Assignment #A: 图论：遍历，树算及栈

Updated 2018 GMT+8 Apr 21, 2024

2024 spring, Complied by 王申睿——物理学院

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 typorahttps://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/

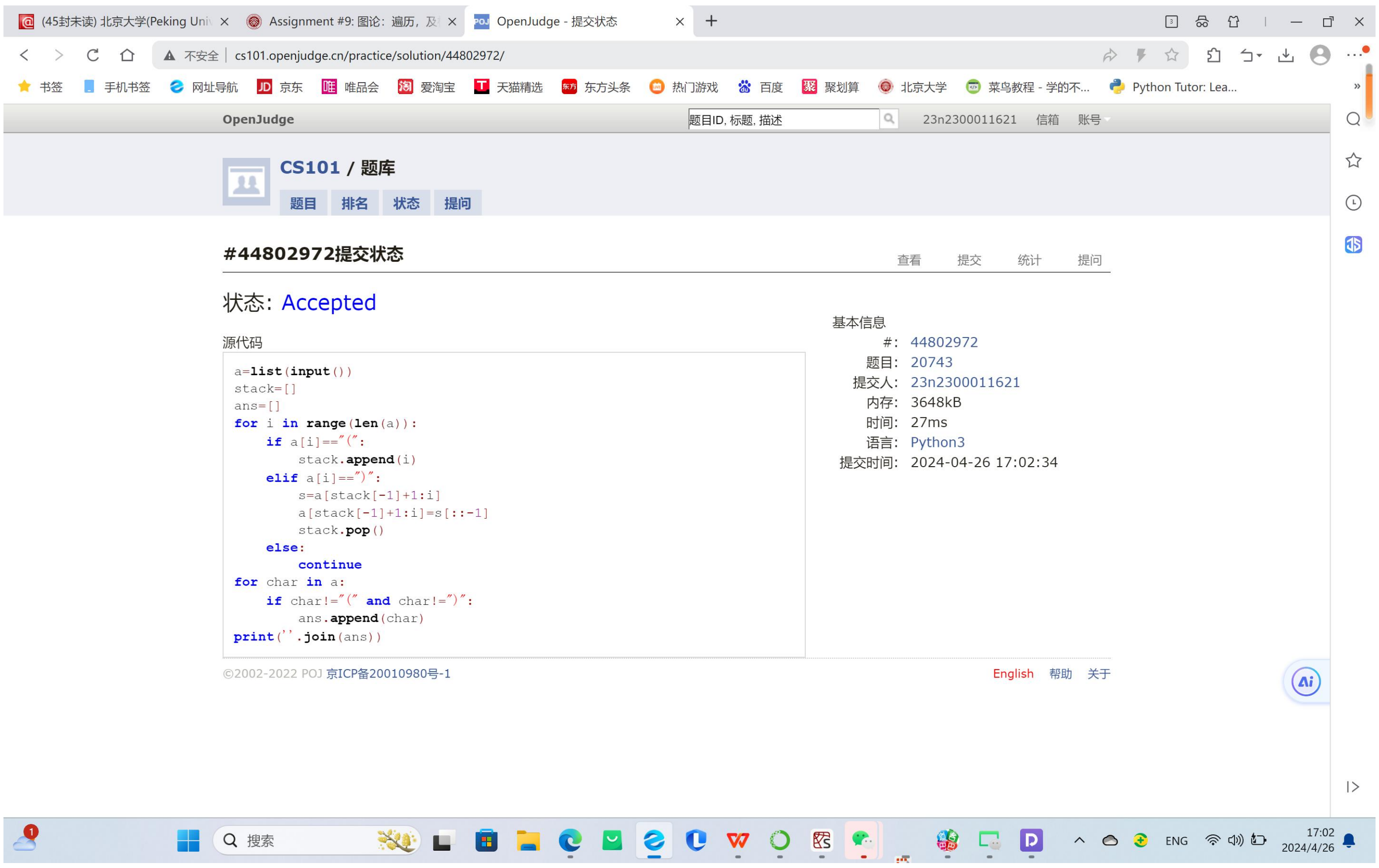
思路：

代码

a=list(input())

stack=[]

```
ans=[]

for i in range(len(a)):

    if a[i]=="(":

        stack.append(i)

    elif a[i]==")":

        s=a[stack[-1]+1:i]

        a[stack[-1]+1:i]=s[::-1]

        stack.pop()

    else:

        continue

for char in a:

    if char!="(" and char!=")":

        ans.append(char)

print(''.join(ans))
```



