CS 161 (Stanford, Winter 2022) Homework 5

Style guide and expectations: Please see the "Homework" part of the "Resources" section on the webpage for guidance on what we are looking for in homework solutions. We will grade according to these standards. You should cite all sources you used outside of the course material.
Note: this homework covers lectures 8-11 and includes additional 0-credit problems for practice. Due to the length, we recommend you work on the problems for credit first.
What we expect: Make sure to look at the "**We are expecting**" blocks below each problem to see what we will be grading for in each problem!

Exercises. The following questions are exercises. We suggest you do these on your own. As with any homework question, though, you may ask the course staff for help.

1 Exercise: Hash Functions

With 466 students currently enrolled in CS161, We would like to keep track of student records. The key for each person will be their 8-digit Stanford student ID.

1.1 (2 pt.)

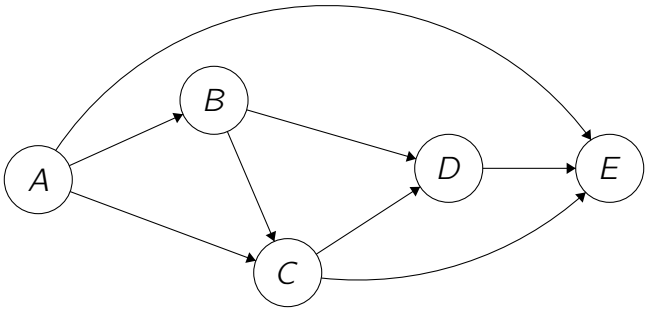
Consider using a hash table of size 100 and the hash function $h(N) =$ sum of each digits of N. For example, $h(01234567) = 0+1+2+3+4+5+6+7 = 28$. Is this a good hash function to use? List two reasons supporting your decision.
[We are expecting: The answer to whether this is a good hash function or not, and two reasons explaining why.**]**

1.2 (2 pt.)

Consider a hash table of 99 buckets. For $m \in \{1, \ldots, 1000\}$, let $h_m(x) =$ last two digits of mx. For example, $h_4(01234567) =$ last two digits of $4938268 = 68$. Define $\mathcal{H} = \{h_i : i \in \{1, \ldots, 1000\}\}$. Is $\mathcal{H}$ a good family of hash functions? Justify your decision.
[We are expecting: The answer to whether $\mathcal{H}$ is a good family of hash functions or not, and a formal justification.**]**

2 Exercise: BFS and DFS Basics

Consider the following directed acyclic graph (DAG):



In class, we saw how to use DFS to find a topological ordering of the the vertices; in the graph above, the unique topological ordering is A, B, C, D, E. We saw an example where we happened to start DFS from the first vertex in the topological order. In this exercise we'll see what happens when we start at a different vertex. Recall that when you run DFS, if it reached a node with no children (i.e. can't go any further), then it will resume the search at an unvisited vertex.

2.1 (3 pt.)

Run DFS starting at vertex C, breaking any ties by alphabetical order.[1]

(a) What do you get when you order the vertices by **ascending** start time?

(b) What do you get when you order the vertices by **descending** finish time?

2.2 (0 pt.)

Run DFS starting at vertex C, breaking any ties by **reverse** alphabetical order.[2]

(a) What do you get when you order the vertices by **ascending** start time?

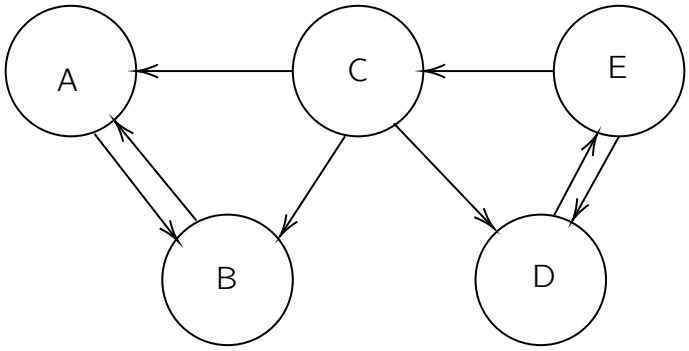
(b) What do you get when you order the vertices by **descending** finish time?

