Project:

# Final project

COP4533 - Algorithms, Abstraction, and Design
University of Florida

Created and updated by Ayush Kumar, Laura Cruz
Last updated by Lisha Zhou May 14, 2025

## Contents

# 1 Project Description:

In this final project, you will have the opportunity to design and implement some algorithms to solve variations of Stock buy sell to maximize profit problem. The problem set is divided into different tasks, each with its own unique objectives and requirements.

## 1.1 Problem Statement - 1:

You are given a matrix $A$ of dimensions $m \times n$, where each element represents the predicted prices of $m$ different stocks for $n$ consecutive days. Your task is to determine a single transaction involving the purchase and sale of a single stock that yields the maximum profit.

### 1.1.1 Input:

- A matrix $A$ of dimensions $m \times n$ ($1 \leq m, n \leq 1000$), where each element $A[i][j]$ ($0 \leq A[i][j] \leq 10^5$) represents the predicted price of the $i$-th stock on the $j$-th day. Rows represent stocks, columns represent days.

### 1.1.2 Output:

Return a tuple $(i, j_1, j_2, \text{profit})$ where:

- $i$ ($1 \leq i \leq m$) is the index of the chosen stock.
- $j_1$ ($1 \leq j_1 \leq n$) is the day when you would sell the stock to maximize profit.
- $j_2$ ($1 \leq j_2 \leq n$) is the day when you would sell the stock to maximize profit.
- profit is the maximum profit achievable through this single transaction (sell price minus buy price).

### 1.1.3 Constraints:

1. You can only perform one transaction, which involves buying and selling a single stock. You can only sell the stock you bought previously.
2. You must buy before selling; in other words, $j_1$ must be less than $j_2$.
3. The prices are non-negative integers and represent the stock's value at a specific time.
4. You want to maximize profit, so you're looking for the highest possible difference between the sell price and the buy price.
5. If no profitable transaction is possible, return a tuple $(0, 0, 0, 0)$ to indicate this.

### 1.1.4 Example:

**Input:**

$$A = \begin{bmatrix} 7 & 1 & 5 & 3 & 6 \\ 2 & 4 & 3 & 7 & 9 \\ 5 & 8 & 9 & 1 & 2 \\ 9 & 3 & 14 & 8 & 7 \end{bmatrix}$$

**Output:**

$$(4, 2, 3, 11)$$

**Explanation:** Choosing the 4th stock (1-based indexing). Buying it in day 2nd day and selling it on the 3rd day (1-based indexing), yields the maximum profit of 11 (sell price 14 minus buy price 3).

## 1.2 Problem Statement - 2:

You are given a matrix $A$ of dimensions $m \times n$, where each element represents the predicted prices of $m$ different stocks for $n$ consecutive days. Additionally, you are given an integer $k$ ($1 \leq k \leq n$). Your task is to find a sequence of at most $k$ transactions, each involving the purchase and sale of a single stock, that yields the maximum profit.

### 1.2.1 Input:

- A matrix $A$ of dimensions $m \times n$ ($1 \leq m, n \leq 1000$), where each element $A[i][j]$ ($0 \leq A[i][j] \leq 10^5$) represents the predicted price of the $i$-th stock on the $j$-th day.
- An integer $k$ ($1 \leq k \leq n$) representing the maximum number of transactions allowed.

### 1.2.2 Output:

Return a sequence of tuples $(i, j_1, j_2)$ representing the transactions where:

- $i$ ($1 \leq i \leq m$) is the index of the chosen stock.
- $j_1$ ($1 \leq j_1 \leq n$) is the day when you would buy the stock to maximize profit in this transaction.
- $j_2$ ($1 \leq j_2 \leq n$) is the day when you would sell the stock to maximize profit in this transaction.

### 1.2.3 Constraints:

1. You can perform at most $k$ transactions, each involving the purchase and sale of a single stock.
2. You must buy before selling; in other words, $j_1$ must be less than $j_2$.
3. The prices are non-negative integers and represent the stock's value at a specific time.
4. You want to maximize profit, so you're looking for the highest possible difference between the sell price and the buy price.
5. If no profitable transaction is possible, return an empty sequence. If there are multiples sequences achieving the same optimal profit, return any one of them.

