University of New Mexico
Department of Computer Science

Final Examination

CS 561 Data Structures and Algorithms
Fall, 2023

Name:
Email:

Directions:

- This exam lasts 2 hours. It is closed book and notes, and no electronic devices are permitted. However, you are allowed to use 2 pages of hand-written "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.

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

1. **Short Answer (4 points each)**

   Answer the following using *simplest possible* Θ *notation.*

   (a) Time to determine if a graph with n nodes and m edges has a 4-clique?

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

   (c) 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 $O(\log n)$? Assume the graph has n nodes and m edges.

(d) A stack has two operations: Push, and a PopGreaterThan(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 less than x. Over a total of n operations on an initially empty stack, what is the amortized cost per operation?

(e) What is the best expected time to solve the activity selection problem on n jobs if the finish times of all activities are selected independently and uniformly at random in the range 0 to M?

2. **Medium Answer**

(a) (10 points) Let $n > 2$. In an n by n grid, a node is an *interior* node if it has 4 neighbors. So, the grid has $(n-2)^2$ interior nodes.

Each node is colored independently red or green, each with probability 1/2. What is the expected number of interior nodes where the node and all 4 of its neighbors have the same color?

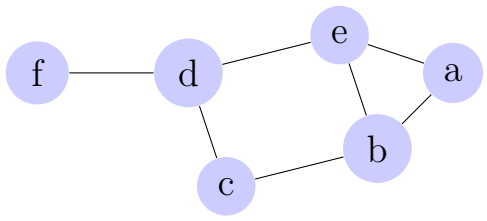
(b) (10 points) Let T be a minimum spanning tree in some graph $G = (V, E)$. Prove that any edge (u, v) in T is a light edge crossing some cut.

(Remember that a cut is a partition of the nodes in V into two disjoint sets. You will get no points on this problem if you don't specify these two sets for which (u, v) is a light crossing edge. Hint: Use both T and (u, v) in defining your cut.)

3. **NP-Completeness**

LONG-PATH takes as input a graph $G = (V, E)$, a start vertex $s \in V$, an end vertex $t \in V$, and an integer k. It returns YES iff there is a path starting at s and ending at t, that traverses k edges and visits at least k different vertices. Note that s and t do not need to be different.

For example, for the following graph, G, LONG-PATH(G,a,a,5) returns YES because of the path a, e, d, c, b, a, while LONG-PATH(G,f,d,2) returns NO, since there is no path of length 2 from f to d.



(a) (16 points) Prove that LONG-PATH is NP-Hard by reduction from one of the following: 3-SAT, CLIQUE, INDEPENDENT SET, VERTEX COVER, 3-COLORABLE, or HAMILTONIAN-CYCLE.

(b) (4 points) Prove that LONG-PATH is NP-COMPLETE by showing that it is also in NP.

4. **Dynamic Programming**

Dance, Dance Revolution (DDR) is played on a platform with 4 squares. In round $i \geq 1$, one of your feet must be on the square $\sigma[i]$, where σ is an input sequence composed of the symbols A, B, C or D, representing the four squares. Your feet must always be in different squares, and you can move at most one foot at the start of each round to any new square. Your left foot starts in square A and right foot in square B.

Your goal is to maximize your *score*: the number of rounds in which neither foot moves. Below is an example game play.

σ	A	C	A	D	C	D	B
Feet position	(A,B)	(A,C)	(A,C)	(D,C)	(D,C)	(D,C)	(B,C)
Point?	0	0	1	0	1	1	0

You scored 3 points since there are 3 rounds where neither foot moved.

(a) (15 points) Write a recurrence relation for the value $m(i, \ell, r)$ which gives the maximum score possible on the first i symbols of σ if your left foot ends in square ℓ and your right foot ends in square r.

(b) (5 points) Describe a dynamic program to return the max score for any input σ of length n based on your recurrence. What are the dimensions of your table? How do you fill it in? What is the final value returned? What is the runtime of your algorithm?

5. **Gradient Descent**

X-STREAM Dance Dance Revolution (XDDR) is played on a pogo-stick while blind-folded. At the beginning of each round, you can hop from your current square to any square. However, the target sequence, σ is not known in advance, and the value $\sigma[i]$ is announced only at the end of round i, for each $i \in [1, n]$ where n is the length of σ.

Your goal is to minimize *cost*: the number of rounds $i \in [1, n]$ in which you hop in a square different than $\sigma[i]$.

Below is an example game; your cost is 5 because there are 5 rounds where the square you hop in does not match the target square in σ.

σ	A	C	A	D	C	D	D
Pogo position	A	B	B	D	A	C	A
Cost?	0	1	1	0	1	1	1

In round i, you let $\vec{x_i}$ be a length 4 vector giving a probability distribution over the 4 possible squares on which you will hop, i.e. $\vec{x_i}[1]$, $\vec{x_i}[2]$, $\vec{x_i}[3]$, $\vec{x_i}[4]$ are the probabilities of hopping into squares A, B, C, D respectively.

Then, for round i, you will define $f_i(x_i)$ as your expected cost in round i. In round i, let c_i be a length 4 vector giving the cost outcome: $c_i[j] = 0$ when j matches the square given by $\sigma[i]$ and $c_i[j] = 1$ otherwise. For example, in round i, if $x_i = [1/8, 1/2, 1/8, 1/4]$ and $c_i = [1, 1, 1, 0]$, then your expected cost for this round is 3/4

(a) (5 points) Give a 1 line definition of f_i as a function of $\vec{x_i}$ and $\vec{c_i}$.

(b) (5 points) Describe, algebraically, the convex search space κ. What is the diameter, D of κ?

(c) (5 points) Zinkevich's theorem says 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 the cost of our algorithm is at most $OPT + \sqrt{n}DG$. Give a precise 1 line definition of OPT for this problem using the c_i values.

(d) (5 points) Now, you want to use some history. In particular, you notice that the current square in σ often depends on the last square. So you want to use the outcome of the last round to help set your probability distribution for the current round. To do this, how would you change the convex search space κ? How many dimensions does it now have? Will OPT, G and D likely increase or decrease? Will your algorithm's expected cost increase or decrease?

5. **Gradient Descent, continued.**