Assignment #9: 图论：遍历，及 树算

Updated 1739 GMT+8 Apr 14, 2024

2024 spring, Complied 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/

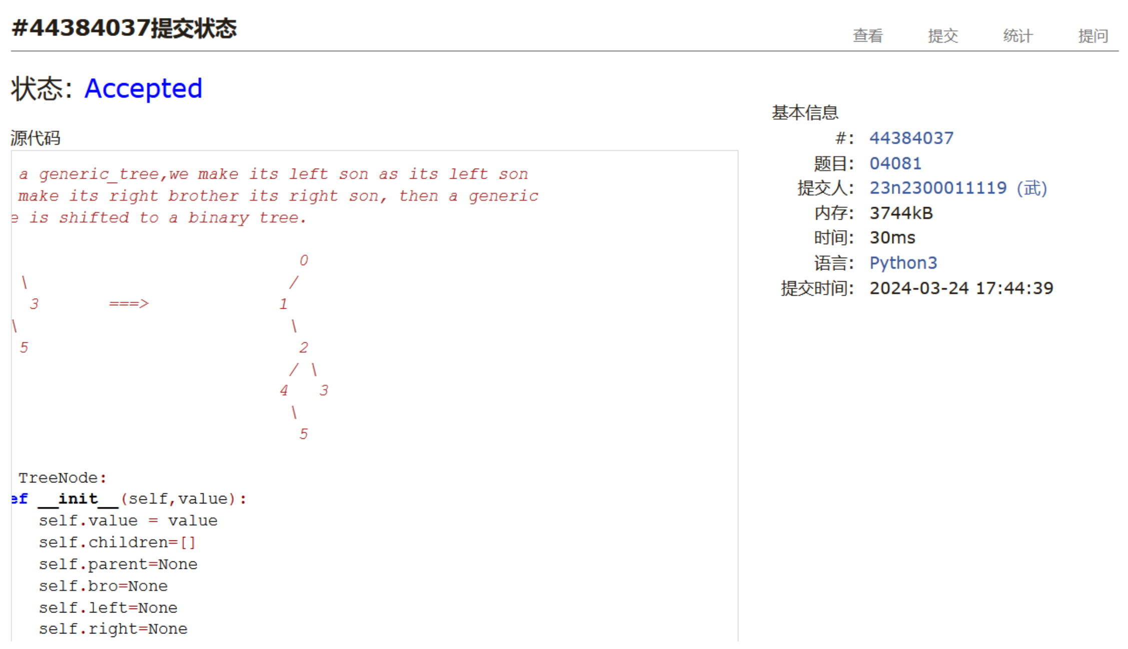
思路：

代码

```
# for a generic_tree,we make its left son as its left son
# and make its right brother its right son, then a generic
# tree is shifted to a binary tree.
'''
    0                             0
  / | \                          /
 1  2  3       ===>             1
   / \                           \
  4   5                           2
                                 / \
                                4   3
                                 \
                                  5
'''
class TreeNode:
    def __init__(self,value):
        self.value = value
        self.children=[]
        self.parent=None
        self.bro=None
        self.left=None
        self.right=None
    def get_max_H(self):
        if not self.children:return 1
        return max(child.get_max_H() for child in self.children)+1
    def get_new_max_h(self):
        if self.left==None and self.right==None:
            return 1
        if self.left==None and self.right:
            return self.right.get_new_max_h()+1
        if self.right==None and self.left:
            return self.left.get_new_max_h()+1
        if self.left and self.right:
```

```python
            return max(self.left.get_new_max_h(),self.right.get_new_max_h())+1

num=0
def build_generic_tree(l:list,current_node):
    global num
    if len(l)==0:
        return
    if l[0]=="d":
        num += 1
        node = TreeNode(num)
        if current_node.children:
            current_node.children[-1].bro=node
        current_node.children.append(node)
        node.parent=current_node
        build_generic_tree(l[1:],node)
    if l[0]=='u':
        build_generic_tree(l[1:],current_node.parent)
