Assignment #A: 图论：遍历，树算及栈

Updated 2018 GMT+8 Apr 21, 2024

2024 spring, Complied by 武昱达

编程环境

Windows 11 PyCharm

1. 题目

20743: 整人的提词本

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

思路:

代码

```
def reverse(s:str):
    return s[::-1]

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

print(f(input()))
```

代码运行截图 （至少包含有"Accepted"）

02255: 重建二叉树

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

思路:

代码

```
class TreeNode:
    def __init__(self,value):
        self.value = value
        self.left = None
        self.right = None

#初始化global变量
node_dict,pre_order,idx,current_node=dict(),[],0,None

# 函数的功能是建立起以name为根的子树，参数是name和中序表达式
def TreeBuilding(name,in_order:list):
    # idx全局变量寻找左子树根
    # current_node指向现在操作的对象
    global idx,current_node,node_dict,pre_order
    #设置递归出口
    if len(in_order)==1:
        node_dict[name]=TreeNode(name)
        if current_node.left==None:
            current_node.left=node_dict[name]
            return
        current_node.right=node_dict[name]
        return

    # 建立树根并存在字典中，便于索引
    node_dict[name]=TreeNode(name)

    # 如果name节点是一个子节点，那current_node!=None
    # 建立起name和current_node的连接。
    if current_node!=None:
        if current_node.left==None:
```

```
            current_node.left=node_dict[name]
            pass
        elif current_node.right==None:
            current_node.right=node_dict[name]

    # 标明现在状态
    current_node=node_dict[name]
    pivot=in_order.index(name)

    # 建立右子树
    ltree_in_order=in_order[:pivot]
    if ltree_in_order:
        idx+=1
        TreeBuilding(pre_order[idx],ltree_in_order)

    # 建立右子树
    current_node=node_dict[name]
    rtree_in_order=in_order[pivot+1:]
    if rtree_in_order:
        idx+=1
        TreeBuilding(pre_order[idx],rtree_in_order)

def post_search(root):
    if root==None:
        return ""
    output=[]
    output.extend(post_search(root.left))
    output.extend(post_search(root.right))
    output.append(root.value)
    return "".join(output)

while True:
    try:
        node_dict = dict()
        pre_order,in_order=input().split()
        pre_order=list(pre_order)
        in_order=list(in_order)
        # 最初的父节点指向None，即根节点的父节点指向None
        current_node = None
        idx = 0
        if len(pre_order) == 1:
            print(pre_order[0])
        else:
            TreeBuilding(pre_order[idx], in_order)
            print(post_search(node_dict[pre_order[0]]))
    except EOFError:
        break
```

代码运行截图 （至少包含有"Accepted"）

#44320525提交状态

查看 提交 统计 提问

状态: Accepted

基本信息

#: 44320525
题目: 02255
提交人: 23n2300011119（武）
内存: 3672kB
时间: 24ms
语言: Python3
提交时间: 2024-03-21 00:38:35

源代码

```
class TreeNode:
    def __init__(self,value):
        self.value = value
        self.left = None
        self.right = None

#初始化global变量
node_dict,pre_order,idx,current_node=dict(),[],0,None

# 函数的功能是建立起以name为根的子树，参数是name和中序表达式
def TreeBuilding(name,in_order:list):
    # idx全局变量寻找左子树根
    # current_node指向现在操作的对象
    global idx,current_node,node_dict,pre_order
    #设置递归出口
    if len(in_order)==1:
        node_dict[name]=TreeNode(name)
        if current_node.left==None:
            current_node.left=node_dict[name]
            return
```

01426: Find The Multiple

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

要求用bfs实现

思路:

