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：

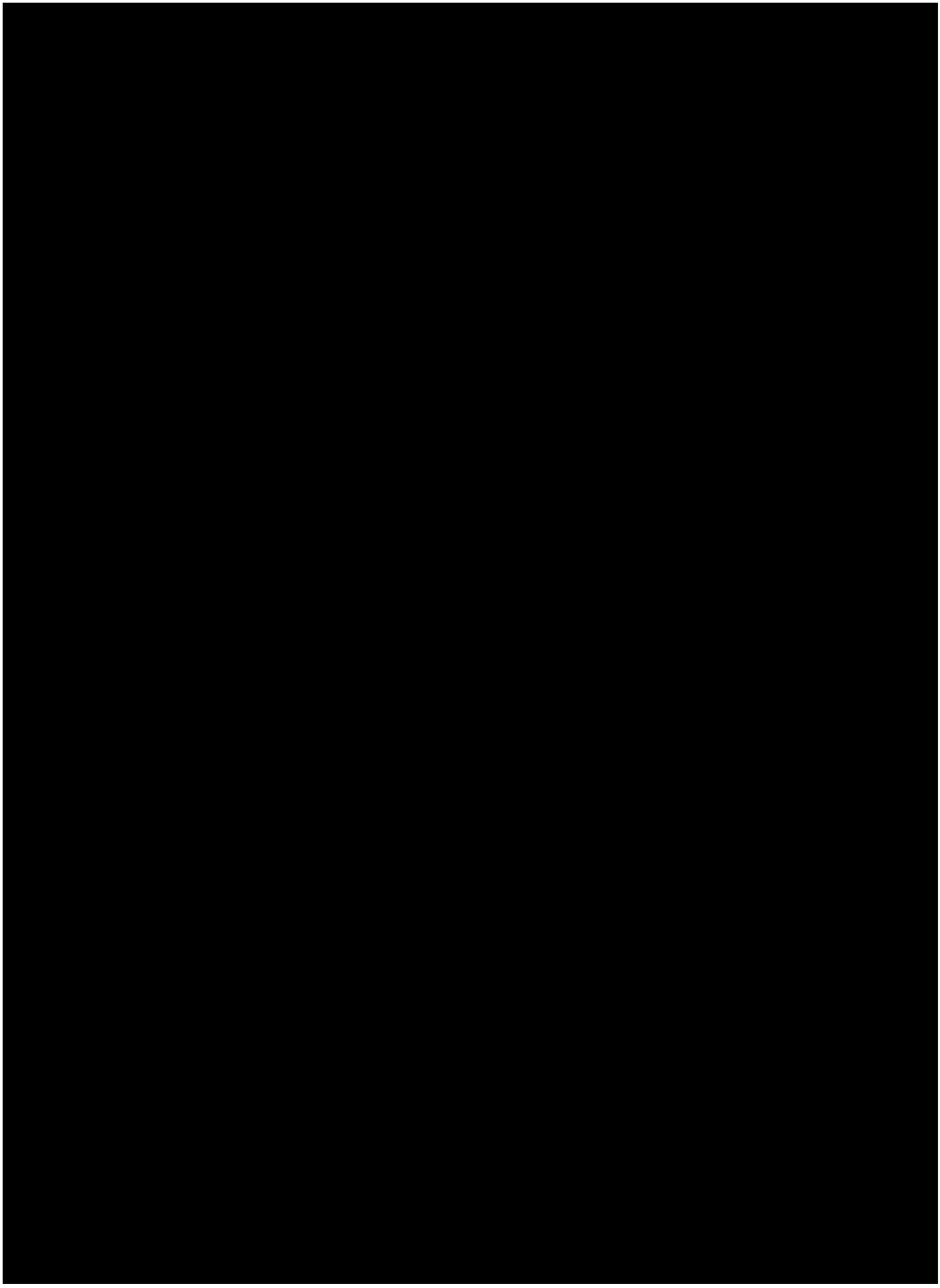
类似上周的题目。

代码

```
class Node:
    def __init__(self, id):
        self.id = id
        self.children = []
        self.father = None

    def add_child(self, child):
```

```
        self.children.append(child)
        child.father = self
        return child

class Tree:
    def __init__(self):
        self.root = Node(0)
        self.nodes = {0: self.root}
        self.len = 1
        self.pointer = self.root

    def move(self, op):
        if op == 'd':
            self.pointer = self.pointer.add_child(Node(self.len))
            self.nodes[self.len] = self.pointer
            self.len += 1
        if op == 'u':
            self.pointer = self.pointer.father

    def depth(self):
        return self._depth(self.root)

    @staticmethod
    def _depth(node):
        if len(node.children) == 0:
            return 0
        return max(map(Tree._depth, node.children)) + 1

class BinaryTree(Tree):
    def __init__(self, tree):
        self.nodes = {}
        for id in range(tree.len):
            self.nodes[id] = Node(id)
        self.root = self.nodes[0]
        for id, node in tree.nodes.items():
            children = node.children
            pointer = self.nodes[id]
            for child in children:
                pointer = pointer.add_child(self.nodes[child.id])

tree = Tree()
for op in input():
    tree.move(op)
binary_tree = BinaryTree(tree)
print(f"{tree.depth()} => {binary_tree.depth()}")
```

