

Assignment #A: 图论：遍历，树算及栈

Updated 2018 GMT+8 Apr 21, 2024

2024 spring, Compiled 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

```
1  def reverse(s:str):
2      return s[::-1]
3
4  def f(s):
5      stack = []
6      for char in s:
7          if char == ')':
8              temp = []
9              while stack and stack[-1] != '(':
10                 temp.append(stack.pop())
11                 stack.pop() # pop the '('
12                 stack.extend(temp)
13             else:
14                 stack.append(char)
15             return ''.join(stack)
16
17  print(f(input()))
```

代码运行截图 (至少包含有"Accepted")

状态: Accepted

源代码

```
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()))
```

基本信息

#: 44167310
题目: 20743
提交人: 23n2300011119 (武)
内存: 3620kB
时间: 22ms
语言: Python3
提交时间: 2024-03-11 11:12:42

02255: 重建二叉树

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

思路:

代码

```
1 class TreeNode:
2     def __init__(self,value):
3         self.value = value
4         self.left = None
5         self.right = None
6
7 #初始化global变量
8 node_dict,pre_order,idx,current_node=dict(),[],0,None
9
10 # 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
11 def TreeBuilding(name,in_order:list):
12     # idx全局变量寻找左子树根
13     # current_node指向现在操作的对象
14     global idx,current_node,node_dict,pre_order
15     #设置递归出口
16     if len(in_order)==1:
17         node_dict[name]=TreeNode(name)
18         if current_node.left==None:
19             current_node.left=node_dict[name]
20             return
21         current_node.right=node_dict[name]
22         return
23
24     # 建立树根并存在字典中，便于索引
25     node_dict[name]=TreeNode(name)
26
27     # 如果name节点是一个子节点，那current_node!=None
28     # 建立起name和current_node的连接。
29     if current_node!=None:
30         if current_node.left==None:
```

```

31         current_node.left=node_dict[name]
32         pass
33     elif current_node.right==None:
34         current_node.right=node_dict[name]
35
36     # 标明现在状态
37     current_node=node_dict[name]
38     pivot=in_order.index(name)
39
40     # 建立右子树
41     ltree_in_order=in_order[:pivot]
42     if ltree_in_order:
43         idx+=1
44         TreeBuilding(pre_order[idx],ltree_in_order)
45
46     # 建立右子树
47     current_node=node_dict[name]
48     rtree_in_order=in_order[pivot+1:]
49     if rtree_in_order:
50         idx+=1
51         TreeBuilding(pre_order[idx],rtree_in_order)
52
53 def post_search(root):
54     if root==None:
55         return ""
56     output=[]
57     output.extend(post_search(root.left))
58     output.extend(post_search(root.right))
59     output.append(root.value)
60     return "".join(output)
61
62 while True:
63     try:
64         node_dict = dict()
65         pre_order,in_order=input().split()
66         pre_order=list(pre_order)
67         in_order=list(in_order)
68         # 最初的父节点指向None，即根节点的父节点指向None
69         current_node = None
70         idx = 0
71         if len(pre_order) == 1:
72             print(pre_order[0])
73         else:
74             TreeBuilding(pre_order[idx], in_order)
75             print(post_search(node_dict[pre_order[0]]))
76     except EOFError:
77         break

```

代码运行截图 (至少包含有"Accepted")

状态: Accepted

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
```

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

01426: Find The Multiple

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

要求用bfs实现

思路:

BFS

代码

```
1  from collections import deque
2  def find_the_Multiple_BFS(n):
3      queue = deque()
4      _01=['0', '1']
5      queue.extend(_01)
6      while True:
7          tmp=queue.popleft()
8          if int(tmp)==0:continue
9          if int(tmp)%n==0:return int(tmp)
10
11         queue.append(tmp+'0')
12         queue.append(tmp+'1')
13
14 while (m:=int(input()))!=0:
15     print(find_the_Multiple_BFS(m))
```

代码运行截图 (AC代码截图, 至少包含有"Accepted")

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

