Assignment #A: 图论：算法，树算及栈

Updated 2018 GMT+8 Apr 21, 2024

2024 spring, Complied by 数学科学学院 王镜廷 2300010724

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

操作系统：Windows11 专业版

Python编程环境：VSCode 1.86.2, with extension Python and python version 3.12.2

1. 题目

20743: 整人的提词本

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

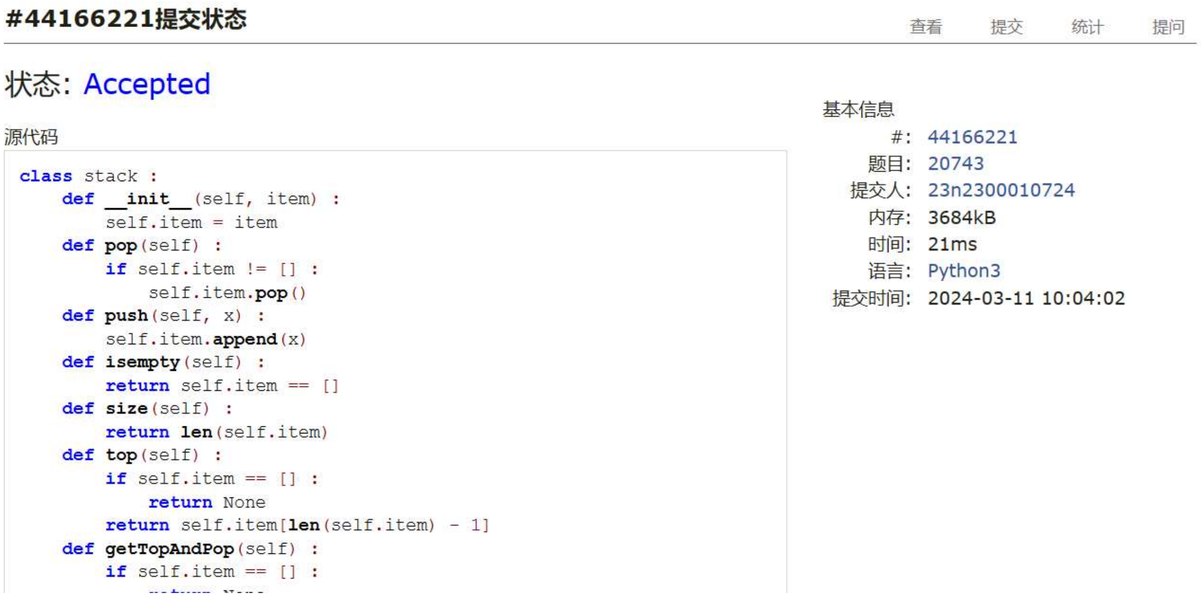
用时：忘记了，这个题是在很早之前的每日选做做的。

思路：直接依题意模拟即可，每次碰到左括号进栈，碰到右括号的时候不断弹栈（同时记录弹出的字母）直到左括号，再将字母按题目所说的顺序压入栈中继续处理。

代码

```python
class stack :
    def __init__(self, item) :
        self.item = item
    def pop(self) :
        if self.item != [] :
            self.item.pop()
    def push(self, x) :
        self.item.append(x)
    def isempty(self) :
        return self.item == []
    def size(self) :
        return len(self.item)
    def top(self) :
        if self.item == [] :
            return None
        return self.item[len(self.item) - 1]
    def getTopAndPop(self) :
        if self.item == [] :
            return None
        x = self.item[len(self.item) - 1]
        self.item.pop()
        return x
    def __str__(self) :
        return str(self.item)

s = input()
h = stack([])
for item in s :
    #print(item)
    if item == ")" :
        tmp = []
        while h.top() != "(" :
            tmp.append(h.getTopAndPop())
        h.pop()
        for i in tmp :
            h.push(i)
    else :
        h.push(item)
    #print("".join(i for i in h.item))
print("".join(i for i in h.item))
```

