COMP3121-Ass3-Q2

z5302513, Kihwan Baek

July 2021

1. Subproblem

For every $i \leq C$ and $j \geq 1$, we assume that opt(i,j) is the smallest total number of moves from lower elevation to higher elevation along the path from (1,R). And also, we assume that the elevation of square(i,j) is e(i,j).

2. Base cases

The base case is opt(1,R) = 0 because we start at squar(1,R) and there is no number of move from lower elevation to higher elevation at the first square.

3. recursion

To get opt(i,j), we need to move from opt(i-1,j) or opt(i,j+1) because w only can move immediately below or to the right. And also, if the heights of these squares (square (i-1,j) and square (i,j+1)) are greater or equal to the square(i,j), the opt value of square(i,j) will be the same as those.
So, if we divide the cases, we can say opt(i,j) is the minimum value from 1), 2), 3) and 4) which are

1) opt(i-1,j), if $e(i-1,j) \geq e(i,j)$
2) opt(i-1,j)+1, if $e(i-1,j) < e(i,j)$
3) opt(i,j+1), if $e(i,j+1) \geq e(i,j)$
4) opt(i,j+1)+1, if $e(i,j+1) < e(i,j)$

3. How to obtain the final solution

We can get the final solution as getting the last value opt(C,1) as we move to square (C,1) from the starting point which is square (1,R).

4. Time complexity

From the start point which is the base case (1,R) We need to search all possible values of i which are between 1 and R. So, it takes O(R) time. And also at the same time, for comparing elevations and getting opt value, we need to search all j values which are between C and 1. So, It takes O(C) time. Therefore, the above algorithm runs in time $O(R) \times O(C) = O(CR)$.