### 1.2.4 Example:

**Input:**

$$A = \begin{bmatrix} 7 & 1 & 5 & 3 & 6 \\ 2 & 9 & 3 & 7 & 9 \\ 5 & 8 & 9 & 1 & 6 \\ 9 & 3 & 4 & 8 & 7 \end{bmatrix}$$

$$k = 3$$

**Output:**

$$[(2, 1, 2), (1, 2, 3), (2, 3, 5)]$$

**Explanation:** Performing at most 3 transactions, selling the 2nd stock on the 2nd day after buying on 1st day, 1st stock selling on the 3rd day buying on 2nd day, 2nd stock selling on 5th day buying on 3rd day yields the maximum profit of 17 (transaction 1: $9 - 2 = 7$, transaction 2: $5 - 1 = 4$, transaction 3: $9 - 3 = 6$). **Note**: In each transaction tuple *(i, $j_1$, $j_2$)*, the buy day $j_1$ is strictly less than the sell day $j_2$. However, when comparing different transactions, we do not consider them overlapping as along as the actual days involved do not conflict. For example, (1,2) and (2,3) are not considered as overlapping because the sell day of the first is the same as the buy day of the second. Similarly, (2,3) and (3,5) are valid together. But if it is (2,4) and (3,5), then we consider them as overlapping.

## 1.3 Problem Statement - 3:

You are given a matrix $A$ of dimensions $m \times n$, where each element represents the predicted prices of $m$ different stocks for $n$ consecutive days. Additionally, you are given an integer $c$ ($1 \leq c \leq n - 2$). Your task is to find the maximum profit achievable by trading stocks, subject to the restriction that you cannot buy any stock for $c$ days after selling any stock. If you sell a stock on day $j_1$, you are not allowed to buy any stock until day $j_1 + c + 1$ in the next transaction.

**Input:**

- A matrix $A$ of dimensions $m \times n$ ($1 \leq m, n \leq 1000$), where each element $A[i][j]$ ($0 \leq A[i][j] \leq 10^5$) represents the predicted price of the $i$-th stock on the $j$-th day.
- An integer $c$ ($1 \leq c \leq n-2$) representing the waiting period after selling a stock before buying another.

**Output:**

Return a sequence of tuples $(i, j_1, j_2)$ representing the transactions where:

- $i$ ($1 \leq i \leq m$) is the index of the chosen stock.
- $j_1$ ($1 \leq j_1 \leq n$) is the day when you would buy the stock to maximize profit in this transaction.
- $j_2$ ($1 \leq j_2 \leq n$) is the day when you would sell the stock to maximize profit in this transaction.

#### 1.3.1 Example:

**Input:**

$$A = \begin{bmatrix} 2 & 9 & 8 & 4 & 5 & 0 & 7 \\ 6 & 7 & 3 & 9 & 1 & 0 & 8 \\ 1 & 7 & 9 & 6 & 4 & 9 & 11 \\ 7 & 8 & 3 & 1 & 8 & 5 & 2 \\ 1 & 8 & 4 & 0 & 9 & 2 & 1 \end{bmatrix}$$

$$c = 2$$

**Output:**

$$[(3, 1, 3), (2, 6, 7)]$$

**Explanation:** To achieve the maximum profit, buy 3rd stock on day 1, sell it on day 3. buy 2nd stock on day 6 and sell it on day 7 adhering to 2 days waiting period.

# 2 Tasks

## 2.1 Task - 1: Brute Force Algorithm for Problem1

Design a brute force algorithm for solving Problem1 that runs in $O(m \cdot n^2)$ time.

## 2.2 Task - 2: Greedy Algorithm for Problem1

Design a greedy algorithm for solving Problem1 that runs in $O(m \cdot n)$ time.

## Task - 3: Dynamic Programming Algorithm for Problem1

Design a dynamic programming algorithm for solving Problem1 that runs in $O(m \cdot n)$ time.

## 2.3 Task - 4: Dynamic Programming Algorithm for Problem2

Design a dynamic programming algorithm for solving Problem2 that runs in $O(m \cdot n^{2k})$ time.

## 2.4 Task - 5: Dynamic Programming Algorithm for Problem2

Design a dynamic programming algorithm for solving Problem2 that runs in $O(m \cdot n \cdot k)$ time.

## 2.5 Task - 6: Dynamic Programming Algorithm for Problem3

Design a dynamic programming algorithm for solving Problem3 that runs in $O(m \cdot n^2)$ time.