代码运行截图 （至少包含有"Accepted"）


#44166221提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
class stack :
    def __init__(self, item) :
        self.item = item
    def pop(self) :
        if self.item != [] :
            self.item.pop()
    def push(self, x) :
        self.item.append(x)
    def isempty(self) :
        return self.item == []
    def size(self) :
        return len(self.item)
    def top(self) :
        if self.item == [] :
            return None
        return self.item[len(self.item) - 1]
    def getTopAndPop(self) :
        if self.item == [] :
```

基本信息

#:	44166221
题目:	20743
提交人:	23n2300010724
内存:	3684kB
时间:	21ms
语言:	Python3
提交时间:	2024-03-11 10:04:02

02255: 重建二叉树

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

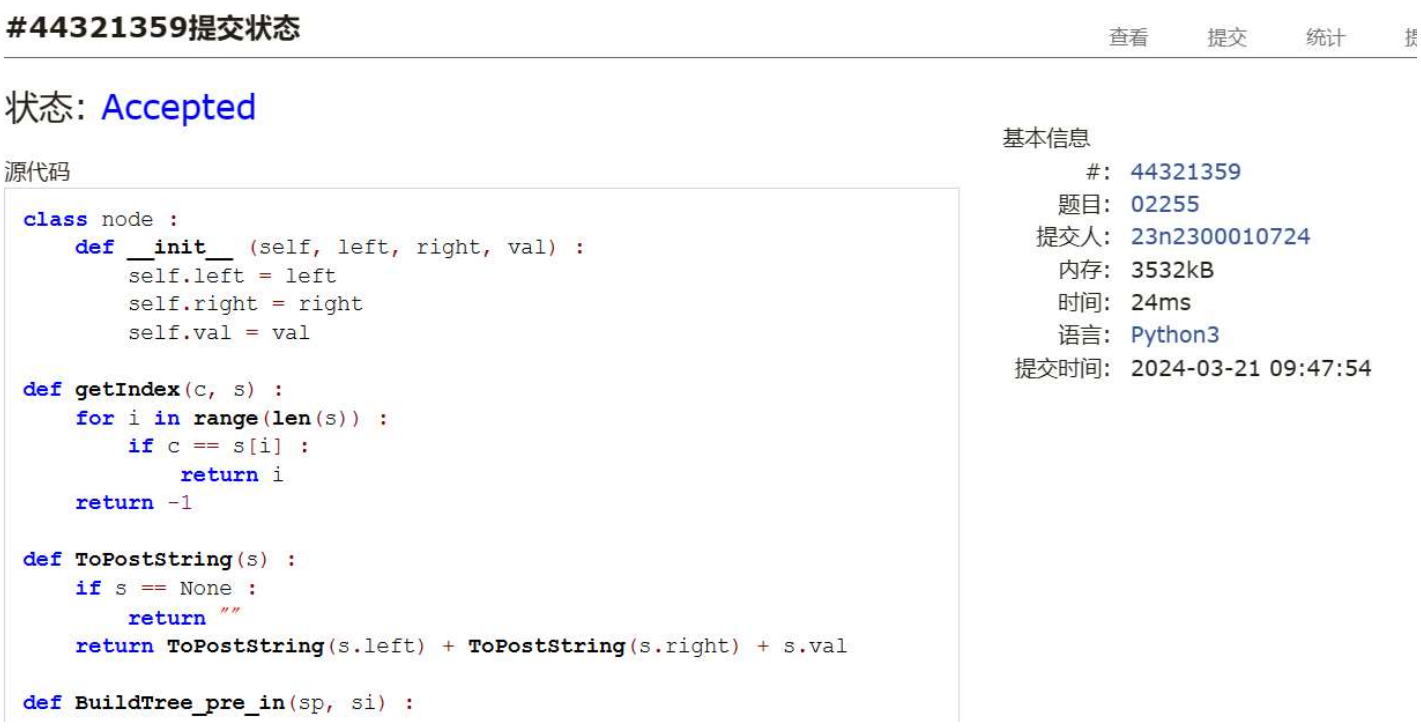
用时：忘记了，这个也是之前的每日选做。

思路：递归处理，每次可以从前序遍历看出根节点，再在中序遍历中分开左右子树

代码