[We are expecting: For all four questions, an ordering of vertices. No justification is required.**]**

[1] For example, if DFS has a choice between B or C, it will always choose B. This includes when DFS is starting a new tree in the DFS forest.

[2] For example, when DFS has a choice between B or C, it will always choose C. This includes when DFS is starting a new tree in the DFS forest.

3 Exercise: Kosaraju's algorithm



In this question, we ask you to run a different version of Kosarju's algorithm than the one we covered in class. This version is algorithm 1 from the lecture note (check out the note to see why it works!). Instead of running DFS on the original graph, we will run DFS on the **reversed** graph starting from C, and then start the second run of DFS on the **original** graph from the node with largest finish time.
When running DFS, we will **break any ties by alphabetical order**. Remember that if DFS has reached a node with no un-visited children (i.e. can't go any further), then it will resume the search at an unvisited vertex. We will decide which unexplored vertex to continue DFS by alphabetical order.

3.1 First DFS (4 pt.)

What are the start and finish times of each node after running the first DFS from node C on the **reversed** graph?

Node	Start Time	Finish Time
A		
B		
C	0	
D		
E		

[We are expecting: Fill in this table**]**

3.2 Second DFS (2 pt.)

Next, we will run DFS from the node with the largest finish time on the **original** graph. What do you get when you order the vertices by **ascending** start time for the second DFS run? **[We are expecting:** an ordering of vertices. No justification is required**]**

3.3 Connected components (2 pt.)

(a) What are the strongly connected component(s) in this graph?

(b) What are the weakly connected component(s) in this graph?

[We are expecting: For each question, a list of sets of nodes. No justification is required**]**

Problems. The following questions are problems. You may talk with your fellow CS 161-ers about the problems. However:

- Try the problems on your own *before* collaborating.
- Write up your answers yourself, in your own words. You should never share your typed-up solutions with your collaborators.
- If you collaborated, list the names of the students you collaborated with at the beginning of each problem.

---

4 Most Reliable Path

You are planning for an exciting trip from the nice little town s to the shining new city t this summer! Sadly there is no direct flight from s to t so you need to take multiple flights to get to t. You created a directed graph $G(V, E)$ where each node represents an airport and each edge represents a flight connection from one airport to another. Each flight connects two airports, but for each flight e there is some probability p_e that this flight can get cancelled due to COVID-19. Each p_e is some value between 0 and 1, where 0 means that this flight will always take place, and 1 means that this flight will always get cancelled. You have to book your flights and hotels in advance. You can assume that all p_e are independent of each other and the graph has no cycles.

4.1 Just to be safe? (6 pt.)

Create an algorithm to find the most reliable path between s and t. By most reliable path, we mean that the path P you find from s to t should have the highest probability of having all flights in P not being cancelled. Having plenty of time for the trip, You **DO NOT** care about how long or how much it takes you to travel, but you hope to reach the destination with **exactly the route you planned**. [Hint: log(AB) = log(A) + log(B)] **[We are expecting:** Pseudocode or a very clear English description of your algorithm, an informal justification that your algorithm is correct, and brief analysis of running time**]**

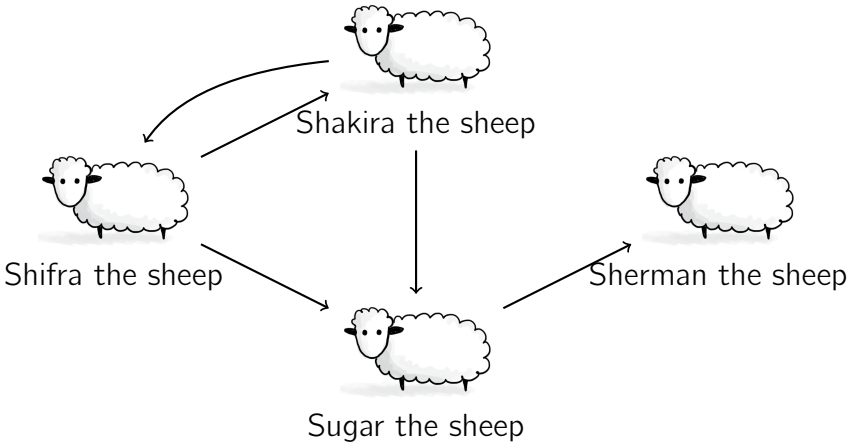
4.2 Ethics (5 pt.)

