{% note info %} **摘要** Title: 2060. 奶牛选美 Tag: BFS、DFS、双端队列广搜 Memory Limit: 64 MB Time Limit: 1000 ms {% endnote %}

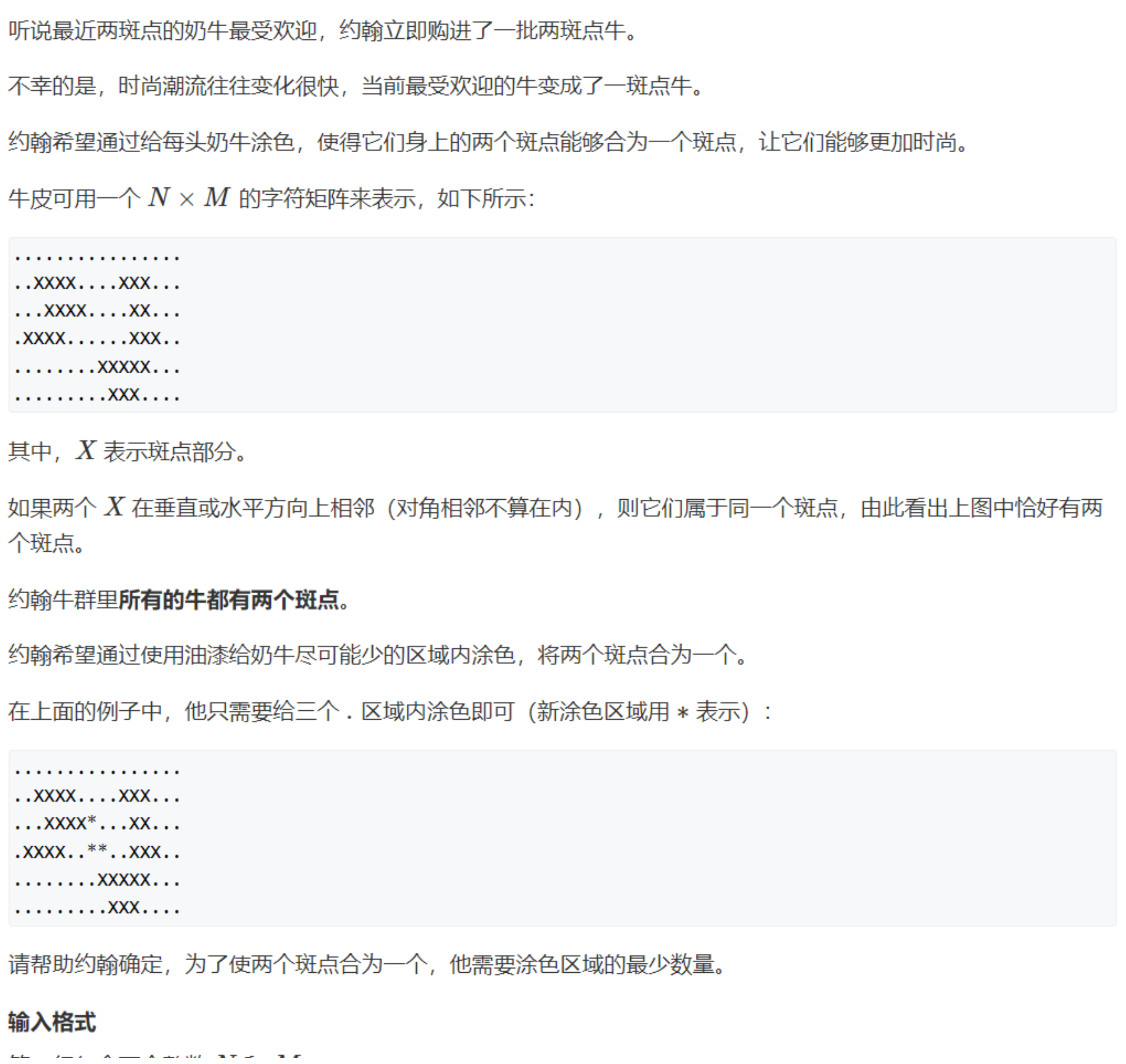
Powered by:**NEFU AB-IN**

Link

@TOC

2060. 奶牛选美

- 题意



听说最近两斑点的奶牛最受欢迎，约翰立即购进了一批两斑点牛。

不幸的是，时尚潮流往往变化很快，当前最受欢迎的牛变成了一斑点牛。

约翰希望通过给每头奶牛涂色，使得它们身上的两个斑点能够合为一个斑点，让它们能够更加时尚。

牛皮可用一个 $N \times M$ 的字符矩阵来表示，如下所示：

```
................
..XXXX....XXX...
...XXXX....XX...
.XXXX......XXX..
........XXXXX...
.........XXX....
```

其中，X 表示斑点部分。

如果两个 X 在垂直或水平方向上相邻（对角相邻不算在内），则它们属于同一个斑点，由此看出上图中恰好有两个斑点。

约翰牛群里**所有的牛都有两个斑点。**

约翰希望通过使用油漆给奶牛尽可能少的区域内涂色，将两个斑点合为一个。

在上面的例子中，他只需要给三个 . 区域内涂色即可（新涂色区域用 * 表示）：

```
................
..XXXX....XXX...
...XXXX*...XX...
.XXXX..**..XXX..
........XXXXX...
.........XXX....
```

请帮助约翰确定，为了使两个斑点合为一个，他需要涂色区域的最少数量。

输入格式

- 思路
  - **双端队列广搜** 随意挑选出一个X点作为起点，点为`.`时点权为1，点为`X`时点权为0，当遍历到**某个X且距离不为0**时结束，此时为最短 因为此时肯定是到了与起点不一样的连通块，而且是求最短路时最先到的，所以是最短距离 复杂度为线性

