Assignment #8: 图论：概念、遍历，及 树算

Updated 1919 GMT+8 Apr 8, 2024

2024 spring, Complied by 周添 物理学院

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。

1. 题目

19943: 图的拉普拉斯矩阵

matrices, http://cs101.openjudge.cn/practice/19943/

请定义Vertex类，Graph类，然后实现

代码

```
class Graph:
    def __init__(self, num_vertices):
        self.num_vertices = num_vertices
        self.adjacency_matrix = [[0] * num_vertices for _ in range(num_vertices)]
        self.edges = []

    def add_edge(self, u, v):
        self.adjacency_matrix[u][v] = 1
        self.adjacency_matrix[v][u] = 1
        self.edges.append((u, v))

    def degree_matrix(self):
        degree_matrix = [[0] * self.num_vertices for _ in
range(self.num_vertices)]
        for i in range(self.num_vertices):
            degree_matrix[i][i] = sum(self.adjacency_matrix[i])
        return degree_matrix


class Matrix:
    def __init__(self, num_rows, num_cols):
        self.num_rows = num_rows
```

```
        self.num_cols = num_cols
        self.data = [[0] * num_cols for _ in range(num_rows)]

    def subtract(self, other):
        result = Matrix(self.num_rows, self.num_cols)
        for i in range(self.num_rows):
            for j in range(self.num_cols):
                result.data[i][j] = self.data[i][j] - other.data[i][j]
        return result

    def __str__(self):
        return '\n'.join([' '.join(map(str, row)) for row in self.data])


n, m = map(int, input().split())
edges = [list(map(int, input().split())) for _ in range(m)]


graph = Graph(n)
for edge in edges:
    u, v = edge
    graph.add_edge(u, v)


degree_matrix = Matrix(graph.num_vertices, graph.num_vertices)
degree_matrix.data = graph.degree_matrix()


adjacency_matrix = Matrix(graph.num_vertices, graph.num_vertices)
adjacency_matrix.data = graph.adjacency_matrix


laplacian_matrix = degree_matrix.subtract(adjacency_matrix)
print(laplacian_matrix)
```

状态: Accepted

源代码

```
class Graph:
    def __init__(self, num_vertices):
        self.num_vertices = num_vertices
        self.adjacency_matrix = [[0] * num_vertices for _ in range(num_v
        self.edges = []

    def add_edge(self, u, v):
        self.adjacency_matrix[u][v] = 1
        self.adjacency_matrix[v][u] = 1
        self.edges.append((u, v))

    def degree_matrix(self):
        degree_matrix = [[0] * self.num_vertices for _ in range(self.nu
        for i in range(self.num_vertices):
            degree_matrix[i][i] = sum(self.adjacency_matrix[i])
        return degree_matrix


class Matrix:
    def __init__(self, num_rows, num_cols):
        self.num_rows = num_rows
        self.num_cols = num_cols
        self.data = [[0] * num_cols for _ in range(num_rows)]

    def subtract(self, other):
        result = Matrix(self.num_rows, self.num_cols)
```

基本信息

#:	44647602
题目:	19943
提交人:	23n2300011538
内存:	3724kB
时间:	29ms
语言:	Python3
提交时间:	2024-04-14 13:13:33

18160: 最大连通域面积

matrix/dfs similar, http://cs101.openjudge.cn/practice/18160

代码

```
def find_largest_connected_area(grid):
    rows, cols = len(grid), len(grid[0])

    def dfs(row, col):
        if row < 0 or col < 0 or row >= rows or col >= cols or grid[row][col] !=
'W':
            return 0
        grid[row][col] = 'X'
        size = 1
        for dr in [-1, 0, 1]:
            for dc in [-1, 0, 1]:
                if dr != 0 or dc != 0:
                    size += dfs(row + dr, col + dc)
        return size

    largest_size = 0
    for i in range(rows):
        for j in range(cols):
            if grid[i][j] == 'W':
                size = dfs(i, j)
                largest_size = max(largest_size, size)

    return largest_size


T = int(input())
for _ in range(T):
    N, M = map(int, input().split())
    grid = [list(input().strip()) for _ in range(N)]
    largest_area = find_largest_connected_area(grid)
    print(largest_area)
```

