COP4533 Algorithms Programming Project

1 Problem Definition

We are given a array of price predictions for m stocks for n consecutive days. The price of stock i for day j is $A[i][j]$ for $i = 1, \ldots, m$ and $j = 1, \ldots, n$. You are tasked with finding the maximum possible profit by buying and selling stocks. The predicted price at any day will always be a non-negative integer. **You can hold at most one share of any stock at any time.** You are allowed to buy a stock on the same day you sell another stock. More formally,

PROBLEM1 Given a matrix A of $m \times n$ integers (non-negative) representing the predicted prices of m stocks for n days, find a single transaction (buy and sell) that gives maximum profit.

PROBLEM2 Given a matrix A of $m \times n$ integers (non-negative) representing the predicted prices of m stocks for n days and an integer k (positive), find a sequence of at most k transactions that gives maximum profit. You are allowed to buy and sell the same stock multiple times, as long as you hold at most one stock at any time.[Hint :- Try to solve for $k = 2$ first and then expand that solution.]

2 Algorithm Design Tasks

You are asked to design three different algorithms for each problem, with varying time complexity requirement in order to conduct an experimental comparative study.

ALG1 Design a $\Theta(m * n^2)$ time brute force algorithm for solving PROBLEM1

ALG2 Design a $\Theta(m * n)$ time greedy algorithm for solving PROBLEM1

ALG3 Design a $\Theta(m * n)$ time dynamic programming algorithm for solving PROBLEM1

ALG4 Design a $\Theta(m * \binom{n}{2k})$ time brute force algorithm for solving PROBLEM2

ALG5 Design a $\Theta(m * n^2 * k)$ time dynamic programming algorithm for solving PROBLEM2

ALG6 Design a $\Theta(m * n * k)$ time dynamic programming algorithm for solving PROBLEM2

3 Programming Tasks

Once you complete the algorithm design tasks, perform the following programming tasks:

TASK1 Give an implementation of ALG1.

TASK2 Give an implementation of ALG2.

TASK3A Give a recursive implementation of ALG3 using **Memoization**.

TASK3B Give an iterative **BottomUp** implementation of ALG3.

TASK4 Give an implementation of ALG4.

TASK5 Give an implementation of ALG5.

TASK6 Give an implementation of ALG6 (either **Memoization** or **BottomUp**, or both ☺).

4 Language/Input/Output Specifications

You may use C++, Java. Your program must compile/run on the Thunder CISE server using gcc/g++ or standard JDK. You may access the server using SSH client on `thunder.cise.ufl.edu`. You must write a `makefile` document that creates an executable named **Stocks**. The task is passed by an argument, e.g., when **Stocks 3b** is called from the terminal, your program needs to execute the implementation of TASK3B. Through out this assignment assume that the indices of the stocks are $1 \dots m$ and the indices of the days are $1 \dots n$.

PROBLEM1: For convenience assume that $1 \leq m < 100$, $1 \leq n < 10^5$ and $\forall i \;\; 0 \leq A[i][j] < 10^4$. If multiple buy/sell transaction pairs yield the maximum profit, output any one of them.

Input. Your program will read input from standard input (`stdin`) in the following order:

- Line 1 consists of two integers m and n separated by a single space.
- Next m lines each consists of n integers (prices for n days) separated by a single space.

Output. Print the optimal transaction info to standard output (`stdout`)

- A single line with 3 integers (*Stock, BuyDay,* & *SellDay* indices) separated by a space.

PROBLEM2: For convenience assume that $1 \leq k < 100$, $1 \leq m < 100$, $1 \leq n < 1000$ and $\forall ij \;\; 0 \leq A[i][j] < 1000$. If multiple sets of transactions yield the maximum profit, output any one of them.

Input. Your program will read input from standard input (`stdin`) in the following order

- Line 1 consists one integer k.
- Line 2 consists of two integers m and n separated by one space character.
- Next m lines each with n integers (predicted prices) separated by a single space.

Output. Print a set of (at most k) transactions that yields the max profit in order of dates.

- Each line with 3 integers (*Stock, BuyDay,* & *SellDay* indices) separated by a single space.

5 Experiments and Report

Conduct an experimental study to test the performance/scalability of your algorithms/implementations. Your report must include at least the following components: i) Team members; ii) Design and Analysis of Algorithms; iii) Experimental Comparative Study; iv) Conclusion.

5.1 Team Members

You are allowed to work as teams of (at most) two students on this assignment. Clearly state the name and the contribution of each team member.

5.2 Design and Analysis of Algorithms

Describe each algorithm and analyze it (correctness, time & space complexity). Also include the recursive formulation expressing optimal substructure for the dynamic programming algorithms.

