COT5405 Programming Project II
Dynamic Programming

1 Problem Definition

Consider a town named Greenvale that has a grid layout of m × n plots of land. Associated with each plot (i, j) where i = 1, . . . , m and j = 1, . . . , n, Greenvale's Parks and Recreation Department assigns a non-negative number p[i, j] indicating the minimum number of trees that must be planted on that plot. A local environmental group named GreenHands is interested in finding the largest possible square shaped area of plots within the town that requires a minimum of h trees to be planted on each plot individually. More formally, problems are stated below, where Problem1 is a special case of Problem2 and Problem3.

Problem1 Given a matrix p of $m \times n$ integers (non-negative) representing the minimum number of trees that must be planted on each plot and an integer h (positive), find the bounding indices of a square area where each plot enclosed requires a minimum of h trees to be planted.

Problem2 Given a matrix p of $m \times n$ integers (non-negative) representing the minimum number of trees that must be planted on each plot and an integer h (positive), find the bounding indices of a square area where all but the corner plots enclosed requires a minimum of h trees to be planted. The corner plots can have any number of trees required.

Problem3 Given a matrix p of $m \times n$ integers (non-negative) representing the minimum number of trees that must be planted on each plot and an integer h (positive), find the bounding indices of a square area where where only up to k enclosed plots can have a minimum tree requirement of less than h.

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 $\Theta(m^3n^3)$ time **Brute Force** algorithm for solving Problem1

Alg2 Design a $\Theta(m^2n^2)$ time algorithm for solving Problem1

Alg3 Design a $\Theta(mn)$ time **Dynamic Programming** algorithm for solving Problem1

Alg4 Design a $\Theta(mn^2)$ time **Dynamic Programming** algorithm for solving Problem2

Alg5 Design a $\Theta(mn)$ time **Dynamic Programming** algorithm for solving Problem2

Alg6 Design a $\Theta(m^3n^3)$ time **Brute Force** algorithm for solving Problem3

Alg7 Design a $\Theta(mnk)$ time **Dynamic Programming** algorithm for solving Problem3

3 Programming Tasks

Once you complete 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.

TASK3 Give an implementation of ALG3.

TASK4 Give an implementation of ALG4.

TASK5A Give a recursive implementation of ALG5 using **Memoization**.

TASK5B Give an iterative **BottomUp** implementation of ALG5.

TASK6 Give an implementation of ALG6.

TASK7A Give a recursive implementation of ALG7 using **Memoization**.

TASK7B Give an iterative **BottomUp** implementation of ALG7.

4 Language/Input/Output Specifications

You may use Java, Python, or C++. Your program must compile/run on the Thunder CISE server using gcc/g++, Python-3 interpreter, or standard JDK. You may access the server using SSH client on thunder.cise.ufl.edu. You must write a makefile document that executes first any compilation tasks then will run each task. First the command **make** will be run. Then each task should have its own command in the format **make run#** where the number corresponds to each task e.g. when **make run3** is called from the terminal, your program needs to execute the implementation of TASK3.

PROBLEM1:

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

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

For convenience assume that $1 \leq m, n \leq 10^4$, $0 \leq h \leq 10^9$, and $\forall i, j \;\; 0 \leq p[i,j] \leq 10^9$.

Output. Print four integers x_1, y_1, x_2, y_2 to standard output (stdout) separated by a space character, where (x_1, y_1) is the upper left corner and (x_2, y_2) is the lower right corner of the optimal solution region.

PROBLEM2:

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

- Line 1 consists of three integers m, n, h separated by one space character.

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

For convenience assume that $1 \leq m, n \leq 10^4$, $0 \leq h \leq 10^9$, and $\forall i, j \;\; 0 \leq p[i,j] \leq 10^9$.

Output. Print four integers x_1, y_1, x_2, y_2 to standard output (stdout) separated by a space character, where (x_1, y_1) is the upper left corner and (x_2, y_2) is the lower right corner of the optimal solution region.

Problem3:

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

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

For convenience assume that $1 \leq m, n \leq 10^4$, $0 \leq h \leq 10^9$, and $\forall i, j \;\; 0 \leq p[i,j] \leq 10^9$.

Output. Print four integers x_1, y_1, x_2, y_2 to standard output (stdout) separated by a space character, where (x_1, y_1) is the upper left corner and (x_2, y_2) is the lower right corner of the optimal solution region.

If there are multiple possible squares that yield the maximum area, output any one.

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, 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 three 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. Then, you

should do a performance comparison among Tasks 1 2 3, among Task 4 5A 5B and among Tasks 6 7A 7B. 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.

You should also include additional plots when plot of one task over shadows another. For instance, add a separate plot comparing Task2 and Task3, not to shadow it by Task1. 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 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 three, 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 60%.** Quality (clarity, details) of the write up on your design, analysis, programming experience, and experimental study.

Note that 60% +60% gives 20% bonus points option.