状态: Accepted

源代码

```
def find_largest_connected_area(grid):
    rows, cols = len(grid), len(grid[0])

    def dfs(row, col):
        if row < 0 or col < 0 or row >= rows or col >= cols or grid[row
            return 0
        grid[row][col] = 'X'
        size = 1
        for dr in [-1, 0, 1]:
            for dc in [-1, 0, 1]:
                if dr != 0 or dc != 0:
                    size += dfs(row + dr, col + dc)
        return size

    largest_size = 0
    for i in range(rows):
        for j in range(cols):
            if grid[i][j] == 'W':
                size = dfs(i, j)
                largest_size = max(largest_size, size)

    return largest_size


T = int(input())
for _ in range(T):
    N, M = map(int, input().split())
    grid = [list(input().strip()) for _ in range(N)]
    largest_area = find_largest_connected_area(grid)
    print(largest_area)
```

基本信息

#:	42808467
题目:	18160
提交人:	23n2300011538
内存:	3824kB
时间:	120ms
语言:	Python3
提交时间:	2023-11-28 16:46:13

sy383: 最大权值连通块

https://sunnywhy.com/sfbj/10/3/383

代码

```
class Graph:
    def __init__(self, num_vertices):
        self.num_vertices = num_vertices
        self.adjacency_matrix = [[0] * num_vertices for _ in range(num_vertices)]

    def add_edge(self, u, v):
        self.adjacency_matrix[u][v] = 1
        self.adjacency_matrix[v][u] = 1


def find_connected_components(graph):
    visited = [False] * graph.num_vertices
    components = []

    def dfs(node, component):
        visited[node] = True
        component.append(node)
        for neighbor in range(graph.num_vertices):
            if graph.adjacency_matrix[node][neighbor] == 1 and not
visited[neighbor]:
                dfs(neighbor, component)

    for vertex in range(graph.num_vertices):
        if not visited[vertex]:
            component = []
```

```
            dfs(vertex, component)
            components.append(component)

    return components


def max_component_weight(graph, weights):
    components = find_connected_components(graph)
    max_weight = 0

    for component in components:
        component_weight = sum(weights[node] for node in component)
        max_weight = max(max_weight, component_weight)

    return max_weight


n, m = map(int, input().split())
weights = list(map(int, input().split()))
edges = [list(map(int, input().split())) for _ in range(m)]

graph = Graph(n)
for edge in edges:
    u, v = edge
    graph.add_edge(u, v)

result = max_component_weight(graph, weights)
print(result)
```

```
        for neighbor in range(graph.num_vertices):
            if graph.adjacency_matrix[node][neighbor] == 1 a
                dfs(neighbor, component)

    for vertex in range(graph.num_vertices):
        if not visited[vertex]:
            component = []
            dfs(vertex, component)
            components.append(component)

    return components


```

测试输入 | 提交结果 | 历史提交

完美通过 查看题解

100% 数据通过测试

运行时长: 0 ms

03441: 4 Values whose Sum is 0

data structure/binary search, http://cs101.openjudge.cn/practice/03441

代码

```
n = int(input())
a = [0]*(n+1)
b = [0]*(n+1)
c = [0]*(n+1)
d = [0]*(n+1)

for i in range(n):
    a[i],b[i],c[i],d[i] = map(int, input().split())

dict1 = {}
for i in range(n):
    for j in range(n):
        if not a[i]+b[j] in dict1:
            dict1[a[i] + b[j]] = 0
        dict1[a[i] + b[j]] += 1

ans = 0
for i in range(n):
    for j in range(n):
```

```
        if -(c[i]+d[j]) in dict1:
            ans += dict1[-(c[i]+d[j])]

print(ans)
```

状态: Accepted

源代码

```
n = int(input())
a = [0]*(n+1)
b = [0]*(n+1)
c = [0]*(n+1)
d = [0]*(n+1)

for i in range(n):
    a[i],b[i],c[i],d[i] = map(int, input().split())

dict1 = {}
for i in range(n):
    for j in range(n):
        if not a[i]+b[j] in dict1:
            dict1[a[i] + b[j]] = 0
        dict1[a[i] + b[j]] += 1

ans = 0
for i in range(n):
    for j in range(n):
        if -(c[i]+d[j]) in dict1:
            ans += dict1[-(c[i]+d[j])]

print(ans)
```

基本信息

#:	44652544
题目:	03441
提交人:	23n2300011538
内存:	171688kB
时间:	5544ms
语言:	Python3
提交时间:	2024-04-14 16:34:12

