

# CS 374 Midterm 2

Aldo Sanjoto

TOTAL POINTS

**40 / 125**

## QUESTION 1

20 pts

### 1.1 Recursion 10 / 10

✓ - 0 pts Correct

- 10 pts Completely Incorrect/IDK

- 7 pts Considered new recurrence relation which would have higher complexity and computed that quantity.  $V-1$   $V-2$

- 7 pts  $v1/v2$ : Missed  $O(n)$  in recurrence and got sth in  $o(n)$

- 7 pts  $v3/v4$ : correct height of recursion tree but wrong running time/correct work per level, wrong height of recursion tree

- 7 pts Computed loose upper and lower bounds.

### 1.2 Graph 10 / 10

✓ - 0 pts Correct

- 10 pts Completely Incorrect/IDK/Assuming  $u, v$  are given for finding a pair

- 10 pts Non-Linear time solution

- 5 pts Partial Solution

- 2 pts Minor mistake

- 2 pts Cycle doesn't necessarily contain  $v$

## QUESTION 2

### 2 Seen in lab 10 / 20

- 0 pts Correct

✓ - 2 pts Did not remove the negative edge before running Dijkstra's

- 2 pts Missing initial vertex for calls to Dijkstra

- 12 pts Incorrect algorithm

- 4 pts Missing/Incorrect running time

- 4 pts Missing the case where the shortest path both exists and goes through the negative edge

✓ - 4 pts Missing/Wrong argument for correctness

✓ - 4 pts Did not correctly find the negative cycle if it

exists

- 20 pts Wrong/IDK

- 10 pts Using Bellman-Ford instead of Dijkstra's

## QUESTION 3

### 3 Array corruption/binary search 0 / 20

- 0 pts Correct

- 4 pts Minor mistake in algorithm

- 8 pts Major mistake in algorithm

- 5 pts Incorrect proof of correctness

- 2.5 pts Incorrect running time

- 20 pts Incorrect algorithm

- 2 pts Inefficient solution

- 16 pts Trivial solution

- 16 pts Worse than or as bad as  $O(n)$

✓ - 20 pts The answer is too long / unreadable

- 20 pts IDK

## QUESTION 4

### 4 Dynamic programming 0 / 20

+ 20 pts Correct

- 8 pts Greedy solution - Might get "stuck" and incorrectly report FALSE if it makes a bad decision early on.

+ 10 pts Correct recursion

+ 2 pts definition of recursive variable

+ 2 pts correct base case for recurrence

+ 6 pts correct recursive case

- 2 pts Checking all  $j$  instead of  $\Delta$  many

+ 5 pts filling table in right order

+ 5 pts correct running time analysis

- 2 pts running time analysis  $O(n^2)$  not tight/algorithm not fast enough

+ 0 pts incorrect; using implicit memoization / recursive function; illegible or answer too long/too messy to follow; solving a different problem due to

misunderstanding of problem

✓ + 0 pts IDK

#### QUESTION 5

Safe vertices - SCC + heaviest path 20 pts

5.1 DAG 0 / 10

- 0 pts Correct

✓ - 10 pts Completely Incorrect

- 5 pts Over counting

- 2 pts Over counting (two paths to same node problem)

- 5 pts under counting

- 10 pts Recurrence Relation for update not specified. Otherwise all correct.

- 3 pts  $O(N^2)$  Runtime

- 3 pts (Vulnerability) DFS based solution, runs into time complexity issues if nodes have more than 1 parent

- 5 pts Sub-optimal runtime

- 5 pts Partial Solution

- 10 pts IDK

- incorrect modification on the graph.  
only considering neighboring nodes instead of reachable nodes.

5.2 Directed graph 0 / 10

- 0 pts Correct

✓ - 10 pts Completely Incorrect

- 3 pts Suboptimal Runtime

- 7 pts Don't consider paths within SCC (You have not specified what the  $r(v)$  values for the metanodes are)

- 10 pts IDK

- 10 pts Same solution (almost) as in part A

- 5 pts Insufficient explanation

- what if there aren't  $k$  infected nodes in a single scc?

#### QUESTION 6

IDK 25 pts

6.1 Q1 (A) 0 / 2.5

✓ - 2.5 pts Not used

- 0 pts Used IDK

6.2 Q1 (B) 0 / 2.5

- 0 pts Correct

✓ - 2.5 pts Not Used

6.3 Q2 0 / 5

- 0 pts IDK

✓ - 5 pts Not IDK

6.4 Q3 5 / 5

✓ - 0 pts Used IDK

- 5 pts Did not use IDK

6.5 Q4 5 / 5

✓ + 5 pts Used idk

+ 0 pts Did not use IDK

6.6 Q5 (A) 0 / 2.5

✓ - 2.5 pts Not Used

- 0 pts Correct

6.7 Q5 (B) 0 / 2.5

✓ - 2.5 pts Not Used

- 0 pts Correct

## Midterm 2: November 13, 7-9pm, 2017

A	B	C	D	E	F	G	H	J	K
9am	10am	11am	noon	1pm	1pm	2pm	2pm	3pm	3pm
Rucha	Rucha	Srihita	Shant	Abhishek	Xilin	Shalan	Phillip	Vishal	Phillip
SC 1404	SC 1404	SC 1404	DCL	DCL	DCL	ECE	ECE	ECE	ECE
			1320	1320	1320	1002	1002	1002	1002