```
class node :
    def __init__ (self, left, right, val) :
        self.left = left
        self.right = right
        self.val = val

def getIndex(c, s) :
    for i in range(len(s)) :
        if c == s[i] :
            return i
    return -1

def ToPostString(s) :
    if s == None :
        return ""
    return ToPostString(s.left) + ToPostString(s.right) + s.val

def BuildTree_pre_in(sp, si) :
    if sp == "" :
        return None
    l = getIndex(sp[0], si)
#    print(sp, si, l, sep = " ")
    return node(BuildTree_pre_in(sp[1 : l + 1], si[0 : l]), BuildTree_pre_in(sp[l + 1 : len(sp)

while True :
    try :
        s1, s2 = input().split()
        print(ToPostString(BuildTree_pre_in(s1, s2)))
    except :
        break
```

代码运行截图 （至少包含有"Accepted"）


#44321359提交状态

查看 提交 统计 提

状态: Accepted

源代码

```
class node :
    def __init__ (self, left, right, val) :
        self.left = left
        self.right = right
        self.val = val

def getIndex(c, s) :
    for i in range(len(s)) :
        if c == s[i] :
            return i
    return -1

def ToPostString(s) :
    if s == None :
        return ""
    return ToPostString(s.left) + ToPostString(s.right) + s.val

def BuildTree_pre_in(sp, si) :
```

基本信息

#:	44321359
题目:	02255
提交人:	23n2300010724
内存:	3532kB
时间:	24ms
语言:	Python3
提交时间:	2024-03-21 09:47:54

01426: Find The Multiple

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

要求用bfs实现

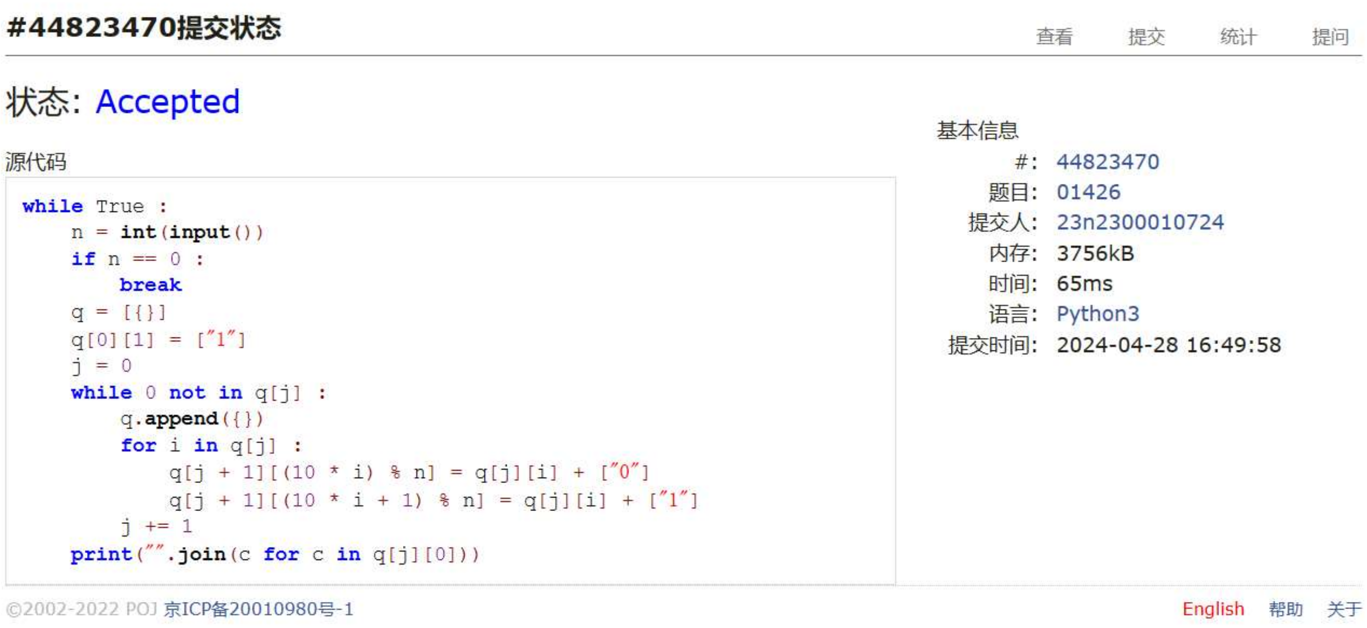
用时：约20分钟

思路：用BFS实现，特别的在这里实现时可以直接明确记录出每一层的情形。

代码

