)$ *by Master Method.*

   (c) Solution to the recurrence: $f(n) = 3f(n-1) - 2f(n-2)$. *Solution:* $\theta(2^n)$ *or* $\theta(c_1 2^n + c_3)$. *Annihilator is* $L^2 - 3L + 2 = (L-2)(L-1)$.

   (d) Suppose you have a hash function that hashes n items into an array of length $\sqrt{n}$. What is the expected number of items in the first bin? *Solution:* $\theta(\sqrt{n})$

   (e) Suppose you have a hash function that hashes n items into an array of length n. What is the expected number of colliding pairs of items (asymptotically, as a function of n)? *Solution:* $\theta(n)$

(f) Runtime of fastest algorithm to solve the fractional knapsack problem over n items if each item has a value independent distributed uniformly between 0 and 1, and a weight that is 1? *Solution:* $\theta(n)$ *(use bucket sort!)*

(g) What is the expected number of nodes at the $(1/2)\log n$ level of a skip list *Solution:* $\theta(\sqrt{n})$

(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 2 times on a newly initialized data structure, what is the worst case cost of the sum of those two calls? *Solution:* $O(1)$

(i) Your friend uses the Union-Find data structure to count the number of connected components in a graph $G = (V, E)$. First, they call Make-Set on every node. Then, they go through each edge (u, v) in E, and if Find-Set(u) does not equal Find-Set(v), they call Union on Find-Set(u) and Find-Set(v). Assume $|V| = n$ and $|E| = m$, that the amortized cost for all operations on the data structure is $O(\log^* n)$ and that the worst case cost of any operation is $O(\log n)$. Then what is the worst case runtime of your friend's algorithm? *Solution:* $\theta((m + n)\log^* n)$

(j) You have computed a max flow f in a network G with n nodes and m edges. What is the cost of the most efficient algorithm to now find a min s, t cut? *Solution: Just use BFS or DFS to find S which equals all vertices reachable from s, and T which is all remaining vertices. This is the min s, t cut*

2. **Reductions and Induction**

Show that the next two problems are NP-Hard via a reduction from one of the following problems: 3-SAT, VERTEX-COVER, INDEPENDENT-SET, 3-COLORABLE, HAMILTONIAN-CYCLE, or CLIQUE

(a) (5 points) **Suspicious Group:** There is a set of n attacks on a computer network. For each attack, there is some subset of m users who were active during that time. You want to find a *suspicious group* of k users such that for each attack, there was some user in the suspicious group active during that attack. *Solution: Reduce from Vertex Cover*

(b) (5 points) **Diverse Subset** There is a set of n customers, and a set of m items, where each customer has purchased some subset of the m items. For marketing purposes, you want to find a diverse subset of customers, S such that no two customers in S have bought the same product. In the Diverse Subset Problem, you are given an integer $k \leq n$ and want to determine if there is a diverse subset of k customers. *Solution: Reduction from Independent Set.*

(c) (5 points) Prove by induction that any tree over n nodes has $n-1$ edges for any $n \geq 1$. Don't forget to include the base case, inductive hypothesis and inductive step. Hint: For the inductive step, what do you get if you remove a leaf node from a tree over n nodes?
Solution: BC: n=1. A tree with 1 *node has* 0 *edges. IH: For* $0 \leq j < n$*, all trees over* j *nodes have* $j-1$ *edges. IS: Consider some tree,* T *with* n *nodes. Remove a leaf node to get a tree over* $n-1$ *nodes. By the I.H., this tree has* $n-2$ *edges. Hence* T *has* $n-1$ *edges*

(d) (5 points) You have an unbounded supply of stamps with values v_1, v_2, v_3 and a target value x. Give an algorithm to determine if it is possible to achieve the value x with your stamps. In particular, for any integer i, let $m(i) = 1$ if it is possible to achieve value i with the stamps and 0 otherwise, and show how to compute $m(x)$. For example, if the stamps have values 3, 6, and 10, then $m(17) = 0$ and $m(29) = 1$.

Solution: For all $j < 0$, $m(j) = 0$, $m(0) = 1$. *For all* $j > 0$, $m(j) = \max(m(j - v_1), m(j - v_2, m(j - v_3))$.

3. **Graph Escape**

In the **graph escape problem**, you are given a directed graph $G = (V, E)$ along with a set of occupied vertices X, and a set of safe vertices Y. You want to find paths from every vertex in X to some vertex in Y such that none of these paths share an edge.