    return current_node

def build_new_tree(root,last_root,is_left:bool):
    new_root=TreeNode(root.value)
    if last_root:
        if is_left:last_root.left=new_root
        else:last_root.right=new_root

    if root.children:
        build_new_tree(root.children[0],new_root,True)
    if root.bro:
        build_new_tree(root.bro,new_root,False)
    return new_root


root=TreeNode(0)
build_generic_tree(list(input()),root)
new_root=build_new_tree(root,None,False)
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"）



#44384037提交状态

查看　提交　统计　提问

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.

                                        0
                                       /
 \                                    1
  3        ===>                        \
\                                       2
 5                                     / \
                                      4   3
                                       \
                                        5

 TreeNode:
ef __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:

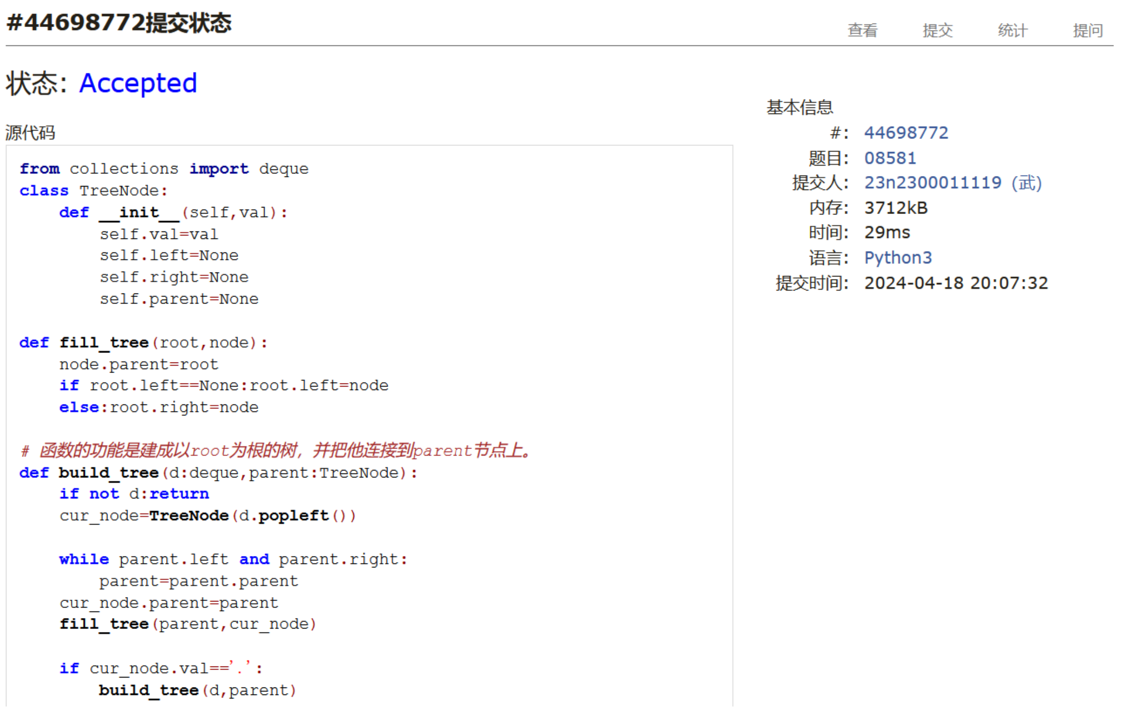
涉及一步循环地找空祖先节点。

代码

```python
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)
    else:build_tree(d,cur_node)
```

```python

def post_search(root):
    if root.val == '.':
        return ""
    output=[]
    output.extend(post_search(root.left))
    output.extend(post_search(root.right))
    output.append(root.val)
    return "".join(output)

def in_search(root):
    if root.val=='.':
        return ""
    output=[]
    output.extend(in_search(root.left))
    output.append(root.val)
    output.extend(in_search(root.right))
    return "".join(output)

raw=deque(input())
root=TreeNode(raw.popleft())
build_tree(raw,root)
print(in_search(root))
print(post_search(root))
```

代码运行截图 （至少包含有"Accepted"）



#44698772提交状态

状态: Accepted

源代码

