

Assignment #9: 图论：遍历，及 树算

Updated 1739 GMT+8 Apr 14, 2024

2024 spring, Compiled by 23工院 武昱达

编程环境

PyCharm Professional Edition

1. 题目

04081: 树的转换

<http://cs101.openjudge.cn/dsapre/04081/>

思路：

代码

```
1  # for a generic_tree,we make its left son as its left son
2  # and make its right brother its right son, then a generic
3  # tree is shifted to a binary tree.
4  '''
5      0                      0
6      / | \                /
7      1 2 3      ==>      1
8      / \                \
9      4  5                2
10                       / \
11                      4  3
12                       \
13                      5
14  '''
15  class TreeNode:
16      def __init__(self,value):
17          self.value = value
18          self.children=[]
19          self.parent=None
20          self.bro=None
21          self.left=None
22          self.right=None
23      def get_max_H(self):
24          if not self.children:return 1
25          return max(child.get_max_H() for child in self.children)+1
26      def get_new_max_h(self):
27          if self.left==None and self.right==None:
28              return 1
29          if self.left==None and self.right:
30              return self.right.get_new_max_h()+1
31          if self.right==None and self.left:
32              return self.left.get_new_max_h()+1
33          if self.left and self.right:
```

```

34         return
35     max(self.left.get_new_max_h(),self.right.get_new_max_h()+1
36
37 num=0
38 def build_generic_tree(l:list,current_node):
39     global num
40     if len(l)==0:
41         return
42     if l[0]=="d":
43         num += 1
44         node = TreeNode(num)
45         if current_node.children:
46             current_node.children[-1].bro=node
47             current_node.children.append(node)
48             node.parent=current_node
49             build_generic_tree(l[1:],node)
50         if l[0]=='u':
51             build_generic_tree(l[1:],current_node.parent)
52     return current_node
53
54 def build_new_tree(root,last_root,is_left:bool):
55     new_root=TreeNode(root.value)
56     if last_root:
57         if is_left:last_root.left=new_root
58         else:last_root.right=new_root
59
60     if root.children:
61         build_new_tree(root.children[0],new_root,True)
62     if root.bro:
63         build_new_tree(root.bro,new_root,False)
64     return new_root
65
66 root=TreeNode(0)
67 build_generic_tree(list(input()),root)
68 new_root=build_new_tree(root,None,False)
69 print('{} => {}'.format(root.get_max_h()-1,new_root.get_new_max_h()-1))

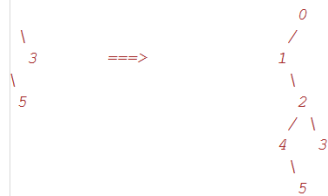
```

代码运行截图 (至少包含有"Accepted")

状态: **Accepted**

源代码

a generic_tree, we make its left son as its left son
make its right brother its right son, then a generic
e is shifted to a binary tree.



```
TreeNode:
def __init__(self, value):
    self.value = value
    self.children = []
    self.parent = None
    self.bro = None
    self.left = None
    self.right = None
```

基本信息

#: 44384037

题目: 04081

提交人: 23n2300011119 (武)

内存: 3744kB

时间: 30ms

语言: Python3

提交时间: 2024-03-24 17:44:39

08581: 扩展二叉树

<http://cs101.openjudge.cn/dsapre/08581/>

思路:

涉及一步循环地找空祖先节点。

代码

```
1 from collections import deque
2 class TreeNode:
3     def __init__(self, val):
4         self.val = val
5         self.left = None
6         self.right = None
7         self.parent = None
8
9 def fill_tree(root, node):
10     node.parent = root
11     if root.left == None: root.left = node
12     else: root.right = node
13
14 # 函数的功能是建成以root为根的树，并把他连接到parent节点上。
15 def build_tree(d: deque, parent: TreeNode):
16     if not d: return
17     cur_node = TreeNode(d.popleft())
18
19     while parent.left and parent.right:
20         parent = parent.parent
21     cur_node.parent = parent
22     fill_tree(parent, cur_node)
23
24     if cur_node.val == '.':
25         build_tree(d, parent)
26     else: build_tree(d, cur_node)
```

```

27
28 def post_search(root):
29     if root.val == '.':
30         return ""
31     output=[]
32     output.extend(post_search(root.left))
33     output.extend(post_search(root.right))
34     output.append(root.val)
35     return "".join(output)
36
37 def in_search(root):
38     if root.val=='.':
39         return ""
40     output=[]
41     output.extend(in_search(root.left))
42     output.append(root.val)
43     output.extend(in_search(root.right))
44     return "".join(output)
45
46 raw=deque(input())
47 root=TreeNode(raw.popleft())
48 build_tree(raw,root)
49 print(in_search(root))
50 print(post_search(root))

