CS 161 (Stanford, Winter 2023) Homework 6

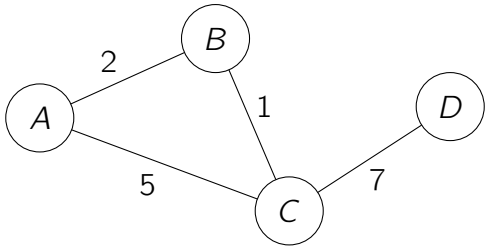
Style guide and expectations: Please see the "Homework" part of the "Resources" section on the webpage for guidance on what we are looking for in homework solutions. We will grade according to these standards. You should cite all sources you used outside of the course material.
What we expect: Make sure 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 Floyd-Warshall Trace (5 pt.)

Given the undirected graph below, trace through the Floyd-Warshall algorithm. 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: This question asks you to trace Floyd-Warshall on an undirected graph, but the algorithm also works on directed ones. 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: You to replace every "??" in the table below with an integer. No explanation is required.**]**

Initial (Direct Edge Weights)				
-	a	b	c	d
a	0	2	5	∞
b	-	0	1	∞
c	-	-	0	7
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	2	3	10
b	-	0	1	8
c	-	-	0	7
d	-	-	-	0

2 Finding Princess Peach (5 pt.)

Princess Peach has been kidnapped again and Mario needs to save her! In order to get to her, Mario needs to go through many kingdoms. The good thing is that Mario knows not just one, but actually, two separate highways to take him from kingdom 0 (where he starts his journey) to kingdom n (the destination where Princess Peach is).

Both highways go linearly through all the kingdoms in sequential order ($0 \to 1 \to 2 \to \cdots \to n$) and Mario cannot go backward. He can, however, jump from highway A onto highway B and vice-versa at any given kingdom. The issue is: every time Mario moves from one kingdom to another on either of the highways, he must pay a toll of a certain number of stars, and switching from highway A to highway B also costs some stars. The cost of switching from highway A to highway B is always fixed regardless of the kingdom in which Mario does it, and there is no price to jump from highway B back to highway A.

The task at hand is to find the minimum cost path going from kingdom 0 to kingdom n,

considering that Mario always starts on kingdom 0 on highway A. It does not matter if Mario finishes his path on kingdom n at highway A or highway B since he can move from highway B to highway A at cost 0.

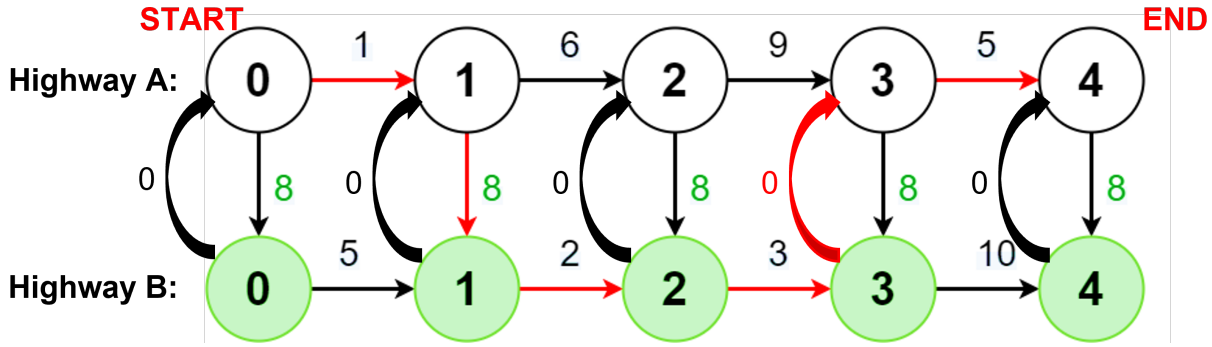


Figure 1: A sample problem with the two available routes to Mario (highway A at the top and highway B at the bottom) and the minimum path solution. Note that Mario can never go backward and that each transition in the graph has a specific cost associated with it. The cost to switch from highway A to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