- **Floyd fill** 一共有两个连通块，那么先用Floyd fill
  - BFS
  - DFS 将两个连通块的所有点全部标记出来 最后算距离时，枚举两个集合的所有点，求两个点的**曼哈顿距离-1**，即是答案，复杂度$O(n^2)$

> **对于距离最近的两个点，最近的距离为曼哈顿距离**

- 代码
  - **双端队列广搜** 1452ms

```python
'''
Author: NEFU AB-IN
Date: 2022-01-29 18:26:56
FilePath: \ACM\Acwing\2060.2.py
LastEditTime: 2022-01-29 19:12:51
'''

from collections import deque

N = 55
INF = int(2e9)
g = []
dist = [[INF for _ in range(N)] for _ in range(N)]
st = [[0 for _ in range(N)] for _ in range(N)]

dx = [-1, 0, 1, 0]
dy = [0, 1, 0, -1]


def bfs(sx, sy):
    global n, m
    q = deque()
    q.append([sx, sy])
    dist[sx][sy] = 0

    while len(q):
        t = q.pop()
        if st[t[0]][t[1]]:
            continue
        st[t[0]][t[1]] = 1

        if g[t[0]][t[1]] == 'X' and dist[t[0]][t[1]] > 0:
            return dist[t[0]][t[1]]

        for i in range(4):
            x = t[0] + dx[i]
            y = t[1] + dy[i]
            if x >= 0 and x < n and y >= 0 and y < m:
                w = 0
                if g[x][y] == '.':
                    w = 1
                if dist[x][y] > dist[t[0]][t[1]] + w:
                    dist[x][y] = dist[t[0]][t[1]] + w
                    if w == 0:
                        q.append([x, y])
                    else:
                        q.appendleft([x, y])


if __name__ == "__main__":
    n, m = map(int, input().split())
    for i in range(n):
        s = input()
        g.append(list(s))
    for i in range(n):
        for j in range(m):
            if g[i][j] == 'X':
                print(bfs(i, j))
                exit(0)
```

- **BFS** 1502ms

  代码长，不容易爆栈，可以求最短路

```
'''
Author: NEFU AB-IN
Date: 2022-01-17 22:29:48
FilePath: \ACM\Acwing\2060.1.py
LastEditTime: 2022-01-17 23:07:00
'''

#BFS

from collections import deque


class Point(object):
    def __init__(self, x, y):
        self.x = x
        self.y = y


n, m = map(int, input().split())

g = []
vis = [[0 for _ in range(m + 1)] for _ in range(n + 1)]
point = [[] for _ in range(2)]
dx = [-1, 0, 1, 0]
dy = [0, 1, 0, -1]
# 上右下左
q = deque()
```

```
def bfs(x, y, p):
    vis[x][y] = 0
    p.append(Point(x, y))
    q.appendleft(Point(x, y)) #左进
    while q.__len__():
        top = q.pop() #右出
        x = top.x
        y = top.y
        for i in range(4):
            xx = x + dx[i]
            yy = y + dy[i]
            if xx >= 0 and yy >= 0 and xx < n and yy < m and vis[xx]
[yy] == 1:
                q.appendleft(Point(xx, yy))
                p.append(Point(xx, yy))
                vis[xx][yy] = 0


if __name__ == "__main__":
    for i in range(n):
        s = input()
        g.append(list(s))
        for j in range(len(s)):
            vis[i][j] = 1 if g[i][j] == 'X' else 0
    k = 0
    for i in range(n):
        for j in range(m):
            if vis[i][j] == 1:
                bfs(i, j, point[k])
                k += 1
    res = int(2e9)
    for i in point[0]:
        for j in point[1]:
            res = min(res, abs(i.x - j.x) + abs(i.y - j.y) - 1)
    print(res)

# 2,6
# 4,8
```

- **DFS** 1152ms

  代码短，容易爆栈，无法求最短路 由于python自己设置了递归层数，所以需要手动修改！！

```
'''
Author: NEFU AB-IN
Date: 2022-01-17 20:05:02
FilePath: \ACM\Acwing\2060.py
LastEditTime: 2022-01-17 22:31:33
'''
# DFS

import sys
```

```
# python设置了默认迭代次数，如果不用以下导入的话，最大迭代次数1e3级别，dfs无
法正常运行
sys.setrecursionlimit(2000000)


class Point(object):
    def __init__(self, x, y):
        self.x = x
        self.y = y


n, m = map(int, input().split())

g = []
vis = [[0 for _ in range(m + 1)] for _ in range(n + 1)]
point = [[] for _ in range(2)]
dx = [-1, 0, 1, 0]
dy = [0, 1, 0, -1]
# 上右下左


def dfs(x, y, p):
    vis[x][y] = 0
    p.append(Point(x, y))
    for i in range(4):
        xx = x + dx[i]
        yy = y + dy[i]
        if xx >= 0 and yy >= 0 and xx < n and yy < m and vis[xx][yy] ==
1:
            dfs(xx, yy, p)


if __name__ == "__main__":
    for i in range(n):
        s = input()
        g.append(list(s))
        for j in range(len(s)):
            vis[i][j] = 1 if g[i][j] == 'X' else 0
    k = 0
    for i in range(n):
        for j in range(m):
            if vis[i][j] == 1:
                dfs(i, j, point[k])
                k += 1
    res = int(2e9)
    for i in point[0]:
        for j in point[1]:
            res = min(res, abs(i.x - j.x) + abs(i.y - j.y) - 1)
    print(res)
```