Homework 5

Mountain Paths - Part I

In this homework, you will read a set of topographic (land elevation) data into a 2D vector and manipulate the data so that it can be visualized. In the next homework, you will extend your code so that it computes some paths through the topographic terrain (mountains) as well as visualize those paths.

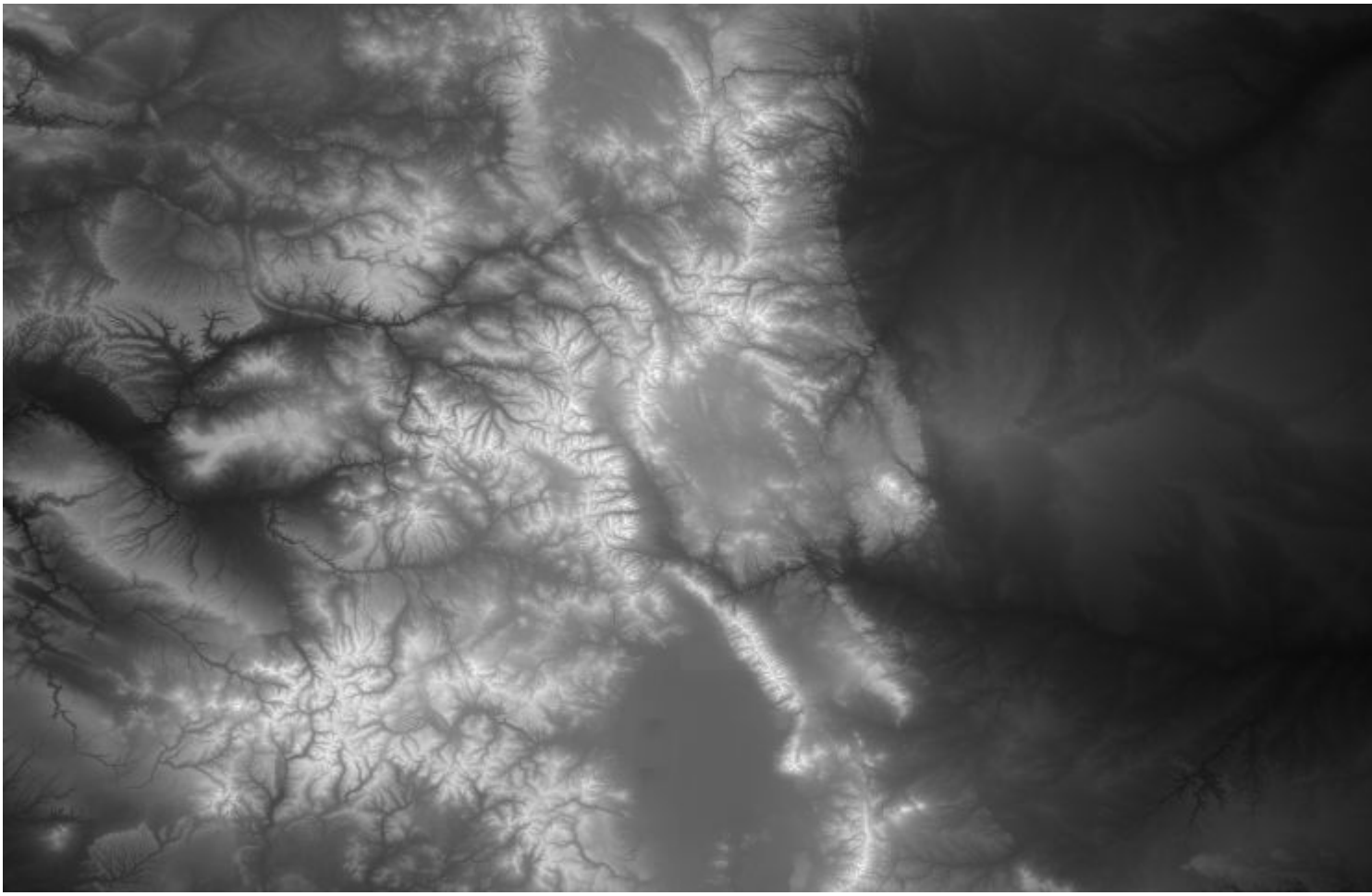
Background

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.
In part I of this homework, you will implement code to read the topographical data and manipulate it so that it can be displayed. In part II (next homework), you will implement the code to identify and displays certain paths.

