University of New Mexico
Department of Computer Science

Final Examination

CS 561 Data Structures and Algorithms
Fall, 2018

Name:
Email:

- This exam lasts 120 minutes. It is closed book and notes, and no electronic devices are permitted. However, you are allowed to use 2 pages of handwritten "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 (2 points each)**

   Answer the following questions using the *simplest possible theta* notation. Assume as usual, that $f(n)$ is $\theta(1)$ for constant values of n.

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

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

   (c) Solution to the following recurrence relation: $f(n) = 4f(n-1) - 3f(n-2)$. *Solution: Annihilator is* $(L-3)(L-1)$. *So solution is* $\theta(3^n)$.

   (d) Expected time to find the minimum item in a skip list with n items? *Solution:* $\theta(1)$

   (e) Assume an operation, FOO, has amortized cost $O(1)$ but worst case cost $O(n)$. What is the worst case time for n total calls to FOO? *Solution:* $\theta(n)$

(f) Runtime of fastest algorithm to compute the shortest path between two nodes in a weighted graph with n nodes and m edges, when the graph does not have negative edges? *Solution:* $O(m + n \log n)$

(g) Runtime of fastest algorithm to find whether a weighted graph, with negative edges, has a negative cycle? *Solution:* $O(nm)$

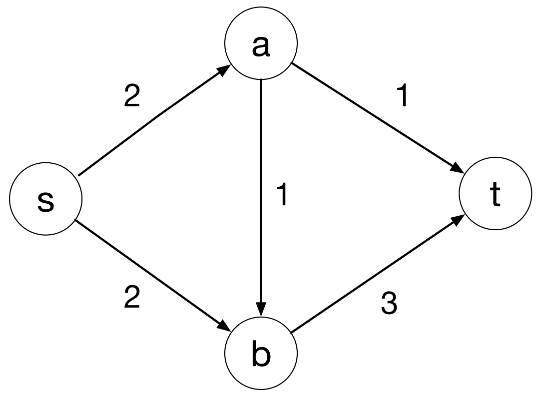
(h) You throw n items into n bins independently and uniformly at random. What is the expected number of pairs of items where both items in the pair fall in the same bin? *Solution:* $\theta(n)$

(i) Using Markov's inequality in the last problem, what is an upper bound on the probability that there are $\theta(n^2)$ pairs of items where both items in the pair fall in the same bin? *Solution:* $\theta(1/n)$

(j) You throw n items into n^4 bins independently and uniformly at random. Using a union bound, what is an upper bound on the probability that *any* pair of items fall in the same bin? *Solution:* $\theta(1/n^2)$

2. **Induction and Linear Programming**

(a) (10 points) Give a linear program to find the maximum flow that can be sent from s to t in this graph. The numbers on each edge represents the capacity of that edge.
Hint: For each edge $u \to v$, let $f_{u\to v}$ be the flow over that edge.



Solution: Maximize: $f_{s\to a} + f_{s\to b}$
Subject to the constraints that:
$0 \le f_{s\to a} \le 2$
$0 \le f_{s\to b} \le 2$
$0 \le f_{a\to b} \le 1$
$0 \le f_{a\to t} \le 1$
$0 \le f_{b\to t} \le 3$
and the conservation of flow constraints:
$f_{s\to a} = f_{a\to b} + f_{a\to t}$
$f_{s\to b} + f_{a\to b} = f_{b\to t}$

(b) (10 points) Prove via induction that any 3-regular graph can be colored with at most 4 colors. Recall that a 3-regular graph is one where every node has degree 3. A *coloring* of a graph G is an assignment of a color to each node in G such that the endpoints of each edge in G are assigned different colors. Don't forget to include BC, IH and IS in your proof.

Hint: Perform induction on, n, the number of nodes in G. In the IS, think about how to make G smaller, so that you can use the IH.

Solution: BC: When $n = 1$ *clearly we can color the graph with* 1 *color, which is at most* 4.

IH: For any graph with j *nodes where* $j < n$, *we can color the graph with at most* 4 *colors.*

IS: Let G *be a graph with* n *nodes, and let* G' *be* G *with an arbitrary node* v *and all incident edges removed. By the IH,* G' *can be colored with at most* 4 *colors. To color* G, *just use this coloring of* G' *and then color* v *with whichever color doesn't appear on its* 3 *neighbors. Thus* G *can be* 4*-colored.*

3. **SENSOR-PLACEMENT**

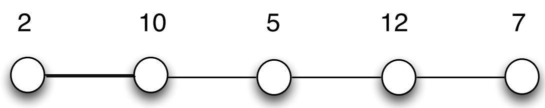
In the SENSOR-PLACEMENT problem, you are given a graph $G = (V, E)$ that represents a sensor network. The nodes of V represent sensors, and each node $v \in V$ has a positive weight $w(v)$ that indicates the importance of the data collected by that sensor. There is an edge $(u, v) \in E$ if sensors u 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.

Your goal is to find a subset $S \subseteq V$ such that (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.

(a) (6 points) Show that SENSOR-PLACEMENT is NP-Hard by a reduction from one of the following: 3-SAT, CLIQUE, INDEPENDENT SET, VERTEX COVER, 3-COLORABLE or HAMILTONIAN CYCLE. *Solution: This is NP-Hard by a reduction from INDEPENDENT-SET. Let $G = (V, E)$, k be an INDEPENDENT-SET problem. Imagine we have an algorithm, A, that can solve the above problem. Then assign weight* 1 *to each node in V and feed this problem to algorithm A. G has an independent set of size k iff A returns a set S with total weight at least k.*

(b) (4 points) Now your boss decides to create a new easier problem by assuming that G is a path (see figure). 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, i.e. a graph G that is a path, and weights for the vertices of G, for which GREEDY does not return the optimal solution.



