

# 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：

正常遍历读树和建树

代码

```
1 class TreeNode:
2     def __init__(self):
3         self.children = []
4         self.first_child = None
5         self.next_sib = None
6
7
8     def build(seq):
9         root = TreeNode()
10        stack = [root]
11        depth = 0
```

```

12     for act in seq:
13         cur_node = stack[-1]
14         if act == 'd':
15             new_node = TreeNode()
16             if not cur_node.children:
17                 cur_node.first_child = new_node
18             else:
19                 cur_node.children[-1].next_sib = new_node
20                 cur_node.children.append(new_node)
21                 stack.append(new_node)
22                 depth = max(depth, len(stack) - 1)
23         else:
24             stack.pop()
25     return root, depth
26
27
28 def cal_h_bin(node):
29     if not node:
30         return -1
31     return max(cal_h_bin(node.first_child), cal_h_bin(node.next_sib)) + 1
32
33
34 seq = input()
35 root, h_orig = build(seq)
36 h_bin = cal_h_bin(root)
37 print(f'{h_orig} => {h_bin}')

```

代码运行截图

# 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]
```

## 08581: 扩展二叉树

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

思路:

代码

```
1 def build_tree(preorder):
2     if not preorder or preorder[0] == '.':
3         return None, preorder[1:]
4     root = preorder[0]
5     left, preorder = build_tree(preorder[1:])
6     right, preorder = build_tree(preorder)
7     return (root, left, right), preorder
8 def inorder(tree):
9     if tree is None:
10        return ''
11    root, left, right = tree
12    return inorder(left) + root + inorder(right)
13 def postorder(tree):
14     if tree is None:
15        return ''
```

```

16     root, left, right = tree
17     return postorder(left) + postorder(right) + root
18 preorder = input().strip()
19 tree, _ = build_tree(preorder)
20 print(inorder(tree))
21 print(postorder(tree))

```

代码运行截图

状态: Accepted

源代码

```

def build_tree(preorder):
    if not preorder or preorder[0] == '.':
        return None, preorder[1:]
    root = preorder[0]
    left, preorder = build_tree(preorder[1:])
    right, preorder = build_tree(preorder)
    return (root, left, right), preorder
def inorder(tree):
    if tree is None:
        return ''
    root, left, right = tree
    return inorder(left) + root + inorder(right)
def postorder(tree):

```

## 22067: 快速堆猪

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

思路:

建一个栈

代码

```

1 import heapq
2
3 class PigStack:
4     def __init__(self):
5         self.s = []
6         self.m = []
7
8     def push(self, w):
9         self.s.append(w)

```

```
10         if not self.m or w <= self.m[-1]:
11             self.m.append(w)
12
13     def pop(self):
14         if not self.s:
15             return
16         w = self.s.pop()
17         if w == self.m[-1]:
18             self.m.pop()
19
20     def get_min(self):
21         if not self.m:
22             return None
23         return self.m[-1]
24
25 if __name__ == "__main__":
26     ps = PigStack()
27     n = int(input().strip())
28
29     for _ in range(n):
30         c = input().strip().split()
31         if c[0] == 'push':
32             w = int(c[1])
33             ps.push(w)
34         elif c[0] == 'pop':
35             ps.pop()
36         elif c[0] == 'min':
37             m = ps.get_min()
38             if m is not None:
39                 print(m)
```

代码运行截图

---

状态: Accepted

源代码

```
import heapq

class PigStack:
    def __init__(self):
        self.stack = []
        self.min_heap = []
        self.popped = set()

    def push(self, weight):
        self.stack.append(weight)
```

## 04123: 马走日

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

思路:

dfs

代码

```
1 def dfs(n, m, x, y, visited):
2     if n <= 0 or m <= 0:
3         return 0
4
5     directions = [(-2, 1), (-1, 2), (1, 2), (2, 1),
6                  (2, -1), (1, -2), (-1, -2), (-2, -1)]
7
8     count = 0
9     visited[x][y] = True
10
11     for dx, dy in directions:
12         new_x, new_y = x + dx, y + dy
13         if is_valid_move(n, m, x, y, visited, new_x, new_y):
14             count += dfs(n, m, new_x, new_y, visited)
15
16     visited[x][y] = False
17
18     return 1 if count == 0 else count
19
```

```

20 T = int(input().strip())
21
22 for _ in range(T):
23     n, m, x, y = map(int, input().strip().split())
24     visited = [[False] * m for _ in range(n)]
25     print(dfs(n, m, x, y, visited))

```

代码运行截图 (AC代码截图, 至少包含有"Accepted")

## 28046: 词梯

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

思路:

bfs

代码

```

1  from collections import deque
2
3  def build_g(words):
4      g = {}
5      for w in words:
6          for i in range(len(w)):
7              p = w[:i] + '*' + w[i + 1:]
8              g.setdefault(p, []).append(w)
9      return g
10
11 def find_p(s, e, g):
12     q = deque([(s, [s])])
13     v = set([s])
14
15     while q:
16         w, p = q.popleft()
17         if w == e:
18             return p
19         for i in range(len(w)):
20             ptn = w[:i] + '*' + w[i + 1:]
21             if ptn in g:
22                 for n in g[ptn]:
23                     if n not in v:
24                         v.add(n)
25                         q.append((n, p + [n]))
26     return None
27
28 def word_trans(wds, s, e):
29     g = build_g(wds)
30     return find_p(s, e, g)

```

```

31
32
33 n = int(input().strip())
34 wds = [input().strip() for _ in range(n)]
35 s, e = input().strip().split()
36
37 r = word_trans(wds, s, e)
38
39 if r:
40     print(' '.join(r))
41 else:
42     print("NO")

```

代码运行截图

状态: Accepted

源代码

```

from collections import deque

def build_g(words):
    g = {}
    for w in words:
        for i in range(len(w)):
            p = w[:i] + '*' + w[i + 1:]
            g.setdefault(p, []).append(w)
    return g

def find_p(s, e, g):
    q = deque([(s, [s])])
    v = set([s])

    while q:
        w, p = q.popleft()
        if w == e:
            return p
        for i in range(len(w)):
            ptn = w[:i] + '*' + w[i + 1:]
            if ptn in g:
                for n in g[ptn]:
                    if n not in v:

```



## 28050: 骑士周游

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

思路：

代码

```
1 #
2
```

代码运行截图 (AC代码截图，至少包含有"Accepted")

## 2. 学习总结和收获

---

这周的bfsdfs在计概的课程中大部分都学习过来了基本就是熟悉一下，最后一道题没来得及做，等期中全考完再补上吧