5.3 Experimental Comparative Study

Test your implementations extensively for correctness and performance. For this purpose, you should create randomly generated input files of various sizes. The exact size of the experimental data sets that your program can handle depends on the quality of your implementation. For instance, you might want to choose $n = 1000, 2000, 3000, 4000, 5000$ for TASK1, and $n = 20, 40, 60, 80, 100$ for TASK4 to create at least five data sets for each experiment. Then, you should conduct and present a performance comparison of at least the following: For each comparison, generate a two dimensional plot of running time (y-axis) against input size (x-axis). These should be included in your report along with additional comments/observations.

PLOT1 Comparison of TASK1, TASK2, TASK3A, TASK3B with variable n and fixed m

PLOT2 Comparison of TASK1, TASK2, TASK3A, TASK3B with variable m and fixed n

PLOT3 Comparison of TASK4, TASK5, TASK6 with variable n and fixed m and k

PLOT4 Comparison of TASK4, TASK5, TASK6 with variable m and fixed n and k

PLOT5 Comparison of TASK4, TASK5, TASK6 with variable k and fixed m and n

Pick an appropriate value for the fixed values. For each comparison, generate a two dimensional plot of running time (y-axis) against input variable (x-axis). Also include additional plots whenever the plot of one task overshadows others. For instance, if you ended up implementing both top-down and bottom-up versions of TASK6, add a separate plot comparing them not to be overshadowed by TASK4 and/or TASK5. Feel free to include additional plots to present your results.

5.4 Conclusion

Summarize your learning experience on this project assignment. For each programming task, comment on the ease of implementation and other potential technical challenges.

6 Submission

The following contents are required for submission:

1. **Makefile**: Your makefile must be directly under the zip folder. No nested directories. Do not locate the executable file in any directory either.
2. **Source code**: should include detailed comments next to each non-trivial block of code.
3. **Report**: The report must be in PDF format.
4. **Bundle**: Compress all your files together using a zip utility and submit it on Canvas. Each team makes only one submission with a single zip file identified with last and first names of team members, i.e., **LName1FName1LName2FName2.zip**

7 Grading Policy

Grades will be based on the correctness & efficiency of algorithms and the quality of the report:

- **Program 50%.** Correct/efficient design and implementation/execution. Also make sure to include comments with your code for clarity.

- **Report 50%.** Quality (clarity, details) of the write up on your design, analysis, programming experience, and experimental study.

8 Bonus [upto %30]

For upto 30% bonus points, solve this problem with an additional constraint. Make sure to include all parts of the report for these bonus tasks as well.

PROBLEM3 Given a matrix A of $m \times n$ integers (non-negative) representing the predicted prices of m stocks for n days and an integer c (positive), find the maximum profit with no restriction on number of transactions. However, you cannot buy any stock for c days after selling any stock. If you sell a stock at day i, you are not allowed to buy any stock until day $i + c$

8.1 Algorithm Design Tasks

ALG7 Design a $\Theta(m * 2^n)$ time brute force algorithm for solving PROBLEM3

ALG8 Design a $\Theta(m * n^2)$ time dynamic programming algorithm for solving PROBLEM3

ALG9 Design a $\Theta(m * n)$ time dynamic programming algorithm for solving PROBLEM3

8.2 Programming Tasks

TASK7 Give an implementation of ALG7.

TASK8 Give an implementation of ALG8.

TASK9 Give an implementation of ALG9.

8.3 Input/Output Specifications

For convenience assume that $1 \leq c < 100$, $1 \leq m < 100$, $1 \leq n < 1000$ and $\forall i \;\; 0 \leq A[i][j] < 1000$. If there are multiple possible sets of transactions that yield the maximum profit, output any one of them.

Input. Your program will read input from standard input (`stdin`) in the following order:

- Line 1 consists one integer c.
- Line 2 consists of two integers m and n separated by one space character.
- Next m lines each with n integers (predicted prices) separated by a single space.

Output. Print a set of transactions that yields the max profit in order of dates.

- Each line with 3 integers (*Stock, BuyDay,* & *SellDay* indices) separated by a single space.

8.4 Experimental Comparative Study

You might want to use the following comparisons for the Experimental Comparative Study:

PLOT6 Comparison of TASK7, TASK8, TASK9 with variable n and fixed m and c

PLOT7 Comparison of TASK7, TASK8, TASK9 with variable m and fixed n and c

PLOT8 Comparison of TASK8, TASK9 with variable n and fixed m and c

PLOT9 Comparison of TASK8, TASK9 with variable m and fixed n and c