04089: 电话号码

trie, http://cs101.openjudge.cn/practice/04089/

Trie 数据结构可能需要自学下。

代码

```
class TrieNode:
    def __init__(self):
        self.children = {}
        self.is_end_of_word = False


def insert(root, word):
    current = root
    for char in word:
        if char not in current.children:
            current.children[char] = TrieNode()
        current = current.children[char]
    current.is_end_of_word = True


def is_consistent(phone_numbers):
    if len(set(phone_numbers)) != len(phone_numbers):
        return "NO"
```

```
    root = TrieNode()
    for phone_number in phone_numbers:
        insert(root, phone_number)

    for phone_number in phone_numbers:
        current = root
        for char in phone_number:
            if char not in current.children:
                break
            if current.is_end_of_word:
                return "NO"
            current = current.children[char]
    return "YES"


t = int(input())
for _ in range(t):
    n = int(input())
    phone_numbers = [input() for _ in range(n)]
    print(is_consistent(phone_numbers))
```

状态: Accepted

源代码

```
class TrieNode:
    def __init__(self):
        self.children = {}
        self.is_end_of_word = False


def insert(root, word):
    current = root
    for char in word:
        if char not in current.children:
            current.children[char] = TrieNode()
        current = current.children[char]
    current.is_end_of_word = True


def is_consistent(phone_numbers):
    if len(set(phone_numbers)) != len(phone_numbers):
        return "NO"

    root = TrieNode()
    for phone_number in phone_numbers:
        insert(root, phone_number)

    for phone_number in phone_numbers:
        current = root
        for char in phone_number:
            if char not in current.children:
                break
            if current.is_end_of_word:
                return "NO"
```

基本信息

#:	44653496
题目:	04089
提交人:	23n2300011538
内存:	25644kB
时间:	346ms
语言:	Python3
提交时间:	2024-04-14 17:12:17

04082: 树的镜面映射

http://cs101.openjudge.cn/practice/04082/

代码

```
from queue import Queue

class BinNode:
    def __init__(self):
        self.data = None
        self.lchild = None
        self.rchild = None

class CSNode:
    def __init__(self, n):
        self.data = n
        self.children = []

def converse(root):
    if root is None:
        return None
    ve = []
    t = CSNode(root.data)
    p = None
    if root.lchild and root.lchild.data != '$':
        p = root.lchild
    while p and p.data != '$':
        ve.append(converse(p))
        p = p.rchild
    t.children = ve
    return t

def printTree(root):
    if root is None:
        return
    print(root.data, end=' ')
    for child in root.children:
        printTree(child)

def revert(root):
    if root is None:
        return
    n = len(root.children)
    for i in range(n // 2):
        root.children[i], root.children[n - i - 1] = root.children[n - i - 1],
root.children[i]

def mirror(root):
    if root is None:
        return
    revert(root)
    for child in root.children:
        mirror(child)

def rebuild(x, num, n):
    stack = []
    root = None
    for i in range(n):
        r = BinNode()
        if i == 0:
```

```
            root = r
        cur = None
        r.data = x[i]
        if stack:
            cur = stack[-1]
            if cur.lchild is None:
                cur.lchild = r
            elif cur.rchild is None:
                cur.rchild = r
            if cur.lchild and cur.rchild:
                stack.pop()
        if num[i] == 0 and x[i] != '$':
            stack.append(r)
    return root

def bfsVisit(root):
    if not root:
        return
    q = Queue()
    q.put(root)
    while not q.empty():
        root = q.get()
        print(root.data, end=' ')
        for child in root.children:
            q.put(child)

def bfsVisitBinary(root):
    q = Queue()
    tmp = root
    if not tmp:
        return
    q.put(tmp)
    while not q.empty():
        tmp = q.get()
        print(tmp.data, end=' ')
        if tmp.lchild:
            q.put(tmp.lchild)
        if tmp.rchild:
            q.put(tmp.rchild)


n = int(input())
a = input().split()
x = []
num = []
for i in a:
    x.append(i[0])
    num.append(int(i[1]))
root = rebuild(x, num, n)
t = converse(root)
mirror(t)
bfsVisit(t)
```

状态: Accepted

源代码

```
from queue import Queue

class BinNode:
    def __init__(self):
        self.data = None
        self.lchild = None
        self.rchild = None

class CSNode:
    def __init__(self, n):
        self.data = n
        self.children = []

def converse(root):
    if root is None:
        return None
    ve = []
    t = CSNode(root.data)
    p = None
    if root.lchild and root.lchild.data != '$':
        p = root.lchild
    while p and p.data != '$':
        ve.append(converse(p))
        p = p.rchild
```

基本信息

#:	44655511
题目:	04082
提交人:	23n2300011538
内存:	3868kB
时间:	29ms
语言:	Python3
提交时间:	2024-04-14 19:07:39

2. 学习总结和收获

最近考试考得很差很差，有点丧失学习动力