Assignment #9: 图论：遍历，及 树算

Updated 1739 GMT+8 Apr 14, 2024

2024 spring, Complied by 王业成 生命科学学院

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

==（请改为同学的操作系统、编程环境等）==

操作系统：macOS Ventura 13.4.1 (c)

Python编程环境：Spyder IDE 5.2.2, PyCharm 2023.1.4 (Professional Edition)

C/C++编程环境：Mac terminal vi (version 9.0.1424), g++/gcc (Apple clang version 14.0.3, clang-1403.0.22.14.1)

1. 题目

04081: 树的转换

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

思路：自己的代码挑了半天一直不对，最后发现还是思路有问题，应该重点找在上一层而不是根据下一层判断是否-1

存一个左儿子，存一个右兄弟，找最大

代码

```
# class TreeNode:
    def __init__(self):
        self.children = []
        self.first_child = None
        self.next_sib = None
def build(seq):
    root = TreeNode()
    stack = [root]
```

```
    depth = 0
    for act in seq:
        cur_node = stack[-1]
        if act == 'd':
            new_node = TreeNode()
            if not cur_node.children:
                cur_node.first_child = new_node
            else:
                cur_node.children[-1].next_sib = new_node
            cur_node.children.append(new_node)
            stack.append(new_node)
            depth = max(depth, len(stack) - 1)
        else:
            stack.pop()
    return root, depth
def cal_h_bin(node):
    if not node:
         return -1
    return max(cal_h_bin(node.first_child), cal_h_bin(node.next_sib)) + 1
seq = input()
root, h_orig = build(seq)
h_bin = cal_h_bin(root)
print(f'{h_orig} => {h_bin}')
```

代码运行截图 ==（至少包含有"Accepted"）==

状态: Accepted

源代码

```
class TreeNode:
    def __init__(self):
        self.children = []
        self.first_child = None
        self.next_sib = None
def build(seq):
    root = TreeNode()
    stack = [root]
    depth = 0
    for act in seq:
        cur_node = stack[-1]
        if act == 'd':
            new_node = TreeNode()
            if not cur_node.children:
                cur_node.first_child = new_node
            else:
                cur_node.children[-1].next_sib = new_node
            cur_node.children.append(new_node)
            stack.append(new_node)
            depth = max(depth, len(stack) - 1)
        else:
            stack.pop()
    return root, depth
def cal_h_bin(node):
    if not node:
         return -1
    return max(cal_h_bin(node.first_child), cal_h_bin(node.next_sib)) +
seq = input()
root, h_orig = build(seq)
h_bin = cal_h_bin(root)
print(f'{h_orig} => {h_bin}')
```

基本信息

#:	44742668
题目:	04081
提交人:	wangyecheng
内存:	3676kB
时间:	29ms
语言:	Python3
提交时间:	2024-04-21 17:55:23

08581: 扩展二叉树

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

思路：建树模拟，感觉用“.”填充之后好像跟容易建了

代码

```
# class Treenode():
    def __init__(self,value):
        self.value=value
        self.left=None
        self.right=None
def buildtree(lst):
    if not lst:
        return None
    value=lst.pop(0)
    if value==".":
        return None
    else:
        node=Treenode(value)
        node.left=buildtree(lst)
        node.right=buildtree(lst)
    return node
def inorder(root):
    if not root:
        return []
    else:
        left=inorder(root.left)
        right=inorder(root.right)
        return left+[root.value]+right

def postorder(root):
    if not root:
        return []
    else:
        left = postorder(root.left)
        right = postorder(root.right)
        return left+right+[root.value]
lst=list(input())
root=buildtree(lst)
a= inorder(root)
b= postorder(root)
print("".join(a))
print("".join(b))
```

代码运行截图 ==（至少包含有"Accepted"）==

源代码

```
class Treenode():
    def __init__(self,value):
        self.value=value
        self.left=None
        self.right=None
def buildtree(lst):
    if not lst:
        return None
    value=lst.pop(0)
    if value=="#":
        return None
    else:
        node=Treenode(value)
        node.left=buildtree(lst)
        node.right=buildtree(lst)
    return node
def inorder(root):
    if not root:
        return []
    else:
        left=inorder(root.left)
        right=inorder(root.right)
        return left+[root.value]+right

def postorder(root):
    if not root:
        return []
    else:
        left = postorder(root.left)
        right = postorder(root.right)
        return left+right+[root.value]
lst=list(input())
root=buildtree(lst)
a= inorder(root)
b= postorder(root)
print("".join(a))
print("".join(b))
```