```

代码运行截图 (至少包含有"Accepted")

#44698772提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```

from collections import deque
class TreeNode:
    def __init__(self, val):
        self.val=val
        self.left=None
        self.right=None
        self.parent=None

def fill_tree(root,node):
    node.parent=root
    if root.left==None:root.left=node
    else:root.right=node

# 函数的功能是建成以root为根的树, 并把他连接到parent节点上。
def build_tree(d:deque,parent:TreeNode):
    if not d:return
    cur_node=TreeNode(d.popleft())

    while parent.left and parent.right:
        parent=parent.parent
        cur_node.parent=parent
        fill_tree(parent,cur_node)

    if cur_node.val=='.':
        build_tree(d,parent)

```

基本信息

#: 44698772
 题目: 08581
 提交人: 23n2300011119 (武)
 内存: 3712kB
 时间: 29ms
 语言: Python3
 提交时间: 2024-04-18 20:07:32

22067: 快速堆猪

<http://cs101.openjudge.cn/practice/22067/>

思路:

懒删除堆实现。

代码

```
1 import heapq
2 from collections import defaultdict
3 p_stack,p_heap,is_out=[],[],defaultdict(int)
4 while True:
5     try:
6         tmp=input()
7         if tmp=="min":
8             if p_stack:
9                 while True:
10                     a=heapq.heappop(p_heap)
11                     if not is_out[a]:
12                         heapq.heappush(p_heap,a)
13                         print(a)
14                         break
15                     else:
16                         is_out[a]-=1
17                         continue
18             elif tmp=="pop":
19                 if p_stack:
20                     is_out[p_stack.pop()]+=1
21             else:
22                 _,num=map(str,tmp.split())
23                 p_stack.append(int(num))
24                 heapq.heappush(p_heap,int(num))
25     except EOFError:
26         break
```

代码运行截图 (AC代码截图, 至少包含有"Accepted")

#44097967提交状态

[查看](#) [提交](#) [统计](#) [提问](#)

状态: **Accepted**

源代码

```
import heapq
from collections import defaultdict
p_stack,p_heap,is_out=[],[],defaultdict(int)
while True:
    try:
        tmp=input()
        if tmp=="min":
            if p_stack:
                while True:
                    a=heapq.heappop(p_heap)
                    if not is_out[a]:
                        heapq.heappush(p_heap,a)
                        print(a)
                        break
                    else:
                        is_out[a]-=1
                        continue
            elif tmp=="pop":
                if p_stack:
                    is_out[p_stack.pop()]+=1
            else:
                _,num=map(str,tmp.split())
                p_stack.append(int(num))
                heapq.heappush(p_heap,int(num))
    except EOFError:
        break
```

基本信息

#: 44097967
题目: 22067
提交人: 23n2300011119 (武)
内存: 7644kB
时间: 366ms
语言: Python3
提交时间: 2024-03-06 21:42:26

04123: 马走日

dfs, <http://cs101.openjudge.cn/practice/04123>

思路:

写DFS的时候务必在写代码之前明确几件事情:

1. 函数的作用
2. 需要传入几个参数
3. 哪些是可以global的, 哪些是传入的
4. 参数的意义。如这里的path, 我的定义是:

对某一层dfs, path是计算当前起点以后总共的访问路径数。

那么返回时返回条件就应该是 $path == X * Y$, 因为当前起点已经被计算进来了。

dfs的作用是从起点开始遍历所有尚未遍历的点。

在这一定义下, 初始时必须把起点visited标记为True, 于实际意义相对应。

代码

