# CS 1083

**Assignment #2**

**Author: Yulong Wang**

**Id: 3713596**

# 1. Source Code

### Cavern.java

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
import java.util.LinkedList;
import java.util.Queue;

/**
 * @author Yulong Wang
 * @date 2021/10/06
 */
public class Cavern {
    /**
     * The width of the array represents the cavern
     */
    private int width;
    /**
     * The height of the array represents the cavern
     */
    private int height;
    /**
     * The array represents the cavern
     */
    private int[][] cavern;

    /**
     * @param height The height of the cavern
     * @param width The width of the cavern
     * @param cavern The array that represents the cavern
     */
    public Cavern(int height, int width, int[][] cavern) {
//      assume the input is valid, this means height and width is always larger than
0;
        this.height = height;
        this.width = width;
        this.cavern = cavern;
    }

    /**
     * This method calculate and print the area of the cavern that is accessible.
     *
     */
    public void showCavernArea(){
        int visited = 0;
        Queue<int[]> q = new LinkedList<>();
//        find the valve
        for(int i=0;i<cavern[0].length;i++){
            if(cavern[0][i] == 0){
```

```
                int[] temp = {0,i};
                cavern[0][i] = 3;
                q.offer(temp);
                visited ++;
                break;
            }
        }
        while(!q.isEmpty()){
//          get a node from the head of the queue.
            int[] cur = q.poll();
            int row = cur[0];
            int col = cur[1];
//          add all possible path in to the end of queue
            if(row-1>0 && cavern[row-1][col]==0){
                int[] temp = {row-1, col};
                q.offer(temp);
                visited ++;
                cavern[row-1][col] = 3;
            }
            if(row+1<height && cavern[row+1][col]==0){
                int[] temp = {row+1, col};
                q.offer(temp);
                visited ++;
                cavern[row+1][col] = 3;
            }
            if(col-1>=0 && cavern[row][col-1]==0){
                int[] temp = {row, col-1};
                q.offer(temp);
                visited ++;
                cavern[row][col-1] = 3;
            }
            if(col+1<width && cavern[row][col+1]==0){
                int[] temp = {row, col+1};
                q.offer(temp);
                visited ++;
                cavern[row][col+1] = 3;
            }
        }
//      print the result
        for(int i=0;i<height;i++){
            for(int j=0;j<width;j++){
                System.out.print(cavern[i][j]+" ");
            }
            System.out.println();
        }
        System.out.println("The area of the cavern is: "+visited);
    }

    /**
     * @param args Test file path
     * @throws FileNotFoundException
     */
```

```
    public static void main(String[] args) throws FileNotFoundException {
        File file = new File(args[0]);
        Scanner sc = new Scanner(file);
        int row = sc.nextInt();
        int column = sc.nextInt();
        int[][] cavernArray = new int[row][column];
        sc.nextLine();
        int r = 0;
        while(sc.hasNextInt()){
            for(int i = 0; i<column;i++){
                cavernArray[r][i] = sc.nextInt();
            }
            r ++;
        }
        Cavern cavern = new Cavern(row, column, cavernArray);
        cavern.showCavernArea();
    }
}
```

# 2.Test

### a.TestCase1.txt

This test case test if the algorithm works at normal condition.

```
8 7
1 1 1 1 1 0 1
1 0 0 1 0 0 1
1 1 1 0 0 0 1
1 1 0 0 1 1 1
1 0 1 0 1 0 1
1 0 1 0 0 0 1
0 0 0 1 1 1 0
1 1 1 0 0 0 1
```

Output:

```
(base) yulongwang@yulongdembp src % java Cavern ../TestCase1.txt
1 1 1 1 1 3 1
1 0 0 1 3 3 1
1 1 1 3 3 3 1
1 1 3 3 1 1 1
1 0 1 3 1 3 1
1 0 1 3 3 3 1
0 0 0 1 1 1 0
1 1 1 0 0 0 1
The area of the cavern is: 13
```

### b.TestCase2.txt

This test case test if the algorithm works at boundary condition.

```
1 1
1
```

Output

```
(base) yulongwang@yulongdembp src % java Cavern ../TestCase2.txt
1
The area of the cavern is: 0
```

### c.TestCase3.txt

This test case test if the algorithm works at boundary condition.

```
1 1
0
```

Output

```
(base) yulongwang@yulongdembp src % java Cavern ../TestCase3.txt
3
The area of the cavern is: 1
```

### d.TestCase4.txt

This test case test if the algorithm can track "0"s in different directions. And whether the algorithm will go across diagonally.

```
7 7
1 1 1 1 1 0 1
1 0 0 0 0 0 1
1 0 0 1 1 1 1
1 0 1 0 1 0 1
1 0 0 1 0 0 1
1 0 0 0 0 0 1
1 1 1 1 1 1 1
```

Output

```
(base) yulongwang@yulongdembp src % java Cavern ../TestCase4.txt
1 1 1 1 1 3 1
1 3 3 3 3 3 1
1 3 3 1 1 1 1
1 3 1 0 1 3 1
1 3 3 1 3 3 1
1 3 3 3 3 3 1
1 1 1 1 1 1 1
The area of the cavern is: 19
```

### e.TestCase5.txt

This test case test if the algorithm can track "0"s at the boundaries.

```
8 7
0 1 1 1 1 1 1
0 1 1 1 1 1 0
0 0 0 0 0 0 0
0 1 1 1 1 1 0
0 1 1 1 1 1 0
0 1 1 1 1 1 0
0 1 1 1 1 1 0
0 0 0 0 0 0 0
```

Output

```
(base) yulongwang@yulongdembp src % java Cavern ../TestCase5.txt
3 1 1 1 1 1 1
3 1 1 1 1 1 3
3 3 3 3 3 3 3
3 1 1 1 1 1 3
3 1 1 1 1 1 3
3 1 1 1 1 1 3
3 1 1 1 1 1 3
3 3 3 3 3 3 3
The area of the cavern is: 25
```