Solution: Consider the graph $V = \{a, b, c\}$ *where* $E = \{(a, b), (b, c)\}$ $w(a) = w(c) = 2$, $w(b) = 3$. *Greedy will always take the node* b*, but the optimal solution takes the nodes* a *and* b.

(c) (6 points) 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.

Solution: $m(0) = 0$, $m(1) = w(v_1)$. *For all* $i > 0$, $m(i) = \max(m(i-1), m(i-2)+w(v_i))$.

(d) (4 points) Now describe how to create a dynamic program to find S. What is the run time of your algorithm as a function of n?

Solution: It is necessary to have an array m *that is filled in from left to right using this recurrence above. We can also create backward edges from* i *to either* $i-1$ *or* $i-2$ *depending on whether or not the maximum value for* $m(i)$ *was achieved using* $m(i-1)$ *or* $m(i-2)$*. Then to reconstruct the set* S*, we simply start at the value* $m(n)$ *and follow the edges backwards to* $m(0)$*. Every time we visit a vertex* v_i *in this path, we output it as being part of* S*. The runtime of this algorithm is* $O(n)$*.*

4. **Fitting a Line**

In this problem, you want to use gradient descent to find the best line that fits a collection of data. You have a collection of n data items, where for $1 \leq i \leq n$, item i is a tuple (u_i, v_i). Your goal is to find a a line $mx + b$ that minimizes

$$\sum_{i=1}^{n}((mu_i + b) - v_i)^2$$

Assume that m and b are both in the range -10 to 10.

(a) (3 points) In one sentence, describe the convex search space κ (dimensions and boundaries)? *Solution: κ is a 2 dimension square with boundaries defined by the points* $(-10, -10)$ *and* $(10, 10)$

(b) (3 points) What is D, the diameter of κ? You can leave square roots in your answer. *Solution: It is* $|(-20, -20)| = \sqrt{800}$

(c) (2 points) What is the function $f(m, b)$ that you are minimizing? *Solution:* $f(m, b) = \sum_{i=1}^{n}((mu_i + b) - v_i)^2$

(d) (4 points) What is the gradient of $f(m, b)$? *Solution:* $\frac{\partial f}{\partial m} = \sum_{i=1}^{n} 2((mu_i + b) - v_i)u_i$; *and* $\frac{\partial f}{\partial b} = \sum_{i=1}^{n} 2((mu_i + b) - v_i)$

(e) (4 points) Assume that for all $1 \leq i \leq n$, u_i and v_i are both in the range -10 to 10. What is G, the max norm of ∇f as a function of n? Feel free to leave squares and square roots in your answer. *Solution: From the above,* $\frac{\partial f}{\partial m}$ *has value at most* $2(1200)$ *and* $\frac{\partial f}{\partial b}$ *has value at most* $2(120)$. *Thus the max norm of the gradient vector is* $|(2400n, 240n)| = \sqrt{(2400n)^2 + (240n)^2} = n\sqrt{(2400^2 + 240^2}$

(f) (4 points) Your boss wants to speed up the algorithm. She suggests at each iteration of gradient descent to choose a random i between 1 and n, set $f_i(m, b) = ((mu_i + b) - v_i)^2$, and then update based on the gradient of $f_i(m, b)$ rather than the gradient of $f(m, b)$. Will this work? Justify your answer in one sentence. *Solution: Yes this will work. It is the stochastic gradient descent algorithm.*

5. **UNIQUE-SET-COVER**

Consider the following problem called UNIQUE-SET-COVER. The input is an n-item set S, and a collection of m subsets $S_1, S_2, \ldots S_m \subseteq S$, where each item of S is in k of the subsets. Our goal is to find a collection of subsets that maximizes the number of items that are *uniquely covered*, where an item is uniquely covered if it is in exactly one subset in the collection.

(a) (6 points) Fix a real number p between 0 and 1 and consider the following algorithm: **For each index i, select set S_i independently with probability p.**
What is the expected number of items that are uniquely covered by this algorithm? Express your answer as a function of p and k. *Solution: Each item is exactly covered with probability $k(1-p)^{k-1}p$. By linearity, the expectation is thus $nk(1-p)^{k-1}p$.*

(b) (6 points) What value of p maximizes this expectation? *Solution: Taking the derivative of the above and setting to 0 we get $-(k-1)(1-p)^{k-2}p + (1-p)^{k-1} = 0$. Solving for p, we get $p = 1/k$.*

(c) (8 points) Now describe a polynomial-time randomized algorithm for UNIQUE-SET-COVER. Prove that its expected approximation ratio is $O(1)$. Hint: Use Bernoulli's inequality which says that for all $t \geq -1$ and $n \geq 1$, $(1+t)^n \geq 1+nt$. *Solution: Setting* $p = 1/k$ *in the algorithm given, we get that the probability that an item is uniquely covered is* $(1-1/k)^{k-1}$. *Using Bernoulli's, we have that* $(1-1/k)^{k-1} \geq 1-(k-1)/k$, *which is at least* $1/2$ *for all* k*!*