### 2.6 Task - 7: Dynamic Programming Algorithm for Problem3

Design a dynamic programming algorithm for solving Problem3 that runs in $O(m \cdot n)$ time.

## 3 Groups

For this project you may choose a group to work with. Groups can be of maximum 5 students. You can work individually, if that is your preference. You can use any communication platform (Canvas or Slack) to talk to your peers and find a group. Instructional team does not oversee communication between teams, it is your responsibility to organize your work and find the best way to work together. There are three milestones in the project and while you can change your mind after your first milestone, the submission members on the second submission must be the members on the third submission.

## 4 Milestone 1: Understanding the problems

### 4.1 Problem 1

Work out the given numerical example for Problem-1. That is, You are given a matrix $A$ of dimensions $m \times n$, where each element represents the predicted prices of $m$ different stocks for $n$ consecutive days. Your task is to manually determine the maximum profit achievable through a single transaction, involving the purchase and sale of a single stock. (Refer to problem statement - 1 for output format).

#### Input Matrix $A$

$$A = \begin{bmatrix} 12 & 1 & 5 & 3 & 16 \\ 4 & 4 & 13 & 4 & 9 \\ 6 & 8 & 6 & 1 & 2 \\ 14 & 3 & 4 & 8 & 10 \end{bmatrix}$$

#### Steps

1. Begin with the input matrix $A$ as provided.
2. For each stock, calculate the potential profit that could be obtained by selling the stock on each day after buying it.
3. Identify the day with the highest potential profit for each stock.
4. Determine the stock and day combination that yields the maximum potential profit.

## Notes

- This milestone involves manually applying the basic logic of calculating potential profits and selecting the optimal transaction for a single stock.
- The goal is to understand the foundational logic of finding the maximum profit for a single transaction by considering the price differences between different days for a specific stock.

### 4.2 Problem 2

Work out the given numerical example for Problem-2. That is, You are given a matrix $A$ of dimensions $m \times n$, where each element represents the predicted prices of $m$ different stocks for $n$ consecutive days. Additionally, you are given an integer $k$ ($1 \leq k \leq n$). Your task is to manually find a sequence of at most $k$ transactions, each involving the purchase and sale of a single stock, that yields the maximum profit. (Refer to problem statement - 2 for output format).

### Input Matrix $A$

$$A = \begin{bmatrix} 25 & 30 & 15 & 40 & 50 \\ 10 & 20 & 30 & 25 & 5 \\ 30 & 45 & 35 & 10 & 15 \\ 5 & 50 & 35 & 25 & 45 \end{bmatrix}$$

$$k = 3$$

### Steps

1. Begin with the input matrix $A$ as provided.
2. Determine the sequence of at-most K non-overlapping transactions. A valid transaction is a buy-sell of the same stock. Different transactions can have different stocks, but one transaction would deal with only a single stock.
3. Output should be a sequence of at most K transactions in the format of ($i$,$j$,$l$) that yields the maximum potential profit by selling $i$th stock on $l$th day that was bought on $j$th day.

## Notes

- This milestone involves manually applying the basic logic of calculating potential profits and selecting the optimal transaction for a single stock.
- The goal is to understand the foundational logic of finding the maximum profit for atmost K non-overlapping transactions by considering the price differences between different days for a specific stock (in a single transaction).

## 4.3 Problem 3

Work out the given numerical example for Problem-3. Work out the given test-case for Problem-3 manually. That is, You are given a matrix $A$ of dimensions $m \times n$, where each element represents the predicted prices of $m$ different stocks for $n$ consecutive days. Additionally, you are given an integer $c$ ($1 \leq c \leq n-2$). Your task is to manually determine the maximum profit achievable under the given trading restrictions, where you cannot buy any stock for $c$ days after selling any stock. If you sell a stock on day $i$, you are not allowed to buy any stock until day $i + c + 1$. (Refer to problem statement - 3 for output format).

