Hill Descent

Andrew Huang (amh877)
Kevin Li (kl2482)
Titas Geryba(tg1404)

April 22, 2018

1 Problem Description

Description: John is an Olympic Skier. He is currently preparing for a competition at a skiing resort and wants to maximize his efficiency of training. He gets his hands on a an altitude mapping of the local mountains represented as a grid of altitudes. Given this mapping, he wants to know what is the longest descent of negative slope that he could achieve. Since these paths are not official slopes, John would like a set of directions that would allow him to find his way along the longest path of descent. John also loves admiring the views from high altitudes, so if more than one longest descent exists, he would like to choose one that starts at the highest point possible.

Input: On the first line are specified the values $n, m(1 \leq n, m \leq 1000)$. Each of the next n lines has m integers describing the altitudes $a_{ij}(1 \leq a_{ij} \leq 10000)$ in one row of the grid.

Output: On the first line output the starting position of the descent (the point (y, x) where the top left corner of the array is $(1, 1)$, x increases rightwards and y increases downwards). On the following line output a sequence of letters that describe the direction John should take to go from the starting position to the ending position of the path. For each direction output N, E, W, S for North, East, West and South respectively. If there are multiple starting coordinates for the path of maximal length, choose the one that starts at the highest point, the input will guarantee that only one such point exists. If there are multiple paths of maximal length from the highest point found, output the directions that would yield the lowest lexicographical ordering.

Example 1:

Input:

```
2 4
1 2 3 4
1 1 1 8
```

Output:

```
(2,4)
NWWW
```

Example 2:

Input:

```
2 2
0 3
4 0
```

Output:

```
(2,1)
E
```

Example 3:

Input:

```
1 9
1
2
3
4
5
4
3
2
1
```

Output:

```
(5,1)
NNNN
```

2 Solutions

There are several ways to solve this problem, ranging to naive to slightly more difficult. For example,

- Naively perform dfs on every spot and record the furthest distance reached while storing the longest path(s) in some data structure.
  Time complexity is $O((nm)^2)$.

- Use dynamic programming to consider a cell up to i, j and whether or not taking a step in any valid direction would add to the distance. So the state would be current distance, i, j. To retrieve the path, at every location we store the neighbor that will yield the longest path. If there are neighbors that that can be taken to achieve maximal length, we store the one that will yield smaller lexicographical order. Once we know the distance of the longest path L, we retrieve the point on the grid that has a length of this size and go from stored neighbor to the following stored neighbor for L steps, each time printing out the direction taken.

  Time complexity is $O(nm)$.

- Find a topological sort and find the length of the longest sub-length. $O(mn)$

3 Anti-Solutions

Some common pitfalls are:

- WA - One may think that the length of the longest path may simply be the furthest DFS traversal from the highest point, that may not be true because it may achieve a very low altitude very quickly. For example:

1	5	4
9	1	3
1	1	2

  starting at altitude 9 the longest path is of length 2. But starting at altitude 5 we achieve a longest path of length 5.

- WA - One may make a simple coding error or a conceptual error in trying to store neighbors by lexicographical order.

- RTE - Out of bounds due to not treating neighbors on the boundary of the matrix effectively

- TLE - If you use the naive algorithm, there is no way a 1000 by 1000 grid can be dealth with.

4 Test Case Design

Ideas for test case generation.

- Three cases are generated randomly
- One case contains only ones
- One case that starts with a common point and branches off into many directions, all yielding paths of the same length.
- One case with one peak and valleys
- One case where the path is of length nm.
- Randomly generated 2D Perlin noise

A total of 8 secret cases will be generated.