Requirements

The minimum requirements for this homework are that you write code to read elevation data and produce a file representing an image that, when visualized, displays each position on the map with a color that represents the position elevation. The visualization will produce a

picture like the one below:



Program Flow

1.Read the data into a 2D vector

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
4. Produce the output file in the specified format (PPM)
5. Use an online free tool to convert your PPM file into a JPG file

Step 1 - Read the data into a 2D vector

Your first goal is to read the values into a 2D vector (i.e., a matrix) of *ints* from a data file. *Note: You should use a vector to benefit from the additional runtime checks.*

This data file contains, in plain text, data representing the average elevations of patches of land (roughly 700x700 meters) in the US. The user of your program will provide as input three values:

1. The number of rows in the map; *(The height of the image to be produced.)*
2. The number of columns in the map; *(The width of the image to be produced.)*
3. The name of the file containing the map data.

After reading these data items from the user, you should read the elevation data from the specified file. We provide several sample data files containing elevation data for most of the state of Colorado (mountains!). Other input files can be obtained from the National Oceanic and Atmospheric Administration (http://maps.ngdc.noaa.gov/viewers/wcs-client/).

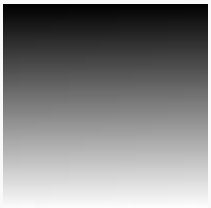
A data file comes as one large, space-separated list of integers. There are 405,120 integers representing a 480-row by 844-col grid. Each integer is the average elevation in meters of each cell in the grid. The data is listed in row-major order, i.e., first the 844 numbers for row 0, then the 844 numbers for row 1, etc.

Hints/Warnings:

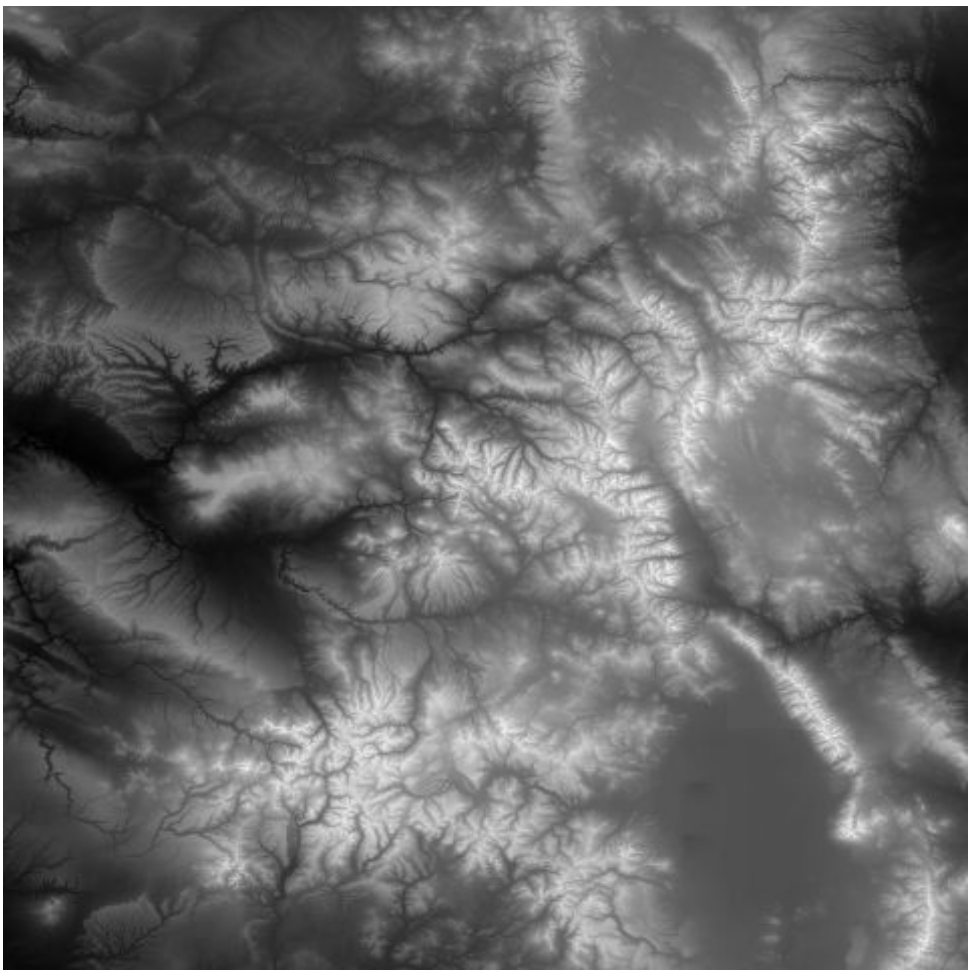
4. The data is given as a continuous stream of numbers - there are no line breaks in the file;
5. You should validate the input from the user (number of rows and columns), and you should also validate that the input file exists and that it contains all the data for the expected numbers of rows and columns. It is ok to exit the program execution with an error

