University of New Mexico
Department of Computer Science

Final Examination

CS 561, Data Structures and Algorithms
Fall, 2021

Name:
Email:

- This exam is open book, notes and Internet. However, you are not allowed to discuss problems with any person, or to post any problem to the internet. **Cite any sources you use.**
- I check sites like chegg and coursehero. If I find a question from this test there, I will work with them to track you down, and give you a failing grade in this class. Answers by "experts" on these sites are often riddled with errors, which aids tracking.
- PLEASE: Start each main problem (i.e. Problems 1-5) at the top of a new page!
- Give yourself extra time to properly upload your solutions. Late exams may be penalized.
- *Show your work!* You will not get full credit if we cannot figure out how you arrived at your answer.

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* Θ notation, unless it specifies otherwise. For recurrences, assume as usual, that $f(n)$ is $\Theta(1)$ for constant values of n.

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

   (b) Expected number of nodes in a skip list storing n items?

   (c) Solution to the recurrence $T(n) = 2T(n/2) + n^2$?

   (d) Solution to the recurrence: $f(n) = 2f(n-1) - f(n-2) + 1$. Answer in big-O notation here.

   (e) Time to determine if a graph with n nodes and m edges has a 5-clique?

(f) A stack has standard push and pop operations, and also a popLessThan(x) operation which repeatedly pops the top item off the stack until either the stack is empty or the top item on the stack has value at least x. Over n calls to push, pop and popLessThan, what is the worst case cost of any single call?

(g) In the problem above, what is the worst case *amortized* cost of any operation?

(h) Worst-case runtime of Kruskal's algorithm when using a union-find data structure where Make-Set, Find-Set and Union all have amortized cost $O(\log^2 n)$? Assume the graph has n nodes and m edges.

(i) You have a graph G with m edges and n nodes. You color each node uniformly at random with some color between 1 and n. What is the expected number of edges that have both endpoints colored the same?

(j) What is the best expected time to solve the activity selection problem if the start and finish times of all activities are selected independently and uniformly at random from time 0 to some time T?

2. **Recursion Cat** You are given a tree with all nodes colored either red or black. Call a path *valid* if at any step of the path, the number of red nodes visited so far is greater than or equal to the number of black nodes visited so far. A cat starts at the root node of the tree and wants to find a valid path to some leaf node.

   For each node v, let $f(v)$ be $-\infty$ if there is no valid path to v. Otherwise, let $f(v)$ be the number of red nodes visited minus the number of black nodes for the path ending at v.

   (a) (8 points) Give a recurrence relation for f. Hint: you may find it useful to let $p(v)$ be the parent of v, for every node v that is not the root.

   (b) (2 points) Briefly describe a dynamic program that uses the recurrence above to return a valid path from root to some leaf, if such a path exists.

(HARD/BONUS) Now recursion cat wants to find valid paths on any graph. Define a *red cycle* to be a cycle that has more red than black nodes in it. Assume you are given a graph, G, with no red cycles. For any pair of nodes, you want to determine if there is a valid path from u to v. Taking inspiration from Floyd-Warshall, you first assign labels 1 to n to all n nodes in the graph. Then you consider paths from nodes u to v that visit intermediate nodes with label at most i. For a given path, let the *black excess* of that path be the maximum over all steps of the path of the number of black nodes minus the number of red nodes at any step. For example, a path of the form R, B, R, B, B, B, R, R has black excess of 2.

Define $f(u, v, i, b) = -\infty$ if there is no path from u to v using intermediate nodes of label at most i, with black excess at most b. Otherwise, define $f(u, v, i, b)$ to be the maximum, over all paths from u to v, with black excess at most b that visit intermediate nodes with label at most i, of the number of red nodes minus the number of black nodes in that path. For example, if the only path from u to v has form R, B, R, B, B, B, R, R, then $f(u, v, n, 2) = 0$.

(c) (8 points) Write a recurrence relation for $f(u, v, i, b)$. It may help to assume that $-\infty + x = -\infty$ for any value x. Hint: Let the base case(s) be $f(u, v, 0, b)$ for any values of u, v and any b, $0 \leq b \leq n$. It may help to define for a node v, $color(v)$ to be 1 if the node is red, and -1 if the node is black.

(d) (2 points) Briefly describe a dynamic program that uses the recurrence above to determine if a valid path exists from u to v for every u and v. What is the runtime as a function of n, the number of nodes, and m the number of edges?

3. **MIN-INSIDE-EDGES** In the MIN-INSIDE-EDGES problem, you are given a graph $G = (V, E)$, and a number $x \leq |V|$, and you must choose a subset $V' \subseteq V$ of size x. Call an edge in E *inside* if both endpoints of the edge are nodes in V'. Your goal is to output the minimum number of inside edges for any set V' of size x.

   (a) (6 points) Show that MIN-INSIDE-EDGES is NP-Hard by a reduction from one of the following: 3-SAT, VERTEX-COVER, CLIQUE, SUBGRAPH-ISOMORPHISM, INDEPENDENT-SET, 3-COLORABLE, HAMILTONIAN-CYCLE, or TSP.

