# Assignment #C: 五味杂陈

Updated 1148 GMT+8 Dec 10, 2024

2024 fall, Complied by 李振硕 、院系：信息管理系

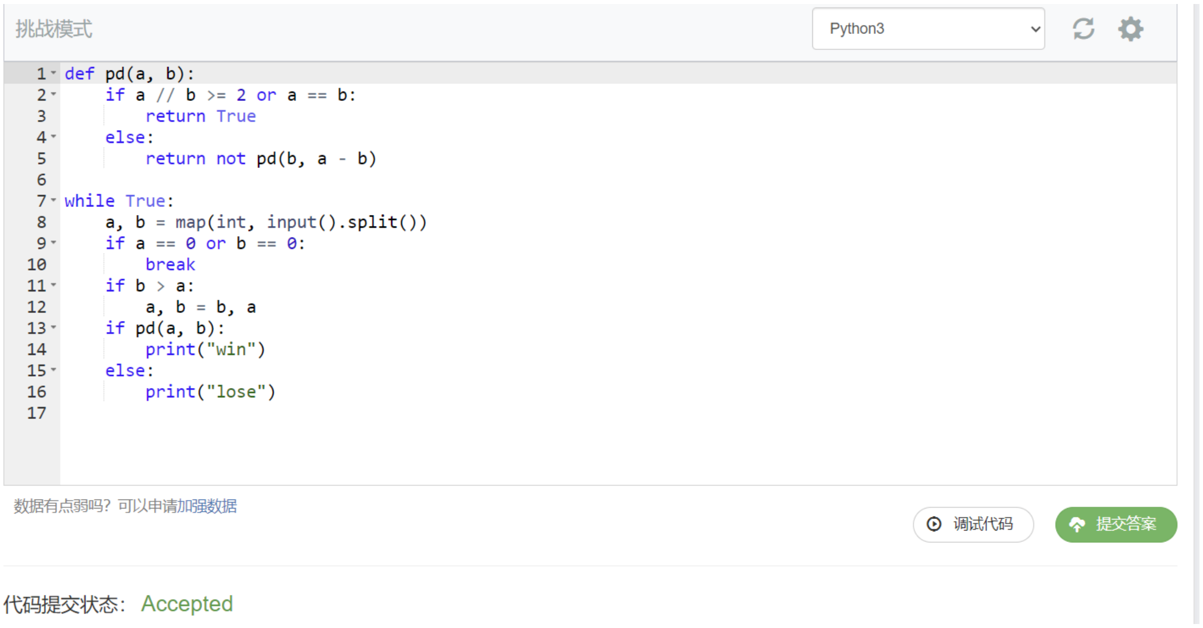
## 1. 题目

### 1115. 取石子游戏

dfs, https://www.acwing.com/problem/content/description/1117/

思路：

代码：



```python
def pd(a, b):
    if a // b >= 2 or a == b:
        return True
    else:
        return not pd(b, a - b)

while True:
    a, b = map(int, input().split())
    if a == 0 or b == 0:
        break
    if b > a:
        a, b = b, a
    if pd(a, b):
        print("win")
    else:
        print("lose")

