# Assignment #A: 图论：算法，树算及栈

Updated 2018 GMT+8 Apr 21, 2024

2024 spring, Complied by 同学的姓名、院系

**说明：**

1）请把每个题目解题思路（可选），源码Python, 或者C++（已经在Codeforces/Openjudge上AC），截图（包含Accepted），填写到下面作业模版中（推荐使用 typora https://typoraio.cn ，或者用word）。AC 或者没有AC，都请标上每个题目大致花费时间。

2）提交时候先提交pdf文件，再把md或者doc文件上传到右侧“作业评论”。Canvas需要有同学清晰头像、提交文件有pdf、"作业评论"区有上传的md或者doc附件。

3）如果不能在截止前提交作业，请写明原因。

**编程环境**

（请改为同学的操作系统、编程环境等）

操作系统：macOS Ventura 13.4.1 (c)

Python编程环境：Spyder IDE 5.2.2, PyCharm 2023.1.4 (Professional Edition)

C/C++编程环境：Mac terminal vi (version 9.0.1424), g++/gcc (Apple clang version 14.0.3, clang-1403.0.22.14.1)

## 1. 题目

### 20743: 整人的提词本

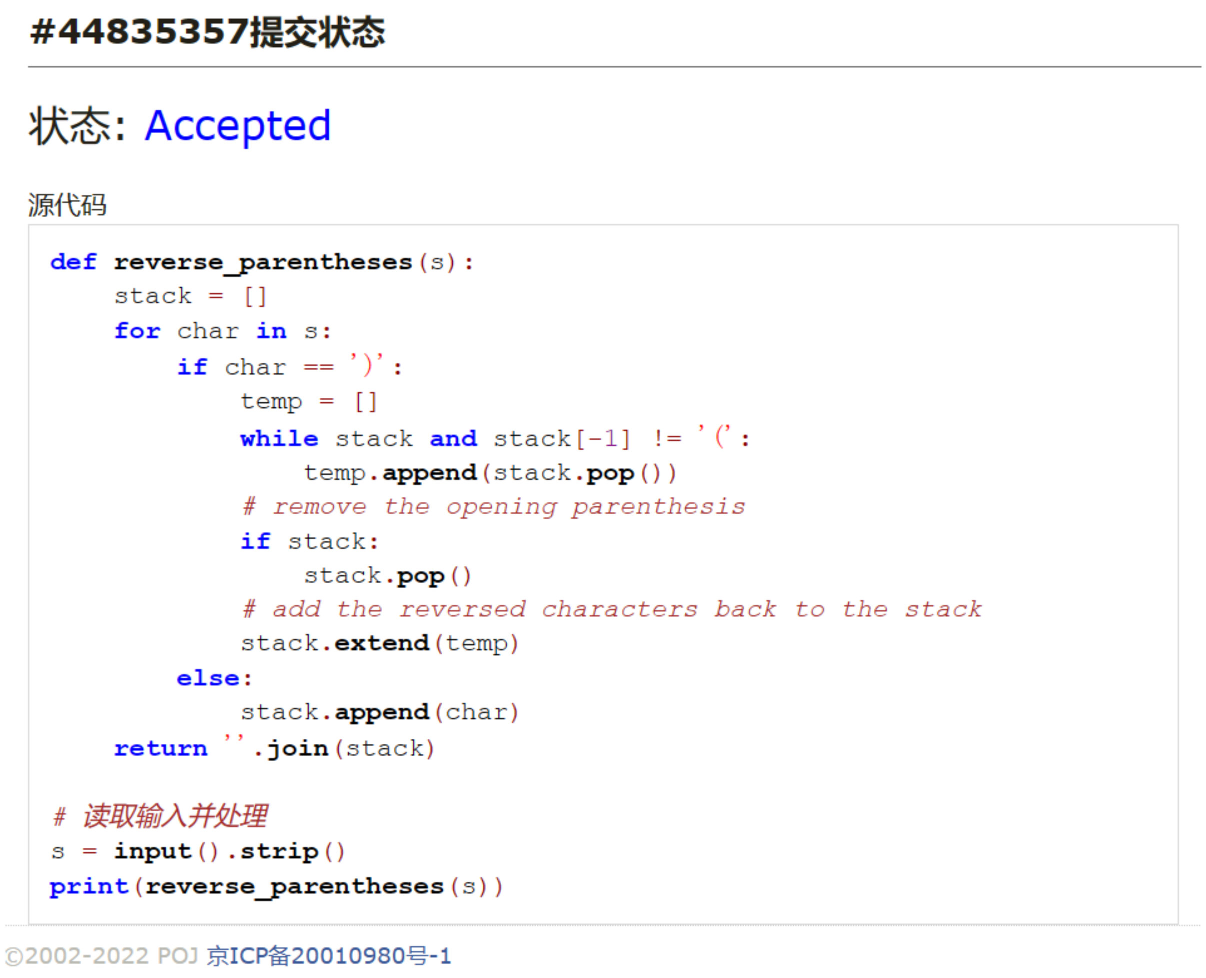
http://cs101.openjudge.cn/practice/20743/