代码运行截图 ==（至少包含有"Accepted"）==



08581: 扩展二叉树

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

思路：

按题目要求建树，然后遍历。

代码

```
class Node:
    def __init__(self, id, is_root=None):
        self.id = id
        self.left = None
        self.right = None
        self.father = None
        self.is_root = is_root

    def is_full(self):
        if self.id == '.':
            return True
        if self.left is None or self.right is None:
            return False
        return self.left.is_full() and self.right.is_full()

    def jump(self):
        if self.is_full() and self.is_root is None:
            return self.father.jump()
        return self

class BinaryTree:
    def __init__(self):
        self.nodes = {}
        self.root = None
        self.pointer = None
        self.placeholder = Node('.')

    def add_node(self, id):
        if id != '.':
            node = Node(id)
            if len(self.nodes) == 0:
                self.root = node
            else:
                self.pointer = self.pointer.jump()
                if self.pointer.left is None:
                    self.pointer.left = node
                    node.father = self.pointer
                elif self.pointer.right is None:
                    self.pointer.right = node
                    node.father = self.pointer
            self.pointer = node
```

```
            self.nodes[id] = node
        else:
            self.pointer = self.pointer.jump()
            if self.pointer.left is None:
                self.pointer.left = self.placeholder
            elif self.pointer.right is None:
                self.pointer.right = self.placeholder

    def middle_traverse(self):
        return self._middle_traverse(self.root)

    def backward_traverse(self):
        return self._backward_traverse(self.root)

    def _middle_traverse(self, node):
        if node == self.placeholder:
            return ''
        return self._middle_traverse(node.left) + node.id +
self._middle_traverse(node.right)

    def _backward_traverse(self, node):
        if node == self.placeholder:
            return ''
        return self._backward_traverse(node.left) + self._backward_traverse(node.right) +
node.id

binary_tree = BinaryTree()
for i in input():
    binary_tree.add_node(i)
print(binary_tree.middle_traverse())
print(binary_tree.backward_traverse())
```

代码运行截图 ==（至少包含有"Accepted"）==

状态: Accepted

源代码

```
#08581:扩展二叉树
class Node:
    def __init__(self, id, is_root=None):
        self.id = id
        self.left = None
        self.right = None
        self.father = None
        self.is_root = is_root

    def is_full(self):
        if self.id == '.':
            return True
        if self.left is None or self.right is None:
            return False
        return self.left.is_full() and self.right.is_full()

    def jump(self):
        if self.is_full() and self.is_root is None:
            return self.father.jump()
        return self

class BinaryTree:
    def __init__(self):
        self.nodes = {}
        self.root = None
        self.pointer = None
        self.placeholder = Node('.')

    def add_node(self, id):
        if id != '.':
            node = Node(id)
            if len(self.nodes) == 0:
                self.root = node
            else:
                self.pointer = self.pointer.jump()
                if self.pointer.left is None:
                    self.pointer.left = node
                    node.father = self.pointer
                elif self.pointer.right is None:
                    self.pointer.right = node
                    node.father = self.pointer
            self.pointer = node
            self.nodes[id] = node
```

22067: 快速堆猪

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

思路：

维护两个栈。

代码

```
pigs = []
mins = []

def push(weight):
    pigs.append(weight)
    mins.append(min(mins[-1], weight) if len(mins) > 0 else weight)

def pop():
    if len(pigs) > 0:
        pigs.pop()
        mins.pop()

while 1:
    try:
        inp = input().split()
        if inp[0] == "push":
            push(int(inp[1]))
        if inp[0] == "pop":
            pop()
        if inp[0] == "min":
            if len(pigs) > 0:
                print(mins[-1])
    except EOFError:
        break
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

状态: Accepted

源代码

```
#22067:快速堆猪
pigs = []
mins = []

def push(weight):
    pigs.append(weight)
    mins.append(min(mins[-1], weight) if len(mins) > 0 else weight)

def pop():
    if len(pigs) > 0:
        pigs.pop()
        mins.pop()

while 1:
    try:
        inp = input().split()
        if inp[0] == "push":
            push(int(inp[1]))
        if inp[0] == "pop":
            pop()
        if inp[0] == "min":
            if len(pigs) > 0:
                print(mins[-1])
    except EOFError:
        break
```

04123: 马走日

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

思路:

dfs。

代码

```
directions = [(1, 2), (2, 1), (1, -2), (2, -1), (-1, 2), (-2, 1), (-1, -2), (-2, -1)]
def dfs(n, m, x, y, traversed):
    if len(traversed) == n * m:
        return 1
    res = 0
    for dx, dy in directions:
        xx, yy = x + dx, y + dy
```

