SWC01. Grid Acid

locked

by rohit_lakhotia

A rectangular grid has N x M size. Each cell is either made of a special metal (type A) or made of stone (type B). If acid is poured on a cell of grid, it can either melt it(for type A: special metal cell) and spread further or do not pass through it(for type B: stone cell). The cell made with special metal (type A) can melt with acid in 1 second and thus allow acid to spread further. The cell made with stone (type B) does not react with acid and hence neither melt nor allow acid to pass through it. There is a third type of a cell that is empty (type C), but has a boundary (all 4 sides) covered with a protective metal. If all 4 sides of it (type C: empty cell) come into contact with acid at any given time, then in that case boundary of it (all 4 sides)melt and allow acid to pass through it. In that case it (type C: empty cell) get filled with acid. There is only one and only one such cell in a given grid. Acid is poured on one of the cell of grid, the cell being made of special metal that can melt with acid. It is guaranteed that acid will be poured on only one cell made up of special metal (type A), not on the stone (type B) or the empty cell (type C). Acid is poured continuously until all the grid cells (except stone - type B) melt completely. You have to tell when the empty cell with special protective boundary will get filled with acid and when whole grid will get filled with acid (except the cells made up of stones). It takes 1sec for acid to dissolve special metal cell and after that it can spread to its 4 orthogonal neighbors (Left, Right, Up, Down).

Input Format

First entry is number of test cases; rest is each test caseinput.

For each test case first row contains N and M as twointegers separated by a space.

Next row contains the location of cell (row and columnnumber separated by space) where acid will be poured continuously until wholegrid (except stone cells) melt.

Next N rows contain M integer values each containing thecell type. Cell type value is of 3 types: - 0 : cell is stone (type B) - 1: cell is made special metal (type A) - 2: cell is special empty cell, having a special boundary (type C)

Note: There is always one and only one cell of typeC (value 2) in a given grid.

Constraints

1. 1<=Number of test cases <= 10

2. The maximum number of rows or columns of grid is 3000

Output Format

Output should contain 2 lines for each test case.

1st line: Case#

2nd line: Count1 Count2

Where:

Count1: time in seconds when special empty cell (type C) will get filled.

Count2: time in seconds when whole grid will get filled with water (note: stone cell can not be filled with acid).

Note:

1. Count2 will be -1if all cells of the grid (except stone cells) cannot be dissolved. Whole grid dissolving term means that all cells except stone get filled with Acid.
2. Count1 will be -1if empty cell cannot be filled. If empty cell cannot be filled the #1 is also applicable, i.e. then Count2 = -1.
3. Once acid enters special cell, it accumulates there for 1 second. After that the acid starts leaking to neighboring (left, right, up, down) cells.
4. The terms “dissolve”, “melt”, “leaking” are used to express similar meaning that the cell starts leaking the acid to its neighbor cells (left, right, up, down).
5. Acid is poured continuously so once a cell starts leaking acid, it may spread further to other cells in further course of time.

Sample Input 0

```
9
4 5
2 4
1 0 1 0 1
1 0 1 1 1
1 1 2 1 1
1 0 1 0 1
3 3
1 2
1 1 0
1 2 1
0 1 1
3 3
1 1
1 1 1
1 2 1
0 1 1
3 3
3 3
1 1 1
1 2 1
0 1 1
4 4
2 3
0 0 1 0
0 1 1 1
1 1 2 1
1 0 1 1
3 3
1 3
0 1 1
1 2 1
1 1 1
4 5
1 3
1 0 1 0 1
1 0 1 1 1
1 1 1 2 1
1 0 0 1 1
4 5
3 5
1 0 1 0 1
1 0 1 1 1
1 1 1 2 1
1 0 0 1 1
4 5
2 4
1 0 1 0 1
1 1 1 1 1
1 1 2 1 1
1 0 1 1 1
```

Sample Output 0

```
Case #1
-1 -1
Case #2
-1 -1
Case #3
6 6
Case #4
```

```
6 6
Case #5
5 5
Case #6
6 6
Case #7
7 7
Case #8
5 9
Case #9
4 6
```

Submissions: 10
Max Score: 100
Difficulty: Medium

Rate This Challenge:
☆☆☆☆☆

More

Current Buffer (saved locally, editable)

C++14

```cpp
#include <cmath>
#include <cstdio>
#include <vector>
#include <iostream>
#include <algorithm>
using namespace std;

int a[3002][3002];
int v[3002][3002];
int p[3][10000000];


void solve(int n,int m,int x,int y,int& total,int& sp ,int c){
    int start=0,end=1;
    p[0][0]=x;
    p[1][0]=y;
    p[2][0]=1;
    int row[4]={-1,0,1,0};
    int col[4]={0,1,0,-1};
    int k=0;
    int c1=1;
    v[x][y]=1;

    while(start!=end)
    {
        int e=p[0][start],f=p[1][start],g=p[2][start];
        for(int i=0;i<4;i++){
            if(e+row[i]>=1 && e+row[i]<=n && f+col[i]>=1 && f+col[i]<=m){
                if(v[e+row[i]][f+col[i]]==0 && a[e+row[i]][f+col[i]]==1){
                    p[0][end]=e+row[i];
                    p[1][end]=f+col[i];
                    p[2][end]=g+1;
                    v[e+row[i]][f+col[i]]=1;
                    end++;
                    c1++;
                }
                else if(v[e+row[i]][f+col[i]]==0 && a[e+row[i]][f+col[i]]==2){
                    k++;
                    if(k==4){
                        sp=g;
                        c1++;
                    }
                }

                if(c==c1){
                    if(g!=sp)
```

```
                        total=g+1;
                    else
                        total=g;
                    return ;
                }
            }
        }
        start++;
    }
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    int t;
    cin>>t;
    int d=1;
    while(t--){
        int n,m;
        cin>>n>>m;
        int e,f;
        cin>>e>>f;
        int x,y;
        int c=0;
        for(int i=1;i<=n;i++){
            for(int j=1;j<=m;j++){
                cin>>a[i][j];
                if(a[i][j]==2){
                    x=i;y=j;
                }
                if(a[i][j]==1 || a[i][j]==2)
                    c++;
            }
        }

        for(int i=1;i<=n;i++){
            for(int j=1;j<=m;j++)
                v[i][j]=0;
        }

        int total=-1,sp=-1;
        if(x==1 || x==n || y==0 || y==m){
            cout<<"Case #"<<(d++)<<endl;
            cout<<-1<<" "<<-1<<endl;

        }
        else if(a[x-1][y]==0 || a[x][y+1]==0 || a[x+1][y]==0 || a[x][y-1]==0){
             cout<<"Case #"<<(d++)<<endl;
            cout<<-1<<" "<<-1<<endl;
        }
        else{
            solve(n,m,e,f,total,sp,c);
            cout<<"Case #"<<(d++)<<endl;
            cout<<sp<<" "<<total<<endl;
        }
    }
    return 0;
}
```

Line: 1 Col: 1

Upload Code as File   Test against custom input   Run Code   Submit Code