```python
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/

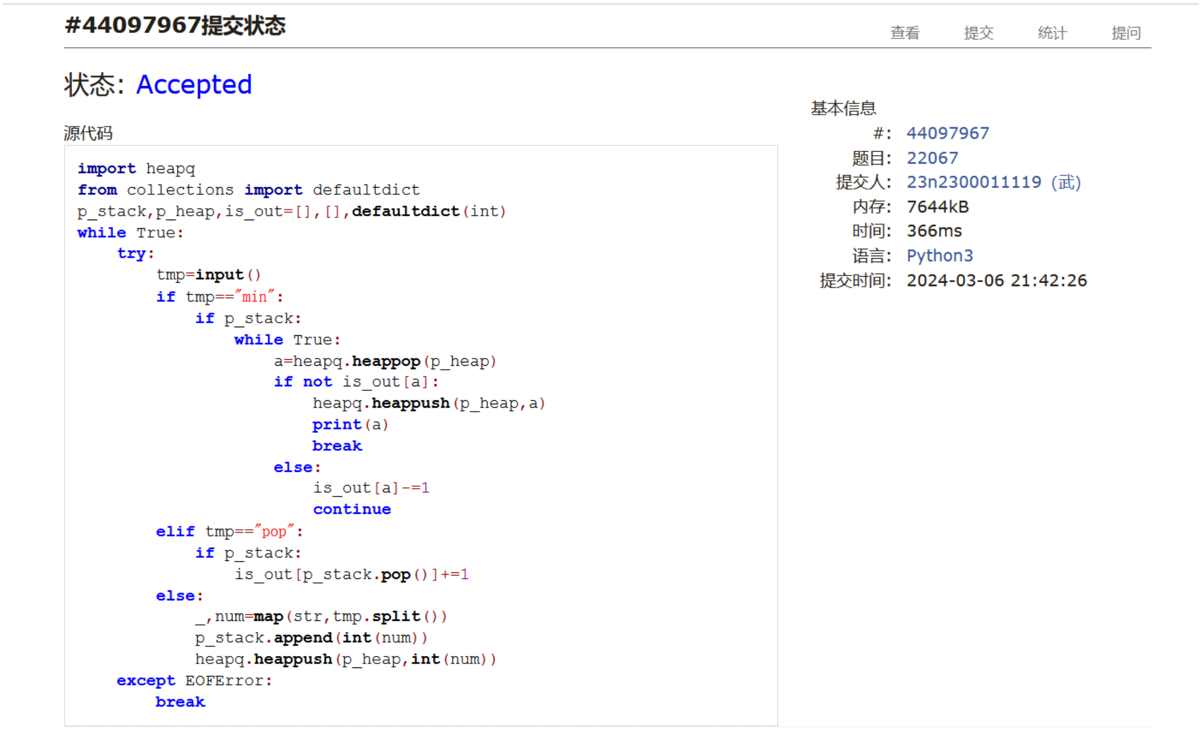
思路：

懒删除堆实现。

代码

```python
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
```

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

```python
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。

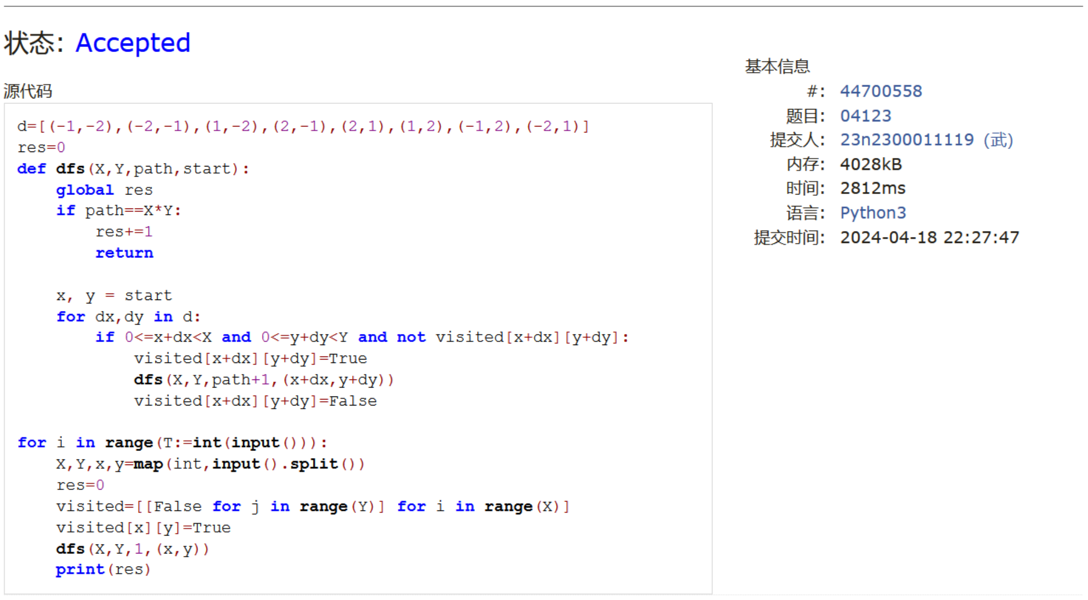
   dfs的作用是从起点开始遍历所有尚未遍历的点。

   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