```

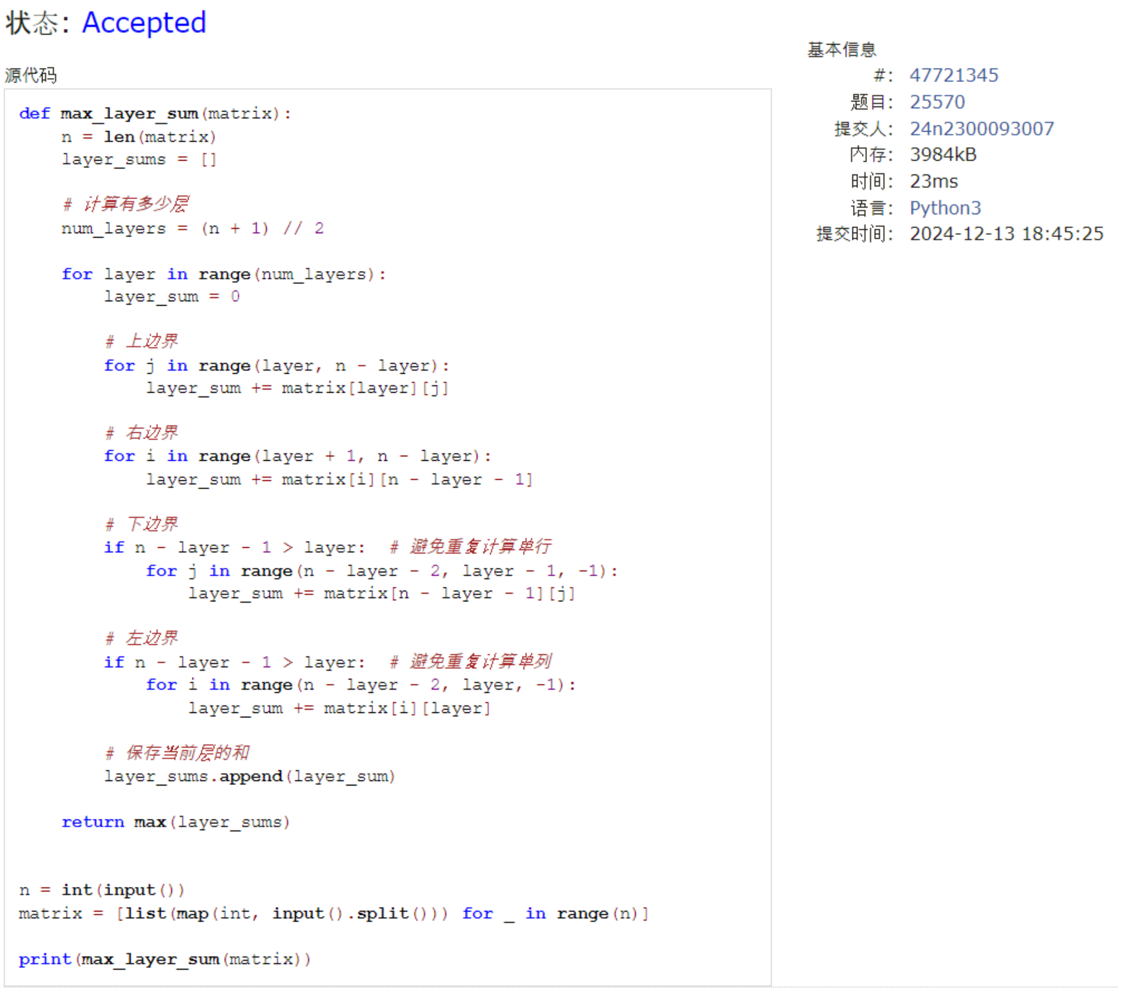
代码提交状态： Accepted

### 25570: 洋葱

Matrices, http://cs101.openjudge.cn/practice/25570

思路：

代码：


状态: Accepted

源代码

```python
def max_layer_sum(matrix):
    n = len(matrix)
    layer_sums = []

    # 计算有多少层
    num_layers = (n + 1) // 2

    for layer in range(num_layers):
        layer_sum = 0

        # 上边界
        for j in range(layer, n - layer):
            layer_sum += matrix[layer][j]

        # 右边界
        for i in range(layer + 1, n - layer):
            layer_sum += matrix[i][n - layer - 1]

        # 下边界
        if n - layer - 1 > layer:  # 避免重复计算单行
            for j in range(n - layer - 2, layer - 1, -1):
                layer_sum += matrix[n - layer - 1][j]

        # 左边界
        if n - layer - 1 > layer:  # 避免重复计算单列
            for i in range(n - layer - 2, layer, -1):
                layer_sum += matrix[i][layer]

        # 保存当前层的和
        layer_sums.append(layer_sum)

    return max(layer_sums)


n = int(input())
matrix = [list(map(int, input().split())) for _ in range(n)]

print(max_layer_sum(matrix))
```