```
1  d=[(-1,-2),(-2,-1),(1,-2),(2,-1),(2,1),(1,2),(-1,2),(-2,1)]
2  res=0
3  def dfs(X,Y,path,start):
4      global res
5      if path==X*Y:
6          res+=1
7          return
8
9      x, y = start
10     for dx,dy in d:
11         if 0<=x+dx<X and 0<=y+dy<Y and not visited[x+dx][y+dy]:
12             visited[x+dx][y+dy]=True
13             dfs(X,Y,path+1,(x+dx,y+dy))
14             visited[x+dx][y+dy]=False
15
16     for i in range(T:=int(input())):
17         X,Y,x,y=map(int,input().split())
18         res=0
19         visited=[[False for j in range(Y)] for i in range(X)]
20         visited[x][y]=True
21         dfs(X,Y,1,(x,y))
22     print(res)
```

代码运行截图 (AC代码截图, 至少包含有"Accepted")

状态: Accepted

源代码

```
d=[(-1,-2),(-2,-1),(1,-2),(2,-1),(2,1),(1,2),(-1,2),(-2,1)]
res=0
def dfs(X,Y,path,start):
    global res
    if path==X*Y:
        res+=1
        return

    x,y=start
    for dx,dy in d:
        if 0<=x+dx<X and 0<=y+dy<Y and not visited[x+dx][y+dy]:
            visited[x+dx][y+dy]=True
            dfs(X,Y,path+1,(x+dx,y+dy))
            visited[x+dx][y+dy]=False

for i in range(T:=int(input())):
    X,Y,x,y=map(int,input().split())
    res=0
    visited=[[False for j in range(Y)] for i in range(X)]
    visited[x][y]=True
    dfs(X,Y,1,(x,y))
    print(res)
```

基本信息

#: 44700558
题目: 04123
提交人: 23n2300011119 (武)
内存: 4028kB
时间: 2812ms
语言: Python3
提交时间: 2024-04-18 22:27:47

28046: 词梯

bfs, <http://cs101.openjudge.cn/practice/28046/>

思路: bucket, BFS

代码

```
1 from collections import deque
2 class Vertex:
3     def __init__(self,id):
4         self.id=id
5         self.neighbors={}
6         self.previous=None
7         self.color='white'
8     def __str__(self):
9         return '*' + self.id
10 class Graph:
11     def __init__(self):
12         self.vertices={}
13         self.num_vertices=0
14
15     def add_vertex(self,id):
16         self.vertices[id]=Vertex(id)
17         self.num_vertices+=1
18
19     def add_edge(self,v1_id,v2_id):
20         # v1_start,v2_end
21         if v1_id not in self.vertices:
22             self.vertices[v1_id]=Vertex(v1_id)
23         if v2_id not in self.vertices:
24             self.vertices[v2_id]=Vertex(v2_id)
25         v1,v2=self.vertices[v1_id],self.vertices[v2_id]
26         v1.neighbors[v2_id]=v2
27         self.num_vertices+=1
28
29 n,graph,buckets=int(input()),Graph(),{}
```

```

30 words=[input() for _ in range(n)]
31 for word in words:
32     for bit in range(1,len(word)+1):
33         tag=word[:bit-1]+'_'+word[bit:]
34         bucket=buckets.setdefault(tag,set())
35         bucket.add(word)
36 # for i,j in buckets.items():
37 #     print(i,j)
38 for bucket in buckets.values():
39     for i in bucket:
40         tmp=bucket-{i}
41         for j in tmp:
42             graph.add_edge(i,j)
43
44
45 start,goal=map(str,input().split())
46 # BFS,这里不用函数实现
47 q=deque()
48 q.append(graph.vertices[start])
49 current=graph.vertices[start]
50 # 注: 标黑色用于把回头路堵死; 标灰色用于把更长的可行路径堵死。
51 # 由于更长的可行路径被堵死且最短路径唯一, 所以每个点的前驱若有有则仅有一个。
52 while q and current.id!=goal:
53     current=q.popleft()
54     for vert in current.neighbors.values():
55         if vert.color=='white':
56             vert.color='grey'
57             vert.previous=current
58             q.append(vert)
59     current.color='black'
60
61 def traverse(start:Vertex):
62     output=[start.id]
63     current=start
64     while current.previous:
65         output.append(current.previous.id)
66         current=current.previous
67     return " ".join(output[::-1])
68
69 if current.id==goal:
70     print(traverse(graph.vertices[goal]))
71 else:
72     print("NO")

```

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
class Vertex:
    def __init__(self, id):
        self.id=id
        self.neighbors={}
        self.previous=None
        self.color='white'
    def __str__(self):
        return '*' +self.id
class Graph:
    def __init__(self):
        self.vertices={}
        self.num_vertices=0
    def add_vertex(self, id):
        self.vertices[id]=Vertex(id)
        self.num_vertices+=1
    def add_edge(self, v1_id, v2_id):
        # v1_start, v2_end
        if v1_id not in self.vertices:
```