message if there is an input error, i.e., you do not need to give the user a chance to enter corrected data;

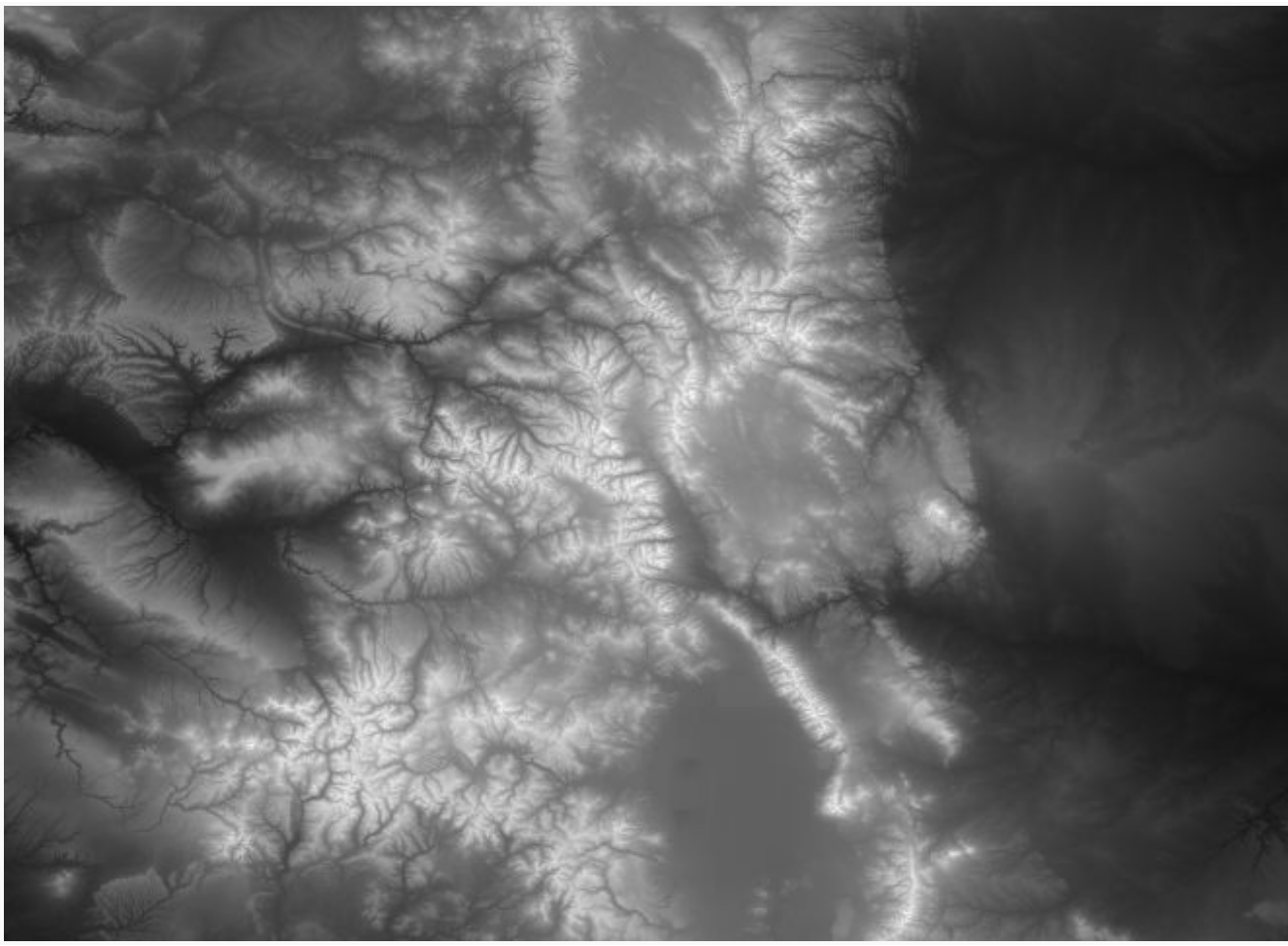
6.You may want to make sure that you are reading the data correctly before you move to the next step of this project. Implement a function to print your map data and try out your code with a file containing a small map first (you can create an input file corresponding to 4 rows and 4 columns, for example.) Then check what happens when you read the sample map data files: (100 by 100, 480 by 480, 844 by 480).