You are given two arrays to solve this problem: A_costs and B_costs, each containing the transition costs on their respective highway in sequential order, such as A_costs = $\{1, 6, 9, 5\}$; and you are given an integer c for the cost to switch from highway A to highway B, 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 kingdoms (considering that a kingdom is reached regardless of the highway 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 "kingdom i on highway A", and $(1, i)$ represents the same idea for highway 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

```
def mario_finding_peach(A_costs, B_costs, c):
    dp_table = generate (2xn) matrix of zeros
    dp[0][0] = 0 # there is no price to reach kingdom 0 on path A
    dp[1][0] = c # price to reach kingdom 0 on path B
    for i in [0, n-1]:
        dp_table[0][i+1] =
              dp_table[0][i] + min{A_costs[i], c + B_costs[i]}
        dp_table[1][i+1] =
```

```
                min{dp_table[1][i] + B_costs[i], dp_table[0][i+1] + c}
    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**]**

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.

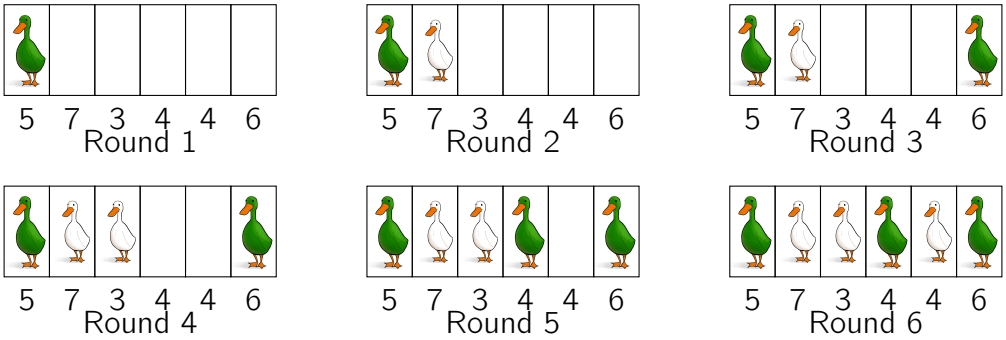
---

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 (a positive number), which 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 each dancer 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 an existing dancer, or next to the edge of the dance floor. 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 (10 pt.)

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

[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 (10 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).**]**

4 Taking Stock

Suppose you are given reliable insider information about the prices for k different stocks over the next n days. That is, for each day $i \in [1, n]$ and each stock $j \in [1, k]$, you're given $p_{i,j}$, the price at which stock j trades at on the ith day. You start with a budget of P dollars, and on each day, you may purchase or sell as many shares of each type of stock as you want, as long as you can afford them. (Assume that all prices are integer-valued and that you can only purchase whole stocks.) You have an earning goal of Q dollars. Here, we will design an algorithm to determine whether you can meet your goal, and if not, how much money you can earn.

4.1 Two days (8 pt.)

Suppose we are only looking at prices over two days (i.e. $n = 2$). Design an $O(kP)$ dynamic programming algorithm that computes the amount of money you can make buying stocks on the first day and selling stocks on the second day. Prove the runtime and correctness of your algorithm.
(*Hint: Let $M[l]$ be the most amount of money you can make by buying l dollars of stock on the first day. Write a recursive relationship for $M[l]$.*)
[We are expecting: Pseudocode or a clear English description of the algorithm, a runtime analysis, and a rigorous proof of correctness**]**

4.2 N days (8 pt.)

Now, suppose you are given prices over n days. Using your solution to part (a) as a guide, design an $O(nkQ)$ time algorithm that determines whether you can reach your goal, and if not, reports how much money you can make. Prove your algorithm's runtime and correctness. (*Hint: It's helpful to reframe each day as 1) selling all the shares you own and 2) then buying a set of shares that you can afford.*)
[We are expecting: Pseudocode or a clear English description of the algorithm, a runtime analysis, and a rigorous proof of correctness**]**

5 Currency conversion

Suppose the various economies of the world use a set of currencies $C_1, C_2, \ldots, C_n$ – think of these as dollars, pounds, bitcoins, etc. Your bank allows you to trade each currency C_i for any other currency C_j, and finds some way to charge you for this service (in a manner to be elaborated in the subparts below). We will devise algorithms to trade currencies to maximize the amount we end up with.

5.1 Flat fees (10 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 (that is, incurring the least in fees) 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**]**

5.2 Exchange rates (10 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, i.e. you receive r_{ij} units of C_j in exchange of one unit of C_i. 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.

[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?

5.3 Making money (5 pt.)

Due to fluctuations in the markets, it is occasionally possible to find a sequence of exchanges that lets you start with currency A, change into currencies, B, C, D, etc., and then end up changing back to currency A in such a way that you end up with more money than you started with—that is, there are 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**]**