```
while True :
    n = int(input())
    if n == 0 :
        break
    q = [{}]
    q[0][1] = ["1"]
    j = 0
    while 0 not in q[j] :
        q.append({})
        for i in q[j] :
            q[j + 1][(10 * i) % n] = q[j][i] + ["0"]
            q[j + 1][(10 * i + 1) % n] = q[j][i] + ["1"]
        j += 1
    print("".join(c for c in q[j][0]))
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）


#44823470提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```python
while True :
    n = int(input())
    if n == 0 :
        break
    q = [{}]
    q[0][1] = ["1"]
    j = 0
    while 0 not in q[j] :
        q.append({})
        for i in q[j] :
            q[j + 1][(10 * i) % n] = q[j][i] + ["0"]
            q[j + 1][(10 * i + 1) % n] = q[j][i] + ["1"]
        j += 1
    print("".join(c for c in q[j][0]))
```

基本信息

#: 44823470
题目: 01426
提交人: 23n2300010724
内存: 3756kB
时间: 65ms
语言: Python3
提交时间: 2024-04-28 16:49:58

©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/

用时：约1小时（因为中间换了写法）

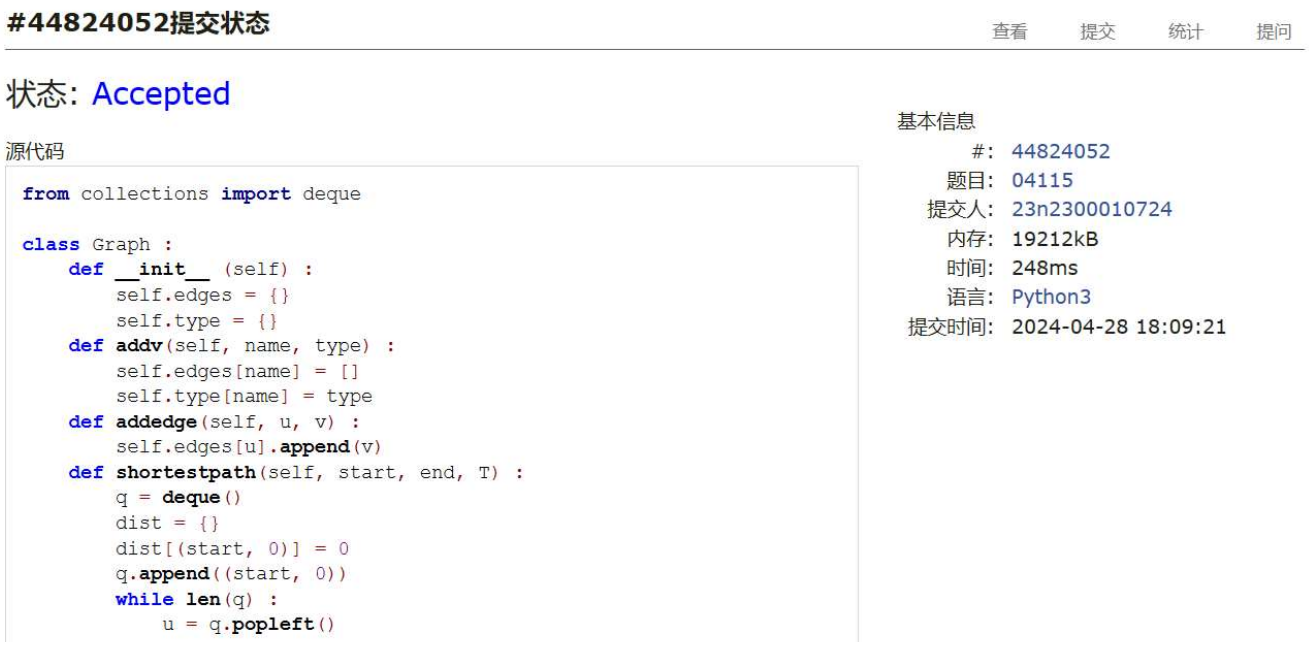
思路：这显然是个最短路问题，一个更直接的想法是通过合理的建图使得这一问题直接对应于最短路，从而使用已有算法完成，这可以通过建立分层图来完成（见下方第一段代码），但是这样写在本题似乎会导致MLE。于是之后采取了直接BFS的做法（见第二段代码），但是其过程事实上与在分层图中求最短路仍是基本上相同的，只是在记录图的结构的时候相对节省空间，从而可以通过本题。

代码