#44802972提交状态

状态: Accepted

源代码

```
a=list(input())
stack=[]
ans=[]
for i in range(len(a)):
    if a[i]=="(":
        stack.append(i)
    elif a[i]==")":
        s=a[stack[-1]+1:i]
        a[stack[-1]+1:i]=s[::-1]
        stack.pop()
    else:
        continue
for char in a:
    if char!="(" and char!=")":
        ans.append(char)
print(''.join(ans))
```

基本信息

#:	44802972
题目:	20743
提交人:	23n2300011621
内存:	3648kB
时间:	27ms
语言:	Python3
提交时间:	2024-04-26 17:02:34

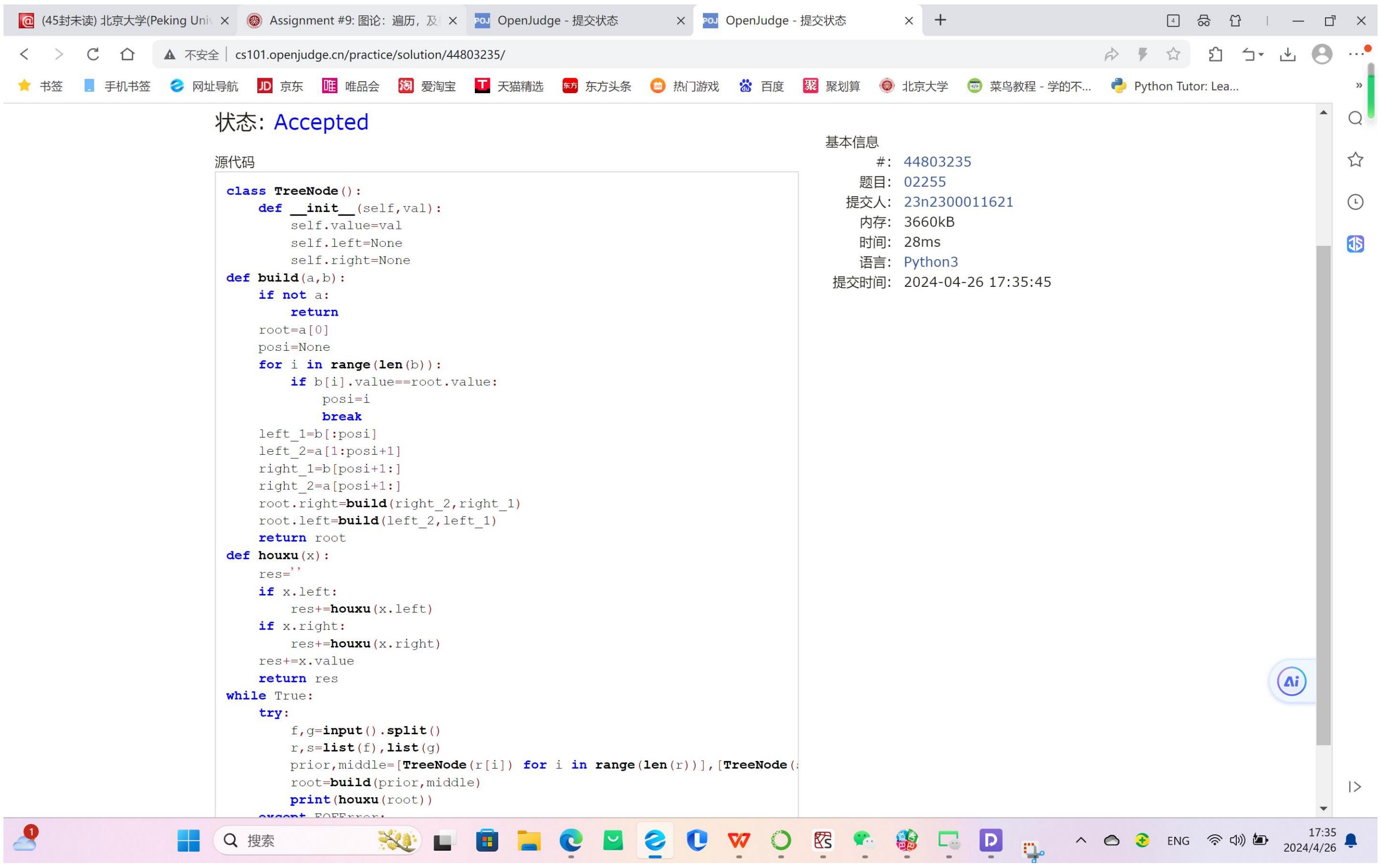
02255: 重建二叉树

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

思路:

代码

```
class TreeNode():

    def __init__(self,val):

        self.value=val

        self.left=None

        self.right=None

def build(a,b):

    if not a:

        return

    root=a[0]

    posi=None
    for i in range(len(b)):
        if b[i].value==root.value:
            posi=i
            break
    left_1=b[:posi]
    left_2=a[1:posi+1]
    right_1=b[posi+1:]
    right_2=a[posi+1:]
    root.right=build(right_2,right_1)
    root.left=build(left_2,left_1)
    return root
def houxu(x):
    res=''
    if x.left:
        res+=houxu(x.left)

    if x.right:

        res+=houxu(x.right)

    res+=x.value

    return res

while True:

    try:

        f,g=input().split()

        r,s=list(f),list(g)

        prior,middle=[TreeNode(r[i]) for i in range(len(r))],[TreeNode(s[j]) for j in range(len(s))]

        root=build(prior,middle)

        print(houxu(root))

    except EOFError:

        break
```


状态: Accepted

源代码

```python
class TreeNode():
    def __init__(self,val):
        self.value=val
        self.left=None
        self.right=None
def build(a,b):
    if not a:
        return
    root=a[0]
    posi=None
    for i in range(len(b)):
        if b[i].value==root.value:
            posi=i
            break
    left_1=b[:posi]
    left_2=a[1:posi+1]
    right_1=b[posi+1:]
    right_2=a[posi+1:]
    root.right=build(right_2,right_1)
    root.left=build(left_2,left_1)
    return root
def houxu(x):
    res=''
    if x.left:
        res+=houxu(x.left)
    if x.right:
        res+=houxu(x.right)
    res+=x.value
    return res
while True:
    try:
        f,g=input().split()
        r,s=list(f),list(g)
        prior,middle=[TreeNode(r[i]) for i in range(len(r))],[TreeNode(
        root=build(prior,middle)
        print(houxu(root))
    except EOFError:
```

基本信息

#:	44803235
题目:	02255
提交人:	23n2300011621
内存:	3660kB
时间:	28ms
语言:	Python3
提交时间:	2024-04-26 17:35:45

01426: Find The Multiple

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

要求用bfs实现

思路:

代码

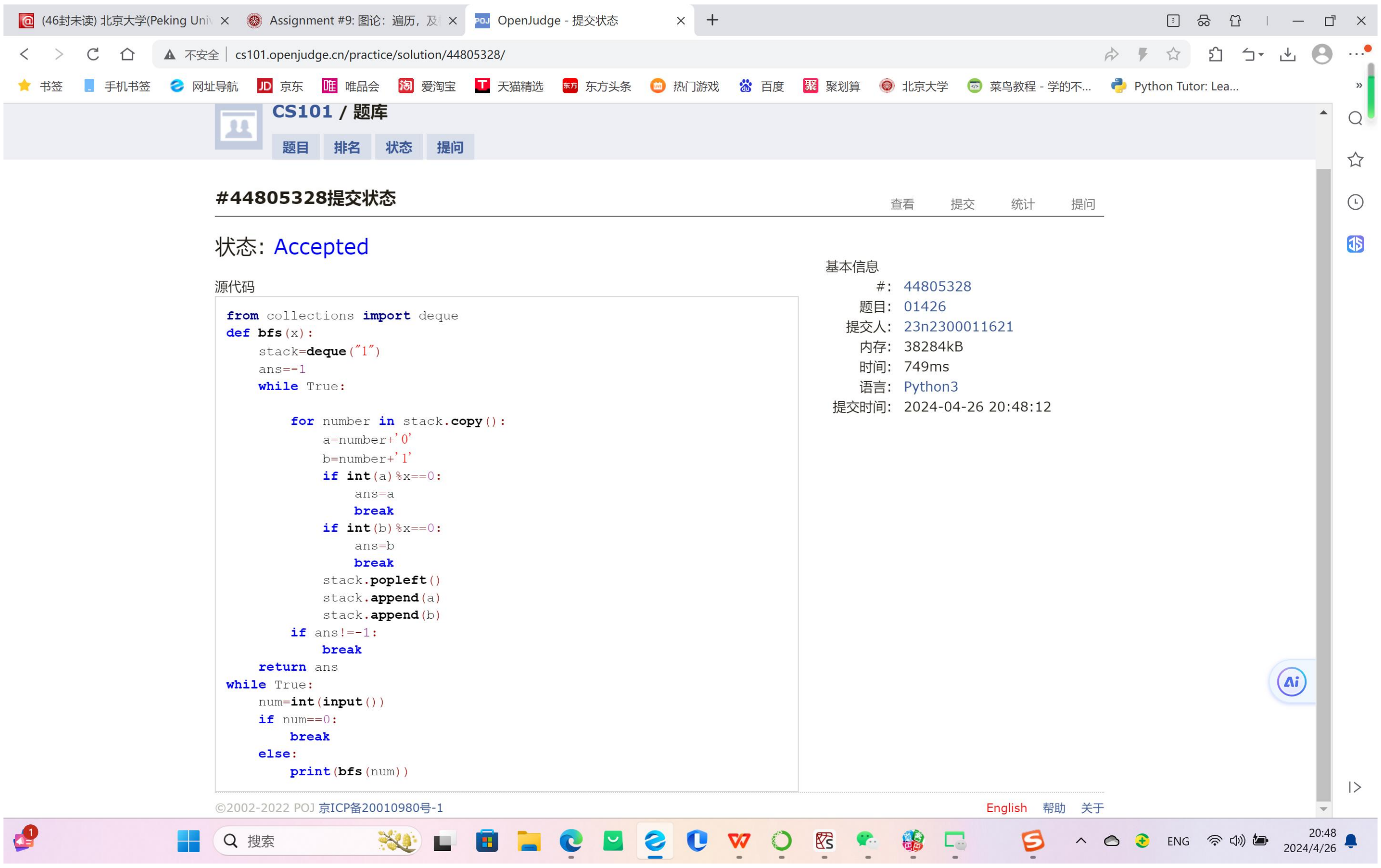
from collections import deque

def bfs(x):

    stack=deque("1")

```
ans=-1
while True:
    for number in stack.copy():
        a=number+'0'
        b=number+'1'
        if int(a)%x==0:
            ans=a
            break
        if int(b)%x==0:
            ans=b
            break
        stack.popleft()
        stack.append(a)
```

```
            stack.append(b)
        if ans!=-1:
            break
    return ans
while True:
    num=int(input())
    if num==0:
        break
    else:
        print(bfs(num))
```



CS101 / 题库

题目 排名 状态 提问

#44805328提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```python
from collections import deque
def bfs(x):
    stack=deque("1")
    ans=-1
    while True:

        for number in stack.copy():
            a=number+'0'
            b=number+'1'
            if int(a)%x==0:
                ans=a
                break
            if int(b)%x==0:
                ans=b
                break
            stack.popleft()
            stack.append(a)
            stack.append(b)
        if ans!=-1:
            break
    return ans
while True:
    num=int(input())
    if num==0:
        break
    else:
        print(bfs(num))
```

基本信息

#:	44805328
题目:	01426
提交人:	23n2300011621
内存:	38284kB
时间:	749ms
语言:	Python3
提交时间:	2024-04-26 20:48:12

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

04115: 鸣人和佐助

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

思路：

代码

```
#

```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

20106: 走山路

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

思路：

代码