思路：栈的运用 较为简单

代码

```python
#
# 提词本  栈的合理运用
# 先用一个函数找到最里层的括号  输入字符串  返回一个修改后的字符串
# 再用一个函数写一个栈进行反向排列

def reverse_se(s):
    stack = []
    for char in s:
        if char == ')':
            temp = []
            while stack and stack[-1] != '(':
                temp.append(stack.pop())
            if stack:
                stack.pop()
            stack.extend(temp)
        else:
            stack.append(char)
    return ''.join(stack)


s = input().strip()
print(reverse_se(s))
```

代码运行截图 (至少包含有"Accepted")



#44835357提交状态

状态: Accepted

源代码

```
def reverse_parentheses(s):
    stack = []
    for char in s:
        if char == ')':
            temp = []
            while stack and stack[-1] != '(':
                temp.append(stack.pop())
            # remove the opening parenthesis
            if stack:
                stack.pop()
            # add the reversed characters back to the stack
            stack.extend(temp)
        else:
            stack.append(char)
    return ''.join(stack)

# 读取输入并处理
s = input().strip()
print(reverse_parentheses(s))
```

©2002-2022 POJ 京ICP备20010980号-1

## 02255: 重建二叉树

http://cs101.openjudge.cn/practice/02255/

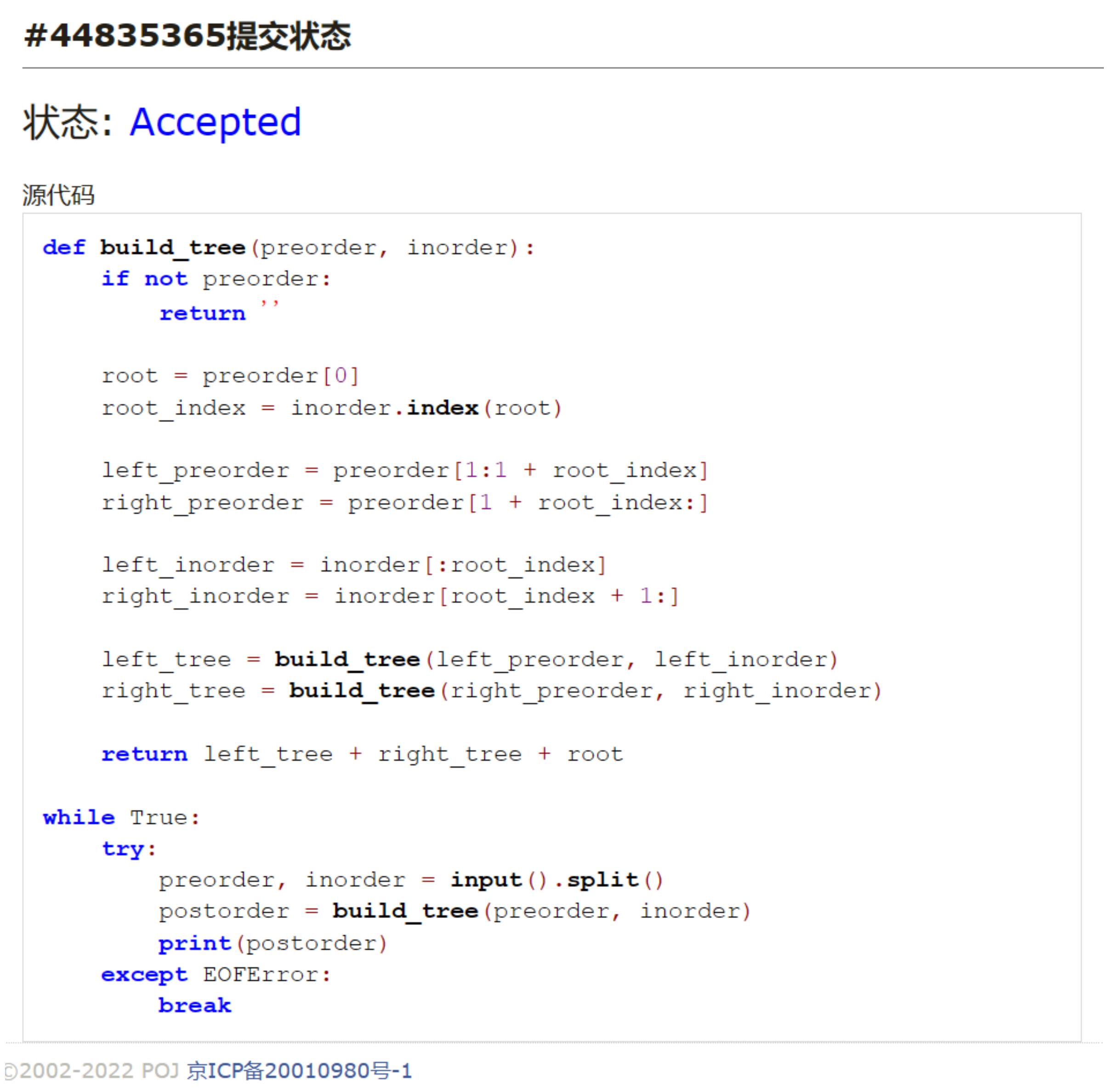
思路：之前做过了 根据前序和中序建树 遍历后续

代码

```
#
def build_tree(preorder, inorder):
```

```python
    if not preorder:
        return ''

    root = preorder[0]
    root_index = inorder.index(root)

    left_preorder = preorder[1:1 + root_index]
    right_preorder = preorder[1 + root_index:]

    left_inorder = inorder[:root_index]
    right_inorder = inorder[root_index + 1:]

    left_tree = build_tree(left_preorder, left_inorder)
    right_tree = build_tree(right_preorder, right_inorder)

    return left_tree + right_tree + root

while True:
    try:
        preorder, inorder = input().split()
        postorder = build_tree(preorder, inorder)
        print(postorder)
    except EOFError:
        break
```

代码运行截图 （至少包含有"Accepted"）



#44835365提交状态

状态: Accepted

源代码

```
def build_tree(preorder, inorder):
    if not preorder:
        return ''

    root = preorder[0]
    root_index = inorder.index(root)

    left_preorder = preorder[1:1 + root_index]
    right_preorder = preorder[1 + root_index:]

    left_inorder = inorder[:root_index]
    right_inorder = inorder[root_index + 1:]

    left_tree = build_tree(left_preorder, left_inorder)
    right_tree = build_tree(right_preorder, right_inorder)

    return left_tree + right_tree + root

while True:
    try:
        preorder, inorder = input().split()
        postorder = build_tree(preorder, inorder)
        print(postorder)
    except EOFError:
        break
```