```
1 import heapq
2 M,N,T=map(int,input().split())
3 graph=[list(input()) for _ in range(M)]
4 visited=[[-1 for _ in range(N)] for _ in range(M)]
5 start,end=None,None
6 for i in range(M):
7     for j in range(N):
8         if graph[i][j]=='@':
9             start=(i,j)
10        if graph[i][j]=='.':
11            end=(i,j)
12
13 def BFS(start,end,tools):
14     directions=[(1,0),(0,1),(-1,0),(0,-1)]
15     x,y=start
16     visited[x][y],pq,steps=tools,[],0
17     heapq.heappush(pq,(steps,x,y))
```

```

18     while pq:
19         tmp_step,x,y=heapq.heappop(pq)
20         if (x,y)==end:
21             return steps
22         for dx,dy in directions:
23             nx,ny=x+dx,y+dy
24             # 不越界
25             if (0<=nx<M and 0<=ny<N):
26                 # 若为'*'
27                 if graph[nx][ny]=='*' and visited[x][y]>visited[nx][ny]:
28                     visited[nx][ny]=visited[x][y]
29                     heapq.heappush(pq,(tmp_step+1,nx,ny))
30                 # 若为'#'
31                 elif graph[nx][ny]=='#' and visited[x][y]-1>visited[nx][ny]:
32                     visited[nx][ny]=visited[x][y]-1
33                     heapq.heappush(pq,(tmp_step+1,nx,ny))
34                 elif graph[nx][ny]=='+':return tmp_step+1
35     return -1
36
37 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

```
1 import heapq
2 m,n,p=map(int,input().split())
3 matrix,test=[["#"]*(n+2)],[]
4 for _ in range(m):
5     matrix.append(['#']+list(map(str,input().split()))+['#'])
6 matrix.append(["#"]*(n+2))
7 for _ in range(p):
8     temp=tuple(map(int,input().split()))
9     start,end=(temp[0]+1,temp[1]+1),(temp[2]+1,temp[3]+1)
10    test.append((start,end))
11 # for _ in matrix:
12 #     print(_)
13 # for _ in test:
14 #     print(_)
15 def bfs(start,end):
16     # 起点或终点在#处, 直接return NO
17     if matrix[start[0]][start[1]]=="#" or matrix[end[0]][end[1]]=="#":
18         return "NO"
19     dx=[1,0,-1,0]
20     dy=[0,1,0,-1]
21     queue,visited,res=[],set(),[]
22     heapq.heapify(queue)
23     heapq.heappush(queue,[0,start[0],start[1]])
24     visited.add((start[0],start[1]))
25     while queue:
26         height,x,y=heapq.heappop(queue)
27         if (x,y)==end:
28             return height
29         for i in range(4):
30             nx,ny=x+dx[i],y+dy[i]
31             if matrix[nx][ny]!="#" and (nx,ny) not in visited:
32                 heapq.heappush(queue,[height+abs(int(matrix[nx][ny])-
33 int(matrix[x][y])),nx,ny])
34                 visited.add((x,y))
35     return "NO"
36 for i in test:
37     print(bfs(i[0],i[1]))
```

代码运行截图 (AC代码截图, 至少包含有"Accepted")

状态: Accepted

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)
```

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

05442: 兔子与星空

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

思路:

MSTs

代码

```
1 import sys,heapq
2 class Vertex:
3     def __init__(self,id):
4         self.id=id
5         self.connectedTo={}
6         self.distance=sys.maxsize
7         self.pre=None
8
9     def __str__(self):
10         return '*' + self.id
11
12     def add_neighbor(self,nbr,weight):
13         self.connectedTo[nbr]=weight
14
15     def getConnections(self):
16         return self.connectedTo.keys()
17
18     def getweight(self,nbr):
19         return self.connectedTo[nbr]
20
21     def __lt__(self, other):
22         return self.distance<other.distance
23
24 class Graph:
25     def __init__(self):
26         self.vertList={}
27         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])

```

```
84         graph.addEdge(curVert,new_VertKey,new_Vertweight)
85
86     start=None
87     for i in graph.vertList.values():
88         start=i
89         break
90     prim(start)
91     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 '*' +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

    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。