```python
# 这是分层图写法，在本题MLE了

from collections import deque

class Graph :
    def __init__ (self) :
        self.edges = {}
    def addv(self, name) :
        self.edges[name] = []
    def addedge(self, u, v) :
        self.edges[u].append(v)
    def shortestpath(self, start, end) :
        q = deque()
        dist = {}
        dist[start] = 0
        q.append(start)
        while len(q) :
            u = q.popleft()
            for v in self.edges[u] :
                if v not in dist :
                    dist[v] = dist[u] + 1
                    q.append(v)
        if end in dist :
            return dist[end]
        else :
            return -1

M, N, T = map(int, input().split())
q = []
for i in range(M) :
    q.append(input())
G = Graph()
for i in range(M) :
    for j in range(N) :
        for k in range(T + 1) :
            G.addv((i, j, k))
G.addv("end")
for i in range(M) :
    for j in range(N) :
        for k in range(T + 1) :
            if q[i][j] == "#" :
                dxs = [(0, 1, -1), (0, -1, -1), (1, 0, -1), (-1, 0, -1)]
            else :
                dxs = [(0, 1, 0), (0, -1, 0), (1, 0, 0), (-1, 0, 0)]
            for dx in dxs :
                u, v, w = i + dx[0], j + dx[1], k + dx[2]
                if (u, v, w) in G.edges :
                    G.addedge((u, v, w), (i, j, k))
for i in range(M) :
    for j in range(N) :
        if q[i][j] == "@" :
            start = (i, j, 0)
        if q[i][j] == "+" :
            end = (i, j)
for k in range(T + 1) :
    G.addedge((end[0], end[1], k), "end")
res = G.shortestpath(start, "end")
print(res - 1 if res != -1 else -1)
```

