Homework 6

Mountain Paths - Part II

In this homework, you will extend part I to compute some paths through the topographic terrain (mountains) as well as visualize these paths.

If you need a working version of Part I, make a private post on piazza with your email address. We cannot give you credit for anything you submit for Part I after we send you the code.

Background

There are many contexts in which you may want to know the most efficient way to travel over land. When traveling through mountains (let's say you're walking), perhaps you want to take the route that requires the least total change in elevation with each step you take — call it the path of least resistance. Given some topographic data, it should be possible to calculate a "greedy lowest-elevation-change walk" from one side of a map to the other.

A Greedy Walk

A "greedy" algorithm is one that, in the face of too many possible choices to choose the best from, pursues a good solution to the problem one step at a time. At each step, a greedy algorithm makes a choice that seems best at that moment, without considering that this choice may make future steps less fruitful, achieving a final solution that may be "good enough" but not the best among all possible solutions. For example, assume you want to drive from city A to city B through highways that require payment of tolls with coins, and you don't have much cash with you (much less in coins!). You would like to take the route from A to B that minimizes the overall spending on toll fees. You could look at all routes you could take, how much each road in a route charges, and spend some time finding the overall best solution. If the route involves many possibilities, this can take quite a while. The "greedy" solution is simpler: at each time you have to choose a new highway, you select the one that

charges less for the toll. It may be that you do very well with this strategy, but there is no guarantee it will work out well enough. It could be that you end up taking "cheap" roads that lead you to a very expensive area, where all following alternatives cost a lot, revealing that your greed choices led to an expensive overall solution.

For the terrain maps that we are dealing with, we can envision a "walk" as starting in some cell at the left-most edge of the map (column 0) and proceeding forward by taking a "step" into one of the 3 adjacent cells in the next column over (column 1). (Well, if you are the top or bottom row of the region, you only have 2 possibilities, but unless your region is very small, most of the time you have 3 choices.) Note that steps can take you uphill or downhill. If you want to move from one side of the region to the other (let's say, from the west to the east edge of the region), there are way too many possible paths to consider: if n steps are necessary to traverse the region, the number of possible paths is close to 3n. For the maps you handled in Homework 4, for example the one with 480 rows and 480 columns, you have 480 possible rows to start your walk and you need 479 steps to get to the other side, resulting in approximately 480 * 3479 possible paths. This is a huge number, so considering each possible path when searching for a good solution (e.g., one that minimizes uphill movement) is a bad idea.

In the figure below, each cell contains the elevation data and the green cells highlight the path resulting from the greed strategy if we start at the 3rd row in the grid:

3011	2900	2852	2808	2791	2818
2972	2937	2886	2860	2830	2748
2937	2959	2913	2864	2791	2742
2999	2888	2986	2910	2821	2754
2909	2816	2893	2997	2962	2798

The diagrams illustrate a few rules for choosing where to move to at each step. In the case of a tie with the straight-forward position (east direction), you should always choose to go straight forward.In the case of a tie between the two non-forward locations, you should always choose the southeast movement (the one on bottom).

	elev. change		elev. change		elev. change		elev. change
109	9	109	9	97	3	96	4
100 107	7	100 97	3	100 97	3	100 105	5
105	5	105	5	105	5	104	4

Case 1: smallest change is 5, go fwd-down

Case 2: smallest change is 3, go fwd

Case 3: smallest change is a tie (3), fwd is an option, so go fwd

Case 4: smallest change is a tie (4), go fwd-down

There are other ways to choose a path through the mountains while optimizing for vertical movement. Such algorithms (and their techniques and insights) are covered in other courses of a computer science degree.

Requirements

You implemented (as required by Part I) code that read the topographical data from an input file, manipulated it to identify the appropriate gray scale value to display it, and generated an output file following a simple format to represent images.

For Homework Part II, you need to compute three paths (from the west to the east side of the terrain) that follow the policy of "path of least resistance".

