



Assignment #9: 图论：遍历，及 树算

Updated 1739 GMT+8 Apr 14, 2024

2024 spring, Compiled by 同学的姓名、院系

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：

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"）

#44769104提交状态

[查看](#)

[提交](#)

[统计](#)

[提问](#)

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)) + 1

seq = input()
root, h_orig = build(seq)
h_bin = cal_h_bin(root)
print(f' {h_orig} => {h_bin}')
```

基本信息

#: 44769104

题目: 04081

提交人: 刘子暄

内存: 3664kB

时间: 27ms

语言: Python3

提交时间: 2024-04-23 22:07:39

08581: 扩展二叉树

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

思路：

代码

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

def build_tree(lst):
    if not lst:
        return None

    value = lst.pop()
    if value == '.':
        return None

    root = BinaryTreeNode(value)
    root.left = build_tree(lst)
    root.right = build_tree(lst)

    return root

def inorder(root):
    if not root:
        return []

    left = inorder(root.left)
    right = inorder(root.right)
    return left + [root.value] + right

def postorder(root):
    if not root:
        return []

    left = postorder(root.left)
    right = postorder(root.right)
    return left + right + [root.value]
```

```
lst = list(input())
root = build_tree(lst[::-1])
in_order_result = inorder(root)
post_order_result = postorder(root)
print(''.join(in_order_result))
print(''.join(post_order_result))
```

代码运行截图（至少包含有"Accepted"）



CS101 / 数算pre每日选做

题目

排名

状态

提问

#44769097提交状态

[查看](#)

[提交](#)

[统计](#)

状态: Accepted

源代码

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

def build_tree(lst):
    if not lst:
        return None

    value = lst.pop()
    if value == '.':
        return None

    root = BinaryTreeNode(value)
    root.left = build_tree(lst)
    root.right = build_tree(lst)

    return root

def inorder(root):
    if not root:
        return []

    left = inorder(root.left)
    right = inorder(root.right)
    return left + [root.value] + right

def postorder(root):
    if not root:
        return []
```

基本信息

#: 44769097

题目: 08581

提交人: 刘子喧

内存: 3616kB

时间: 34ms

语言: Python3

提交时间: 2024-04-23 22:06:20

22067: 快速堆猪

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

思路：

代码

```
a = []
m = []

while True:
    try:
        s = input().split()

        if s[0] == "pop":
            if a:
                a.pop()
            if m:
                m.pop()
        elif s[0] == "min":
            if m:
                print(m[-1])
        else:
            h = int(s[1])
            a.append(h)
            if not m:
                m.append(h)
            else:
                k = m[-1]
                m.append(min(k, h))
    except EOFError:
        break
```

代码运行截图（AC代码截图，至少包含有"Accepted"）



#44769125提交状态

[查看](#) [提交](#) [统计](#) [提问](#)

状态: Accepted

源代码

```
a = []
m = []

while True:
    try:
        s = input().split()

        if s[0] == "pop":
            if a:
                a.pop()
            if m:
                m.pop()
        elif s[0] == "min":
            if m:
                print(m[-1])
        else:
            h = int(s[1])
            a.append(h)
            if not m:
                m.append(h)
            else:
                k = m[-1]
                m.append(min(k, h))
    except EOFError:
        break
```

基本信息

#: 44769125
题目: 22067
提交人: 刘子喧
内存: 10376kB
时间: 320ms
语言: Python3
提交时间: 2024-04-23 22:10:09

04123: 马走日

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

思路：

代码

```

maxn = 10;
sx = [-2,-1,1,2, 2, 1,-1,-2]
sy = [ 1, 2,2,1,-1,-2,-2,-1]

ans = 0;

def Dfs(dep: int, x: int, y: int):
    if n*m == dep:
        global ans
        ans += 1
        return

    for r in range(8):
        s = x + sx[r]
        t = y + sy[r]
        if chess[s][t]==False and 0<=s<n and 0<=t<m :
            chess[s][t]=True
            Dfs(dep+1, s, t)
            chess[s][t] = False;

for _ in range(int(input())):
    n,m,x,y = map(int, input().split())
    chess = [[False]*maxn for _ in range(maxn)]
    ans = 0
    chess[x][y] = True
    Dfs(1, x, y)
    print(ans)

```

代码运行截图（AC代码截图，至少包含有"Accepted"）

状态: Accepted

源代码

```
maxn = 10;
sx = [-2,-1,1,2, 2, 1,-1,-2]
sy = [ 1, 2,2,1,-1,-2,-2,-1]

ans = 0;

def Dfs(dep: int, x: int, y: int):
    if n*m == dep:
        global ans
        ans += 1
        return

    for r in range(8):
        s = x + sx[r]
        t = y + sy[r]
        if chess[s][t]==False and 0<=s<n and 0<=t<m :
            chess[s][t]=True
            Dfs(dep+1, s, t)
            chess[s][t] = False;

for _ in range(int(input())):
    n,m,x,y = map(int, input().split())
    chess = [[False]*maxn for _ in range(maxn)]
    ans = 0
    chess[x][y] = True
    Dfs(1, x, y)
    print(ans)
```

基本信息

#: 44769197
题目: 04123
提交人: 刘子喧
内存: 3632kB
时间: 3329ms
语言: Python3
提交时间: 2024-04-23 22:14:56

28046: 词梯

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

思路 :

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.distance = 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"）

#44769246提交状态

[查看](#) [提交](#) [统计](#) [提问](#)

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):
```

基本信息

#: 44769246
题目: 28046
提交人: 刘子喧
内存: 9564kB
时间: 84ms
语言: Python3
提交时间: 2024-04-23 22:18:22

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_ertex = Vertex(key)
        self.vertices[key] = new_ertex
        return new_ertex

    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), # left-down-down
        (-1, 2), # left-up-up
        (-2, -1), # left-left-down
        (-2, 1), # left-left-up
        (1, -2), # right-down-down
        (1, 2), # right-up-up
        (2, -1), # right-right-down
        (2, 1), # right-right-up
    ]
    for r_off, c_off in move_offsets:
        if (
            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 main():
    def NodeToPos(id):
        return ((id//8, id%8))

    bdSize = int(input())
    *start_pos, = map(int, input().split())
    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=" ")

if __name__ == '__main__':
    main()
```

代码运行截图（AC代码截图，至少包含有"Accepted"）

#44769309提交状态

[查看](#) [提交](#) [统计](#) [提问](#)

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_ertex = Vertex(key)
        self.vertices[key] = new_ertex
        return new_ertex

    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)
```

基本信息

#: 44769309
题目: 28050
提交人: 刘子暄
内存: 4032kB
时间: 31ms
语言: Python3
提交时间: 2024-04-23 22:23:06

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。

好难，真的挺难的