(a) (5 points) Describe an algorithm to solve this problem. *Solution: Create a flow network where there is an edge from s to every node in X with capacity 1, an edge from every node in Y to t with capacity ∞ and all other edges in G are given capacity 1. We then ask if there is a flow with value $|X|$ in this network.*

(b) (7 points) Now imagine the problem is changed so that none of the paths can share a **node**. Give an algorithm to solve this new escape problem.*Solution: Create the same network as above, but replace every node* $v \in V$ *with two nodes* v_i *and* v_o*. For all directed edges* $u \rightarrow v \in E$*, create an edge* $u_o \rightarrow v_i$ *with capacity* 1*.*

(c) (8 points) Now imagine the escape starts at time 0 with a person at each vertex in X, and in every time step, a person **must** traverse some edge. You want to determine if there are paths for each person such that 1) no two paths traverse the same edge in the same time step; 2) all paths end at a safe vertex in Y; and 3) all paths end in at most $n = |V|$ time steps. Concisely describe an algorithm for this problem. *Solution: Create $n+1$ copies of G, $G_0, \ldots G_n$. Add edges with capacity 1 from s to each vertex X in G_0. For each $u \to v$ in G, and for each $0 \le i \le n-1$, create edge of capacity 1 from u in G_i to v in G_{i+1}. Finally, create edges of infinite capacity from all vertices in Y in each copy G_i to t.*

4. **Parenthesis Puzzle**

Consider a puzzle where you are trying to place parenthesis in a math formula in order to maximize the value. The formulas always contain positive numbers, and the two operators addition (+) and multiplication (·). For example, parenthesize $6+0\cdot 6$, has solution $(6+0)\cdot 6 = 36$. Another example is that parenthesize: $.1 + .1 \cdot .1$ has solution $.1 + (.1 \cdot .1) = .11$.

In general, the input to your problem is $x_0, o_0, x_1, o_1, \ldots o_{n-1}, x_n$, where the x_i are positive numbers and the o_i are operators that are either addition or multiplication.

(a) (15 points) For $0 \le i \le j \le n$, let $m(i,j)$ be the maximum value achievable by optimally parenthesizing the formula $x_i, o_i, \ldots, o_{j-1}, x_j$. Write a recurrence relation for $m(i,j)$. Don't forget the base case(s). *Solution:* $m(i,j) = \max_{i \le k \le j-1} m(i,k) o_k m(k+1,j)$, *and* $m(i,i) = x_i$

(b) (5 points) Briefly describe a dynamic program based on the above recurrence relation. What is the runtime of your program? *Solution: Use a table, proceed by increasing length of substring. Keep pointers to value of* k *that achieves each optimal value. Runtime is* $\theta(n^3)$

5. **Challenge Problem**

In the MAX-EDGE-COVER problem, you are given a graph $G = (V, E)$ and an integer k, and you want to find a set of k vertices that **maximizes** the number of covered edges in G. In this problem, you will develop an approximation algorithm for MAX-EDGE-COVER, based on a randomized rounding of a linear program (LP).

(a) (6 points) Write an integer program for MAX-EDGE-COVER. Hint: Create variables x_i for every every vertex, whose value depends on whether or not the vertex is in a cover, and additional variables z_j for every edge, whose value depends on whether or not the edge is covered. *Solution: Maximize* $\sum_\ell z_\ell$ *subject to* $x_i, z_\ell \in \{0, 1\}$ *for all vertices* i x_i, *and* $\sum_i x_i$ *Finally, for all edges* ℓ *between vertices* i *and* j, *there is a variable* z_ℓ *such that* $z_\ell \leq x_i + x_j$ *and* $z_\ell \leq 1$.

(b) (8 points) Now consider a relaxation of your integer program above to a linear program(LP), where the x_i variables can take on real numbers in the range 0 to 1. Let x_i^* and z_ℓ^* be the solution found by the LP. For each vertex i, with probability x_i^*, include vertex i in the cover. Let OPT be the optimal max edge cover possible for G using k vertices. Give a good lower bound on the expected number of edges covered by your rounding. Hint: Recall the geometric/arithmetic mean inequality: $\sqrt{xy} \leq (1/2)(x+y)$. *Solution: For each edge* j*, let* Y_ℓ *be a random variable that is* 1 *if the edge is covered in the rounding and* 0 *otherwise. Let the edge* ℓ *be between vertex* i *and vertex* j*. Then* $E(Y_\ell) = 1 - (1-x_i)(1-x_j) = x_i + x_j - x_i x_j \geq x_i + x_j - 1/4(x_i + x_j)^2$*. Where the last step holds by geometric/arithmetic inequality. Then we have* $x_i + x_j - 1/4(x_i+x_j)^2 \geq z_\ell - 1/4 z_\ell^2 \geq 3/4 z_\ell$ *where the last step holds since* $z_\ell^2 \leq z_\ell$*. Finally, by the linearity of expectation, the expected number of edges covered is* $\sum_\ell (3/4) z_\ell = (3/4) OPT$*.*

(c) (6 points) Imagine that in the solution to you LP, $\sum_\ell z^*_\ell = |E|$. After the randomized rounding described above, let X be a random variable giving the number of vertices included in your cover, and let Y be a random variable giving the number of edges covered. Show that with constant probability, $X \leq 2k$ and $Y \geq |E|/3$. Hint: Markov's inequality and Union bound.*Solution: Let X be the number of vertices in the rounding. Then $E(X) = k$, and by Markov's inequality, $Pr(X \geq 2E(X)) \leq 1/2$. Next, let Z be the number of edges that are *not* covered. By the previous problem, $E(Z) = |E|/4$, and so by Markov's, $Pr(Z \geq (8/3)E(Z)) \leq 3/8$ (note that $(8/3)(|E|/4) = (2/3)|E|$). Now by a Union bound, the probability that either of these bad events happen is no more than $1/2 + 3/8 \leq 7/8$.*