   (b) (8 points) Consider the randomized algorithm that picks a subset V' of size x, uniformly at random from all subsets of V of size x, when given graph $G = (V, E)$ and number x. Compute the expected number of inside edges for this algorithm using indicator random variables and linearity of expectation. Let $n = |V|$ and $m = |E|$.

(c) (4 points) Let μ be the expected number of inside edges for the randomized algorithm (i.e. your answer from part (b)). Now use Markov's inequality to bound the probability that there are greater than or equal to $(11/10)\mu$ inside edges after running the algorithm in part (b).

(d) (4 points) Using your result from part (c), bound the expected number of times you would need to run the randomized algorithm before you get a solution that has less than $(11/10)\mu$ inside edges. *Hint: Recall that for a random variable Y taking on positive integer values, $E(Y) = \sum_{i=1}^{\infty} Pr(Y \geq i)$ (see slides 32-33 from our "Randomized Data Structures" lecture).*

4. **Cake Cutting**

After the holidays, you buy a large cake wholesale, and you want to cut it into smaller cakes to resell for as much profit as possible. The original cake is a large rectangle that has width m inches and height n inches. You want to cut this cake into smaller rectangles of various integer dimensions, so as to maximize the total resale price of all rectangles. You have a lookup array that tells you the resale price, $P[x, y]$, of any x-inch by y-inch rectangle. The resale prices are unusual, depending on aesthetics, customer demand, etc., so don't make any assumptions about them.

You can make horizontal or vertical cuts across any rectangle with your knife, *but you must cut all the way through the rectangle.*

(20 points) Given integers m and n, and an array P, describe a dynamic program to compute how to subdivide the original cake to maximize your profit. Be sure to include the following:

(a) An English description of the subproblems (e.g. "minimimum edit distance of first i characters of string A and first j characters of string B").

(b) A mathematical description of the recurrence (e.g. "Base case(s): e(0,i) = i, e(j,0) = j; Recurrence e(i,j) = min(e(i,j-1), e(i-1,j), e(i-1,j-1) + 1 - I(A[i]=B[j])").

(c) How to compute the final answer using the recurrence (e.g. "Return e(m,n)").

(d) How to solve the subproblems and analysis of your runtime (e.g. "Fill in a table left to right, top down. Runtime is $O(nm)$")

4. **Cake Cutting, continued.**

5. **Gradient Descent**

In this problem, you are trying to minimize your average commute time from a source node s to a sink node t, over multiple days, where weights can vary from day to day due to traffic. To start, assume there are exactly 4 disjoint paths from s to t. You must choose one path each day, and after you have made your choice, the weight for each path for that day are revealed. You decide to use online gradient descent as follows. For each day i and path j, you let $\vec{x}_i$ be the probability distribution over the path for day i, i.e. $\vec{x}_i[j]$ is the probability that you take path j on day i. Then let $\vec{w}_i$ be the weight vector revealed on day i, i.e. $\vec{w}_i[j]$ is the weight of path j on day i. Your goal is to minimize your average expected weights of the path you take over all days.

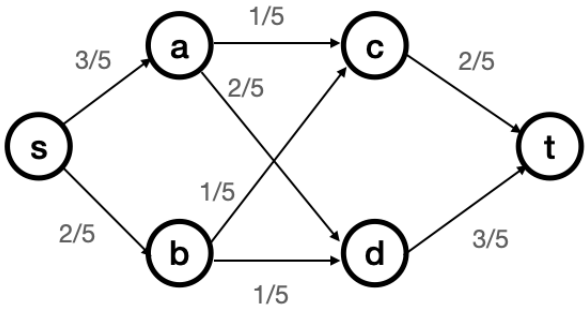
(a) (3 points) Let $f_i(\vec{x}_i)$ be the cost function for day i. Write this out as a function of $\vec{w}$ and $\vec{x}$.

(b) (3 points) Define $\nabla f_i(\vec{x}_i)$, the gradient of f_i.

(c) (3 points) Describe, algebraically, the convex search space κ. What is the diameter D of κ?

(Hard) Now you want to generalize the previous problem to the case of choosing a route from s to t in an directed acyclic graph (DAG), $G = (V, E)$. On day i, you must choose a path from s to t, and then you are given the weight values for all edges, i.e. you learn for all $(u \to v) \in E$, the value $w_i[(u \to v)]$.

The number of paths from s to t can now be exponential in V, so you need a new approach. You decide to let the x values specify probabilities on traversing each edge. In particular, for each day i and edge $(u \to v)$, you set $x_i[(u \to v)]$ to be the probability that you will traverse edge $(u \to v)$ on day i. The figure gives example x values.



(d) (4 points) Describe the set of linear inequalities that define the convex search space κ for this new problem.

(e) (4 points) Given the x_i values, describe how you can use them to choose a path on day i that has expected cost that equals $\sum_{(u \to v) \in E} x_i[(u \to v)] w_i[(u \to v)]$.

(f) (3 points) In class, we proved that the cost of online gradient descent tracks the cost of the best offline solution, x^*. In particular, if OPT is the cost of the best offline solution, then Zinkevich's theorem tells us that the cost of our algorithm is at most $OPT + \sqrt{T}DG$. But what is OPT here? For this problem, give a precise definition of OPT in terms of the w_i values, T, and the graph.