JIE MEI z5173405

Question 2

Since the direction of the path can only be down or right, each cell in the first column of the grid can only be reached from the left corner element, and each element in the first row of the grid can only be reached from the left corner. Because the path is unique, the shortest path corresponding to each cell is the sum of elevation on the corresponding path.

For cells that are not in the first row and first column, you can move one step to the right from the adjacent cell on the left, or move one step down from the adjacent cell above it, so the minimum value between the current cell value and the adjacent top and left is the shortest path value of the current cell plus the current cell value.

We need to create a two-dimensional array res[R][C] store the shortest path for each cell. res[i][j] represents the shortest distance from the upper left corner to (i,j).

If i = 0 and j > 0, res[0][j] = res[0][j-1] + grid[0][j]

If i > 0 and j = 0, res[i][0] = res[i-1][0] + grid[i][0]

If I > 0 and j > 0, res[i][j] = min(res[i-1][j],res[i][j-1]) + grid[i][j]

Sample code(java)

```java
class Q2 {
    public static int shortestPath(int rows,int columns,int[][] grid) {
        int[][] res = new int[rows][columns];
        res[0][0] = grid[0][0];
        for (int i = 1; i < rows; i++) {
            res[i][0] = res[i - 1][0] + grid[i][0];
```

```
        }
        for (int j = 1; j < columns; j++) {
            res[0][j] = res[0][j - 1] + grid[0][j];
        }
        for (int i = 1; i < rows; i++) {
            for (int j = 1; j < columns; j++) {
                res[i][j] = Math.min(res[i - 1][j], res[i][j - 1]) + grid[i][j];
            }
        }
        return res[rows - 1][columns - 1];
    }

    public static void main(String[] args) {
        int rows = 3;
        int columns = 3;
        int[][] grid = new int[rows][columns];
        for(int i = 0; i < 3; i++) {
            for(int j = 0; j < 3; j++) {
                grid[i][j] = 2;
            }
        }
        grid[0][1] = 1;
        grid[0][2] = 1;
        grid[1][2] = 1;
        int res = shortestPath(rows, columns, grid);
        System.out.println(res);
    }

}
```

The example grid[3][3] like that:

2	1	1
2	2	1
2	2	2

So, the shortest path should be 2-1-1-1-2.Get the result 7

```
80000gn/T/vscodesws_4e361/jdt_ws/jdt.ls-java-project/bin Q2
7
bash-3.2$
```