```
        if (xx, yy) in traversed or xx < 0 or xx >= n or yy < 0 or yy >= m:
            continue
        else:
            res += dfs(n, m, xx, yy, traversed + [(xx, yy)])
    return res

l = int(input())
for _ in range(l):
    n, m, x, y = map(int, input().split())
    print(dfs(n, m, x, y, [(x, y)]))
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

状态: Accepted

源代码

```
#04123:马走日
directions = [(1, 2), (2, 1), (1, -2), (2, -1), (-1, 2), (-2, 1), (-1, 
def dfs(n, m, x, y, traversed):
    if len(traversed) == n * m:
        return 1
    res = 0
    for dx, dy in directions:
        xx, yy = x + dx, y + dy
        if (xx, yy) in traversed or xx < 0 or xx >= n or yy < 0 or yy >=
            continue
        else:
            res += dfs(n, m, xx, yy, traversed + [(xx, yy)])
    return res

l = int(input())
for _ in range(l):
    n, m, x, y = map(int, input().split())
    print(dfs(n, m, x, y, [(x, y)]))
```

28046: 词梯

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

思路：

bfs。主要耗时出现在建图的过程中，使用了书上的桶的方法。

代码

```
from queue import deque

class Vertex:
    def __init__(self, id, word):
        self.id = id
        self.word = word
        self.connected = []
        self.visited = False
        self.father = None

class Graph:
    def __init__(self, words):
        self.words = words
        self.word2vertex = {}
        self.len = len(words)
        for id, word in enumerate(words):
            self.word2vertex[word] = Vertex(id, word)

    def add_edge(self, word1, word2):
        self.word2vertex[word1].connected.append(self.word2vertex[word2])
        self.word2vertex[word2].connected.append(self.word2vertex[word1])

    def connect(self):
        buckets = {}
        for word in self.words:
            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}:
                    self.add_edge(word1, word2)

    def search(self, start, end):
        queue = deque()
        queue.append(self.word2vertex[start])
        self.word2vertex[start].visited = True
        while len(queue) != 0:
            now = queue.popleft()
            if now == self.word2vertex[end]:
                break
            for child in now.connected:
                if child.visited == False:
                    queue.append(child)
                    child.father = now
                    child.visited = True
        if self.word2vertex[end].father is None:
            return "NO"
        res = [end]
        now = self.word2vertex[end]
        while now.father is not None:
            res.append(now.father.word)
```

```
            now = now.father
        return ' '.join(reversed(res))

n = int(input())
graph = Graph([input() for _ in range(n)])
graph.connect()
print(graph.search(*input().split()))
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

状态: Accepted

源代码

```
#28046:词梯
from queue import deque

class Vertex:
    def __init__(self, id, word):
        self.id = id
        self.word = word
        self.connected = []
        self.visited = False
        self.father = None

class Graph:
    def __init__(self, words):
        self.words = words
        self.word2vertex = {}
        self.len = len(words)
        for id, word in enumerate(words):
            self.word2vertex[word] = Vertex(id, word)

    def add_edge(self, word1, word2):
        self.word2vertex[word1].connected.append(self.word2vertex[word2]
        self.word2vertex[word2].connected.append(self.word2vertex[word1]

    def connect(self):
        buckets = {}
        for word in self.words:
            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}:
                    self.add_edge(word1, word2)

    def search(self, start, end):
        queue = deque()
        queue.append(self.word2vertex[start])
        self.word2vertex[start].visited = True
        while len(queue) != 0:
            now = queue.popleft()
            if now == self.word2vertex[end]:
                break
            for child in now.connected:
```

28050: 骑士周游

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

思路：

剪枝。n 为奇数时直接返回 fail。

代码

```
directions = [(1, 2), (2, 1), (1, -2), (2, -1), (-1, 2), (-2, 1), (-1, -2), (-2, -1)]
def dfs(n, x, y, traversed):
    if n % 2 == 1:
        return True
    if len(traversed) == n ** 2:
        return True
    for dx, dy in directions:
        xx, yy = x + dx, y + dy
        if (xx, yy) in traversed or xx < 0 or xx >= n or yy < 0 or yy >= n:
            continue
        if dfs(n, xx, yy, traversed + [(xx, yy)]):
            return True
    return False

n = int(input())
x, y = map(int, input().split())
print("success" if dfs(n, x, y, [(x, y)]) else "fail")
```

代码运行截图 ==（AC代码截图，至少包含有"Accepted"）==

状态: Accepted

源代码

```
#28050:骑士周游
directions = [(1, 2), (2, 1), (1, -2), (2, -1), (-1, 2), (-2, 1), (-1, 
def dfs(n, x, y, traversed):
    if n % 2 == 1:
        return True
    if len(traversed) == n ** 2:
        return True
    for dx, dy in directions:
        xx, yy = x + dx, y + dy
        if (xx, yy) in traversed or xx < 0 or xx >= n or yy < 0 or yy >=
            continue
        if dfs(n, xx, yy, traversed + [(xx, yy)]):
            return True
    return False

n = int(input())
x, y = map(int, input().split())
print("success" if dfs(n, x, y, [(x, y)]) else "fail")
```

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。==

复习了dfs和bfs，从词梯问题中学到了很多。