©2002-2022 POJ 京ICP备20010980号-1

## 01426: Find The Multiple

http://cs101.openjudge.cn/practice/01426/

要求用bfs实现

思路：典型的bfs算法 属于是第一次接触后逐渐熟悉了起来

代码

```
# 
from collections import deque

def find_multiple(n):
    # 使用队列实现BFS
    q = deque()
    # 初始化队列，存储的是(模n值, 对应的数字字符串)
    q.append((1 % n, "1"))
    visited = set([1 % n])  # 用于记录访问过的模n值，避免重复搜索

    while q:
        mod, num_str = q.popleft()

        # 检查当前模n值是否为0，是则找到答案
        if mod == 0:
            return num_str

        # 尝试在当前数字后加0或加1，生成新的数字，并计算模n值
        for digit in ["0", "1"]:
            new_num_str = num_str + digit
            new_mod = (mod * 10 + int(digit)) % n

            # 如果新模n值未访问过，则加入队列继续搜索
            if new_mod not in visited:
                q.append((new_mod, new_num_str))
                visited.add(new_mod)

def main():
    while True:
        n = int(input())
        if n == 0:
            break
        print(find_multiple(n))

if __name__ == "__main__":
    main()
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

## #44835504提交状态

状态: Accepted

源代码

```python
from collections import deque

def find_multiple(n):
    # 使用队列实现BFS
    q = deque()
    # 初始化队列，存储的是(模n值, 对应的数字字符串)
    q.append((1 % n, "1"))
    visited = set([1 % n])  # 用于记录访问过的模n值，避免重复搜索

    while q:
        mod, num_str = q.popleft()

        # 检查当前模n值是否为0，是则找到答案
        if mod == 0:
            return num_str

        # 尝试在当前数字后加0或加1，生成新的数字，并计算模n值
        for digit in ["0", "1"]:
            new_num_str = num_str + digit
            new_mod = (mod * 10 + int(digit)) % n

            # 如果新模n值未访问过，则加入队列继续搜索
            if new_mod not in visited:
                q.append((new_mod, new_num_str))
                visited.add(new_mod)

def main():
    while True:
        n = int(input())
        if n == 0:
            break
        print(find_multiple(n))

if __name__ == "__main__":
    main()
```

## 04115: 鸣人和佐助

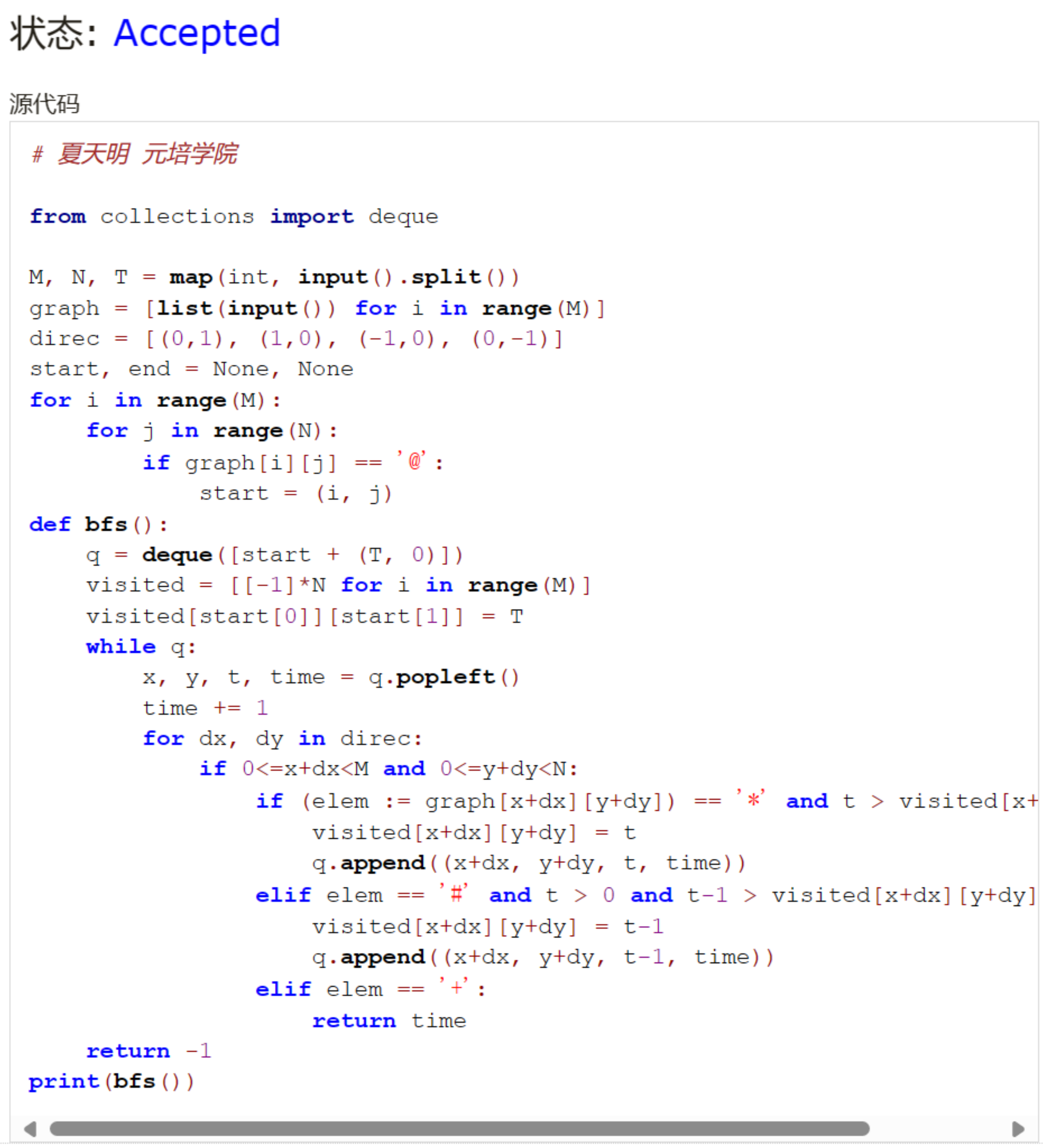
bfs, http://cs101.openjudge.cn/practice/04115/

思路：bfs的术语还不是很清楚 影响理解程序 阅读同学代码

代码