```python
# 这个是AC代码

from collections import deque

class Graph :
    def __init__ (self) :
        self.edges = {}
        self.type = {}
    def addv(self, name, type) :
        self.edges[name] = []
        self.type[name] = type
    def addedge(self, u, v) :
        self.edges[u].append(v)
    def shortestpath(self, start, end, T) :
        q = deque()
        dist = {}
        dist[(start, 0)] = 0
        q.append((start, 0))
        while len(q) :
            u = q.popleft()
            for v in self.edges[u[0]] :
                if self.type[v] == 1 :
                    if (v, u[1] + 1) in dist :
                        continue
                    if u[1] == T :
                        continue
                    dist[(v, u[1] + 1)] = dist[u] + 1
                    q.append((v, u[1] + 1))
                else :
                    if (v, u[1]) in dist :
                        continue
                    if v == end :
                        return dist[u] + 1
                    dist[(v, u[1])] = dist[u] + 1
                    q.append((v, u[1]))
        return -1
M, N, T = map(int, input().split())
q = []
for i in range(M) :
    q.append(input())
G = Graph()
for i in range(M) :
    for j in range(N) :
        for k in range(T + 1) :
            G.addv((i, j), 1 if q[i][j] == "#" else 0)
for i in range(M) :
    for j in range(N) :
            dxs = [(0, 1), (0, -1), (1, 0), (-1, 0)]
            for dx in dxs :
                u, v = i + dx[0], j + dx[1]
                if (u, v) in G.edges :
                    G.addedge((u, v), (i, j))
for i in range(M) :
    for j in range(N) :
        if q[i][j] == "@" :
            start = (i, j)
        if q[i][j] == "+" :
            end = (i, j)
res = G.shortestpath(start, end, T)
print(res)
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）


#44824052提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
from collections import deque

class Graph :
    def __init__ (self) :
        self.edges = {}
        self.type = {}
    def addv(self, name, type) :
        self.edges[name] = []
        self.type[name] = type
    def addedge(self, u, v) :
        self.edges[u].append(v)
    def shortestpath(self, start, end, T) :
        q = deque()
        dist = {}
        dist[(start, 0)] = 0
        q.append((start, 0))
        while len(q) :
            u = q.popleft()
```

基本信息

#: 44824052
题目: 04115
提交人: 23n2300010724
内存: 19212kB
时间: 248ms
语言: Python3
提交时间: 2024-04-28 18:09:21

20106: 走山路

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

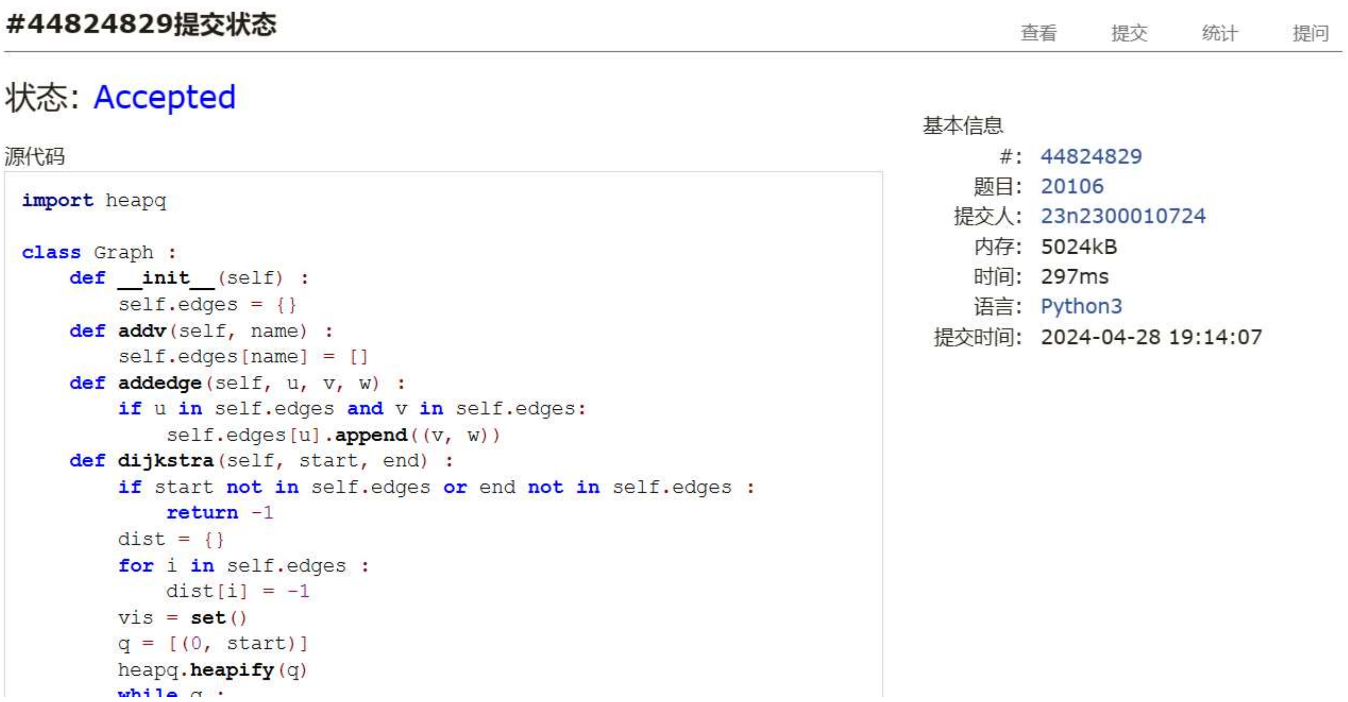
用时：约30分钟

思路：直接模拟

代码