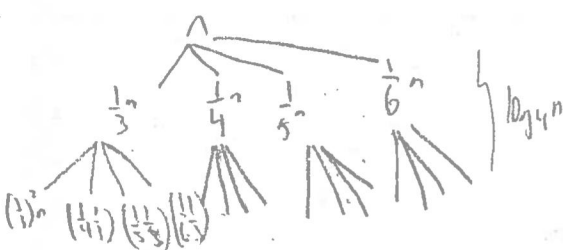
Name:	ALDO SANTO
NetID:	SANTO2
Name on Gradescope:	ALDO SANTO

- **Don't panic!**
- Please print your name and NetID in each page in the appropriate fields, and circle your discussion section in the boxes above. We will return your exam at the indicated section.
- If you brought anything except your writing implements, your double-sided **handwritten** (in the original)  $8\frac{1}{2} \times 11$ " cheat sheet, and your university ID, please put it away for the duration of the exam. In particular, please turn off and put away *all* medically unnecessary electronic devices.
  - Submit your cheat sheet together with your exam. An exam without your cheat sheet attached to it will not be graded.
  - If you are NOT using a cheat sheet, please indicate so in large friendly letters on this page.
- Please ask for clarification if any question is unclear.
- **This exam lasts 120 minutes.**
- If you run out of space for an answer, feel free to use the blank pages at the back of this booklet, but please tell us where to look.
- As usual, answering any (sub)problem with "I don't know" (and nothing else) is worth 25% partial credit. Correct, complete, but sub-optimal solutions are *always* worth more than 25%. A blank answer is not the same as "I don't know".
- Total IDK points for the whole exam would not exceed 10.
- **Beware of the Four Deadly Sins.** Give complete solutions, not examples. Declare all your variables. If you don't know the answer admit it and use IDK. Do not write unnecessary text.
- **Style counts.** Please use the backs of the pages or the blank pages at the end for scratch work, so that your actual answers are clear.
- Try to avoid writing in the margins of the pages, as it might not be scanned in.
- Please return *all* paper with your answer booklet: your cheat sheet, and all scratch paper. We will **not** return the cheat sheet.
- **Good luck!**

## 1 (20 PTS.) Short questions.

1.A. (10 PTS.) Give an asymptotically tight solution to the following recurrence, where  $T(n) = O(1)$  for  $n < 10$ , and otherwise:

$$T(n) = T(n/3) + T(n/4) + T(n/5) + T(n/6) + O(n).$$



For every level we have  $4^l$  nodes at  $l$ -th level. Therefore, for each  $n$  we have  $\sum (\log_4 n)$ . Total running time will be  $O(n \log_4 n)$ .

1.B. (10 PTS.) Given a directed graph  $G$ , describe a linear time algorithm that computes a pair of vertices  $u, v$  such that there is no path from  $u$  to  $v$ . If no such pair exists, then the algorithm should output "no such pair".

Use Kosaraju's algorithm to compute the metagraph  $G^{SCC}$  of  $G$  in  $O(n+nm)$  time.

Then we do topological sort which also takes  $O(n+nm)$  time.

If in the SCC component  $(S_1, S_2, \dots, S_n)$  there's no edge  $S_i \rightarrow S_j$  then we know  $u \in S_i$  has no path to  $v \in S_j$ . This takes  $O(m)$ .

Overall, running time is  $O(n+nm)$ .

- 2 (20 PTS.) (Seen in lab) You are given a directed graph  $G$  with  $n$  vertices and  $m$  edges. Every edge  $x \rightarrow y \in E(G)$  has a weight  $w(x \rightarrow y)$  associated with it. Here, exactly one edge  $u \rightarrow v$  has negative weight (all other weights are positive numbers). Describe an algorithm, as fast as possible, that decides if there is a negative cycle in  $G$ , and if not, it computes the shortest path between the two given vertices  $s$  and  $t$  in  $G$ . What is the running time of your algorithm? Argue that your algorithm is correct.

We can use Dijkstra's algorithm to compute the shortest path from  $s$  to  $t$  in  $G$  given that exactly one edge  $u \rightarrow v$ . However, we need to handle the case to avoid a path that consist of a negative cycle. To do this, we run Dijkstra's algorithm twice.

First, consider the case that Dijkstra's do not visit  $u$  and  $v$ . Then our shortest path will be  $\text{dist}(s, t)$ .