基本信息

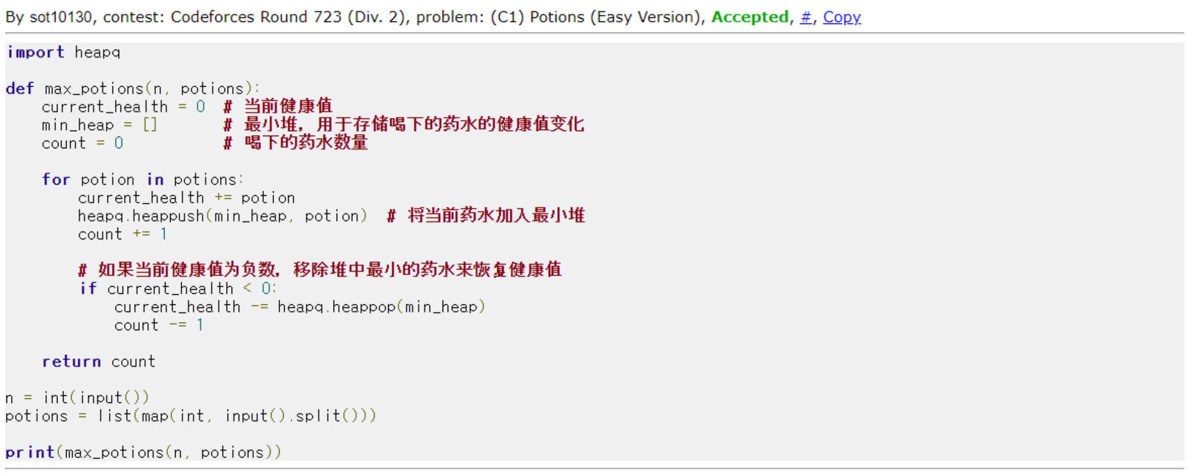
#:	47721345
题目:	25570
提交人:	24n2300093007
内存:	3984kB
时间:	23ms
语言:	Python3
提交时间:	2024-12-13 18:45:25

### 1526C1. Potions(Easy Version)

greedy, dp, data structures, brute force, *1500, https://codeforces.com/problemset/problem/1526/C1

思路：

代码：



By sot10130, contest: Codeforces Round 723 (Div. 2), problem: (C1) Potions (Easy Version), **Accepted**, #, Copy

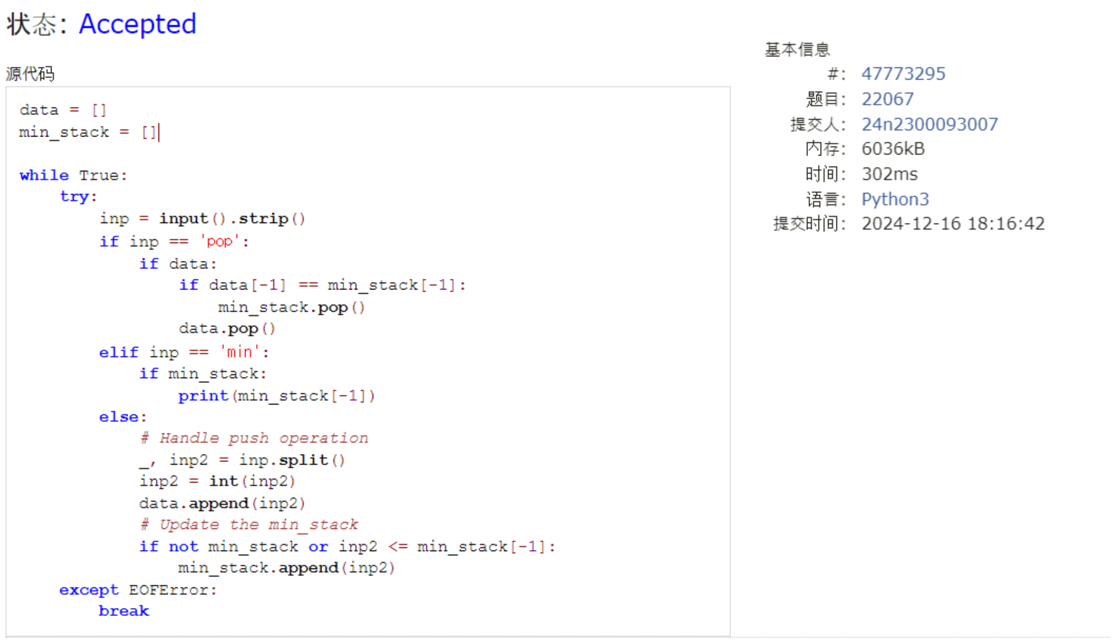
```python
import heapq

def max_potions(n, potions):
    current_health = 0  # 当前健康值
    min_heap = []       # 最小堆，用于存储喝下的药水的健康值变化
    count = 0           # 喝下的药水数量

    for potion in potions:
        current_health += potion
        heapq.heappush(min_heap, potion)  # 将当前药水加入最小堆
        count += 1

        # 如果当前健康值为负数，移除堆中最小的药水来恢复健康值
        if current_health < 0:
            current_health -= heapq.heappop(min_heap)
            count -= 1

    return count

n = int(input())
potions = list(map(int, input().split()))

print(max_potions(n, potions))
```

