{% note info %} **摘要** Title: 2019. 拖拉机 Tag: 双端队列广搜、最短路 Memory Limit: 64 MB Time Limit: 1000 ms {% endnote %}

Powered by:**NEFU AB-IN**

Link

@TOC

2019. 拖拉机

- 题意

> 干了一整天的活，农夫约翰完全忘记了他把拖拉机落在田地中央了。 他的奶牛非常调皮，决定对约翰来场恶作剧。 她们在田地的不同地方放了 N 捆干草，这样一来，约翰想要开走拖拉机就必须先移除一些干草捆。 拖拉机的位置以及 N 捆干草的位置都是二维平面上的整数坐标点。 拖拉机的初始位置上没有干草捆。 当约翰驾驶拖拉机时，他只能沿平行于坐标轴的方向（北，南，东和西）移动拖拉机，并且拖拉机必须每次移动整数距离。 例如，驾驶拖拉机先向北移动 2 单位长度，然后向东移动 3 单位长度。 拖拉机无法移动到干草捆占据的位置。 请帮助约翰确定他需要移除的干草捆的最小数量，以便他能够将拖拉机开到二维平面的原点。

- 思路

问从起点到终点需要最少移开多少障碍物，其实也就等价于最少经过多少障碍物，那么就可以转化为**图论的最短路问题**，两者等价性成立（原问题的每个方案与起点到终点的路径相互对应）

  - 点：格子
  - 边：上下左右四个方向
  - 边权：此题为**点权**（所以采取**矩阵**的方式存图）
    - **如果为障碍物，权值为1**
    - **如果为空地，权值为0**

问题转化为了**边权只有0或1的最短路问题**

  - **边权只有0或1的最短路问题：双端队列广搜**
    - 如果边权为**0**的话，将此点加到**队头**
    - 如果边权为**1**的话，将此点加到**队尾**
    - 拓展时每个点，**出队只出一次**，**入队可入多次**
      - 为什么？因为需要入队多次，从而判断出此点最小的权值
    - 本质为**简化版的dijkstra算法（单源最短路，一个起点到所有点的最短路）**
      - 因边权只有0和1，把**堆**换成了**双端队列**
      - 性质：**在任何时刻，在双端队列的所有距离都是升序**，所以第一个值一定是最小值，可以起到堆的作用
      - 时间复杂度线性
  - **边权只有1的最短路问题：广搜**

此题可以搜0-1001的点，即原图扩大一圈(1-1000 -> 0-1001)

- 代码

```
'''
Author: NEFU AB-IN
Date: 2022-01-28 13:00:59
FilePath: \ACM\Acwing\2019.py
LastEditTime: 2022-01-29 13:34:54
'''

from collections import deque

N = 1010
INF = int(2e9)
g = [[0] * N for _ in range(N)]
dist = [[INF] * N for _ in range(N)]
st = [[0] * N for _ in range(N)]

dx = [-1, 0, 1, 0]
dy = [0, 1, 0, -1]
#上右下左


def bfs(sx, sy):
    q = deque()
    q.append([sx, sy])
    dist[sx][sy] = 0 # dist数组代表此点到原点（1，1）的最短距离为多少

    while q:
        t = q.popleft() # 取出队头元素，因为边的权值为0的在队头
        if st[t[0]][t[1]]:  #出队去重
            continue
        st[t[0]][t[1]] = 1

        if t[0] == 0 and t[1] == 0:  #起点已经走完了
            break

        for i in range(4): #遍历每个点连的边
            x = t[0] + dx[i]
            y = t[1] + dy[i]
            if x >= 0 and x < N and y >= 0 and y < N:
                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.appendleft([x, y])  #加到队头
                    else:
                        q.append([x, y])  #加到队尾
    return dist[1][1]
```

```
if __name__ == "__main__":
    n, sx, sy = map(int, input().split())
    for _ in range(n):
        x, y = map(int, input().split())
        # 障碍为1
        g[x][y] = 1
    print(bfs(sx, sy))
```