BFS

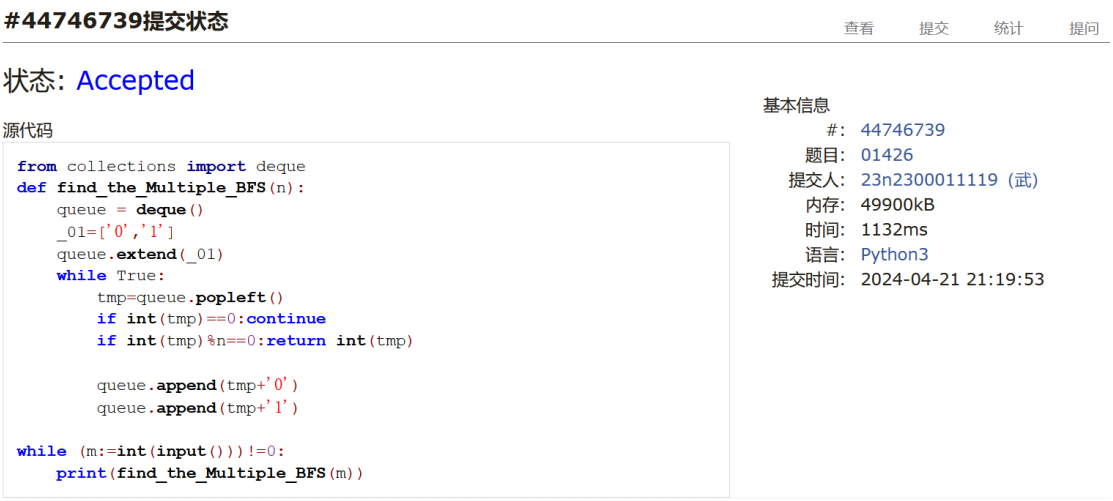
代码

```python
from collections import deque
def find_the_Multiple_BFS(n):
    queue = deque()
    _01=['0','1']
    queue.extend(_01)
    while True:
        tmp=queue.popleft()
        if int(tmp)==0:continue
        if int(tmp)%n==0:return int(tmp)

        queue.append(tmp+'0')
        queue.append(tmp+'1')

while (m:=int(input()))!=0:
    print(find_the_Multiple_BFS(m))
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）



#44746739提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
from collections import deque
def find_the_Multiple_BFS(n):
    queue = deque()
    _01=['0','1']
    queue.extend(_01)
    while True:
        tmp=queue.popleft()
        if int(tmp)==0:continue
        if int(tmp)%n==0:return int(tmp)

        queue.append(tmp+'0')
        queue.append(tmp+'1')

while (m:=int(input()))!=0:
    print(find_the_Multiple_BFS(m))
```

基本信息

#: 44746739
题目: 01426
提交人: 23n2300011119 (武)
内存: 49900kB
时间: 1132ms
语言: Python3
提交时间: 2024-04-21 21:19:53

04115: 鸣人和佐助

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

思路:

非常有趣的BFS，多了一个指标"查克拉数"，记为tools。

我们考虑类比背包问题的一维滚动数组来节省空间。（*）

在一般BFS问题中，一个顶点在队列中至多出现一次，因为其没有多余指标；现在有了多余的指标tools，我们允许其出现多次。

我们按照时间顺序进行思考：当某一个顶点A成为待遍历顶点时，其已经出现在了visited数组里，说明其被访问过，那么当前访问步数一定大于原访问步数。那么为什么允许其入队呢？**必然因为其消耗的查克拉数比较少**，所以**假使原来的路径因为查克拉数不够而行不通**，还有一个备用的步数长但是消耗查克拉数少的路径可能走通。反之，如果访问到当前顶点时其步数又长，消耗的查克拉数又多，可谓是又长又臭毫无优势，那么就不入队。

