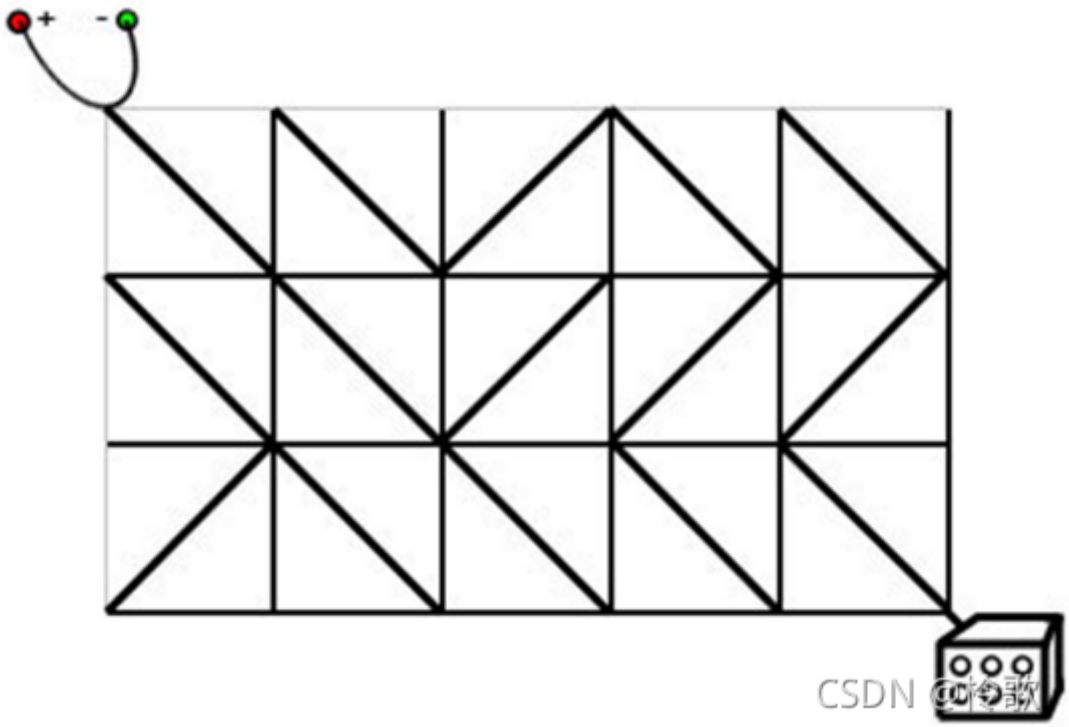
搜索-双端队列广搜

一、AcWing 175. 电路维修

【题目描述】

电路板的整体结构是一个R行C列的网格$(R, C \leq 500)$，如下图所示。



每个格点都是电线的接点，每个格子都包含一个电子元件。

电子元件的主要部分是一个可旋转的、连接一条对角线上的两个接点的短电缆。

在旋转之后，它就可以连接另一条对角线的两个接点。

电路板左上角的接点接入直流电源，右下角的接点接入飞行车的发动装置。

达达发现因为某些元件的方向不小心发生了改变，电路板可能处于断路的状态。

她准备通过计算，旋转最少数量的元件，使电源与发动装置通过若干条短缆相连。

注意：只能走斜向的线段，水平和竖直线段不能走。

【输入格式】

输入文件包含多组测试数据。

第一行包含一个整数T，表示测试数据的数目。

对于每组测试数据，第一行包含正整数R和C，表示电路板的行数和列数。

之后R行，每行C个字符，字符是`/`和`\`中的一个，表示标准件的方向。

【输出格式】

对于每组测试数据，在单独的一行输出一个正整数，表示所需的最小旋转次数。

如果无论怎样都不能使得电源和发动机之间连通，输出`NO SOLUTION`。

【数据范围】

$1 \leq R, C \leq 500$

$1 \leq T \leq 5$

【输入样例】

```
1
3 5
\\/\\
\\///
/\\\\
```

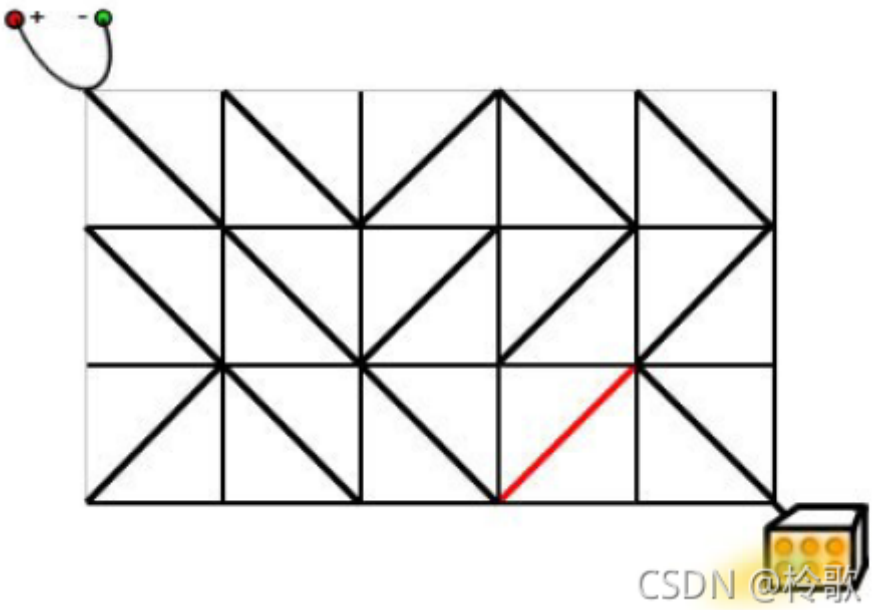
【输出样例】

```
1
```

【样例解释】

样例的输入对应于题目描述中的情况。

只需要按照下面的方式旋转标准件，就可以使得电源和发动机之间连通。



【分析】

首先把电路板上每一个格子点（交叉点）看作无向图中的节点，我们认为两个节点x和y分别是某个小方格的左上角和右下角，那么如果该小方格的线段为`\`，那么我们可以认为边权为**0**,反之该小方格线段是`/`，那么我们的边权视为**1**，说明要旋转一次才能够连上。

现在我们得到了一张完美的边权**0**或**1**的无向图，那么和普通广搜一样，我们唯一的改变就是，如果说当前新状态的边权为**0**，那么我们就放到**队头**先走，因为我们要满足两段性和单调性，而为了这个单调性，如果说当前新状态边权为**1**，那么我们就只能压入到**队尾**。

【代码】

```
#include <iostream>
#include <algorithm>
#include <cstring>
#include <deque>
using namespace std;

typedef pair<int, int> PII;
const int N = 510;
char g[N][N];
int dis[N][N];
bool st[N][N];
int n, m;
int dx[4] = { -1, -1, 1, 1 }, dy[4] = { -1, 1, 1, -1 };//分别往左上、右上、右下、左下四个点走