基本信息

#: 44718761
题目: 28046
提交人: 23n2300011119 (武)
内存: 9444kB
时间: 87ms
语言: Python3
提交时间: 2024-04-20 14:34:05

28050: 骑士周游

dfs, <http://cs101.openjudge.cn/practice/28050/>

思路:

启发式算法+马走日DFS

代码

```
1 d=[(-1,-2),(-2,-1),(1,-2),(2,-1),(2,1),(1,2),(-1,2),(-2,1)]
2 def avail(vert):
3     x,y=vert
4     return (0 <=x<X and 0<=y<Y and not visited[x][y])
5
6 def ordered_by_avail(start):
7     x,y=start
8     steps=[]
9     for dx,dy in d:
10         next_step=(x+dx,x+dy)
11         available=0
12         for step in d:
13             ddx,ddy=step
14             if avail((x+dx+ddx,y+dy+ddy)):available+=1
15         steps.append((available,(dx,dy)))
16     steps.sort(key=lambda x:x[0])
17     return [i[1] for i in steps]
18
19 def dfs(X,Y,path,start):
20     if path==X*Y:
21         print('success')
22         exit()
23
24     x, y = start
25     new_d=ordered_by_avail(start)
26     for dx,dy in new_d:
27         if avail((x+dx,y+dy)):
28             visited[x+dx][y+dy]=True
```

```

29         dfs(X,Y,path+1,(x+dx,y+dy))
30         visited[x+dx][y+dy]=False
31
32 X=Y=int(input())
33 x,y=map(int,input().split())
34 visited=[[False for j in range(Y)] for i in range(X)]
35 visited[x][y]=True
36 dfs(X,Y,1,(x,y))
37 print('fail')

```

代码运行截图 (AC代码截图, 至少包含有"Accepted")

#44719775提交状态

[查看](#) [提交](#) [统计](#) [提问](#)

状态: Accepted

源代码

```

d=[(-1,-2),(-2,-1),(1,-2),(2,-1),(2,1),(1,2),(-1,2),(-2,1)]
def avail(vert):
    x,y=vert
    return (0 <=x<X and 0<=y<Y and not visited[x][y])

def ordered_by_avail(start):
    x,y=start
    steps=[]
    for dx,dy in d:
        next_step=(x+dx,y+dy)
        available=0
        for step in d:
            ddx,ddy=step
            if avail((x+dx+ddx,y+dy+ddy)):available+=1
        steps.append((available,(dx,dy)))
    steps.sort(key=lambda x:x[0])
    return [i[1] for i in steps]

def dfs(X,Y,path,start):
    if path==X*Y:
        print('success')
        exit()

    x, y = start
    new_d=ordered_by_avail(start)
    for dx,dy in new_d:
        if avail((x+dx,y+dy)):
            visited[x+dx][y+dy]=True
            dfs(X,Y,path+1,(x+dx,y+dy))
            visited[x+dx][y+dy]=False

X=Y=int(input())
x,y=map(int,input().split())
visited=[[False for j in range(Y)] for i in range(X)]
visited[x][y]=True
dfs(X,Y,1,(x,y))
print('fail')

```

基本信息

#: 44719775
 题目: 28050
 提交人: 23n2300011119 (武)
 内存: 4044kB
 时间: 81ms
 语言: Python3
 提交时间: 2024-04-20 15:12:29

2. 学习总结和收获

当图变得越来越抽象(指从计概的矩阵变成点和边的集合), 过程写得越来越规范, 实现的复杂度越来越低, 但是随之而来的是代码量越来越大.....

本周题目难度不小, 前面的题比较熟悉, 暂且不提。

两个图算法的题目, 词梯巧妙在用桶的方法快速建图(夏佬狂喜), 骑士周游巧妙在利用启发式算法大幅降低时间成本。

学习下来, 能感受到自己变抽象了, 也变强了。