1.First row (row index 0)

2.Middle row (row index numRows/2 - don't forget this is integer division)

3.Last row (row index numRows -1)

The output file generated by your program will produce a visualizationsimilar to the ones depicted below. These are based on the sample input files from part I.
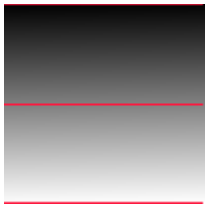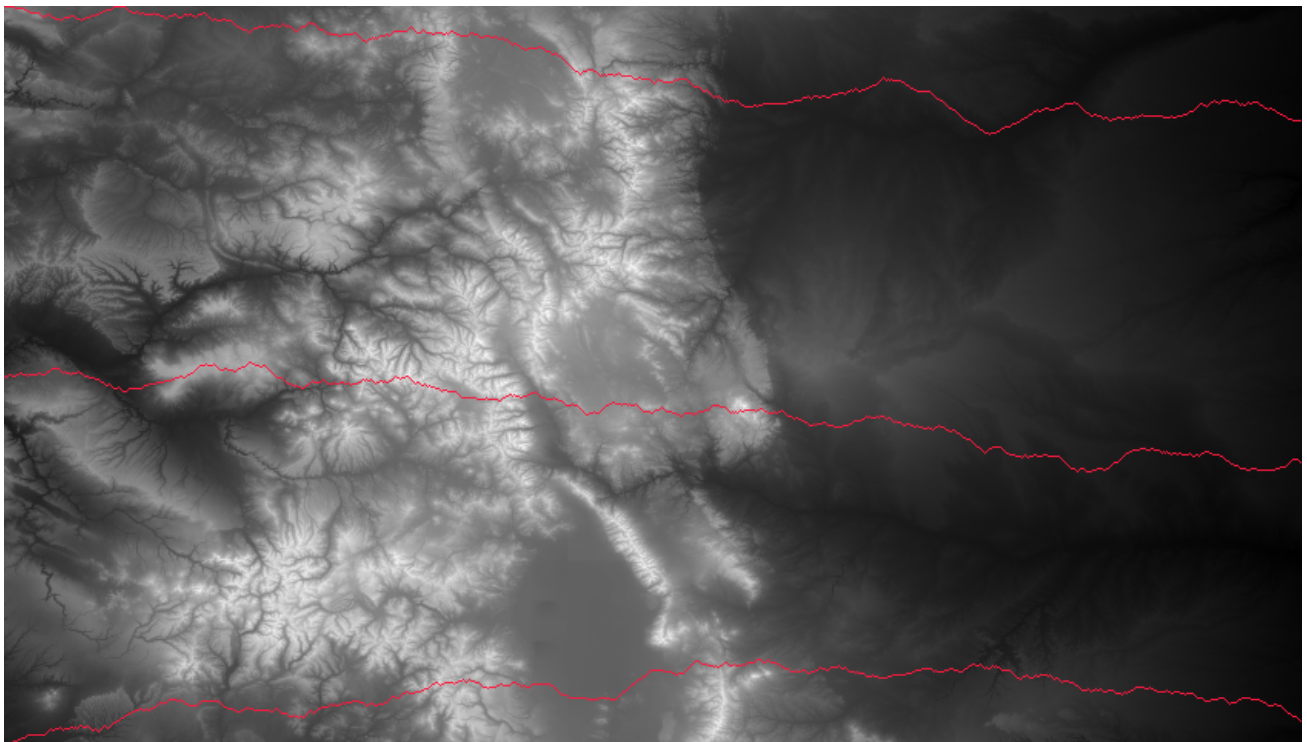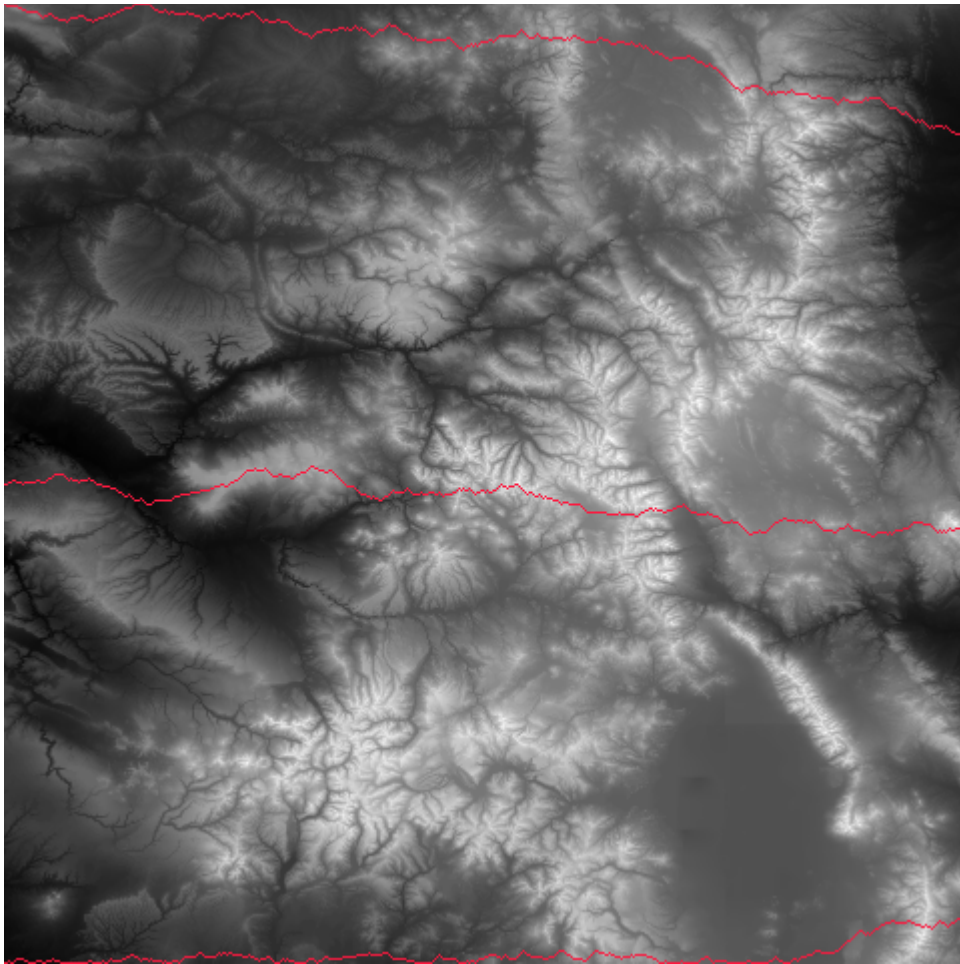
Program Flow