int ix[4] = { -1, -1, 0, 0 }, iy[4] = { -1, 0, 0, -1 };//对应的方格坐标
```

```cpp
char e[5] = "\\/\\/";//对应的方格中的线路

int bfs()
{
    memset(st, 0, sizeof st);
    memset(dis, 0x3f, sizeof dis);
    deque<PII> Q;
    Q.push_back({ 0, 0 });
    dis[0][0] = 0;
    while (Q.size())
    {
        auto t = Q.front();
        Q.pop_front();
        int x = t.first, y = t.second;
        if (x == n && y == m) return dis[n][m];
        if (st[x][y]) continue;
        st[x][y] = true;
        for (int i = 0; i < 4; i++)
        {
            int nx = x + dx[i], ny = y + dy[i];
            int w = (g[x + ix[i]][y + iy[i]] != e[i]);//如果已知路径与所需路径不同那么花费为1
            if (nx >= 0 && nx <= n && ny >= 0 && ny <= m && dis[x][y] + w < dis[nx][ny])
            {
                dis[nx][ny] = dis[x][y] + w;
                if (w) Q.push_back({ nx, ny });//如果需要修改线路那么放到队尾
                else Q.push_front({ nx, ny });//否则放到队头
            }
        }
    }
    return 20030925;
}

int main()
{
    int T;
    cin >> T;
    while (T--)
    {
        cin >> n >> m;

        for (int i = 0; i < n; i++)
```

```
55              for (int j = 0; j < m; j++)
56                  cin >> g[i][j];
57          if (n + m & 1) puts("NO SOLUTION");//由于起点为偶点，因此终点如果为奇点必定无解
58          else cout << bfs() << endl;
59      }
60      return 0;
61  }
```