```python
from collections import deque

M, N, T = map(int, input().split())
graph = [list(input()) for i in range(M)]
direc = [(0,1), (1,0), (-1,0), (0,-1)]
start, end = None, None
for i in range(M):
    for j in range(N):
        if graph[i][j] == '@':
            start = (i, j)
def bfs():
    q = deque([start + (T, 0)])
    visited = [[-1]*N for i in range(M)]
    visited[start[0]][start[1]] = T
    while q:
        x, y, t, time = q.popleft()
        time += 1
        for dx, dy in direc:
            if 0<=x+dx<M and 0<=y+dy<N:
                if (elem := graph[x+dx][y+dy]) == '*' and t > visited[x+dx]
[y+dy]:
                    visited[x+dx][y+dy] = t
                    q.append((x+dx, y+dy, t, time))
                elif elem == '#' and t > 0 and t-1 > visited[x+dx][y+dy]:
                    visited[x+dx][y+dy] = t-1
                    q.append((x+dx, y+dy, t-1, time))
                elif elem == '+':
                    return time
    return -1
print(bfs())
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）


状态: Accepted

源代码

```
# 夏天明 元培学院

from collections import deque

M, N, T = map(int, input().split())
graph = [list(input()) for i in range(M)]
direc = [(0,1), (1,0), (-1,0), (0,-1)]
start, end = None, None
for i in range(M):
    for j in range(N):
        if graph[i][j] == '@':
            start = (i, j)
def bfs():
    q = deque([start + (T, 0)])
    visited = [[-1]*N for i in range(M)]
    visited[start[0]][start[1]] = T
    while q:
        x, y, t, time = q.popleft()
        time += 1
        for dx, dy in direc:
            if 0<=x+dx<M and 0<=y+dy<N:
                if (elem := graph[x+dx][y+dy]) == '*' and t > visited[x+
                    visited[x+dx][y+dy] = t
                    q.append((x+dx, y+dy, t, time))
                elif elem == '#' and t > 0 and t-1 > visited[x+dx][y+dy]
                    visited[x+dx][y+dy] = t-1
                    q.append((x+dx, y+dy, t-1, time))
                elif elem == '+':
                    return time
    return -1
print(bfs())
```