Imagine that, based on the excellent results of your flight planning algorithm, you have been hired by an online travel agency to give your expert advice on their route recommendation strategies. You discover that, by maximizing the reliability of suggested routes, the company would be overlooking other considerations such as carbon and time efficiency. Write a one-paragraph memo to the programming team explaining why these (or other) values should be taken into account. **[We are expecting:** four to six sentences explaining (1) what other measures are overlooked by an algorithm that maximizes reliability; and (2) the reasons why

the airline should care about these measures, including some potential consequences of the omission**]**

5 Wake up, Sheeple!

You arrive on an island with n sheep. The sheep have developed a pretty sophisticated society, and have a social media platform called Baaaahtter (it's like Twitter but for sheep[3]). Some sheep follow other sheep on this platform. Being sheep, they believe and repeat anything that they hear. That is, they will re-post anything that any sheep they are following said. We can represent this by a graph, where $(a) \to (b)$ means that (b) will re-post anything that (a) posted. For example, if the social dynamics on the island were:



then Sherman the Sheep follows Sugar the Sheep, and Sugar follows both Shakira and Shifra, and so on. This means that Sherman will re-post anything that Sugar posts, Sugar will re-post anything by Shifra and Shikira, and so on. (If there is a cycle then each sheep will only re-post a post once). For the parts below, let G denote this directed, unweighted graph on the n sheep. Let m denote the number of edges in G.

5.1 The influencer circle (6 pt.)

Call a sheep an **influencer** if anything that they post eventually gets re-posted by every other sheep on the island. In the example above, both Shifra and Shakira are influencers. Prove that, if there is at least one influencer, then all influencers are in the same strongly connected component of G, and every sheep in that component is an influencer. **[We are expecting:** A short but rigorous proof.**]**

5.2 Who is the influencer? (8 pt.)

Suppose that there is at least one influencer. Give an algorithm that runs in time $O(n + m)$ and finds an influencer. You may use any algorithm we have seen in class as a subroutine.

[3] Also my new start-up idea

[We are expecting: Pseudocode or a very clear English description of your algorithm, an informal justification that your algorithm is correct , an informal justification that the running time is $O(n+m)$.**]**

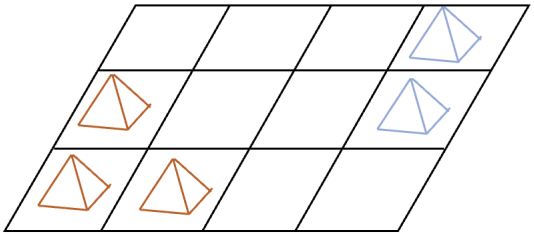
5.3 Is there an influencer? (0 pt.)

Suppose that you don't know whether or not there is an influencer. Give an algorithm that runs in time $O(n+m)$ and either returns an influencer or returns the text `"no influencer"`. You may use any algorithm we have seen from class as a subroutine, and you may also use your algorithm from previous part as a subroutine. **[We are expecting:** Pseudocode or a very clear English description of your algorithm, an informal justification that your algorithm is correct, an informal justification that the running time is $O(n+m)$**]**

6 Summer Camp (6 pt.)

Some Stanford raccoons decided to hold a summer camp event at the Portola Redwoods State Park. The camping ground is divided into an $m \times n$ grid, where each cell can accommodate one standard-sized tent. The Stanford raccoons put up their tents connected on the grid (considering four cardinal directions). The following day, they discover another connected collection of tents put up by raccoons from Berkeley. Two groups of raccoons are willing to make friends with each other and connect their living places into a single collection by putting up some spare tents. Please help design an $\mathcal{O}(mn)$ algorithm that calculates the minimum number of spared tents that the raccoons need to put up.

Example of camping ground layout If run your algorithm on this, the output should be 2 (We need to put up at least 2 tents to connect the two groups of tents).



Input: A nested array A of $m \times n$ where 1 represents a tent and 0 represents an empty

camping ground. [hint: You might want to locate all tents in one connected collection of tent first.] **[We are expecting:** Pseudocode of the algorithm and a clear English description, **the algorithm should run in** $\mathcal{O}(mn)$ **time]**