CS 161 (Stanford, Winter 2025) Homework 6

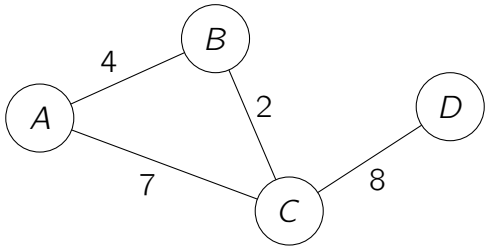
Style guide and expectations: We do NOT accept handwritten solutions. Please see the "Homework" part of the "Resources" section on the webpage for guidance on what we look for in homework solutions. We will grade according to these standards. You should cite all sources you used outside of the course 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 (5 pt.)

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 ($0 \to 1 \to 2 \to$

$\cdots \rightarrow n$). 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.

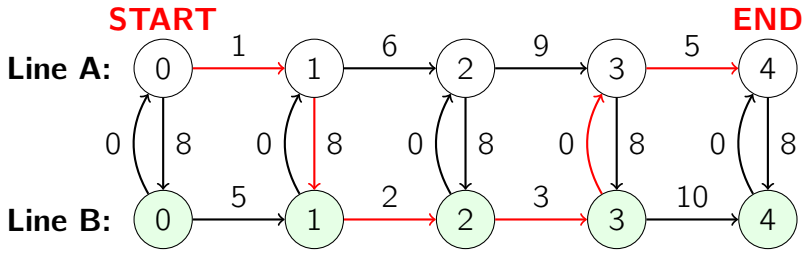


Figure 1: A sample problem with the two available routes to Plucky (line A at the top and line B at the bottom) and the minimum path solution. Note that Plucky can never go backward and that each transition in the graph has a specific cost associated with it. The cost to switch from line A to line B 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 $\{1, 7, 14, 19\}$

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] + min{A_costs[i], c + B_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

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

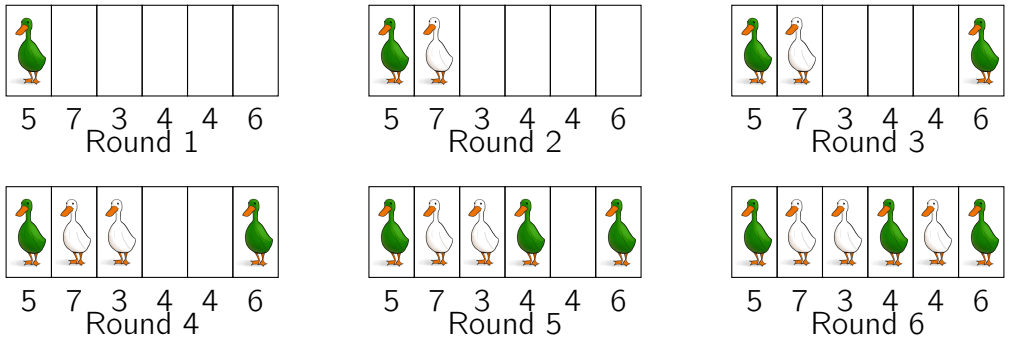
- Try the problems on your own *before* collaborating.
- Write up your answers yourself, in your own words. You should never share your typed-up solutions with your collaborators.
- If you collaborated, list the names of the students you collaborated with at the beginning of each problem.

3 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 troupe 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 $O(n^2)$.

3.1 Solution Outline (5 pt.)

What sub-problems will you use in your dynamic programming algorithm? What is the recursive relationship which 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.**]**

3.2 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

4 Most Reliable Path

After a wonderful trip from town s to city t, you decided to open a tourism company and provide the same trip plan to tourists. There is no direct flight from s to t so you need to take multiple flights to reach t, and the probability of cancellation for each flight varies from month to month. You created a directed graph $G(V, E)$ where each node is an airport and each edge is a flight from one airport to another.