#: 44747442
题目: 08581
提交人: wangyecheng
内存: 3664kB
时间: 26ms
语言: Python3
提交时间: 2024-04-21 22:13:08

22067: 快速堆猪

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

思路：本来想用heap堆的懒删除做的，结果看到题解有一个绝妙的双栈做法，感觉有点类似于并查集的思路把，light栈一直添加最小元素，相当于把他做一个代表，学到了

代码

```
# stack=[]
light=[]
while True:
    try:
        s=input().split()
        if s[0]=="pop":
            if stack:
                stack.pop()
                light.pop()
        elif s[0]=="min":
            if light:
                print(light[-1])
        else:
            value=int(s[1])
            stack.append(value)
            if light:
```

```
                light.append(min(light[-1],value))
            else:
                light.append(value)
    except EOFError:
        break
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

源代码

```
stack=[]
light=[]
while True:
    try:
        s=input().split()
        if s[0]=="pop":
            if stack:
                stack.pop()
                light.pop()
        elif s[0]=="min":
            if light:
                print(light[-1])
        else:
            value=int(s[1])
            stack.append(value)
            if light:
                light.append(min(light[-1],value))
            else:
                light.append(value)
    except EOFError:
        break
```

#: 44747732
题目: 22067
提交人: wangyecheng
内存: 6672kB
时间: 313ms
语言: Python3
提交时间: 2024-04-21 22:38:

04123: 马走日

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

思路：dfs回溯

代码

```
# num=0
def dfs(chess,x,y,dirs,step):
    global num
    n=len(chess)
    m=len(chess[0])
    if step==n*m:
        num+=1
        return
    chess[x][y]=1
    for dx,dy in dirs:
        x1,y1=x+dx,y+dy
        if 0<=x1<n and 0<=y1<m and chess[x1][y1]==0:
            dfs(chess,x1,y1,dirs,step+1)
            chess[x1][y1]=0
s=int(input())
dirs=[(1,-2),(1,2),(-1,2),(-1,-2),(2,1),(2,-1),(-2,1),(-2,-1)]
```

```
for _ in range(s):
    num=0
    n,m,x,y=map(int,input().split())
    chess=[[0]*m for _ in range(n)]
    dfs(chess,x,y,dirs,1)
    print(num)
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

状态: Accepted

源代码

```
num=0
def dfs(chess,x,y,dirs,step):
    global num
    n=len(chess)
    m=len(chess[0])
    if step==n*m:
        num+=1
        return
    chess[x][y]=1
    for dx,dy in dirs:
        x1,y1=x+dx,y+dy
        if 0<=x1<n and 0<=y1<m and chess[x1][y1]==0:
            dfs(chess,x1,y1,dirs,step+1)
            chess[x1][y1]=0
s=int(input())
dirs=[(1,-2),(1,2),(-1,2),(-1,-2),(2,1),(2,-1),(-2,1),(-2,-1)]
for _ in range(s):
    num=0
    n,m,x,y=map(int,input().split())
    chess=[[0]*m for _ in range(n)]
    dfs(chess,x,y,dirs,1)
    print(num)
```

基本信息

#:	44747984
题目:	04123
提交人:	wangyecheng
内存:	3680kB
时间:	2846ms
语言:	Python3
提交时间:	2024-04-21 23:11:02