Your program starts by executing the first 3 steps from part I:

1.Read the data into a 2D array

2.Find min and max elevation to correspond to darkest and brightest color, respectively

3.Compute the shade of gray for each cell in the map

In part I, you completed the assignment by producing the output file (image file in the PPM format) and used a tool to look at the visualization of your output.

For homework 5, your new steps are:

4.Write a function that calculates distance of the path created for a row and colors it a particular color. The function signature/prototype is:

```
int colorPath(const vector<vector<int>>& heightMap, vector<vector<int>>& r,
```

```
vector<vector<int>>& g, vector<vector<int>>& b, int color_r, int color_g, int color_b, int
```

```
start_row)
```

5.For the first (index 0), last (index numRows-1) and middle (index numRows/w -- integer division) rows, compute the greedy path starting on the west edge of the row and color in red [RGB(252,25,63)] the corresponding cells in your output RGB data;

Then you conclude by carrying out steps you already implemented in part I:

6.Produce the output file in the specified format (PPM)

7.Use an online free tool to convert your PPM file into a JPG file

Step 4: Computing a greedy path

For a row, you color the cell you are at in red [RGB(252,25,63)], and compute your next position (following the specified movement rules) and color it, until you get to the last column. To choose the direction to move at each step, you look at the elevation data (that you read from a file) and compute the difference. Remember that you can be going downhill or uphill, therefore when you compute your vertical movement/distance, notice that it may

end up negative, and you will need to compare the absolute value of vertical differences. As you color the path, you also have to compute the total distance. This can be used in Part III to identify the best among all the greedy paths. You will need to return this distance for the function.

The function will compute the greedy path starting at the given row and color it with the provided RGB color value, returning the total vertical movement. Every "step" you take should follow the greedy choice strategy we described (including what to do when there are ties.) As you move through the path, keep a running total of the total elevation change that would be 'experienced' by a person walking this path. Since we consider an elevation change the absolute value (i.e. going 'uphill' 10 meters is the same amount of change as going 'downhill' 10 meters), this running total will be non-decreasing and may end up being a pretty large positive number. The function should return this running total, so that you can compare it with other totals.

Step 5: Call Function

Call the function for the three rows indicated below. Use the rgb values from this red [RGB(252,25,63)] to pass into the function.

1. First row (row index 0)

2. Middle row (row index numRows/2 - don't forget this is integer division)

3. Last row (row index numRows -1)

Steps 6 & 7

You are now almost done. You need to generate the PPM file (you already did this in part I) and use an online tool to visualize your output to make sure it shows as expected (as you did in part I.)