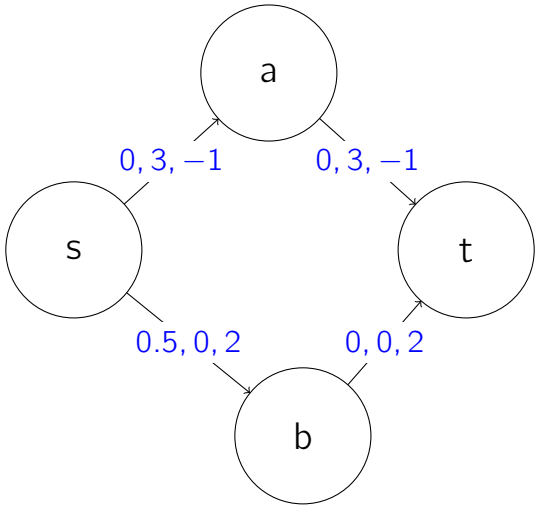
You have calculated *reliability scores* based on the history of cancellation and time efficiency for the flights in each month (can be either positive or negative or zero), and you want to maximize the total reliability score along the trip. However, your customers expect to visit exactly the same stops as they see on social media posts from you, so they don't like changes.

The reliability score of a route R is denoted $S_i(R)$, and is the total reliability score of flights in the route corresponding to the i-th month. $\text{Differences}(R_1, \ldots, R_k)$ is the number of months $i > 1$ such that $R_i \neq R_{i-1}$. The cumulative tourist satisfaction from a sequence of k routes is given by:

$$\text{Satisfaction}(R_1, \ldots, R_k) := \left(\sum_{i=1}^{k} S_i(R_i) \right) - \text{Differences}(R_1, \ldots, R_k).$$

Assumption $S_i(R)$ is finite for every i and any route R from s to t (i.e. no positive cycles).

Example



In this example, the tuples represent reliability scores of the flights over months. The optimal tour is to use route $(s \to a \to t) = R_1 = R_2$ for the first two trips, and then $(s \to b \to t) = R_3$ for the third trip. The total satisfaction is:

$$\text{Satisfaction}(R_1, R_2, R_3) = (0 + 0) + (3 + 3) + (2 + 2) - 1 = 9$$

where the -1 represents switching route once between R_2 and R_3. Notice that $(s \to b \to t)$ would have a higher score (0.5) for the first trip, but it wouldn't offset the cost (1) of adding another switch.

1. **(4 pt.)** If all routes had to be exactly the same, describe an algorithm that would maximize only the first term, namely $\sum_{i=1}^{k} S_i(R_i)$. What is the runtime of this algorithm?

   [We are expecting: A short English description. You only need to give an $O(\cdot)$ form for the runtime.**]**

   Solution

2. **(6 pt.)** Consider $\ell \in \{1, 2, \ldots, k\}$, any partition of the k months into ℓ pairwise disjoint intervals $I_1 \cup I_2 \cup \cdots \cup I_\ell = \{1, 2, \ldots, k\}$, and any ℓ routes $\mathcal{R}_1, \ldots, \mathcal{R}_\ell$. Consider the schedule in which we let $R_i = \mathcal{R}_j$ for each $j \in \{1, 2, \ldots, \ell\}$ and $i \in I_j$; in other words, we use route $\mathcal{R}_j$ for each month i in the interval I_j. Prove that

   (a) The total satisfaction of this specific schedule is at least

$$\sum_{j=1}^{\ell} \sum_{i \in I_j} S_i(\mathcal{R}_j) - (\ell - 1) \ .$$

   (b) The best total satisfaction equals to

$$\max_{1 \le \ell \le k} \ \max_{\substack{\text{disjoint intervals} \\ I_1 \cup \cdots \cup I_\ell = \{1, \ldots, k\}}} \ \max_{\mathcal{R}_1, \ldots, \mathcal{R}_\ell} \quad \sum_{j=1}^{\ell} \sum_{i \in I_j} S_i(\mathcal{R}_j) - (\ell - 1) \ .$$

   [We are expecting: Formal mathematical proofs.**]**

   Solution

3. **(10 pt.)** Design a $O(k^2 \cdot mn)$ dynamic programming algorithm to find the routes that maximize the satisfaction score, where m is the number of flights and n is the number of airports.

   Hint: What sub-problems will you use in your dynamic programming algorithm? How should we use the facts we have proved in part 2?

   [We are expecting: A short English description, pseudocode, and running time analysis. For simplicity, your pseudocode only needs to output the best satisfaction score. **]**

   Solution

5 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.

5.1 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**]**

Solution

5.2 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}, 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 there are no negative cycles.

[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?

Solution

5.3 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**. 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**]**

Solution