28046: 词梯

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

思路：本来想写一个不用图的代码的，但不是超内存就是超时，还是老老实实用题解中图的代码写吧

代码

```
# import sys
from collections import deque
class Graph:
    def __init__(self):
        self.vertices = {}
        self.num_vertices = 0
    def add_vertex(self, key):
        self.num_vertices = self.num_vertices + 1
        new_vertex = Vertex(key)
        self.vertices[key] = new_vertex
        return new_vertex
    def get_vertex(self, n):
        if n in self.vertices:
```

```
            return self.vertices[n]
        else:
            return None
    def __len__(self):
        return self.num_vertices
    def __contains__(self, n):
        return n in self.vertices
    def add_edge(self, f, t, cost=0):
        if f not in self.vertices:
            nv = self.add_vertex(f)
        if t not in self.vertices:
            nv = self.add_vertex(t)
        self.vertices[f].add_neighbor(self.vertices[t], cost)
    def get_vertices(self):
        return list(self.vertices.keys())
    def __iter__(self):
        return iter(self.vertices.values())
class Vertex:
    def __init__(self, num):
        self.key = num
        self.connectedTo = {}
        self.color = 'white'
        self.distance = sys.maxsize
        self.previous = None
        self.disc = 0
        self.fin = 0
    def add_neighbor(self, nbr, weight=0):
        self.connectedTo[nbr] = weight
    def get_neighbors(self):
        return self.connectedTo.keys()
def build_graph(all_words):
    buckets = {}
    the_graph = Graph()
    for line in all_words:
        word = line.strip()
        for i, _ in enumerate(word):
            bucket = f"{word[:i]}_{word[i + 1:]}"
            buckets.setdefault(bucket, set()).add(word)
    for similar_words in buckets.values():
        for word1 in similar_words:
            for word2 in similar_words - {word1}:
                the_graph.add_edge(word1, word2)
    return the_graph
def bfs(start, end):
    start.distnce = 0
    start.previous = None
    vert_queue = deque()
    vert_queue.append(start)
    while len(vert_queue) > 0:
        current = vert_queue.popleft()
        if current == end:
            return True
        for neighbor in current.get_neighbors():
            if neighbor.color == "white":
                neighbor.color = "gray"
                neighbor.distance = current.distance + 1
```

```
                neighbor.previous = current
                vert_queue.append(neighbor)
        current.color = "black"
    return False
def traverse(starting_vertex):
    ans = []
    current = starting_vertex
    while (current.previous):
        ans.append(current.key)
        current = current.previous
    ans.append(current.key)

    return ans
n = int(input())
all_words = []
for _ in range(n):
    all_words.append(input().strip())
g = build_graph(all_words)
s, e = input().split()
start, end = g.get_vertex(s), g.get_vertex(e)
if start is None or end is None:
    print('NO')
    exit(0)
if bfs(start, end):
    ans = traverse(end)
    print(' '.join(ans[::-1]))
else:
    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

```
import sys
from collections import deque
class Graph:
    def __init__(self):
        self.vertices = {}
        self.num_vertices = 0
    def add_vertex(self, key):
        self.num_vertices = self.num_vertices + 1
        new_vertex = Vertex(key)
        self.vertices[key] = new_vertex
        return new_vertex
    def get_vertex(self, n):
        if n in self.vertices:
            return self.vertices[n]
        else:
            return None
    def __len__(self):
        return self.num_vertices
    def __contains__(self, n):
        return n in self.vertices
    def add_edge(self, f, t, cost=0):
        if f not in self.vertices:
            nv = self.add_vertex(f)
        if t not in self.vertices:
            nv = self.add_vertex(t)
        self.vertices[f].add_neighbor(self.vertices[t], cost)
    def get_vertices(self):
        return list(self.vertices.keys())
    def __iter__(self):
        return iter(self.vertices.values())
class Vertex:
    def __init__(self, num):
        self.key = num
```

基本信息

#:	44750192
题目:	28046
提交人:	wangyecheng
内存:	9508kB
时间:	82ms
语言:	Python3
提交时间:	2024-04-22 11:43:06

28050: 骑士周游

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

思路:

代码

```
# import sys
class Graph:
    def __init__(self):
        self.vertices = {}
        self.num_vertices = 0
    def add_vertex(self, key):
        self.num_vertices = self.num_vertices + 1
        new_vertex = Vertex(key)
        self.vertices[key] = new_vertex
        return new_vertex
    def get_vertex(self, n):
        if n in self.vertices:
            return self.vertices[n]
        else:
            return None
    def __len__(self):
        return self.num_vertices
    def __contains__(self, n):
        return n in self.vertices
    def add_edge(self, f, t, cost=0):
        if f not in self.vertices:
            nv = self.add_vertex(f)
        if t not in self.vertices:
            nv = self.add_vertex(t)
        self.vertices[f].add_neighbor(self.vertices[t], cost)
    def getVertices(self):
        return list(self.vertices.keys())
    def __iter__(self):
        return iter(self.vertices.values())
class Vertex:
    def __init__(self, num):
        self.key = num
        self.connectedTo = {}
        self.color = 'white'
        self.distance = sys.maxsize
        self.previous = None
        self.disc = 0
        self.fin = 0
    def __lt__(self,o):
        return self.key < o.key
    def add_neighbor(self, nbr, weight=0):
        self.connectedTo[nbr] = weight
    def get_neighbors(self):
```