### Input Matrix $A$

$$A = \begin{bmatrix} 7 & 1 & 5 & 3 & 6 & 8 & 9 \\ 2 & 4 & 3 & 7 & 9 & 1 & 8 \\ 5 & 8 & 9 & 1 & 2 & 3 & 10 \\ 9 & 3 & 4 & 8 & 7 & 4 & 1 \\ 3 & 1 & 5 & 8 & 9 & 6 & 4 \end{bmatrix}$$

$c = 2$

### Steps

1. Begin with the input matrix $A$ and integer $c$ as provided.
2. For each day $j$, identify the maximum price to sell the stock $l$. You can only buy another stock after $c + 1$ days (i.e., on day $l + c + 1$ or later)
3. Determine the sequence ($i$,$j$,$l$) that yields the maximum potential profit by selling $i$th stock on $l$th day that was bought on $j$th day.

## Notes

- This milestone involves manually applying the logic of identifying the optimal days to buy and sell stocks based on the given trading restrictions.
- The goal is to understand the foundational logic of finding the maximum profit under trading restrictions while considering the waiting period between selling and buying stocks.

### 4.4 TODO:

- Create a private GitHub Repository and add instructional team members as contributors
- Coordinate with your team members : communication methods, internal deadlines, responsibilities, meetings, and roles.
- Solve together the numerical examples provided
- Push into the GitHub repository a pdf document titled: *Project - Milestone 1.pdf* . The document must contain the following sections: Group members (names, emails and GitHub accounts), Member Roles, Communication method, Project Gantt Chart (including internal deadlines and meetings), GitHub repository link, and solution to the three working examples.
- One member of the group must submit the pdf document to Canvas too

# 5 Milestone 2: Algorithm Design

Formulate the problems and design an algorithms. Provide pseudocode for the following tasks:

1. Task - 1
2. Task - 2
3. Task - 3
4. Any two tasks between task - 4, task - 5, task - 6, task - 7.

You must assure that all variables and definitions required are provided before the pseudo-code, for each task.

### 5.1 TODO:

- Push into the GitHub repository a pdf document titled: *Project - Milestone 2.pdf* . The document must contain all sections from Milestone 1, and in addition to it: the programming language to be used in the implementation of the algorithms, all definitions and assumptions necessary to design the algorithm of each task, and the pseucode of each algorithm.
- One member of the group must submit the pdf document to Canvas too

# 6 Milestone 3: Algorithm Implementation

In Milestone 2, you designed algorithms and provided pseudocode for various tasks. Now, it's time to move to Milestone 3, where you will implement those algorithms in a programming language of your choice, such as C++, Python, or Java. In this assignment, you will focus on implementing the algorithms for the tasks you picked in milestone 2. Note: You may modify the algorithm proposed in milestone 2, however, you must note any changes in the design and provide the rationale for those changes. **Instructions:**

1. For each of the tasks from Milestone 2, implement the algorithm in your preferred programming language (C++ or Python).
2. Provide the code for the algorithm along with comments to explain your implementation.

3. Test your implementation with appropriate test cases (you will be in charge of creating these test cases) to ensure it works correctly.

**Additional Information:**

- You are encouraged to use suitable data structures, loops, and conditional statements as needed to implement the algorithm effectively.
- Be sure to provide comments within your code to explain key steps and logic.
- Test your implementation thoroughly with various inputs to ensure correctness.
- Document any assumptions or limitations in your code comments.
- Provide citations in your code and document containing any source information that you are using in your project

### 6.1 TODO:

- Push into the GitHub repository a pdf document titled: *Project - Milestone 3.pdf* . The document must contain all sections from Milestone 1, and Milestone 2 and in addition to it description of the implementation of the algorithm, limitations, comparative analysis, trade-off discussion, analysis of the algorithms, lessons learned, limitations of your algorithms.
- One member of the group must submit the pdf document in Canvas too

## 7 Presentation

In addition to the algorithm analysis, you are required to create a presentation video that explains the pseudocode for each of the tasks. The video should be concise and should not exceed 10 minutes in total.

**Instructions for the Pseudocode Presentation:**

(a) Create one slide per task that was implemented. On each slide, present the pseudocode for the respective task. Note: make sure to provide a detailed explanation of the pseudocode, step by step.

(b) Discuss key algorithmic concepts, data structures used, and any critical decisions made during the implementation of the tasks.

(c) Make sure your explanations are clear, concise, and easy to follow.

(d) Record your presentation as a video, ensuring it does not exceed the 15-minute time limit.

(e) You may use presentation software or screen recording tools to create your video.

(f) Submit the video in a common format such as MP4.