Second, to avoid negative cycle, we need to compute the shortest path from  $s$  to  $u$  and  $v$  to  $t$ .

The result will be  $d(s, u) + w(u \rightarrow v) + d(v, t)$ .

To find the shortest path, we need to take the minimum between these two as the following:

$$\text{dist}(s, t) = \min(d(s, t), d(s, u) + w(u \rightarrow v) + d(v, t))$$

The running time of the algorithm could be  $O(n \log n + m)$

NETID:

NAME:

---

NETID: SANJITO2

NAME: ALDO SANJITO

- 3 (20 PTS.) Let  $A[1 \dots n]$  be an array of  $n$  distinct numbers that is not sorted, but it is  $k$ -scrambled. Here, being  $k$ -scrambled means that for any  $i$ , we have that  $i - k \leq \text{rank}(A[i]) \leq i + k$ , where  $\text{rank}(A[i])$  is the location of  $A[i]$  in the sorted version of  $A$ .

You are given the value of  $k$  as part of the input, and a number  $x$ . Describe an algorithm, as fast as possible, that reports whether or not  $x$  is stored in  $A$  somewhere.

What is the running time of your algorithm? Explain (shortly) why your algorithm is correct.

(Hint: What can you say about the relation between  $\text{rank}(A[i])$  and  $\text{rank}(A[i + 3k])$ ?)

IDK

kz

AC · 3

NETID:

NAME:

---




NETID: sanjib2NAME: Aldo Snyda

- 4 (20 PTS.) There are  $n$  people living along Purple street in Shampoo-Banananana. The  $n$  people live in locations specified by a given array  $t[1 \dots n]$ , where  $0 < t[1] < \dots < t[n]$ . Here  $t[i]$  is the location of the  $i$ th person, which is the distance in meters from the start of Purple street,

The district needs to break the street into blocks. A block can have between  $\Delta$  and  $2\Delta$  people living in it, for some prespecified parameter  $\Delta$  (where  $n/3 > \Delta > 1$ ). A block is a consecutive interval along Purple street. Every person is assigned to a block containing it. A portion of Purple street that has no people living on it, does not necessarily has to be in a block.

The price of a block, covering the  $i$ th to  $j$ th person, is the total length of the block (in meters), which is  $t[j] - t[i]$  (the next block would start at  $t[j + 1]$ ). The *total price* of a solution is the total length of the blocks in the solution.

Here is an instance and a suggested solution (of total price 10), with  $\Delta = 2$ :

$t = [1, 3, 4, 8, 9, 10, 11, 13, 16, 17]$  

Describe an algorithm, as fast as possible, that computes the minimum price way of breaking the street into blocks. What is the running time of your algorithm as a function of  $n$  and  $\Delta$ ? The algorithm you provide should be iterative (i.e., please do not use dictionary/hashig or recursion).

10k

NETID:

NAME:

---

- 5 (20 PTS.) You are given a graph  $G$  with  $n$  vertices and  $m$  edges. Think about this graph as a network of windows machines. For every vertex  $v \in V(G)$ , you also given a value  $b(v)$  which is one if it is *infected* (with a virus), and 0 otherwise. A vertex is *vulnerable*, if it is already infected or there are at least  $k$  distinct infected nodes that can reach it (here, think about  $k$  as being a small number compared to  $n$ ).

- 5.A. (10 PTS.) For the case that  $G$  is a DAG, describe an algorithm, as fast as possible, that computes all the vulnerable vertices of  $G$ . What is the running time of your algorithm?

Argue (shortly) that your algorithm is correct.

- Remove all edges s.t.  $u \rightarrow v$  and  $u.b, v.b = 0$ .
  - Use this new graph  $G'$ , and do topological sort  $O(nm)$ .
  - Get  $G'$  REV.
  - We traverse from the first vertices to last vertices which is a sink and source in the original graph.
  - If  $A[n].b = 1$ , we increment our count value.
  - Then if  $A[n].b = 0$ , we check if the number of outgoing edge  $= k$  then increment count.
  - Count will be the number of vulnerable vertices.
  - $O(nm^2)$
- Add an edge  $u \rightarrow w$  for every consecutive edge that connects  $u \rightarrow v \rightarrow w$ .

- 5.B. (10 PTS.) Describe an algorithm, as fast as possible, for the case that  $G$  is a general directed graph. What is the running time of your algorithm?

- Create a metagraph using Kosaraju's algorithm,  $G^{SCC}$ .
- for every vertex in  $G^{SCC}$ ,  $\{S_1, \dots, S_n\}$ . We check if the number of the vulnerable vertices in each SCC is  $\geq k$ , then the total vulnerable vertices in that  $S_i$  is the total number of vertex  $\in S_i$ .  $O(n)$
- If the  $S_i$  has only one vertex, we can just check if that vertex value 0 or 1. If it's 1 then increment total number of vertex.
- We go through each vertex and total running time will be  $O(nm)$ .

NETID:

NAME:

---