Dheeraj Satya Sushant Viswanadham
(z5204820)

COMP3121 Assignment 4 – Question 2

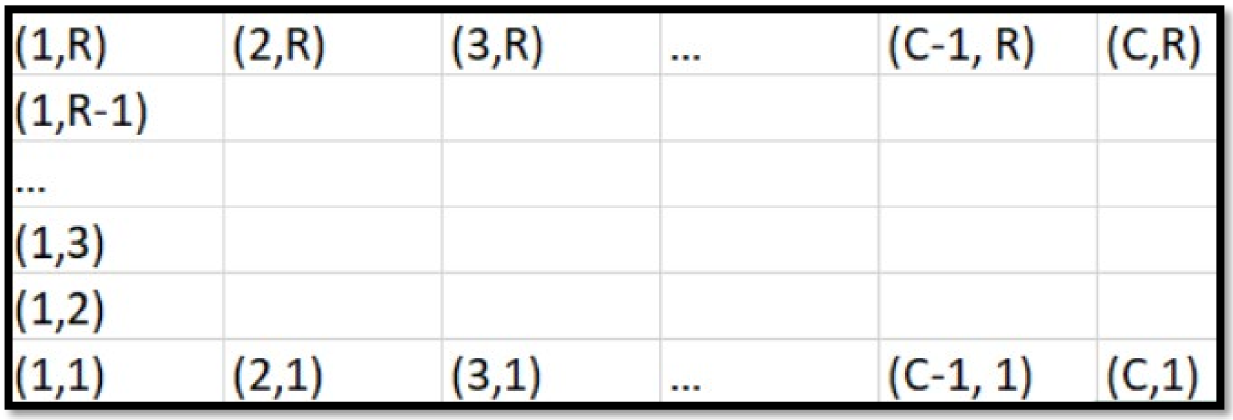
2) In this question we are trying to find a path from (1, R) which is the top-left corner of the map to (C, 1) which is the bottom-left corner of the map so that the number of moves that will result in an elevation increase is MINIMISED. Additionally, we can only move down one square or right one square. So, if we had two paths (amongst others) as below:

1) 6 -> 3 -> 5 -> 1; and

2) 6 -> 5 -> 2 -> 1.

We would choose the second path as there are 0 moves which would increase the elevation compared to the first path where we have gone from 3 -> 5 which would be 1 elevation move. Note that any path will take the same number of moves and that in this case, our 'cost' would be an elevation increase from the previous move.

See given diagram to provide a visual representation of our map (acquired from Piazza):



(1,R)	(2,R)	(3,R)	...	(C-1, R)	(C,R)
(1,R-1)					
...					
(1,3)					
(1,2)					
(1,1)	(2,1)	(3,1)	...	(C-1, 1)	(C,1)

We can solve this question through dynamic programming where we simply fill a table of the size of the map and calculate the cost incurred (+1 if we increase the elevation, otherwise 0 if there are no elevation increases).

To reach a cell (x, y), we can either go to the cell to the left of it (x – 1, y) or above it (x, y – 1). Each cell will have a number that represents the elevation of that square, and we are trying to find the least moves in a given path that will result in an elevation increase from the previous move before it. We can reconstruct this using the back-tracking technique (described in the lecture notes). For a bottom-up dynamic programming approach, we need to know our given cell position and its associated elevation. Then, we need to solve all subproblems of the form "What is the best path that has minimal moves that result in an elevation increase to reach cell (x, y)". We then need to find the lowest elevation increase cost (as explained before: +1 if elevation increases otherwise 0). The base case is that opt(1, R) (in our diagram above) = 0 (as no moves resulted in an elevation increase) and cell (x, y) = ∞ for all moves that are outside our grid. The recursion would be:

$$\text{opt}(x, y) = \text{board}(x, y) + \min\{\text{opt}(x-1, y), \text{opt}(x, y-1)\}.$$

The complexity will be $O(n^2)$ as a result.

Simply put, given the associated elevation for each cell, we will then find the least elevation increases in a path (so we are looking for a path that has the same elevation or less). If we cannot avoid an elevation increase (i.e. both the cell below it and the cell to the right has an increase in elevation), we will then split our path there and repeat the previous step: have the paths run recursively in parallel to determine the minimal moves that will result in elevation increase to reach cell (x, y).

End of Solution