```
#

```

（已参考题解）

05442: 兔子与星空

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

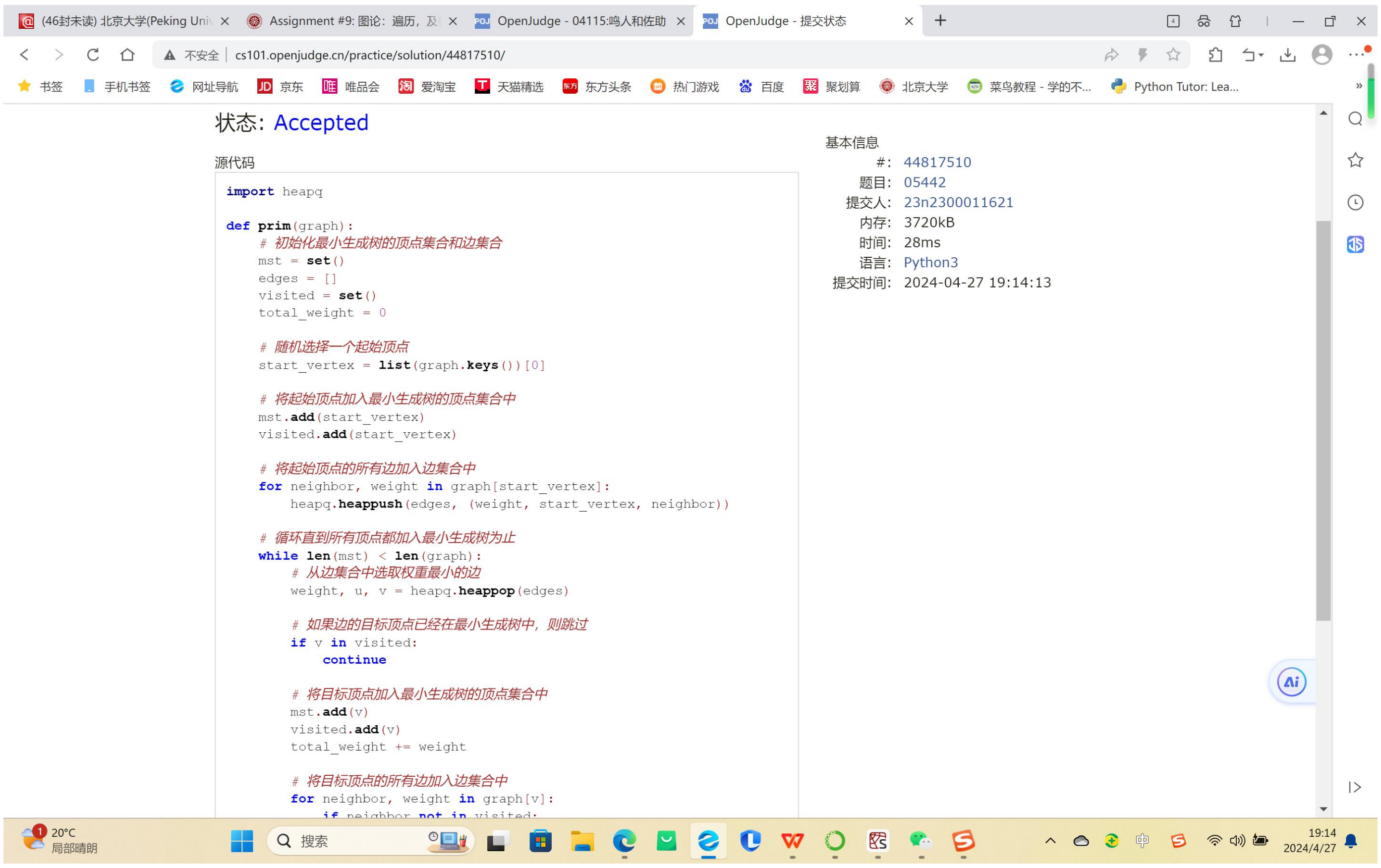
思路：

代码

。

```
import heapq


def prim(graph):

    # 初始化最小生成树的顶点集合和边集合

    mst = set()

    edges = []

    visited = set()

    total_weight = 0


    # 随机选择一个起始顶点

    start_vertex = list(graph.keys())[0]


    # 将起始顶点加入最小生成树的顶点集合中

    mst.add(start_vertex)

    visited.add(start_vertex)


    # 将起始顶点的所有边加入边集合中

    for neighbor, weight in graph[start_vertex]:
```

```
    heapq.heappush(edges, (weight, start_vertex, neighbor))



# 循环直到所有顶点都加入最小生成树为止

while len(mst) < len(graph):

    # 从边集合中选取权重最小的边

    weight, u, v = heapq.heappop(edges)



    # 如果边的目标顶点已经在最小生成树中，则跳过

    if v in visited:

        continue



    # 将目标顶点加入最小生成树的顶点集合中

    mst.add(v)

    visited.add(v)

    total_weight += weight



    # 将目标顶点的所有边加入边集合中

    for neighbor, weight in graph[v]:
            if neighbor not in visited:
                heapq.heappush(edges, (weight, v, neighbor))

    return total_weight


n = int(input())
graph = {}
for _ in range(n - 1):
    alist = list(input().split())
    if alist[0] not in graph.keys():
        graph[alist[0]] = []
    for i in range(1, int(alist[1]) + 1):
        if alist[2 * i] not in graph.keys():
            graph[alist[2 * i]] = []
        graph[alist[0]].append((alist[2 * i], int(alist[2 * i + 1])))
        graph[alist[2 * i]].append((alist[0], int(alist[2 * i + 1])))
print(prim(graph))
```


状态: Accepted

源代码

```python
import heapq

def prim(graph):
    # 初始化最小生成树的顶点集合和边集合
    mst = set()
    edges = []
    visited = set()
    total_weight = 0