Download Link for Map Data Files	**Resulting Image**
100X100	

480X480



844X480



Step 2 - Find the min and max values

In order to draw the map with a gray scale that indicates the elevation, you will need to determine which scale to use, i.e., which will be shown as a dark area (with low elevation) and which ones will be shown as bright areas (high elevation.) This means you need to find the min and max values in the map first, so that you know which values to associate with the brightest and darkest colors. You can compute the min and max elevation values by writing code that scans the entire map data and keeps track of the smallest and largest values found so far.

Test this functionality to make sure you're getting the correct values for min and max, before you proceed to implement the rest of the program. You may want to check with a friend or colleagues in the Piazza forum to see if other students are getting the same min and max values for the provided input files.

Step 3 - Compute the color for each part of the map

The input data file contains the elevation value for each cell in the map. Now you need to compute the color (in a gray scale between white and black) to use to represent these evaluation values. The shade of gray should be scaled to the elevation of the map.

Traditionally, images are represented and displayed in electronic systems (such as TVs and computers) through the RGB color model, an additive color model in which red, green, and blue light are added together in various ways to reproduce a broad array of colors. In this model, colors are represented through three integers (R, G, and B) values between 0 and 255. For example, (0, 0, 255) represents blue and (255, 255, 0) represents yellow. In RGB color, if each of the three RGB values are the same, we get a shade of gray. Thus, there are 256 possible shades of gray from black (0,0,0) to middle gray (128,128,128), to white (255,255,255).

To make the shade of gray, you should use the min and max values in the 2D vector to scale each integer (elevation data) to a value between 0 and 255 inclusive. Check your math to ensure that you are scaling correctly. Check your code to make sure that your arithmetic operations are working as you want. Recall that if a and b are variables declared as integers, the expression a/b will be 0 if a==128 and b==256.

In the next homework you will be coloring certain paths, so you will need to keep track of R, G and B values for each location. We recommend that you create three parallel vectors for each color that correlate to the vector containing the elevation data.

Step 4 - Produce the output file in the PPM format

PPM (portable pixel map) format is a specification for representing images using the RGB color model. PPM is not used widely because it is very inefficient (for example, it does not apply any data compression to reduce the space required to represent an image.) But PPM is very simple, and there are programs available for Windows, Mac, and Linux that can be used to view ppm images. Even more conveniently, you can use an online tool with your browser

to convert a PPM file into a widely used format such as JPG. We will be using Plain PPM, which has the information in text format (readable as decimal numbers instead of binary.)

In Step 3, you computed the RGB values for the shades of gray. All you will need to do to get a PPM image for these RDB values is to write a preamble before writing the RGB numbers into a file. We will follow the convention that if the input file is named *input.dat*, then your program will generate a file named *input.dat.ppm*

A PPM file has the following format:

7.First line: string “P3”

8.Second line: width (number of columns) and height (number of rows)

9.Third line: max color value (for us, 255)

10.Rest of the file: list of RGB values for the image, expressed as a raster of rows, from top to bottom. Each row contains the RGB values (i.e., three values) for each column.

Example:

P3

4 4

15

0 0 0 0 0 0 0 0 0 15 15 15

0 0 0 7 7 7 0 0 0 0 0 0

0 0 0 0 0 0 0 15 15 0 0 0

15 15 15 0 0 0 0 0 0 0 0 0

Sample PPM File:

blocks.ppm 

Step 5 - Use an online free tool to visualize the PPM file you generated

In this step you do not write any code: you simply use a tool to convert the .ppm file into a .jpg file. The tool is available at:

https://www.coolutils.com/online/PPM-to-PNG#

The usage is very intuative: