CS 161 (Stanford, Winter 2026) Homework 6

Style guide and expectations: We do NOT accept handwritten solutions. Please see the "Homework" part of the "Resources" section on the webpage for guidance on what we look for in homework solutions. We will grade according to these standards. You should cite all sources you used outside of the course material. Please do not distribute this material on any public forum.

Note about tagging your pages on gradescope: Please tag all of your pages to the correct question number on gradescope. We will apply a **5% deduction** to all untagged answers.

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!

Pair submissions: You can submit in pairs for this assignment. If you choose to do this, please submit **one** Gradescope assignment per pair and be sure to tag both partners on your submission. Note that we still encourage exercises to be done solo first.

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 Floyd-Warshall Trace (4 pt.)

Your task is to trace through the Floyd-Warshall algorithm for the following undirected graph. Fill in the tables below which keep track of the shortest distances as Floyd-Warshall does. Assume that the algorithm looks at nodes in alphabetical order (as suggested by the table).

Note: We only need to fill in the upper diagonal of our distances matrix because, for an undirected graph, the distances are symmetric.
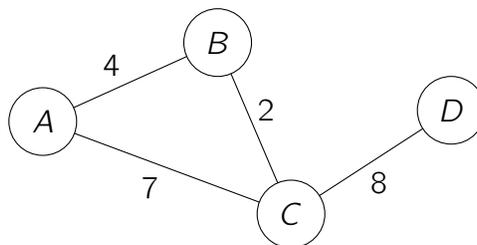
[We are expecting: Complete the tables below by replacing each "??" with the correct integer. No explanation is needed.**]**

Initial (Direct Edge Weights)				
-	a	b	c	d
a	0	4	7	∞
b	-	0	2	∞
c	-	-	0	8
d	-	-	-	0

vertices so far:{A}				
-	a	b	c	d
a	0	??	??	??
b	-	0	??	??
c	-	-	0	??
d	-	-	-	0

vertices so far:{A, B}				
-	a	b	c	d
a	0	??	??	??
b	-	0	??	??
c	-	-	0	??
d	-	-	-	0

vertices so far:{A, B, C}				
-	a	b	c	d
a	0	??	??	??
b	-	0	??	??
c	-	-	0	??
d	-	-	-	0

vertices so far:{A, B, C, D}				
-	a	b	c	d
a	0	4	6	14
b	-	0	2	10
c	-	-	0	8
d	-	-	-	0

Solution

2 Plucky's Commuting Plans

Plucky the Pedantic Penguin loves public transportation and commutes everywhere by train. He uses the new electric PenguinTrain, which offers two services: line A (local) and line B (express). Both lines go linearly through all the stations in sequential order ($0 \rightarrow 1 \rightarrow 2 \rightarrow$

$\cdots \to n$). Plucky never travels backward, but he can switch from line A to line B and visa versa at any given station. The issue is: every time Plucky moves from line A to line B, he must pay a fixed transfer fee, regardless of the station where the switch happens. Additionally, switching from line B back to line A is free.

Your task is to help Plucky find the minimum cost path going from station 0 to station n, considering that Plucky always starts at station 0 on line A. Note that it does not matter which line Plucky arrives on at Station n, since he can always transfer to line A for free.
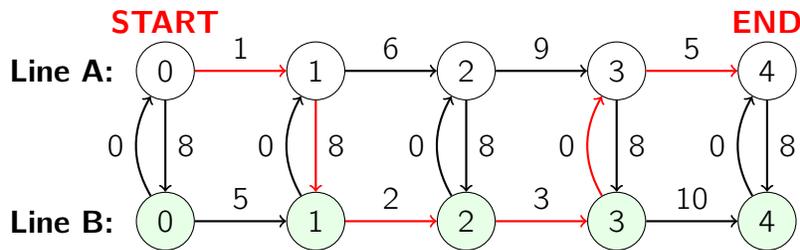

Figure 1: A sample problem with the two available routes to Plucky (line A at the top and line B at the bottom) and the minimum path solution. Note that Plucky can never go backward and that each transition in the graph has a specific cost associated with it. The cost to switch from line A to line B is always fixed, and there is no cost to move back from line B to line A. Keep in mind that Plucky always starts on station 0 on line A!