### 22067: 快速堆猪

辅助栈，http://cs101.openjudge.cn/practice/22067/

思路：

代码：


状态: Accepted

源代码

```python
data = []
min_stack = []

while True:
    try:
        inp = input().strip()
        if inp == 'pop':
            if data:
                if data[-1] == min_stack[-1]:
                    min_stack.pop()
                data.pop()
        elif inp == 'min':
            if min_stack:
                print(min_stack[-1])
        else:
            # Handle push operation
            _, inp2 = inp.split()
            inp2 = int(inp2)
            data.append(inp2)
            # Update the min_stack
            if not min_stack or inp2 <= min_stack[-1]:
                min_stack.append(inp2)
    except EOFError:
        break
```

基本信息

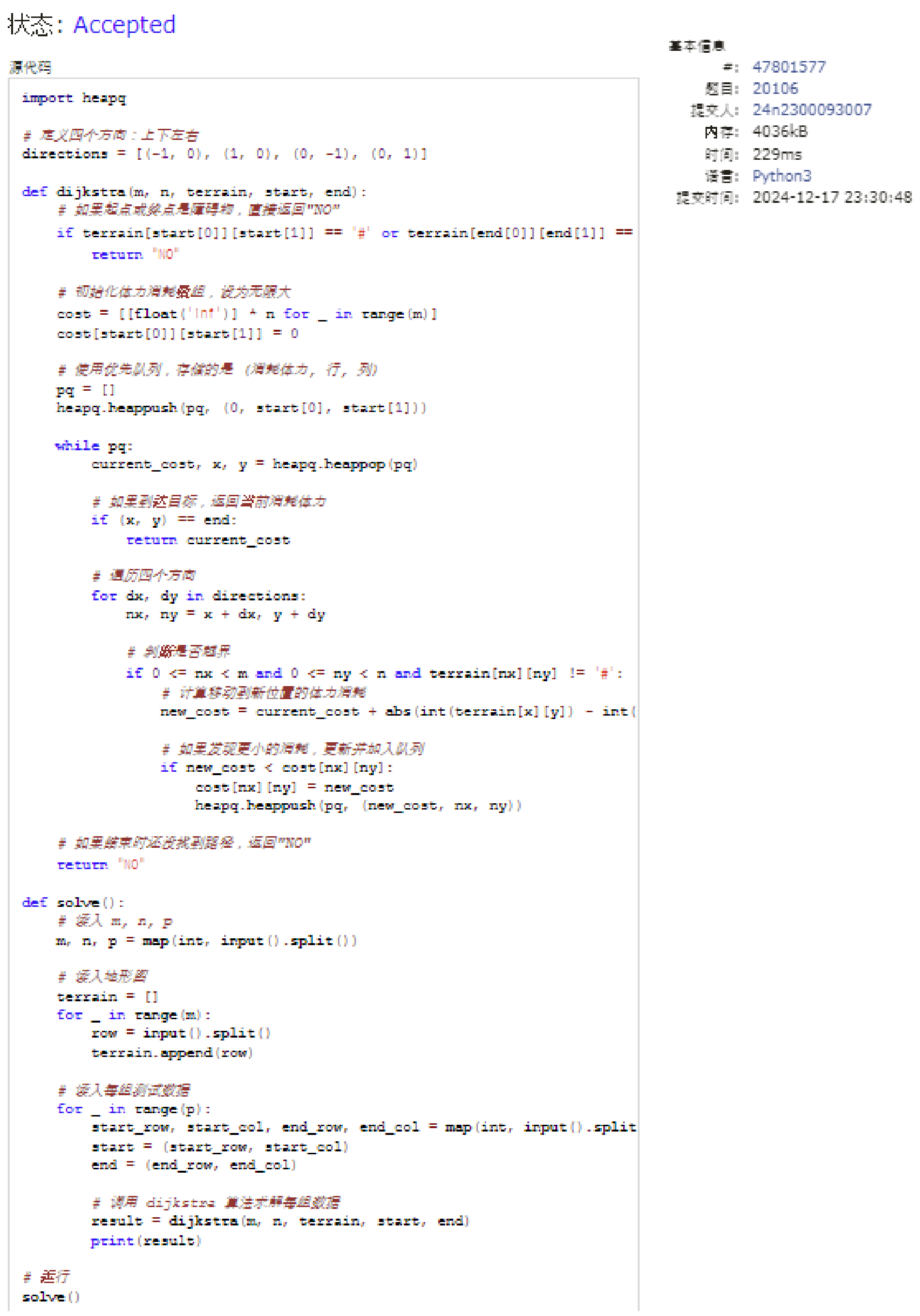
#:	47773295
题目:	22067
提交人:	24n2300093007
内存:	6036kB
时间:	302ms
语言:	Python3
提交时间:	2024-12-16 18:16:42