## 20106: 走山路

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

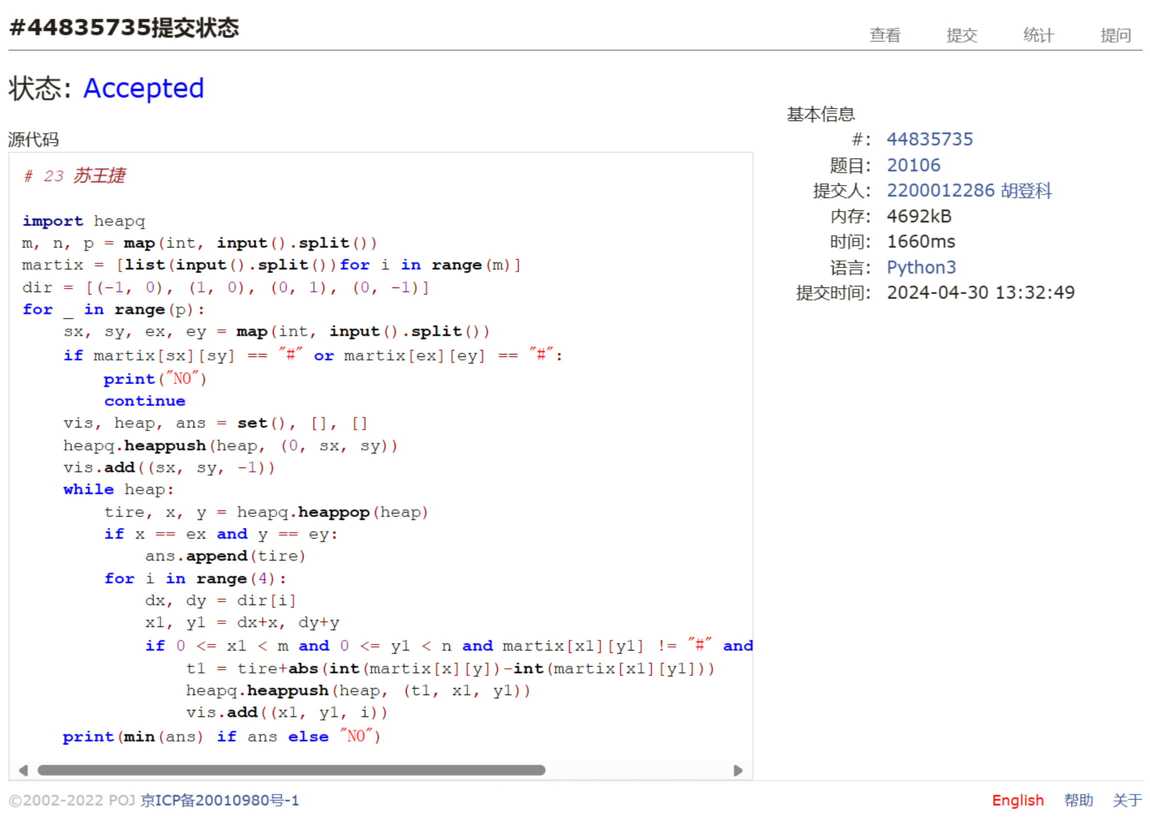
思路：与上一道题目的思考过程类似 但是需要对于#的处理

代码

```
#
# 23 苏王捷

import heapq
m, n, p = map(int, input().split())
martix = [list(input().split())for i in range(m)]
```

```python
dir = [(-1, 0), (1, 0), (0, 1), (0, -1)]
for _ in range(p):
    sx, sy, ex, ey = map(int, input().split())
    if martix[sx][sy] == "#" or martix[ex][ey] == "#":
        print("NO")
        continue
    vis, heap, ans = set(), [], []
    heapq.heappush(heap, (0, sx, sy))
    vis.add((sx, sy, -1))
    while heap:
        tire, x, y = heapq.heappop(heap)
        if x == ex and y == ey:
            ans.append(tire)
        for i in range(4):
            dx, dy = dir[i]
            x1, y1 = dx+x, dy+y
            if 0 <= x1 < m and 0 <= y1 < n and martix[x1][y1] != "#" and (x1, y1,
i) not in vis:
                t1 = tire+abs(int(martix[x][y])-int(martix[x1][y1]))
                heapq.heappush(heap, (t1, x1, y1))
                vis.add((x1, y1, i))
    print(min(ans) if ans else "NO")
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）



**#44835735提交状态**　　查看　提交　统计　提问

状态: Accepted

源代码