You are given two arrays to solve this problem: A_costs and B_costs, each containing the transition costs on their respective line in sequential order, such as A_costs = $\{1, 6, 9, 5\}$; and you are given an integer c for the cost to switch from line A to line B at any station, such as $c = 8$. We want to output an array of length n with the sequence of lowest possible costs to reach each of the n stations (considering that a station is reached regardless of the line we reach it through). For reference, the solution to the image above is $\{1, 7, 14, 19\}$

2.1 Bug Finding (5 pt.)

Below is an attempted Dynamic Programming solution in which the student keeps a 2D table, where the index $(0, i)$ represents the minimum cost to reach the state "station i on line A", and $(1, i)$ represents the same idea for line B. **The presented solution contains a bug**, meaning it will not yield the correct answer for all cases.

1. Explain what is incorrect about the pseudocode given and why it may lead to incorrect solutions

2. Provide a correction to the mistake

```
1 def plucky_commute_cost(A_costs, B_costs, c):
2     dp_table = generate (2xn) matrix of zeros
3     # there is no cost to reach station 0 on path A
4     dp_table[0][0] = 0
5     # cost to reach station 0 on path B
```

```
6        dp_table[1][0] = c
7        for i in [0, n-1]:
8            dp_table[0][i+1] =
9                  dp_table[0][i] + A_costs[i]
10           dp_table[1][i+1] =
11                 min{dp_table[1][i] + B_costs[i], dp_table[0][i+1]+c}
12       return first row of dp_table
```

[**We are expecting:** English description of the mistake and why/how it may lead to incorrect solutions, plus line(s) of code to be added or edited in the pseudocode to correct it**]**

> **Solution**
>
> **English description.**
>
> **Provided code to correct the mistake.**

2.2 Abstractions (2 pt.)

A city-wide version of Pluckys algorithm is implemented to help all residents find the minimum cost path going from station 0 to station n. As part of the implementation, the algorithm idealizes all passengers as having the same ability to transfer between the lines. Describe a group for whom transferring between lines poses additional barriers than just financial. Explain how the algorithm made an abstraction in this scenario. How might this lead to injustice for the group that you chose?

[**We are expecting:** 3-4 sentences identifying a group, an explanation of how the algorithm made an abstraction, and an explanation of how the abstraction might lead to injustice for the group chosen.**]**

> **Solution**

2.3 Discretion (2 pt.)

At each train station, there is a human station manager who oversees the activities of the station. Define one situation in which the station manager should exercise discretion to waive a transfer fee.

[**We are expecting:** 2-3 sentences defining a situation in which the station manager should exercise discretion to waive a transfer fee.**]**

> **Solution**

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.

3 Finding shortest paths for special graphs

3.1 Graphs with few negative edges (5 pt.)

Let $G = (V, E)$ be a directed weighted graph, and $s \in V$ a vertex. Suppose there are no negative-weight cycles, and only the edges going out of s may have negative weights.

As seen in class, Bellman-Ford can solve the shortest path problem on any graph (without negative cycles) including the graph above. However, since the given graph is special and only a specific set of edges may be negative, modify the graph suitably to use a faster algorithm from lecture to solve the shortest paths from s on this graph?

[We are expecting: Algorithm name, English description stating the steps to modify the graph and a concise but clear explanation for correctness.**]**

> Solution
>
> **Algorithm name.**
>
> **English description.**
>
> **Explanation for correctness.**

3.2 Graphs with small integer weights (5 pt.)

Let $G = (V, E)$ be a directed weighted graph, where all the edge weights are from $\{1, 2, 3\}$. Design an algorithm that modifies the graph suitably and solves the shortest path problem for the given graph asymptotically faster than the algorithm you used in the previous part. Mention the time complexity of both the algorithms.