```python
import heapq

class Graph :
    def __init__(self) :
        self.edges = {}
    def addv(self, name) :
        self.edges[name] = []
    def addedge(self, u, v, w) :
        if u in self.edges and v in self.edges:
            self.edges[u].append((v, w))
    def dijkstra(self, start, end) :
        if start not in self.edges or end not in self.edges :
            return -1
        dist = {}
        for i in self.edges :
            dist[i] = -1
        vis = set()
        q = [(0, start)]
        heapq.heapify(q)
        while q :
            d, u = heapq.heappop(q)
            #print(d, u)
            if u == end :
                return d
            if u in vis :
                continue
            dist[u] = d
            vis.add(u)
            for e in self.edges[u] :
                v, w = e
                if dist[u] + w < dist[v] or dist[v] == -1 :
                    dist[v] = dist[u] + w
                heapq.heappush(q, (dist[v], v))
        return -1

abs = lambda x : x if x >= 0 else -x
n, m, T = map(int, input().split())
q = []
for i in range(n) :
    q.append(input().split())
G = Graph()
for i in range(n) :
    for j in range(m) :
        if q[i][j] != "#" :
            G.addv((i, j))
for i in range(n) :
        for j in range(m) :
            dxs = [(1, 0), (0, 1), (-1, 0), (0, -1)]
            for dx in dxs :
                u, v = i + dx[0], j + dx[1]
                if (i, j) in G.edges and (u, v) in G.edges :
                    G.addedge((i, j), (u, v), abs(int(q[i][j]) - int(q[u][v])))
    for i in range(T) :
        u1, v1, u2, v2 = map(int, input().split())
        res = G.dijkstra((u1, v1), (u2, v2))
        print(res if res != -1 else "NO")
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）


