Project 2: Dynamic programming

COT 4400, Fall 2017

Due November 12, 2017

1 Overview

For this project, you will develop an algorithm to find the most efficient paths across rough terrain. Designing and implementing this solution will require you to model the problem using dynamic programming, develop a method to extract the most efficient paths from your model, and implement your solution.

You are only allowed to consult the class slides, the textbook, the TAs, and the professor. In particular, you are not allowed to use the Internet. This is a group project. The only people you can work with on this project are your group members. This policy is strictly enforced.

In addition to the group submission, you will also evaluate your teammates' cooperation and contribution. These evaluations will form a major part of your grade on this project, so be sure that you respond to messages promptly, communicate effectively, and contribute substantially to your group's solution. Details for your team evaluations are in Section 5.2. You will submit the peer evaluations to another assignment on Canvas, labelled "Project 2 (individual)."

A word of warning: this project is team-based, but it is a nontrivial task. You are highly encouraged to start working on (and start asking questions about) this project early; teams who wait to start until the week before the due date may find themselves unable to complete it in time.

2 Problem Description

For this project, you will be computing the most efficient paths across a given matrix, representing some region to be crossed. Each path should start at the northern (top) border of the region and make some combination of south, southeast, and southwest moves until it reaches the southern (bottom) border, and you should compute the most efficient path that ends at every location along the southern border of the region.

Each cell in the input matrix contains value representing the amount of time required to traverse that location. The total amount of time to traverse a path is the sum of time required to traverse each location in the path; however, locations are closer to the cells immediately to their north than to those to the northwest and northeast. In order to account for this, moves to the southwest and southeast add a 40% penalty to the time required to traverse the next location. (You would not pay this penalty on the first location to cross.) The "most efficient" path, of course, is the path that takes the least amount of time to cross according to this model.

In order to compute these paths efficiently, you should compute, for every cell in the matrix, the minimum distance from that cell to the northern border of the area (when considering the diagonal movement penalty). You should develop an iterative dynamic programming algorithm to compute

this value for every cell in the matrix, and then develop an algorithm to compute the most efficient traversal path for every cell on the southern border of the area based on these distances.

3 Project report

In your project report, you should include brief answers to 7 questions. Note that you must use dynamic programming to solve these problems; other solutions will not receive significant credit.

1. How you can break down the problem of finding the minimum distance to one of the cells on the bottom row of the matrix into one or more smaller subproblems? Your answer should include how the solution to the original problem is constructed from the subproblems and why this breakdown makes sense.
2. What recurrence can you use to model the minimum distance problem using dynamic programming?
3. What are the base cases of this recurrence?
4. Describe a pseudocode algorithm that uses memoization to compute the solution to this problem.
5. What is the time complexity of your memoized algorithm?
6. Describe an iterative algorithm for the same problem.
7. Describe an algorithm that computes the optimal route that ends at a given column on the first row of the matrix based on the dynamic programming data structure computed by your algorithm. If it is helpful, you may choose to modify your dynamic programming algorithm to make this problem easier to solve; however, you must describe any modifications that are necessary.

4 Coding your solutions

In addition to the report, you should implement an *iterative* dynamic programming algorithm that can solve the path finding problem described above. Your code may be in C++ or Java, but it must compile and run on the C4 Linux Lab machines.

For C++ users, if compiling your code cannot be accomplished by the command

```
g++ -o pathfinder *.cpp
```

you should include a Makefile that capable of compiling the code via the `make` command.

If you choose to implement your code in Java, you should submit an executable jar file with your source. In either case, your source code may be split into any number of files.

Your code will not need to handle invalid input nor problems with more than 1000 rows or columns.

4.1 Input format

Your program should read its input from the file `input.txt`, in the following format. The first row of the file will contain two positive integers, r and c, specifying the number of rows and columns

in the matrix, respectively. The following r lines of the file will contain c positive real numbers, indicating the traversal time for all of the locations in the matrix. The first of these represents the top of the matrix, where the traversals should start, and the last line represents the bottom of the matrix, where the traversals should end. The first value on each row represents the westernmost location and the last, the easternmost.

4.2 Output format

Your program should write its output to the file `output.txt`. You should write one line for each column in the matrix. Each line should should start with an integer indicating the starting column for the most efficient path. (Number columns from 0 to $c - 1$.). The starting column should be followed by $r - 1$ instances of the strings `S`, `SE`, and `SW`, representing moves to the south, southeast, and southwest, respectively, where r is the number of rows in the matrix. The first line should end on the first column of the matrix, the second on the second, and so forth.

As an example, a path that started on the third column, moved south, then southwest twice would be output as:

```
2 S SW SW
```

5 Submission

Your submission for this project will be in two parts, the group submission and your individual peer evaluations.

5.1 Group submission

The submission for your group should be a zip archive containing 1) your report (described in Section 3) as a PDF document, and 2) your code (described in Section 4). If your code requires more than a simple command to compile and run then you must also provide a Makefile and/or shell script. You should submit this zip archive to the "Project 2 (group)" assignment on Canvas.

Be aware that your project report and code will be checked for plagiarism.

5.2 Teamwork evaluation

The second part of your project grade will be determined by a peer evaluation. Your peer evaluation should be a text file that includes 1) the names of all of your teammates (including yourself), 2) the team member responsibilities, 3) whether or not your teammates were cooperative, 4) a numeric rating indicating the proportional amount of effort each of you put into the project, and 5) other issues we should be aware of when evaluating your and your teammates' relative contribution. The numeric ratings must be integers that sum to 30.

It's important that you be honest in your evaluation of your peers. In addition to letting your team members whether they do (or do not) need to work on their teamwork and communication skills, we will also evaluate your group submission in light of your team evaluations. For example, a team in which one member refused to contribute would be assessed differently than a team with three functioning members.

You should submit your peer evaluation to the "Project 2 (individual)" assignment on Canvas.

6 Grading

Report	**50 points**
Questions 1, 4, and 7	10 each
Questions 2, 3, 5, and 6	5 each
Code	**20 points**
Compiles and is correct	15
Good coding style	5
Teamwork	**30 points**

Note that if your algorithm is inefficient, you may lose points for both your pseudocode and your submission. Also, in extreme cases, the teamwork portion of your grade may become negative or greater than 30. In particular, if you do not contribute to your group's solution at all, you can expect to receive a total grade of 0.