[We are expecting: Short English description stating the steps to modify the graph, algorithm name (from class) to solve shortest path problem on the modified graph, time complexity of the proposed algorithm along with a short justification of the time complexity, and time complexity of the algorithm from previous part. (No pseudocode necessary)**]**

4 Duck Dance-Off

Two dancing duck troupes are having a dance-off. The rules are as follows. There is a dance floor, which is laid out as a row of n squares, where n is an even number. Each square has a score, which is positive and is given by an array D of length n. Each duck receives the score of the square it dances in, and the score for the whole team is the sum of the scores of all dancers in that team.

The two dancing duck troupes take turns adding dancers to the dance floor; but the rules are that a new dancer can only join next to the edge of the dance floor, or next to an existing dancer. The two troupes are colored green and white, and green goes first.

For example, the following would be a legal dance-off, with a dance-floor array $D = [5, 7, 3, 4, 4, 6]$.
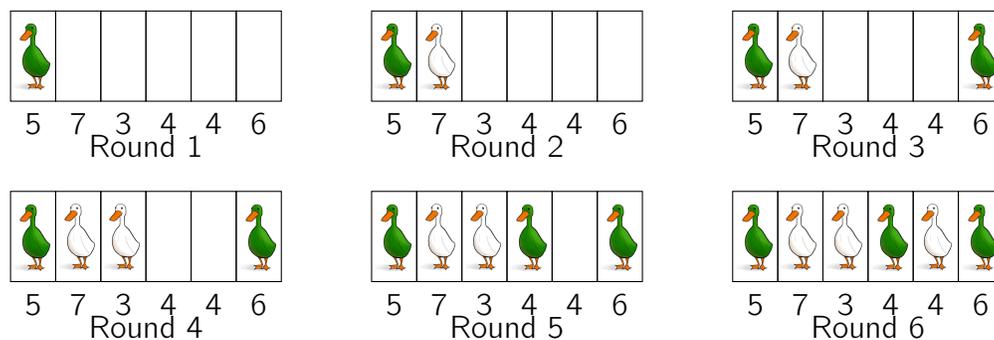
At the end of this dance-off, the green ducks have a score of $5 + 4 + 6 = 15$, while the white ducks have a score of $7 + 3 + 4 = 14$, so the green ducks win. Notice that in the above example, the ducks may not have been using the optimal strategy.

For the questions below, "green ducks" refers to the dance troupe that goes first in this dance-off.

In this problem, you will design an algorithm to compute the best score that the green ducks can obtain, assuming that the white ducks are playing optimally. Your algorithm should run in time $O(n^2)$.

4.1 Solution Outline (5 pt.)

What sub-problems will you use in your dynamic programming algorithm? What is the recursive relationship that is satisfied between the sub-problems?

Hint: Consider the sub-problems where we restrict the dance floor to an interval.

[We are expecting: A clear description of your sub-problems, a recursive relationship that they satisfy (including a base case), and an informal justification that the recursive relationship is correct.**]**

> Solution
>
> **Description.**
>
> **Recursive relationship and base case.**
>
> **Justification.**

4.2 Pseudocode (8 pt.)

Write pseudocode for your algorithm. Your algorithm should take as input the array D, and return a single number which is the best score the green team can achieve. Your algorithm does not need to output the optimal strategy. It should run in time $O(n^2)$.

[We are expecting: Pseudocode **AND** a clear English description. You do not need to justify that your algorithm is correct, but correctness should follow from your reasoning in part (a).**]**

> Solution
>
> **Pseudocode.**
>
> **English description.**

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 X but for sheep[1]). 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 $(a) \rightarrow (b)$ means that (b) will re-post anything that (a) posted. For example, if the social dynamics on the island were:

---
[1]Also my new start-up idea
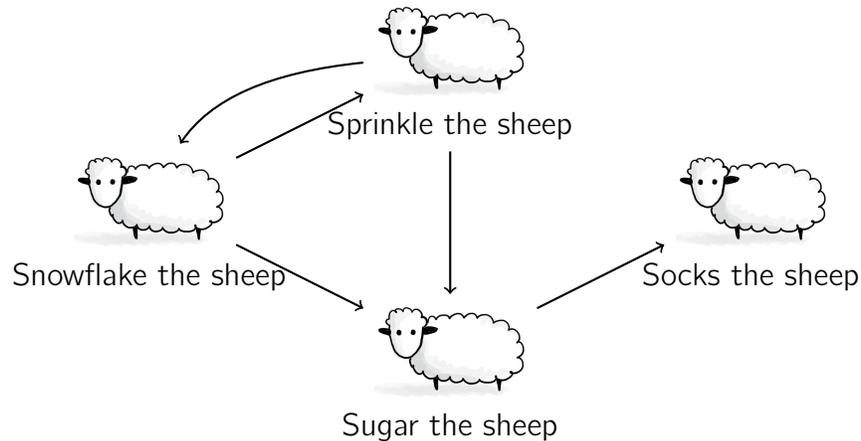
then Socks the Sheep follows Sugar the Sheep, and Sugar follows both Sprinkle and Snowflake, and so on. This means that Socks will re-post anything that Sugar posts, Sugar will re-post anything by Snowflake and Sprinkle, and so on. (If there is a cycle then each sheep will only re-post a post once).

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 (4 pt.)

A sheep is an **influencer** if anything that they post eventually gets re-posted by every other sheep on the island. In the example above, both Snowflake and Sprinkle are influencers.

Prove that, if there is at least one influencer, then (1) all influencers are in the same strongly connected component of G, and (2) every sheep in that component is an influencer.

[We are expecting: A short but rigorous proof.**]**

Solution

5.2 Who is the influencer? (4 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.

[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? (3 pt.)

Suppose that you do **not** know whether there is an influencer. Give an algorithm that runs in time $O(n+m)$ and returns either an influencer or 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 the 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 Currency conversion

Assume the world economy is comprised of a set of currencies $C_1, C_2, \ldots, C_n$ – think of these as dollars, pounds, bitcoins, etc. You can trade each currency C_i for any other currency C_j through your bank, but you will be charged in some way (described in the subproblems below). In these subproblems, you will devise algorithms to trade currencies to maximize the amount you end up with.

6.1 Flat fees (6 pt.)

Suppose that for each ordered pair of currencies (C_i, C_j), the bank charges a flat fee of $f_{ij} > 0$ dollars to exchange C_i for C_j (regardless of the quantity of currency being exchanged). Devise an efficient algorithm which, given a starting currency C_s, a target currency C_t, and a list of fees f_{ij} for all $i, j \in \{1, 2, \ldots, n\}$, computes the cheapest way to exchange all of our currency in C_s to currency C_t. Justify the correctness of your algorithm and its runtime.

[We are expecting: English description of the algorithm, followed by short paragraphs describing its correctness and running time**]**

6.2 Exchange rates (6 pt.)

Consider the more realistic setting where the bank does not charge flat fees, but instead uses exchange rates. In particular, for each ordered pair (C_i, C_j), the bank lets you trade one unit of C_i for r_{ij} units of C_j. Devise an efficient algorithm which, given starting currency C_s, target currency C_t, and a list of rates r_{ij} for all $i, j \in \{1, 2, \ldots, n\}$, computes a sequence of exchanges that results in the greatest amount of C_t. Justify the correctness of your algorithm and its runtime. You may assume that, for any currency C_i, it's impossible to start with one unit of C_i, go through a sequence of exchanges, and end up with strictly more than one unit of C_i.

[We are expecting: English description of the algorithm, followed by short paragraphs describing its correctness and running time**]**

Hint: How can you turn a product of terms into a sum?

6.3 Making money (6 pt.)

Due to fluctuations in the markets, it is occasionally possible to find a sequence of exchanges that results in a net gain. For example, you may start with a certain currency A, change into currencies, B, C, D, etc., and then end up changing back to currency A with more money than you started with. Mathematically, there may exist a sequence of currencies $C_{i_1}, \ldots, C_{i_k}$ such that

$$r_{i_1 i_2} \times r_{i_2 i_3} \times \ldots \times r_{i_{k-1} i_k} \times r_{i_k i_1} > 1.$$

Devise an efficient algorithm that finds such an anomaly **if one exists**. If there are multiple anomalies, your algorithm only needs to return any one of them. Justify the correctness of your algorithm and its runtime.

[We are expecting: English description of the algorithm, followed by short paragraphs describing its correctness and running time**]**