#44824829提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
import heapq

class Graph :
    def __init__(self) :
        self.edges = {}
    def addv(self, name) :
        self.edges[name] = []
    def addedge(self, u, v, w) :
        if u in self.edges and v in self.edges:
            self.edges[u].append((v, w))
    def dijkstra(self, start, end) :
        if start not in self.edges or end not in self.edges :
            return -1
        dist = {}
        for i in self.edges :
            dist[i] = -1
        vis = set()
        q = [(0, start)]
        heapq.heapify(q)
        while q :
```

基本信息

#:	44824829
题目:	20106
提交人:	23n2300010724
内存:	5024kB
时间:	297ms
语言:	Python3
提交时间:	2024-04-28 19:14:07

05442: 兔子与星空

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

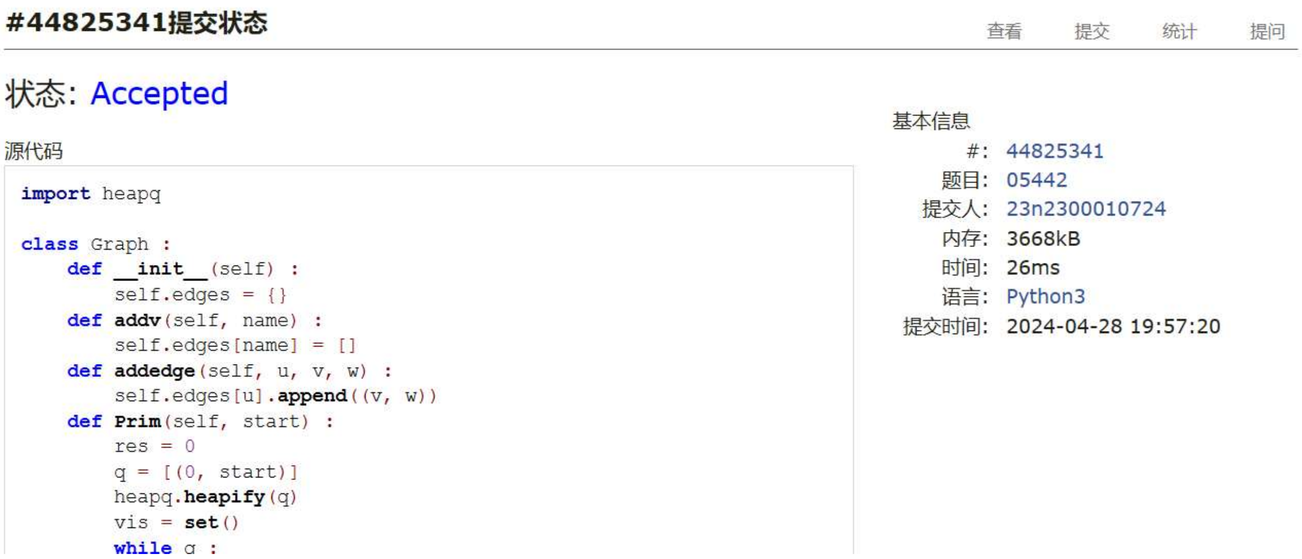
用时：约40分钟

思路：使用Prim求最小生成树

代码

```python
import heapq

class Graph :
    def __init__(self) :
        self.edges = {}
    def addv(self, name) :
        self.edges[name] = []
    def addedge(self, u, v, w) :
        self.edges[u].append((v, w))
    def Prim(self, start) :
        res = 0
        q = [(0, start)]
        heapq.heapify(q)
        vis = set()
        while q :
            d, u = heapq.heappop(q)
            if u in vis :
                continue
            res += d
            for e in self.edges[u] :
                v, w = e
                heapq.heappush(q, (w, v))
            vis.add(u)
            #print(q, d, u)
        return res

n = int(input())
G = Graph()
for i in range(n) :
    G.addv(chr(ord('A') + i))
for i in range(n - 1) :
    s = input().split()
    for j in range(1, int(s[1]) + 1) :
        G.addedge(s[0], s[2 * j], int(s[2 * j + 1]))
        G.addedge(s[2 * j], s[0], int(s[2 * j + 1]))
print(G.Prim('A'))
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）


#44825341提交状态

状态: Accepted

源代码

```
import heapq

class Graph :
    def __init__(self) :
        self.edges = {}
    def addv(self, name) :
        self.edges[name] = []
    def addedge(self, u, v, w) :
        self.edges[u].append((v, w))
    def Prim(self, start) :
        res = 0
        q = [(0, start)]
        heapq.heapify(q)
        vis = set()
        while q :
```

基本信息

#:	44825341
题目:	05442
提交人:	23n2300010724
内存:	3668kB
时间:	26ms
语言:	Python3
提交时间:	2024-04-28 19:57:20

2. 学习总结和收获

这周还是有点忙，只是把这个作业写完了，不过这之后应该时间能稍微多一些，可以再熟悉熟悉图论的这些算法。