    # 随机选择一个起始顶点
    start_vertex = list(graph.keys())[0]

    # 将起始顶点加入最小生成树的顶点集合中
    mst.add(start_vertex)
    visited.add(start_vertex)

    # 将起始顶点的所有边加入边集合中
    for neighbor, weight in graph[start_vertex]:
        heapq.heappush(edges, (weight, start_vertex, neighbor))

    # 循环直到所有顶点都加入最小生成树为止
    while len(mst) < len(graph):
        # 从边集合中选取权重最小的边
        weight, u, v = heapq.heappop(edges)

        # 如果边的目标顶点已经在最小生成树中，则跳过
        if v in visited:
            continue

        # 将目标顶点加入最小生成树的顶点集合中
        mst.add(v)
        visited.add(v)
        total_weight += weight

        # 将目标顶点的所有边加入边集合中
        for neighbor, weight in graph[v]:
            if neighbor not in visited:
```

基本信息

#:	44817510
题目:	05442
提交人:	23n2300011621
内存:	3720kB
时间:	28ms
语言:	Python3
提交时间:	2024-04-27 19:14:13

2. 学习总结和收获

本周第一次独立完成了bfs！计概学bfs的时候其实已经没有精力再理解新的算法了，直到学到二叉树的层级遍历时才理解了BFS的逻辑——队列。

鸣人和佐助的思想是第一次理解，所以参考了题解。下面贴出GPT和题解，讨论其不同：

GPT：

```
from collections import deque


def min_time_to_reach_target(grid, start_row, start_col, chakra):
    directions = [(1, 0), (-1, 0), (0, 1), (0, -1)]
    rows = len(grid)
    cols = len(grid[0])


    visited = set()
    queue = deque([(start_row, start_col, chakra, 0)])  # (row, col, chakra, time)


    while queue:
        row, col, current_chakra, time = queue.popleft()
        if grid[row][col] == '+':  # 发现了佐助
            return time

        for dr, dc in directions:
            new_row, new_col = row + dr, col + dc

            if 0 <= new_row < rows and 0 <= new_col < cols and (new_row, new_col) not in visited:
                visited.add((new_row, new_col))

                if grid[new_row][new_col] == '*':  # 如果是通路
                    queue.append((new_row, new_col, current_chakra, time + 1))
                elif grid[new_row][new_col] == '#':  # 如果是大蛇丸的手下
                    if current_chakra > 0:  # 修正此处判断条件
                        queue.append((new_row, new_col, current_chakra - 1, time + 1))
                    else:
                        continue  # 没有足够的查克拉无法通过


    return -1  # 没有找到佐助

# 读取输入
M, N, T = map(int, input().split())
grid = [input().strip() for _ in range(M)]

# 找到鸣人和佐助的起始位置
start_row, start_col = None, None
for i in range(M):
    for j in range(N):
        if grid[i][j] == '@':
            start_row, start_col = i, j
            break
    if start_row is not None:
        break
```

```
# 计算结果
result = min_time_to_reach_target(grid, start_row, start_col, T)
print(result)
```

这里的问题在于程序会先判断入队与否后消耗，会导致查克拉为-1的数组进入。

相比之下，先消耗查克拉再判断是否为通路的方法就能避免：

```
# 高铭泽 物理学院
from collections import deque


dire = [(-1, 0), (0, -1), (1, 0), (0, 1)]
flag = 0
ans = 0



def bfs(x, y, t):
    visited = set()
    global ans, flag
    q = deque()
    q.append((t, x, y, 0))
    while q:
        t, x, y, ans = q.popleft()
        for dx, dy in dire:
            nx = x + dx
            ny = y + dy
            if 0 <= nx < m and 0 <= ny < n:
                if g[nx][ny] != "#":
                    nt = t
                else:
                    nt = t - 1
                if nt >= 0 and (nt, nx, ny) not in visited:


                    newans = ans + 1
                    if g[nx][ny]=="+":
                        flag = 1
                        return flag,newans
                q.append((nt, nx, ny, newans))
                visited.add((nt, nx, ny))
    return flag,ans



m, n, t = map(int, input().split())
g = []
for i in range(m):
    g.append(list(input()))
for i in range(m):
    for j in range(n):
        if g[i][j] == "@":
            x = i
            y = j
flag,newans=bfs(x, y, t)
if flag:
    print(newans)
else:
    print(-1)
```

感觉闫老师的作业我实在难以完全驾驭，上个学期还能硬刚，这段时间基本只能结合着题解去学习。希望下次能做对同类型题目吧。