```python
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)
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）


状态: Accepted

源代码

```python
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/

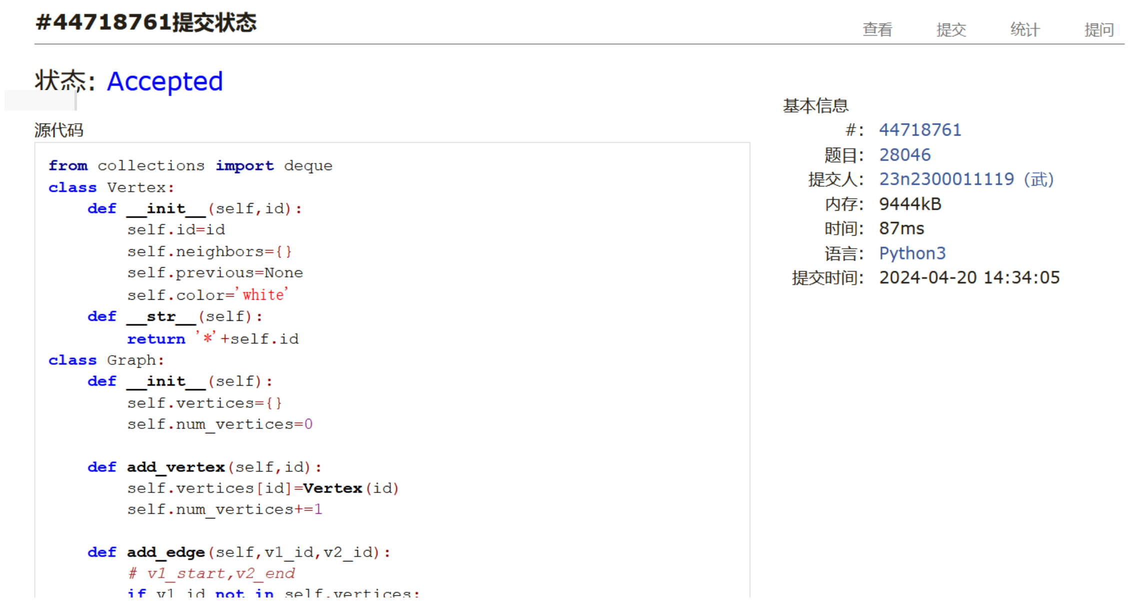
思路：bucket，BFS

代码

```python
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:
            self.vertices[v1_id]=Vertex(v1_id)
        if v2_id not in self.vertices:
            self.vertices[v2_id]=Vertex(v2_id)
        v1,v2=self.vertices[v1_id],self.vertices[v2_id]
        v1.neighbors[v2_id]=v2
        self.num_vertices+=1

n,graph,buckets=int(input()),Graph(),{}
```

```python
words=[input() for _ in range(n)]
for word in words:
    for bit in range(1,len(word)+1):
        tag=word[:bit-1]+'_'+word[bit:]
        bucket=buckets.setdefault(tag,set())
        bucket.add(word)
# for i,j in buckets.items():
#     print(i,j)
for bucket in buckets.values():
    for i in bucket:
        tmp=bucket-{i}
        for j in tmp:
            graph.add_edge(i,j)


