Algorithms Programming Project

1 Problem Definition

Humans are planning to build their second home on Mars. Suppose that the Mars rovers have detected an area of $m \times n$ cells for us, and the air quality index of each cell is $M[i, j]$ for $i = 1, \dots, m$ and $j = 1, \dots, n$. You are tasked to find a rectangle area to build a new base where the total air quality index is maximized. For this you shall solve the following two problems. The solution of the first problem should help for designing an efficient solution for the second problem. For the rest of this document assume that m is $O(n)$.

PROBLEM1 Given an array A of n integers (positive or negative), find a contiguous subarray whose sum is maximum.

PROBLEM2 Given a two dimensional array M of size $m \times n$ consisting of integers (positive or negative), find a rectangle (two dimensional sub-array) whose sum is maximum.

2 Algorithm Design Tasks

You are asked to design three different algorithms with each problem, with varying time complexity requirement.

ALG1 Design a $\Theta(n^3)$ time brute force algorithm for solving PROBLEM1

ALG2 Design a $\Theta(n^2)$ time dynamic programming algorithm for solving PROBLEM1

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

ALG4 Design a $\Theta(n^6)$ time brute force algorithm for solving PROBLEM2

ALG5 Design a $\Theta(n^4)$ time algorithm for solving PROBLEM2 using dynamic programming ALG3

ALG6 Design a $\Theta(n^3)$ time algorithm for solving PROBLEM2 using dynamic programming ALG3

3 Programming Tasks

Once you compelte the algorithm design tasks, you should have an implementation for each of the following programming procedures:

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 using $O(1)$ extra space.

TASK5 Give an implementation of ALG5 using $O(mn)$ extra space.

TASK6 Give an implementation of ALG6 using $O(mn)$ extra space.

4 Language/Input/Output Specifications

You may use Java or C++. 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 **MarsBase**. The task is passed by an argument, e.g., when **MarsBase 3b** is called from the terminal, your program needs to execute the implementation of TASK3B.

PROBLEM1:

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

- Line 1 consists one integer n.
- Line 2 consists of n integers $A[1], A[2], ..., A[n]$ separated by one space character.

For convenience assume that $1 \leq n < 2^{31}$, and $\forall i \quad -2^{15} \leq A[i] < 2^{15}$.

Output. Print three integers l, r, sum to standard output (stdout) separated by a space character, where l is the leftmost index, r is the rightmost index of the optimal solution region, and sum is the summation of the elements in that subarray.

PROBLEM2:

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

- Line 1 consists two integers m, n separated by one space character.
- For the next m lines, line $i+1$ consist of n integers $M[i,1], M[i,2], ..., M[i,n]$ in this particular order separated by one space character.

For convenience assume that $1 \leq m, n < 2^{15}$, and $\forall i, j \quad -2^{15} \leq M[i,j] < 2^{15}$.

Output. Print five integers x_1, y_1, x_2, y_2, sum to standard output (stdout) separated by a space character, where (x_1, y_1) is the upper left corner, (x_2, y_2) is the lower right corner of the optimal solution region, and sum is the summation of the matrix entries in that rectangle.

5 Experiments and Report

You should conduct an experimental study to test the performance and scalability of your algorithms/implementations. Your report should 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 programming assignment. If you decide to work as a team of two, clearly state the names of team members and describe the main contributions of each member.

5.2 Design and Analysis of Algorithms

For each of the six algorithm design, you should clearly describe your method and present and analysis (correctness, time and space complexity). For the dynamic programming algorithms, make sure to clearly state a mathematical recursive formulation expressing the optimal substructure property; and argue its correctness, in addition to the time/space complexity analysis.

5.3 Experimental Comparative Study

You are expected to 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 $m \simeq n = 20, 40, 60, 80, 100$ for Task4 to create at least five data sets for each experiment. Then, you should conduct and present a performance comparison among Task1, Task2, Task3a, and Task3b and another one among Task4 Task5, and Task6. 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. Feel free to include additional plots to present your results. For instance, you might consider having a separate plot comparing Task3a and Task3b, not to shadow it by Task1 and Task2.

5.4 Conclusion

Summarize your learning experience on this project assignment. For each programming task, comment on 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 through the Canvas system. Your submission should be identified with your last name and first name, i.e., **LNameFName.zip**. For teams of two, use **LName1FName1LName2FName2.zip** and make only one submission on canvas.

7 Grading Policy

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

- **Program 60%.** Correct/efficient design and implementation/execution. Also make sure to include comments with your code for clarity.

- **Report 40%.** Quality (clarity, details) of the write up on your design, analysis, programming experience, and experimental study.