### 20106: 走山路

Dijkstra, http://cs101.openjudge.cn/practice/20106/

思路：

代码：


状态: Accepted

源代码

```python
import heapq

# 定义四个方向：上下左右
directions = [(-1, 0), (1, 0), (0, -1), (0, 1)]

def dijkstra(m, n, terrain, start, end):
    # 如果起点或终点是障碍物，直接返回"NO"
    if terrain[start[0]][start[1]] == '#' or terrain[end[0]][end[1]] ==
        return "NO"

    # 初始化体力消耗数组，设为无限大
    cost = [[float('inf')] * n for _ in range(m)]
    cost[start[0]][start[1]] = 0

    # 使用优先队列，存储的是 (消耗体力, 行, 列)
    pq = []
    heapq.heappush(pq, (0, start[0], start[1]))

    while pq:
        current_cost, x, y = heapq.heappop(pq)

        # 如果到达目标，返回当前消耗体力
        if (x, y) == end:
            return current_cost

        # 遍历四个方向
        for dx, dy in directions:
            nx, ny = x + dx, y + dy

            # 判断是否越界
            if 0 <= nx < m and 0 <= ny < n and terrain[nx][ny] != '#':
                # 计算移动到新位置的体力消耗
                new_cost = current_cost + abs(int(terrain[x][y]) - int(

                # 如果发现更小的消耗，更新并加入队列
                if new_cost < cost[nx][ny]:
                    cost[nx][ny] = new_cost
                    heapq.heappush(pq, (new_cost, nx, ny))

    # 如果结束时还没找到路径，返回"NO"
    return "NO"

def solve():
    # 读入 m, n, p
    m, n, p = map(int, input().split())

    # 读入地形图
    terrain = []
    for _ in range(m):
        row = input().split()
        terrain.append(row)

    # 读入每组测试数据
    for _ in range(p):
        start_row, start_col, end_row, end_col = map(int, input().split
        start = (start_row, start_col)
        end = (end_row, end_col)

        # 调用 dijkstra 算法求解每组数据
        result = dijkstra(m, n, terrain, start, end)
        print(result)

# 运行
solve()
```

基本信息

#:	47801577
题目:	20106
提交人:	24n2300093007
内存:	4036kB
时间:	229ms
语言:	Python3
提交时间:	2024-12-17 23:30:48