start,goal=map(str,input().split())
# BFS,这里不用函数实现
q=deque()
q.append(graph.vertices[start])
current=graph.vertices[start]
# 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。
# 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。
while q and current.id!=goal:
    current=q.popleft()
    for vert in current.neighbors.values():
        if vert.color=='white':
            vert.color='grey'
            vert.previous=current
            q.append(vert)
    current.color='black'

def traverse(start:Vertex):
    output=[start.id]
    current=start
    while current.previous:
        output.append(current.previous.id)
        current=current.previous
    return " ".join(output[::-1])

if current.id==goal:
    print(traverse(graph.vertices[goal]))
else:
    print("NO")
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）


#44718761提交状态　　查看　提交　统计　提问

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/

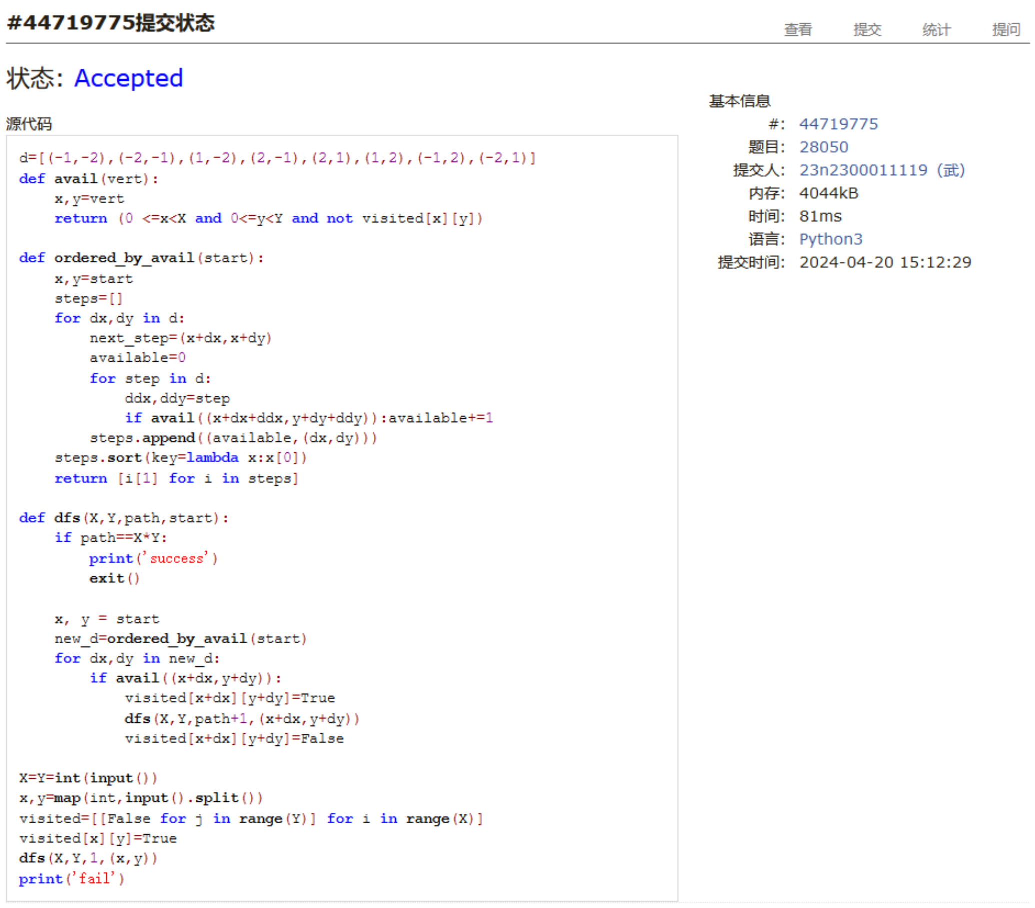
思路:

启发式算法+马走日DFS

代码

```python
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,x+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')
```

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

```python
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,x+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。