Nkosinathi Nkosi

Parallel Programming with the Java Fork/Join framework: Terrain Classification

It is becoming common to undertake scans of the earth's surface for a variety of purposes, including agricultural and forest management, studies of climate change, and other scientific applications. This also includes so-called bare earth surveys, which map the height of the earth's surface, stripped of building and trees. In fact, most of the earth's surface has already been mapped in this way, albeit at a relatively course resolution. The datasets involved can be quite large and their analysis complex, which makes them good candidates for acceleration using parallel computing.

One such analysis is to determine how water flows across the landscape in order to protect against flooding and manage crops. As a precursor to this kind of analysis, in this assignment, you will take a scanned input terrain and find local minima (small-scale basins) where water might accumulate. Figure 1 shows an example of the type of terrain data that you will be analysing.

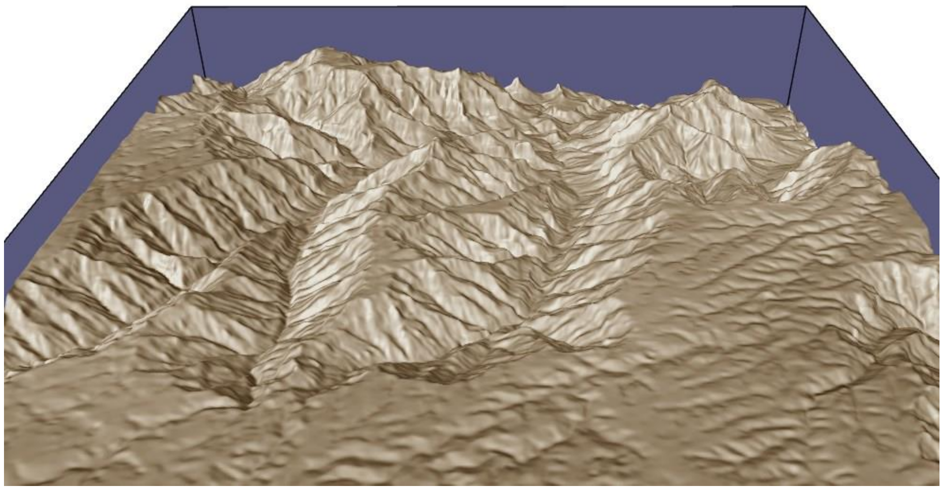


Figure 1: A rendering of a scanned terrain from the Grand Canyon area, in the United States.

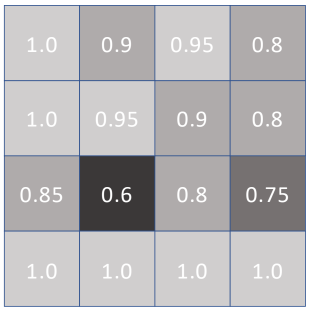
You will be provided with data for a terrain represented as a regular grid of height values in meters above sea level, with each height encoded as a single floating point value.

The file input format is as follows:
<terrain num rows – INT> <terrain num cols – INT>
<height at grid pos (0,0) - FLOAT> <height at grid pos (0,1) - FLOAT> … etc.

For example, an unrealistically small 4 x 4 terrain might be encoded in an input file as:

```
4 4
1.0   0.9   0.95 0.8
1.0   0.95 0.9   0.8
0.85 0.6   0.8   0.75
1.0   1.0   1.0   1.0
```

This corresponds in a simple visualization to the following:



1.0	0.9	0.95	0.8
1.0	0.95	0.9	0.8
0.85	0.6	0.8	0.75
1.0	1.0	1.0	1.0

This trivial example terrain has a local minimum or basin at (2, 1) or row 2, column 1 with the column and row indices beginning from 0. Your task is to categorize every grid point as either a basin or non-basin. A basin is determined for a given grid position by checking the ring of neighbouring height values. If all of the neighbours are at least 0.01m higher than the grid point then it is classified as a basin. (Using an offset of 0.01m allows for some noise in the measurement data). In this case the neighbours around index (2,1), namely 1.0, 0.95, 0.9, 0.8, 1.0, 1.0, 1.0, 0.85 are all greater than or equal to 0.6+0.01=0.61 and so it is classified as a basin. There is no need to classify points on the outer edge of the grid as they are automatically considered non-basins.

To finish, a pass over the classified grid values is used to extract a list of basins, which is then output to file. Note that this final pass can be excluded from your parallelization efforts and any performance timings.

The file output format is as follows:
<number of basins – INT>
<first basin row index – INT> <first basin column index – INT>
<second basin row index – INT> <second basin column index – INT>
…

For the toy problem above (which only has a single basin) this would mean an output of:
1
2 1