Assignment #9: 图论：遍历，及 树算

Updated 1739 GMT+8 Apr 14, 2024

2024 spring, Complied 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/

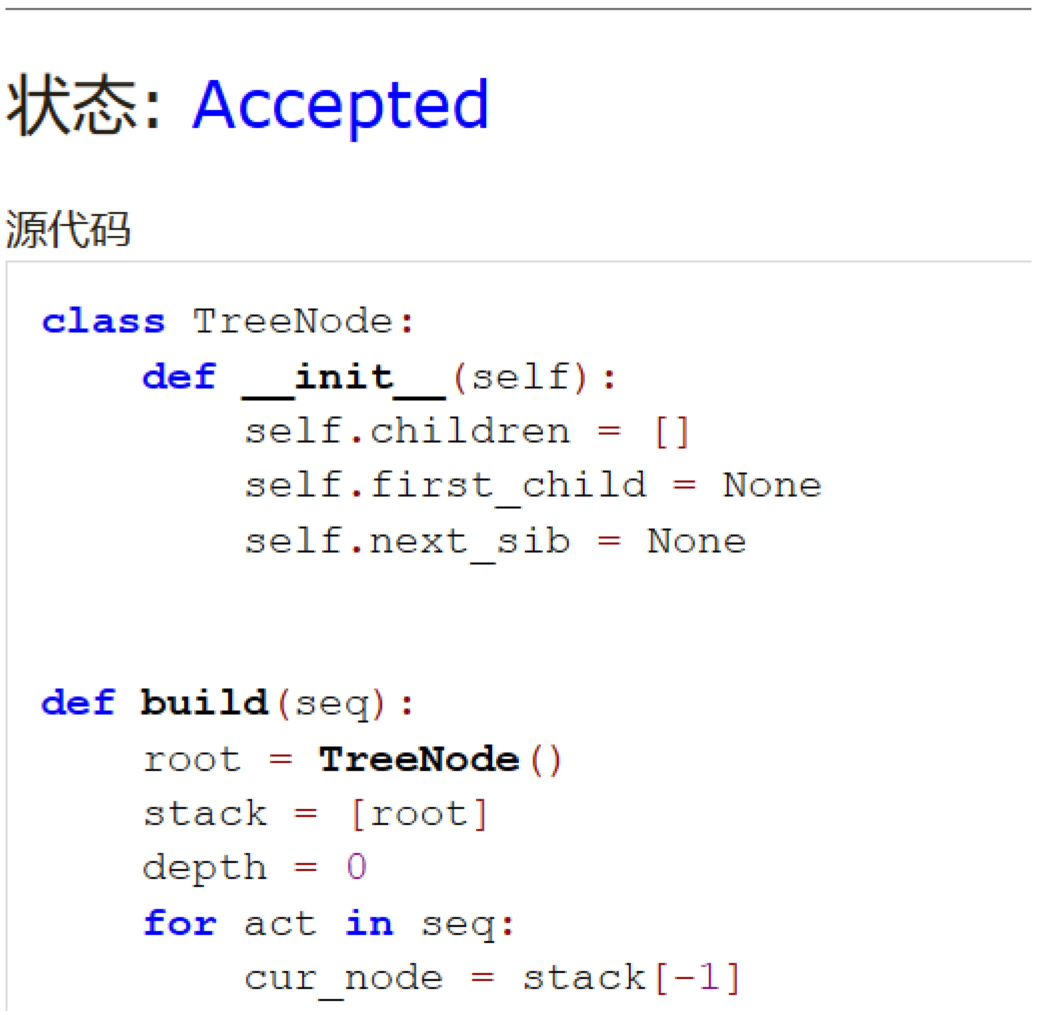
思路：

正常遍历读树和建树

代码

```python
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


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

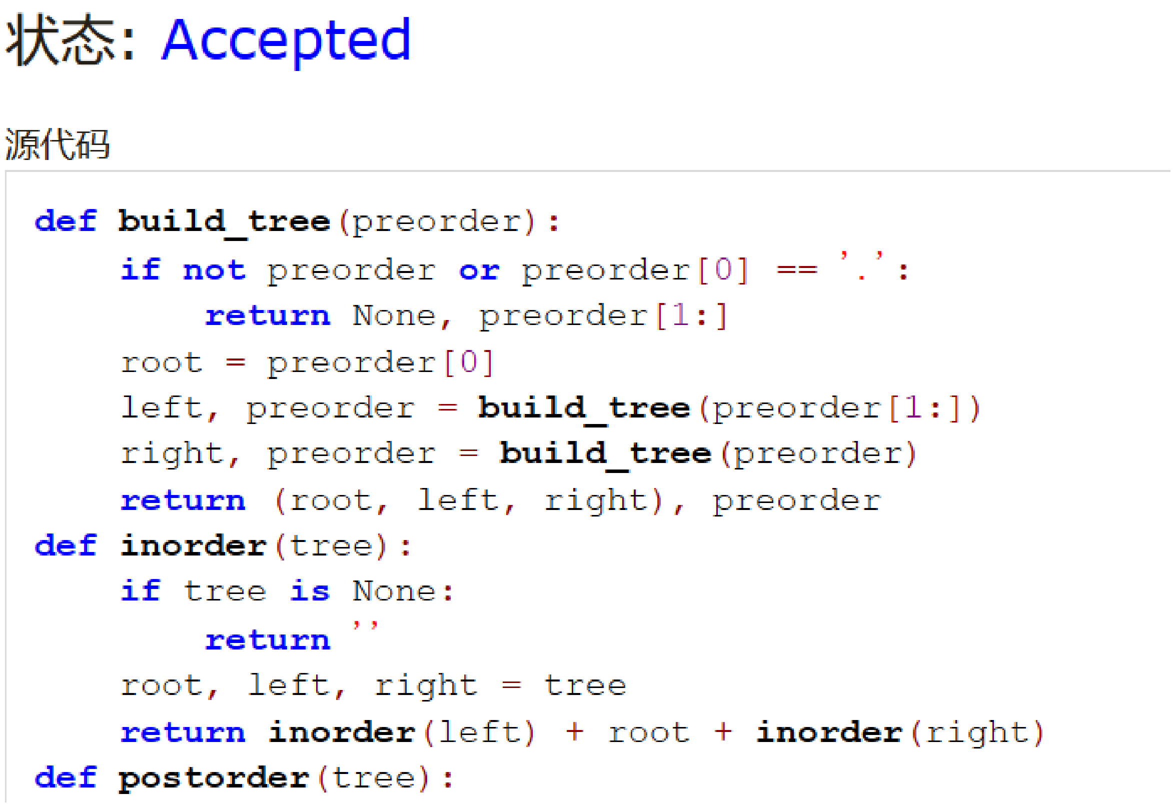
http://cs101.openjudge.cn/dsapre/08581/

思路:

代码

```python
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):
    if tree is None:
        return ''
    root, left, right = tree
    return postorder(left) + postorder(right) + root
preorder = input().strip()
tree, _ = build_tree(preorder)
print(inorder(tree))
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

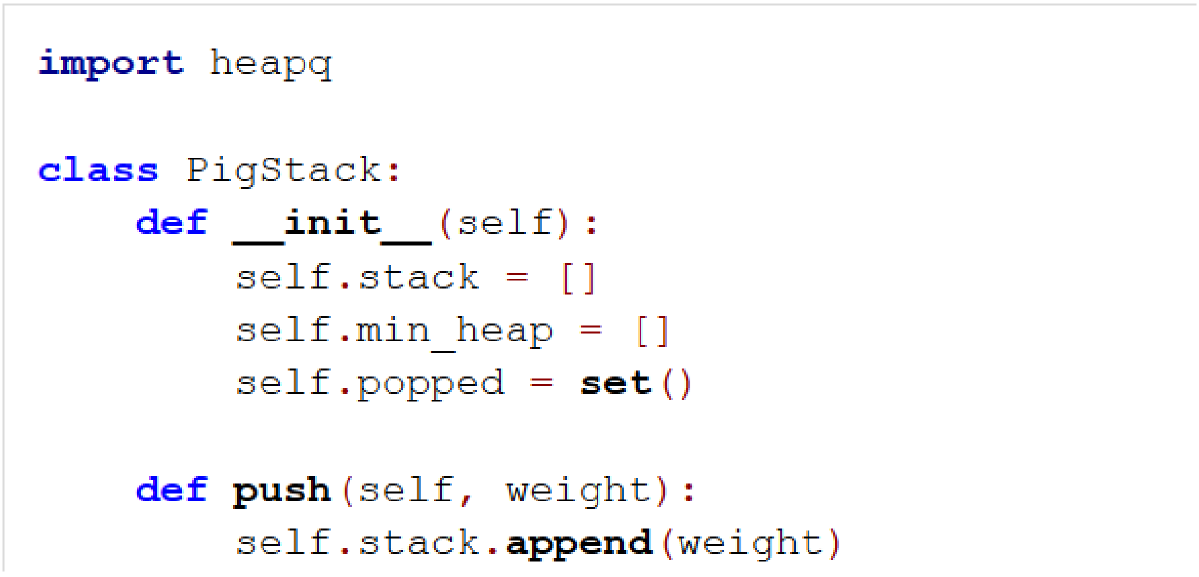
```
import heapq

class PigStack:
    def __init__(self):
        self.s = []
        self.m = []

    def push(self, w):
        self.s.append(w)
```

```python
        if not self.m or w <= self.m[-1]:
            self.m.append(w)

    def pop(self):
        if not self.s:
            return
        w = self.s.pop()
        if w == self.m[-1]:
            self.m.pop()

    def get_min(self):
        if not self.m:
            return None
        return self.m[-1]

if __name__ == "__main__":
    ps = PigStack()
    n = int(input().strip())

    for _ in range(n):
        c = input().strip().split()
        if c[0] == 'push':
            w = int(c[1])
            ps.push(w)
        elif c[0] == 'pop':
            ps.pop()
        elif c[0] == 'min':
            m = ps.get_min()
            if m is not None:
                print(m)
```

代码运行截图

状态: Accepted

源代码



```python
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

```python
def dfs(n, m, x, y, visited):
    if n <= 0 or m <= 0:
        return 0

    directions = [(-2, 1), (-1, 2), (1, 2), (2, 1),
                  (2, -1), (1, -2), (-1, -2), (-2, -1)]

    count = 0
    visited[x][y] = True

    for dx, dy in directions:
        new_x, new_y = x + dx, y + dy
        if is_valid_move(n, m, x, y, visited, new_x, new_y):
            count += dfs(n, m, new_x, new_y, visited)

    visited[x][y] = False

    return 1 if count == 0 else count

```

```python
T = int(input().strip())

for _ in range(T):
    n, m, x, y = map(int, input().strip().split())
    visited = [[False] * m for _ in range(n)]
    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：

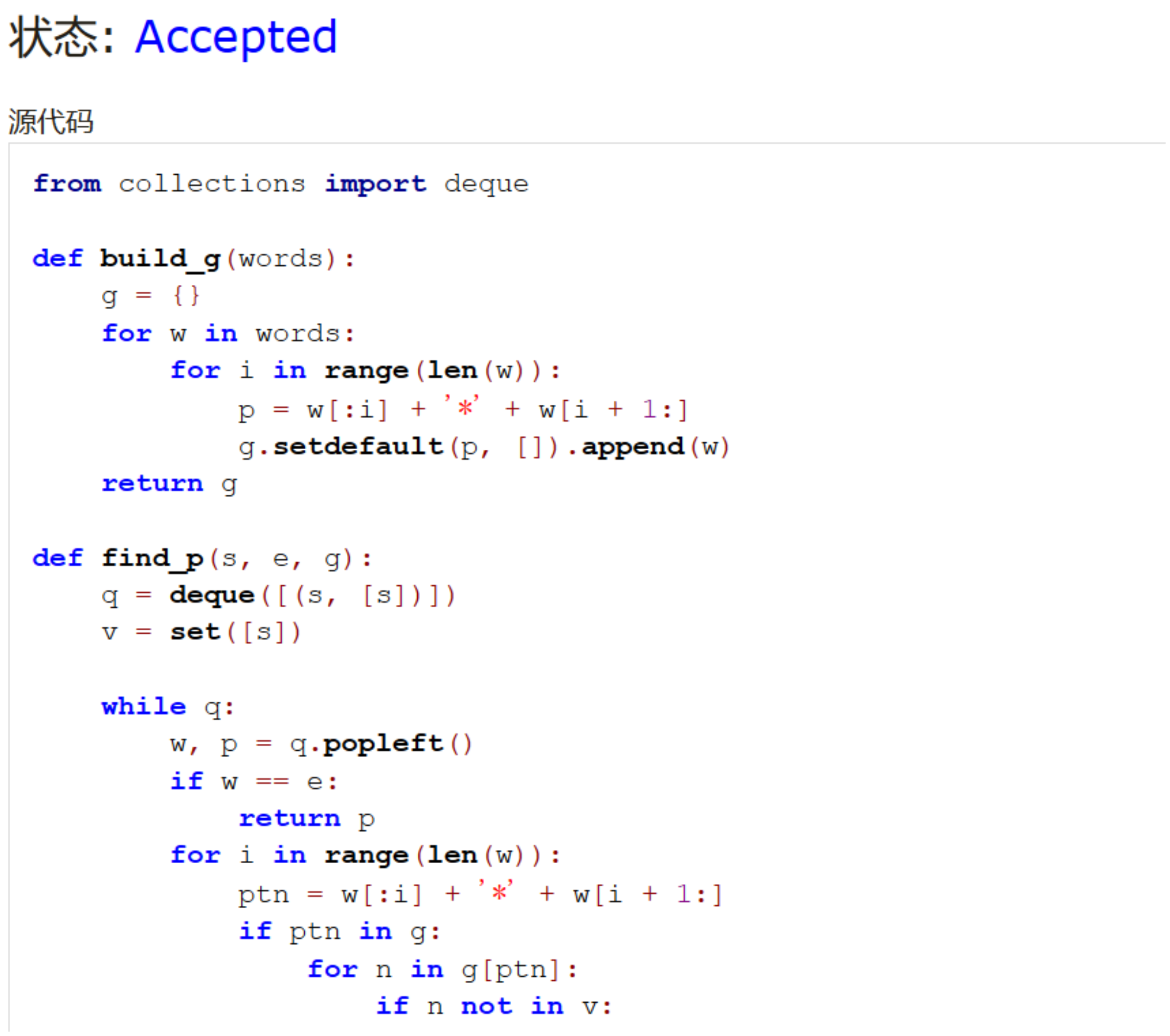
bfs

代码

```python
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:
                        v.add(n)
                        q.append((n, p + [n]))
    return None

def word_trans(wds, s, e):
    g = build_g(wds)
    return find_p(s, e, g)
```

```python


n = int(input().strip())
wds = [input().strip() for _ in range(n)]
s, e = input().strip().split()

r = word_trans(wds, s, e)

if r:
    print(' '.join(r))
else:
    print("NO")
```

代码运行截图



状态: Accepted

源代码

```python
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

```
#

```

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