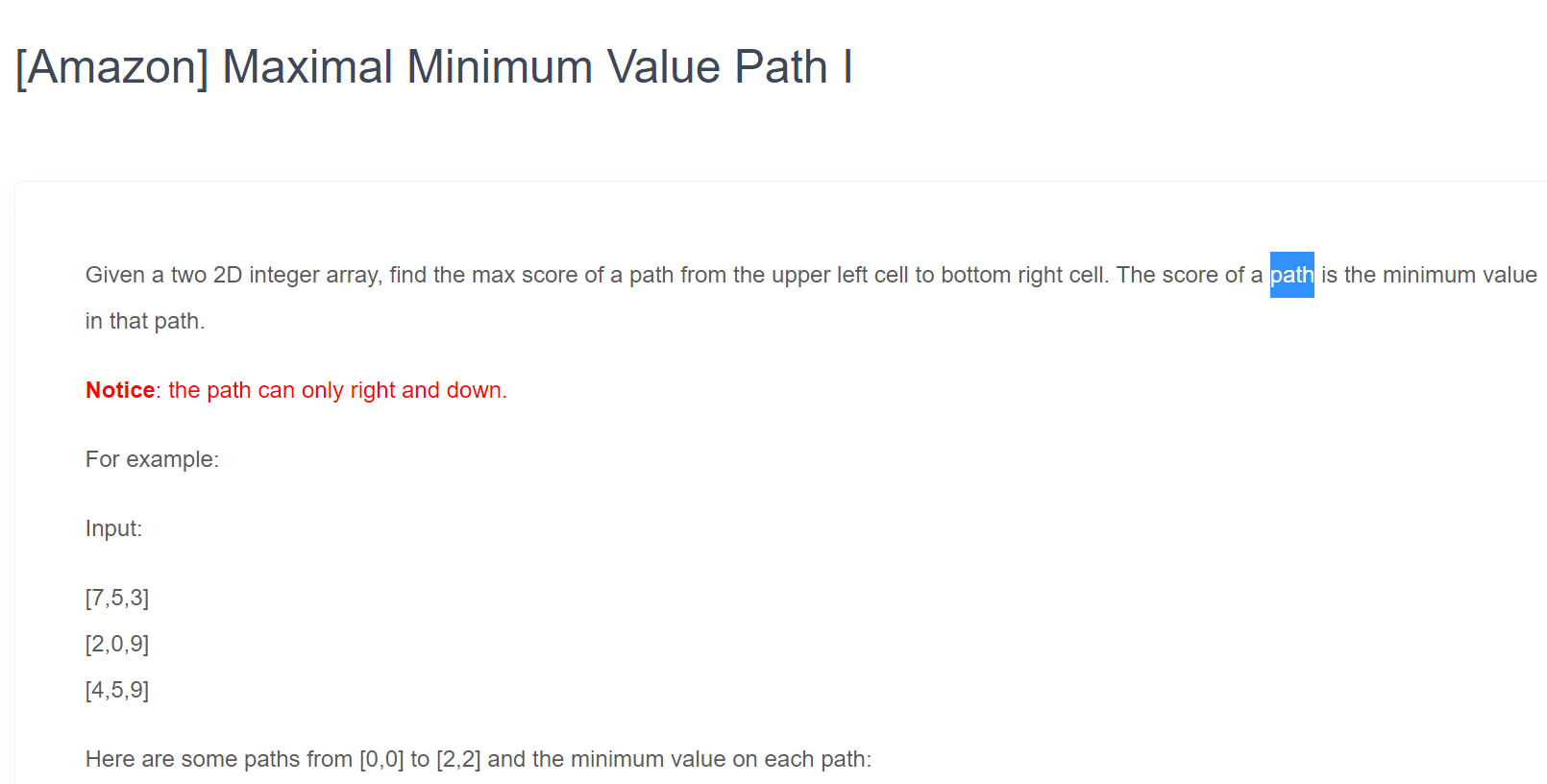
<https://leetcode.com/discuss/interview-question/383669/>

https://leetcode.com/discuss/interview-question/383669/



public class Main {

/\* DRIVER CODE \*/

public static void main(String[] args) {

System.out.println("AMAZON OA2 - DRONE FLYING");

int[][] grid = {{1,2,3},{4,5,1}};

//int[][] grid = {{5,1},{4,5}};

int result = getMaxOfMin(grid);

System.out.println("paths(not including start & end):");

for(ArrayList<Integer> list : Wrapper.paths){

System.out.println(list.toString());

}

System.out.println("mins of every path: "+Wrapper.mins.toString());

System.out.println("result: "+result);

}

/\* SOLUTION DOWN HERE \*/

public static class Wrapper{

public static ArrayList<ArrayList<Integer>> paths;

public static ArrayList<Integer> mins;

public static int max;

}

public static int getMaxOfMin(int[][] grid){

Wrapper.paths = new ArrayList<ArrayList<Integer>>();

Wrapper.mins = new ArrayList<Integer>();

Wrapper.max = 0;

ArrayList<Integer> path = new ArrayList<>();

getMins(grid, 0, 1, grid[0][1], path);

path.clear();

getMins(grid, 1, 0, grid[1][0], path);

return Wrapper.max;

}

public static void getMins(int[][] grid, int r, int c, int min, ArrayList<Integer> path){

//if end found, save values.

if(r == grid.length-1 && c == grid[0].length-1) {

Wrapper.paths.add(path);

Wrapper.mins.add(min);

if(min > Wrapper.max) Wrapper.max = min;

return;

}

//check if we have a new min for current brach path

int newMin = min;

if(grid[r][c] < min){

newMin = grid[r][c];

}

//Add current value to branch current path

path.add(grid[r][c]);

//GO RIGHT

if(c+1 < grid[0].length) {

ArrayList<Integer> newPath = new ArrayList<Integer>(path);

getMins(grid, r, c+1, newMin, newPath);

}

//GO DOWN

if(r+1 < grid.length) {

ArrayList<Integer> newPath = new ArrayList<Integer>(path);

getMins(grid, r+1, c, newMin, newPath);

}

}

}