Assignment #9: 图论：遍历，及 树算

Updated 1739 GMT+8 Apr 14, 2024

2024 spring, Complied by 夏天明 元培学院

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

操作系统：Windows 10 | 22H2

Python编程环境：Spyder IDE 5.4.3 | Python 3.11.4 64-bit

1. 题目

04081: 树的转换

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

思路：根据dfs序列递归建树，但并不需要把树转换为实体二叉树，可以递归地求高度(Height)和转换后高度(NewH)，如以下代码中的关键推导式所示

代码

```
class Node:
    def __init__(self):
        self.child = []

    def getHeight(self):
        return 1 + max([nd.getHeight() for nd in self.child], default=-1)

    def getNewH(self):
        return 1 + max([nd.getNewH() + i for i, nd in enumerate(self.child)],
default=-1)
```

```python
def call():
    res = Node()
    while s and s.pop() == 'd':
        res.child.append(call())
    return res

s = list(input())[::-1]
root = call()
print(f"{root.getHeight()} => {root.getNewH()}")
```

代码运行截图

08581: 扩展二叉树

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

思路:

递归地建树：遇到字母就建立新的节点，再递归一层求子节点；遇到“.”就return

同样递归地得到中序和后序遍历

代码

```python
class Node:
    def __init__(self, name, child):
```

```
        self.name = name
        self.child = child

def getSeq(node, pos):
    if node:
        sub = [getSeq(nd, pos) for nd in node.child]
        sub.insert(pos, node.name)
        return ''.join(sub)
    return ''

def generateTree():
    if (token := s.pop()) != '.':
        return Node(token, [generateTree() for i in 'lr'])

s = list(input())[::-1]
root = generateTree()
for i in range(1, 3):
    print(getSeq(root, i))
```

代码运行截图

22067: 快速堆猪

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

思路：维护已经入栈的猪的最小值的栈

代码

```
minpig = []
while True:
    try:
        s = input()
    except EOFError:
        break
    if s == 'min' and minpig:
        print(minpig[-1])
    elif s == 'pop' and minpig:
        minpig.pop()
    elif s[-1].isdigit():
        p = int(s.split()[1])
        if not minpig:
            minpig.append(p)
        minpig.append(min(minpig[-1], p))
```

代码运行截图

04123: 马走日

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

思路：直接dfs

代码

```
```

```python
direc = [(j*i, k*(3-i)) for i in [1,2] for j in [-1,1] for k in [-1,1]]
for o in range(int(input())):
    n, m, x, y = map(int, input().split())
    graph = [[1]*m for i in range(n)]
    ans = 0
    def dfs(x, y, num):
        graph[x][y] = 0
        if num == n*m:
            global ans
            ans += 1
            return
        for dx, dy in direc:
            if 0<=x+dx<n and 0<=y+dy<m and graph[x+dx][y+dy]:
                dfs(x+dx, y+dy, num+1)
                graph[x+dx][y+dy] = 1
    dfs(x, y, 1)
    print(ans)
```

代码运行截图

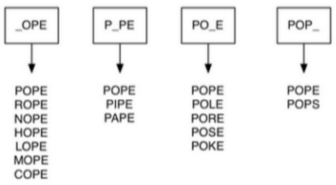
28046: 词梯

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

思路：参考谢老师讲义，用桶辅助建图，然后直接bfs

词梯问题：构建单词关系图-边的发现

- 单词关系图可以通过不同的算法来构建，以 4 个字母的单词表为例，单词表 vocabulary.txt，共 3993 个。
  - 首先是将所有单词作为顶点加入图中，再设法建立顶点之间的边
- 建立边的最直接算法，是对每个顶点（单词），与其它所有单词进行比较，如果相差仅 1 个字母，则建立一条边
  - 时间复杂度是 $O(n^2)$，对于所有 4 个字母的 3993 个单词，需要超过 1547 万次比较
- 改进的算法是创建大量的桶，每个桶可以存放若干单词，桶的标记是去掉 1 个字母，以通配符“_”占空的单词，所有匹配标记的单词都放到这个桶里，所有单词就位后，再在同一个桶的单词之间建立边即可。
  - 最多有多少个桶？以空间换时间。



代码

```
from collections import defaultdict, deque
from itertools import permutations

bucket = defaultdict(list)
for o in range(int(input())):
    word = input()
    for i in range(4):
        label = list(word)
        label[i] = '_'
        bucket[''.join(label)].append(word)
graph = defaultdict(list)
for words in bucket.values():
    for a, b in permutations(words, 2):
        graph[a].append(b)
start, end = input().split()
q = deque([start])
pre = dict()
used = set(q)
def bfs():
    while q:
        word = q.popleft()
        for nex in graph[word]:
            if nex not in used:
                used.add(nex)
                pre[nex] = word
```

```
                if nex == end:
                    return True
                q.append(nex)
res = bfs()
if res:
    ans = [end]
    while ans[-1] in pre:
        ans.append(pre[ans[-1]])
    print(*ans[::-1])
else:
    print('NO')
```

代码运行截图

#44686424提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
from collections import defaultdict, deque
from itertools import permutations

bucket = defaultdict(list)
for o in range(int(input())):
    word = input()
    for i in range(4):
        label = list(word)
        label[i] = '_'
        bucket[''.join(label)].append(word)
graph = defaultdict(list)
for words in bucket.values():
    for a, b in permutations(words, 2):
        graph[a].append(b)
start, end = input().split()
q = deque([start])
pre = dict()
used = set(q)
def bfs():
    while q:
        word = q.popleft()
        for nex in graph[word]:
            if nex not in used:
                used.add(nex)
                pre[nex] = word
```

基本信息

#: 44686424
题目: 28046
提交人: 23n2300017735(夏天明BrightSummer)
内存: 6724kB
时间: 56ms
语言: Python3
提交时间: 2024-04-17 16:53:18

28050: 骑士周游

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

思路：和马走日相同，但是Warnsdorff优化

代码

```
def getNeighbor(pos):
    x, y = pos
```

```
    return [(x+dx, y+dy) for dx, dy in direc if 0<=x+dx<n and 0<=y+dy<n and
graph[x+dx][y+dy]]

def dfs(x, y, num):
    graph[x][y] = 0
    if num == n*n:
        return True
    for x2, y2 in sorted(getNeighbor((x, y)), key=lambda p:len(getNeighbor(p))):
        if graph[x2][y2]:
            if dfs(x2, y2, num+1):
                return True
            graph[x2][y2] = 1

n = int(input())
x, y = map(int, input().split())
direc = [(j*i, k*(3-i)) for i in [1,2] for j in [-1,1] for k in [-1,1]]
graph = [[1]*n for i in range(n)]
ans = 0
print('success' if dfs(x, y, 1) else 'fail')
```

代码运行截图

2. 学习总结和收获

这次作业涉及面比较广，有之前学的树和栈，递归思想的应用，但是与之前作业中较多模板题不同，本次作业的递归问题更灵活，比如树的转换需要一定的观察能力和数学直觉（）

图的问题也不是典型模板，涉及到额外的优化，例如词梯需要使用桶预处理；骑士周游需要引入排序，调整dfs的顺序。因此可以看出，课程中所学典型的数据结构虽然是普适的，但具体问题往往又有其独特性，可以结合学过的各种其他数据结构和算法进行处理和优化。