具体实现是（*）：visited数组记录上次访问时的剩余查克拉数，如果当前访问时（必然步数多于上次）的剩余查克拉数多，则当前顶点再次入队。

代码

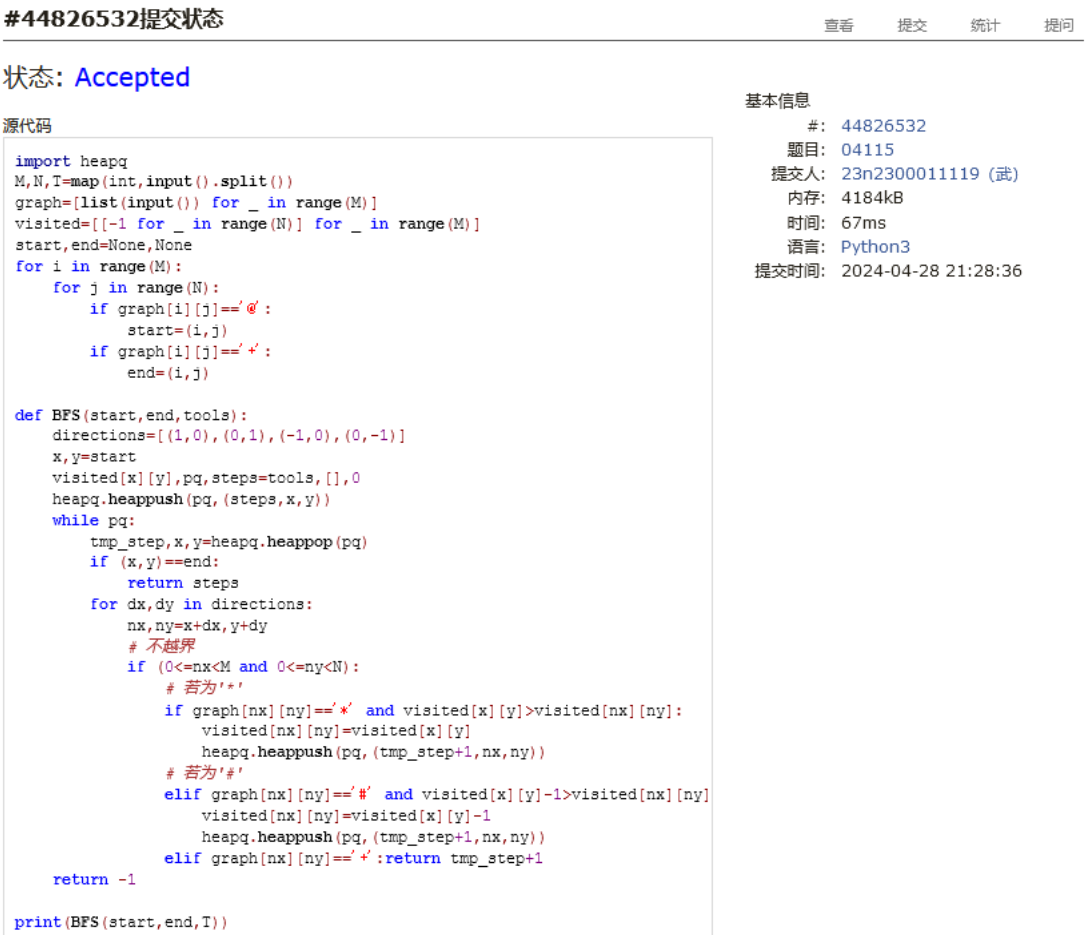
```
import heapq
M,N,T=map(int,input().split())
graph=[list(input()) for _ in range(M)]
visited=[[-1 for _ in range(N)] for _ in range(M)]
start,end=None,None
for i in range(M):
    for j in range(N):
        if graph[i][j]=='@':
            start=(i,j)
        if graph[i][j]=='+':
            end=(i,j)

def BFS(start,end,tools):
    directions=[(1,0),(0,1),(-1,0),(0,-1)]
    x,y=start
    visited[x][y],pq,steps=tools,[],0
    heapq.heappush(pq,(steps,x,y))
```

```
    while pq:
        tmp_step,x,y=heapq.heappop(pq)
        if (x,y)==end:
            return steps
        for dx,dy in directions:
            nx,ny=x+dx,y+dy
            # 不越界
            if (0<=nx<M and 0<=ny<N):
                # 若为'*'
                if graph[nx][ny]=='*' and visited[x][y]>visited[nx][ny]:
                    visited[nx][ny]=visited[x][y]
                    heapq.heappush(pq,(tmp_step+1,nx,ny))
                # 若为'#'
                elif graph[nx][ny]=='#' and visited[x][y]-1>visited[nx][ny]:
                    visited[nx][ny]=visited[x][y]-1
                    heapq.heappush(pq,(tmp_step+1,nx,ny))
                elif graph[nx][ny]=='+':return tmp_step+1
    return -1

print(BFS(start,end,T))
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）



#44826532提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
import heapq
M,N,T=map(int,input().split())
graph=[list(input()) for _ in range(M)]
visited=[[-1 for _ in range(N)] for _ in range(M)]
start,end=None,None
for i in range(M):
    for j in range(N):
        if graph[i][j]=='@':
            start=(i,j)
        if graph[i][j]=='+':
            end=(i,j)

def BFS(start,end,tools):
    directions=[(1,0),(0,1),(-1,0),(0,-1)]
    x,y=start
    visited[x][y],pq,steps=tools,[],0
    heapq.heappush(pq,(steps,x,y))
    while pq:
        tmp_step,x,y=heapq.heappop(pq)
        if (x,y)==end:
            return steps
        for dx,dy in directions:
            nx,ny=x+dx,y+dy
            # 不越界
            if (0<=nx<M and 0<=ny<N):
                # 若为'*'
                if graph[nx][ny]=='*' and visited[x][y]>visited[nx][ny]:
                    visited[nx][ny]=visited[x][y]
                    heapq.heappush(pq,(tmp_step+1,nx,ny))
                # 若为'#'
                elif graph[nx][ny]=='#' and visited[x][y]-1>visited[nx][ny]:
                    visited[nx][ny]=visited[x][y]-1
                    heapq.heappush(pq,(tmp_step+1,nx,ny))
                elif graph[nx][ny]=='+':return tmp_step+1
    return -1

print(BFS(start,end,T))
```