```python
# 23 苏王捷

import heapq
m, n, p = map(int, input().split())
martix = [list(input().split())for i in range(m)]
dir = [(-1, 0), (1, 0), (0, 1), (0, -1)]
for _ in range(p):
    sx, sy, ex, ey = map(int, input().split())
    if martix[sx][sy] == "#" or martix[ex][ey] == "#":
        print("NO")
        continue
    vis, heap, ans = set(), [], []
    heapq.heappush(heap, (0, sx, sy))
    vis.add((sx, sy, -1))
    while heap:
        tire, x, y = heapq.heappop(heap)
        if x == ex and y == ey:
            ans.append(tire)
        for i in range(4):
            dx, dy = dir[i]
            x1, y1 = dx+x, dy+y
            if 0 <= x1 < m and 0 <= y1 < n and martix[x1][y1] != "#" and
                t1 = tire+abs(int(martix[x][y])-int(martix[x1][y1]))
                heapq.heappush(heap, (t1, x1, y1))
                vis.add((x1, y1, i))
    print(min(ans) if ans else "NO")
```

基本信息

| | |
|---|---|
| #: | 44835735 |
| 题目: | 20106 |
| 提交人: | 2200012286 胡登科 |
| 内存: | 4692kB |
| 时间: | 1660ms |
| 语言: | Python3 |
| 提交时间: | 2024-04-30 13:32:49 |

©2002-2022 POJ 京ICP备20010980号-1　　English　帮助　关于

## 05442: 兔子与星空

Prim, http://cs101.openjudge.cn/practice/05442/

思路：

最小生成树 MST算法

代码

```
#
import heapq

def prim(graph, start):
    mst = []
    used = set([start])
    edges = [
        (cost, start, to)
        for to, cost in graph[start].items()
    ]
    heapq.heapify(edges)

    while edges:
        cost, frm, to = heapq.heappop(edges)
        if to not in used:
            used.add(to)
            mst.append((frm, to, cost))
            for to_next, cost2 in graph[to].items():
                if to_next not in used:
                    heapq.heappush(edges, (cost2, to, to_next))

    return mst

def solve():
    n = int(input())
    graph = {chr(i+65): {} for i in range(n)}
    for i in range(n-1):
        data = input().split()
        star = data[0]
        m = int(data[1])
        for j in range(m):
            to_star = data[2+j*2]
            cost = int(data[3+j*2])
            graph[star][to_star] = cost
            graph[to_star][star] = cost
    mst = prim(graph, 'A')
    print(sum(x[2] for x in mst))

solve()
```

代码运行截图（AC代码截图，至少包含有"Accepted"）

### 状态：Accepted

源代码

```python
import heapq

def prim(graph, start):
    mst = []
    used = set([start])
    edges = [
        (cost, start, to)
        for to, cost in graph[start].items()
    ]
    heapq.heapify(edges)

    while edges:
        cost, frm, to = heapq.heappop(edges)
        if to not in used:
            used.add(to)
            mst.append((frm, to, cost))
            for to_next, cost2 in graph[to].items():
                if to_next not in used:
                    heapq.heappush(edges, (cost2, to, to_next))

    return mst

def solve():
    n = int(input())
    graph = {chr(i+65): {} for i in range(n)}
    for i in range(n-1):
        data = input().split()
        star = data[0]
        m = int(data[1])
        for j in range(m):
            to_star = data[2+j*2]
            cost = int(data[3+j*2])
            graph[star][to_star] = cost
            graph[to_star][star] = cost
    mst = prim(graph, 'A')
    print(sum(x[2] for x in mst))

solve()
```

## 2. 学习总结和收获

如果作业题目简单，有否额外练习题目，比如：OJ“2024spring每日选做”、CF、LeetCode、洛谷等网站题目。

本周题目后四个基本为bsf的练习 在规定时间内没有完成 自认为原因是对于概念和基本代码格式理解不到位。打算看一下老师上课提及的教材 然后再回过来看看题目。