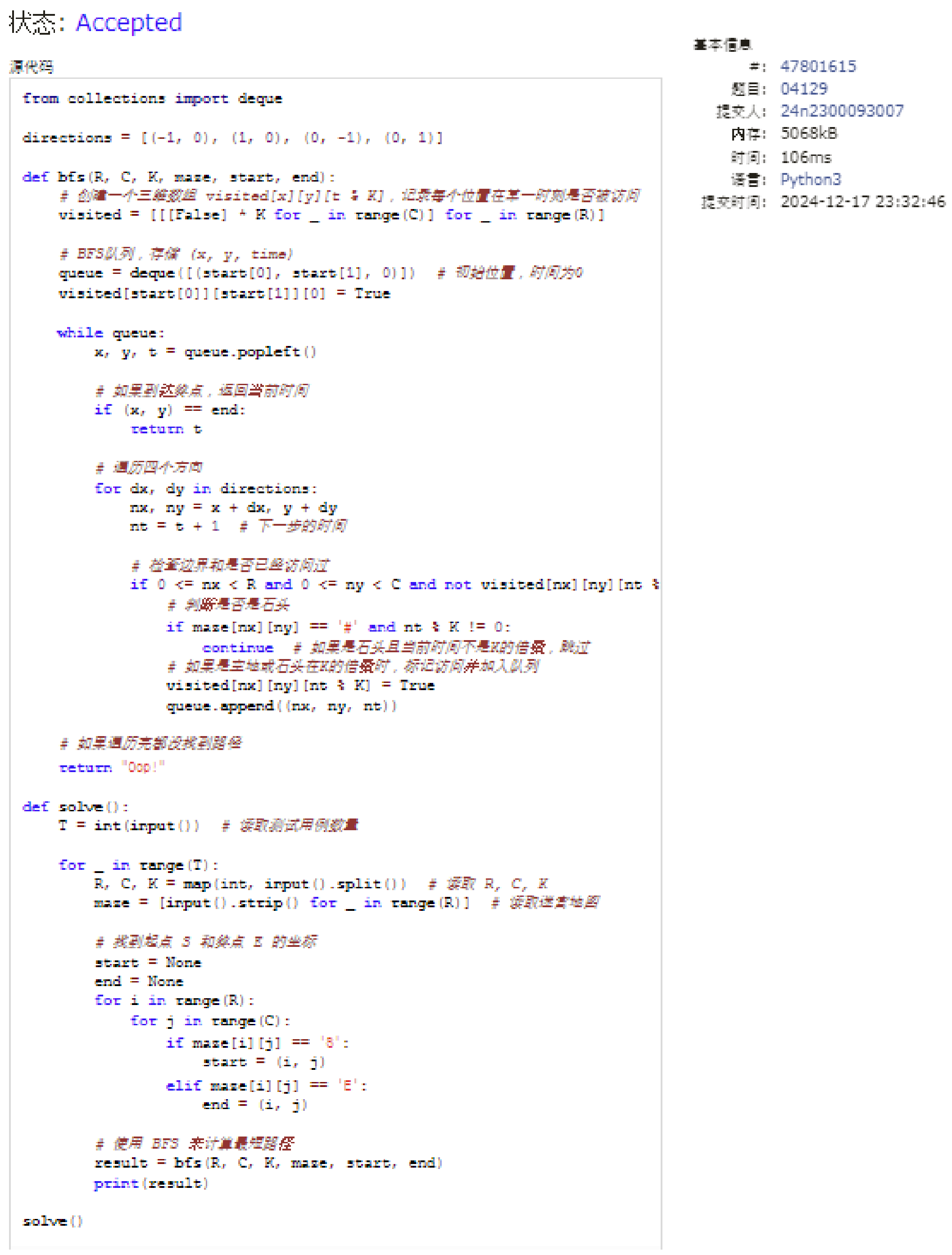
代码运行截图 <mark>（至少包含有"Accepted"）</mark>

### 04129: 变换的迷宫

bfs, http://cs101.openjudge.cn/practice/04129/

思路：

代码：



状态: Accepted

源代码

```python
from collections import deque

directions = [(-1, 0), (1, 0), (0, -1), (0, 1)]

def bfs(R, C, K, maze, start, end):
    # 创建一个三维数组 visited[x][y][t % K]，记录每个位置在某一时刻是否被访问
    visited = [[[False] * K for _ in range(C)] for _ in range(R)]

    # BFS队列，存储 (x, y, time)
    queue = deque([(start[0], start[1], 0)])  # 初始位置，时间为0
    visited[start[0]][start[1]][0] = True

    while queue:
        x, y, t = queue.popleft()

        # 如果到达终点，返回当前时间
        if (x, y) == end:
            return t

        # 遍历四个方向
        for dx, dy in directions:
            nx, ny = x + dx, y + dy
            nt = t + 1  # 下一步的时间

            # 检查边界和是否已经访问过
            if 0 <= nx < R and 0 <= ny < C and not visited[nx][ny][nt %
                # 判断是否是石头
                if maze[nx][ny] == '#' and nt % K != 0:
                    continue  # 如果是石头且当前时间不是K的倍数，跳过
                # 如果是空地或石头在K的倍数时，标记访问并加入队列
                visited[nx][ny][nt % K] = True
                queue.append((nx, ny, nt))

    # 如果遍历完都没找到路径
    return "Oop!"

def solve():
    T = int(input())  # 读取测试用例数量

    for _ in range(T):
        R, C, K = map(int, input().split())  # 读取 R, C, K
        maze = [input().strip() for _ in range(R)]  # 读取迷宫地图

        # 找到起点 S 和终点 E 的坐标
        start = None
        end = None
        for i in range(R):
            for j in range(C):
                if maze[i][j] == 'S':
                    start = (i, j)
                elif maze[i][j] == 'E':
                    end = (i, j)

        # 使用 BFS 来计算最短路径
        result = bfs(R, C, K, maze, start, end)
        print(result)

solve()
```

基本信息

#:	47801615
题目:	04129
提交人:	24n2300093007
内存:	5068kB
时间:	106ms
语言:	Python3
提交时间:	2024-12-17 23:32:46

## 2. 学习总结和收获

最后两道题完全不会，这次作业感觉最难，马上要考机考，所以应该好好复习。。