基本信息

#:	44826532
题目:	04115
提交人:	23n2300011119 (武)
内存:	4184kB
时间:	67ms
语言:	Python3
提交时间:	2024-04-28 21:28:36

20106: 走山路

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

思路：

代码

```
import heapq
m,n,p=map(int,input().split())
matrix,test=[["#"]*(n+2)],[]
for _ in range(m):
    matrix.append(['#']+list(map(str,input().split()))+['#'])
matrix.append(["#"]*(n+2))
for _ in range(p):
    temp=tuple(map(int,input().split()))
    start,end=(temp[0]+1,temp[1]+1),(temp[2]+1,temp[3]+1)
    test.append((start,end))
# for _ in matrix:
#     print(_)
# for _ in test:
#     print(_)
def bfs(start,end):
    # 起点或终点在#处，直接return NO
    if matrix[start[0]][start[1]]=="#" or matrix[end[0]][end[1]]=="#":
        return "NO"
    dx=[1,0,-1,0]
    dy=[0,1,0,-1]
    queue,visited,res=[],set(),[]
    heapq.heapify(queue)
    heapq.heappush(queue,[0,start[0],start[1]])
    visited.add((start[0],start[1]))
    while queue:
        height,x,y=heapq.heappop(queue)
        if (x,y)==end:
            return height
        for i in range(4):
            nx,ny=x+dx[i],y+dy[i]
            if matrix[nx][ny]!="#" and (nx,ny) not in visited:
                heapq.heappush(queue,[height+abs(int(matrix[nx][ny])-
int(matrix[x][y])),nx,ny])
                visited.add((x,y))
    return "NO"
for i in test:
    print(bfs(i[0],i[1]))
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

#43303112提交状态

查看 提交 统计 提问

状态: Accepted

基本信息

#: 43303112
题目: 20106
提交人: 23n2300011119 (武)
内存: 4168kB
时间: 1089ms
语言: Python3
提交时间: 2023-12-22 21:44:55

源代码

```
import heapq
m,n,p=map(int,input().split())
matrix,test=[["#"]*(n+2)],[]
for _ in range(m):
    matrix.append(['#']+list(map(str,input().split()))+['#'])
matrix.append(["#"]*(n+2))
for _ in range(p):
    temp=tuple(map(int,input().split()))
    start,end=(temp[0]+1,temp[1]+1),(temp[2]+1,temp[3]+1)
    test.append((start,end))
# for _ in matrix:
#     print(_)
# for _ in test:
#     print(_)
def bfs(start,end):
    # 起点或终点在#处，直接return NO
    if matrix[start[0]][start[1]]=="#" or matrix[end[0]][end[1]]=="#":
        return "NO"
    dx=[1,0,-1,0]
    dy=[0,1,0,-1]
    queue,visited,res=[],set(),[]
    heapq.heapify(queue)
```

05442: 兔子与星空

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

思路:

MSTs

代码

```python
import sys,heapq
class Vertex:
    def __init__(self,id):
        self.id=id
        self.connectedTo={}
        self.distance=sys.maxsize
        self.pre=None

    def __str__(self):
        return '*'+self.id

    def add_neighbor(self,nbr,weight):
        self.connectedTo[nbr]=weight

    def getConnections(self):
        return self.connectedTo.keys()

    def getWeight(self,nbr):
        return self.connectedTo[nbr]

    def __lt__(self, other):
        return self.distance<other.distance

class Graph:
    def __init__(self):
        self.vertList={}
        self.numVertices=0
```