(g) One member of the group must submit the video to Canvas

## 8 Rubrics

### 8.1 Milestone 1

### 8.2 Milestone 2

### 8.3 Milestone 3

## 9 General Notes

- Include any document required in the repository by each Milestone before the due date.
- Commit and push your files to the repository frequently.

| Item | Points | Observation |
|---|---|---|
| Github Repository | 5 | Full points if repository is complete and shared with instructional team |
| Logistics | 5 | The document contains all the parts mentioned in section 4.4 TODO |
| Problem 1 - correctness | 4 | The manual calculation is entirely correct and matches the expected result precisely. |
| Problem 2 - correctness | 4 | The manual calculation is entirely correct and matches the expected result precisely. |
| Problem 3 - correctness | 4 | The manual calculation is entirely correct and matches the expected result precisely. |
| Problem 1 - Clarity and Explanation | 6 | The manual calculation is clear and well-explained, making it easy to understand the steps taken to arrive at the result. |
| Problem 2 - Clarity and Explanation | 6 | The manual calculation is clear and well-explained, making it easy to understand the steps taken to arrive at the result. |
| Problem 3 - Clarity and Explanation | 6 | The manual calculation is clear and well-explained, making it easy to understand the steps taken to arrive at the result. |
| Total | 40 | |

Table 8.1: Points distribution for milestone 1 (Total Points: 40)

- Make sure the repository is accessible to the instructional team members (syllabus) and contains all the necessary files.
- Seek guidance from your instructor or classmates if you encounter difficulties during any stage of the project. Plagiarism is strictly prohibited; ensure that all work is original and properly cited.
- Remember that the goal of this project is not just to obtain correct results, but also to understand and explain the underlying concepts and strategies.

**Your project will be evaluated based on the following aspects:**

- Correctness and completeness of algorithm design and implementation.
- Accuracy of the comparative analysis and trade-off discussion.
- Clarity and quality of project documentation and presentation.
- Demonstrated understanding of algorithmic concepts and their application.
- Effort. As demonstrated by your commits to the project. This is not only the amount but also the steady work on the project during the semester. This is an individual evaluation.

| Item | Points | Observation |
|---|---|---|
| Github Repository | 5 | Full points if the repository is complete and shared with instructional team and contains the submission |
| Logistics | 5 | The document contains all the parts mentioned in section 5.1 TODO |
| Problem 1 - correctness | 15 | Pseudocode is entirely correct, with no logical errors, accurately represents the algorithm's logic, is clear and understandable. |
| Problem 2 - correctness | 15 | Pseudocode is entirely correct, with no logical errors, accurately represents the algorithm's logic, is clear and understandable. |
| Problem 3 - correctness | 15 | Pseudocode is entirely correct, with no logical errors, accurately represents the algorithm's logic, is clear and understandable. |
| Problem 1 - Clarity and Explanation | 15 | Pseudocode is exceptionally clear, with precise descriptions of each step and logical flow. |
| Problem 2 - Clarity and Explanation | 15 | Pseudocode is exceptionally clear, with precise descriptions of each step and logical flow. |
| Problem 3 - Clarity and Explanation | 15 | Pseudocode is exceptionally clear, with precise descriptions of each step and logical flow. |
| Total | 100 | |

Table 8.2: Points distribution for milestone 2 (Total Points: 100)

| Item | Points | Observation |
|---|---|---|
| Github Repository | 10 | Repository is complete and shared with instructional team and contains the submission |
| Logistics | 40 | The document contains all the parts mentioned in section 6.1 TODO |
| Problem 1 - code correctness | 25 | The code is correct, it solves the given problem statement and passes all hidden test cases without any errors. |
| Problem 2 - correctness | 25 | The code is correct, it solves the given problem statement and passes all hidden test cases without any errors. |
| Problem 3 - correctness | 25 | The code is correct, it solves the given problem statement and passes all hidden test cases without any errors. |
| Problem 1 - Readability and Efficiency | 25 | The code is well-structured, easy to read, and includes thorough comments explaining critical parts. |
| Problem 2 - Readability and Efficiency | 25 | The code is well-structured, easy to read, and includes thorough comments explaining critical parts. |
| Problem 3 - Readability and Efficiency | 25 | The code is well-structured, easy to read, and includes thorough comments explaining critical parts. |
| Total | 100 | |

Table 8.3: Points distribution for milestone 3 (Total Points: 200)