```
        return self.connectedTo.keys()
    def __str__(self):
        return str(self.key) + ":color " + self.color + ":disc " + str(self.disc)
+ ":fin " + str(
            self.fin) + ":dist " + str(self.distance) + ":pred \n\t[" +
str(self.previous) + "]\n"
def knight_graph(board_size):
    kt_graph = Graph()
    for row in range(board_size):
        for col in range(board_size):
            node_id = pos_to_node_id(row, col, board_size)
            new_positions = gen_legal_moves(row, col, board_size)
            for row2, col2 in new_positions:
                other_node_id = pos_to_node_id(row2, col2, board_size)
                kt_graph.add_edge(node_id, other_node_id)
    return kt_graph
def pos_to_node_id(x, y, bdSize):
    return x * bdSize + y
def gen_legal_moves(row, col, board_size):
    new_moves = []
    move_offsets = [
        (-1, -2),
        (-1, 2),
        (-2, -1),
        (-2, 1),
        (1, -2),
        (1, 2),
        (2, -1),
        (2, 1),
    ]
    for r_off, c_off in move_offsets:
        if (                                    # #检查，不能走出棋盘
            0 <= row + r_off < board_size
            and 0 <= col + c_off < board_size
        ):
            new_moves.append((row + r_off, col + c_off))
    return new_moves
def knight_tour(n, path, u, limit):
    u.color = "gray"
    path.append(u)
    if n < limit:
        neighbors = ordered_by_avail(u)
        i = 0
        for nbr in neighbors:
            if nbr.color == "white" and \
                knight_tour(n + 1, path, nbr, limit):
                return True
        else:
            path.pop()
            u.color = "white"
            return False
    else:
        return True
def ordered_by_avail(n):
    res_list = []
    for v in n.get_neighbors():
```

```
        if v.color == "white":
            c = 0
            for w in v.get_neighbors():
                if w.color == "white":
                    c += 1
            res_list.append((c,v))
    res_list.sort(key = lambda x: x[0])
    return [y[1] for y in res_list]
def NodeToPos(id):
    return ((id//8, id%8))
bdSize = int(input())  # 棋盘大小
*start_pos, = map(int, input().split())  # 起始位置
g = knight_graph(bdSize)
start_vertex = g.get_vertex(pos_to_node_id(start_pos[0], start_pos[1], bdSize))
if start_vertex is None:
    print("fail")
    exit(0)
tour_path = []
done = knight_tour(0, tour_path, start_vertex, bdSize * bdSize-1)
if done:
    print("success")
else:
    print("fail")

exit(0)
cnt = 0
for vertex in tour_path:
    cnt += 1
    if cnt % bdSize == 0:
        print()
    else:
        print(vertex.key, end=" ")
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

状态：Accepted

源代码

```
import sys
class Graph:
    def __init__(self):
        self.vertices = {}
        self.num_vertices = 0
    def add_vertex(self, key):
        self.num_vertices = self.num_vertices + 1
        new_vertex = Vertex(key)
        self.vertices[key] = new_vertex
        return new_vertex
    def get_vertex(self, n):
        if n in self.vertices:
            return self.vertices[n]
        else:
            return None
    def __len__(self):
        return self.num_vertices
    def __contains__(self, n):
        return n in self.vertices
    def add_edge(self, f, t, cost=0):
        if f not in self.vertices:
            nv = self.add_vertex(f)
        if t not in self.vertices:
            nv = self.add_vertex(t)
        self.vertices[f].add_neighbor(self.vertices[t], cost)
    def getVertices(self):
        return list(self.vertices.keys())
    def __iter__(self):
        return iter(self.vertices.values())
class Vertex:
    def __init__(self, num):
        self.key = num
```

基本信息

#: 44750285
题目: 28050
提交人: wangyecheng
内存: 4060kB
时间: 31ms
语言: Python3
提交时间: 2024-04-22 11:57:24

2. 学习总结和收获

==如果作业题目简单，有否额外练习题目，比如：OJ“2024spring每日选做”、CF、LeetCode、洛谷等网站题目。==

本周作业好难啊，由于之前的dfs和bfs基础比较薄弱，还专门花了一天时间来学习二者，结果发现题目还是不怎么会，好多题目基本都是看着题解一步一步理解，再一步一步写的，最后一题骑士周游还是直接粘了题解的代码慢慢理解的，感觉图的内容还是不够熟练，建图老是少写函数，实在不行机考的时候再cheeting-sheet上抄一份完整地图代码把，sigh！