```
28
29      def __str__(self):
30          return " ".join(map(str,self.vertList.values()))
31
32      def addVertex(self,key):
33          if key in self.vertList:return
34
35          newVertex=Vertex(key)
36          self.vertList[key]=newVertex
37          self.numVertices+=1
38          return newVertex
39
40      def getVertex(self,key):
41          return self.vertList[key]
42
43      def addEdge(self,f:str,t:str,weight):
44          if f not in self.vertList:
45              self.addVertex(f)
46          if t not in self.vertList:
47              self.addVertex(t)
48          self.vertList[f].add_neighbor(self.vertList[t],weight)
49          self.vertList[t].add_neighbor(self.vertList[f],weight)
50
51  def prim(start:Vertex):
52      pq,visited=[],set()
53      start.distance=0
54      heapq.heappush(pq,(0,start))
55      while pq:
56          curDist,curVert=heapq.heappop(pq)
57          if curVert in visited:
58              continue
59          visited.add(curVert)
60          for nextVert in curVert.getConnections():
61              weight=curVert.getWeight(nextVert)
62              if nextVert not in visited and weight<nextVert.distance:
63                  nextVert.distance=weight
64                  nextVert.pre=curVert
65                  heapq.heappush(pq,(weight,nextVert))
66      return start
67
68  def Tree_summing(graph:Graph):
69      res=0
70      for vert in graph.vertList.values():
71          if vert.pre:res += vert.connectedTo[vert.pre]
72      return res
73
74  n=int(input())
75  graph=Graph()
76  for i in range(n-1):
77      raw=list(input().split())
78      curVert=raw[0]
79      graph.addVertex(curVert)
80      data=raw[2:]
81      for j in range(len(data)//2):
82          new_VertKey=data[2*j]
83          new_VertWeight=int(data[2*j+1])
```

```
        graph.addEdge(curVert,new_VertKey,new_VertWeight)

start=None
for i in graph.vertList.values():
    start=i
    break
prim(start)
print(Tree_summing(graph))
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

#44821808提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
import sys,heapq
class Vertex:
    def __init__(self,id):
        self.id=id
        self.connectedTo={}
        self.distance=sys.maxsize
        self.pre=None

    def __str__(self):
        return str(self.id)+' '+str(self.distance)

    def add_neighbor(self,nbr,weight):
        self.connectedTo[nbr]=weight

    def getConnections(self):
        return self.connectedTo.keys()

    def getWeight(self,nbr):
        return self.connectedTo[nbr]

    def __lt__(self, other):
        return self.distance<other.distance

class Graph:
    def __init__(self):
        self.vertList={}
        self.numVertices=0

    def __str__(self):
        return ' '.join(map(str,self.vertList.values()))

    def addVertex(self,key):
        if key in self.vertList:return

        newVertex=Vertex(key)
        self.vertList[key]=newVertex
        self.numVertices+=1
        return newVertex

    def getVertex(self,key):
        return self.vertList[key]

    def addEdge(self,f:str,t:str,weight):
        if f not in self.vertList:
            self.addVertex(f)
        if t not in self.vertList:
            self.addVertex(t)
        self.vertList[f].add_neighbor(self.vertList[t],weight)
        self.vertList[t].add_neighbor(self.vertList[f],weight)

def prim(start:Vertex):
    pq,visited=[],set()
    start.distance=0
    heapq.heappush(pq,(0,start))
    while pq:
        curDist,curVert=heapq.heappop(pq)
        if curVert in visited:
            continue
        visited.add(curVert)
```

基本信息

#: 44821808
题目: 05442
提交人: 23n2300011119 (武)
内存: 3800kB
时间: 26ms
语言: Python3
提交时间: 2024-04-28 11:45:03

2. 学习总结和收获

鸣人和佐助非常巧妙，写完之后心情舒畅；

兔子和星空非常标准的prim，但是代码太长不太好写；

走山路限时回归，梦回上学期Dijkstra；

五一抽点时间复习基础知识，书面的知识还非常不牢固。