Assignment #A: 图论：算法，树算及栈

Updated 2018 GMT+8 Apr 21, 2024

2024 spring, Complied by 周添 物理学院

1. 题目

20743: 整人的提词本

http://cs101.openjudge.cn/practice/20743/

代码

```
def reverse_parentheses(s):
    stack = []
    for char in s:
        if char == ')':
            temp = ''
            while stack and stack[-1] != '(':
                temp += stack.pop()
            if stack:
                stack.pop()
                stack.extend(temp)
            else:
                stack.extend(temp[::-1])
        else:
            stack.append(char)
    return ''.join(stack)


s = input()
result = reverse_parentheses(s)
print(result)
```

状态: Accepted

源代码

```
def reverse_parentheses(s):
    stack = []
    for char in s:
        if char == ')':
            temp = ''
            while stack and stack[-1] != '(':
                temp += stack.pop()
            if stack:
                stack.pop()
                stack.extend(temp)
            else:
                stack.extend(temp[::-1])
        else:
            stack.append(char)
    return ''.join(stack)


s = input()
result = reverse_parentheses(s)
print(result)
```

基本信息

#: 44827971
题目: 20743
提交人: 23n2300011538
内存: 3608kB
时间: 20ms
语言: Python3
提交时间: 2024-04-29 01:20:49

02255: 重建二叉树

http://cs101.openjudge.cn/practice/02255/

代码

```
def build_tree(preorder, inorder):
    if not preorder or not inorder:
        return []

    root_val = preorder[0]
    root_index = inorder.index(root_val)

    left_preorder = preorder[1:root_index + 1]
    right_preorder = preorder[root_index + 1:]

    left_inorder = inorder[:root_index]
    right_inorder = inorder[root_index + 1:]

    left_tree = build_tree(left_preorder, left_inorder)
    right_tree = build_tree(right_preorder, right_inorder)

    return left_tree + right_tree + [root_val]


def postorder_traversal(preorder, inorder):
    if not preorder or not inorder:
        return ''

    tree = build_tree(list(preorder), list(inorder))
    return ''.join(tree)


while True:
    try:
```

```
        preorder, inorder = input().split()
        result = postorder_traversal(preorder, inorder)
        print(result)
    except EOFError:
        break
```

状态: Accepted

源代码

```
def build_tree(preorder, inorder):
    if not preorder or not inorder:
        return []

    root_val = preorder[0]
    root_index = inorder.index(root_val)

    left_preorder = preorder[1:root_index+1]
    right_preorder = preorder[root_index+1:]

    left_inorder = inorder[:root_index]
    right_inorder = inorder[root_index+1:]

    left_tree = build_tree(left_preorder, left_inorder)
    right_tree = build_tree(right_preorder, right_inorder)

    return left_tree + right_tree + [root_val]

def postorder_traversal(preorder, inorder):
    if not preorder or not inorder:
        return ''

    tree = build_tree(list(preorder), list(inorder))
    return ''.join(tree)


while True:
    try:
        preorder, inorder = input().split()
        result = postorder_traversal(preorder, inorder)
        print(result)
```

基本信息

#:	44827994
题目:	02255
提交人:	23n2300011538
内存:	3540kB
时间:	20ms
语言:	Python3
提交时间:	2024-04-29 02:11:31

01426: Find The Multiple

http://cs101.openjudge.cn/practice/01426/

要求用bfs实现

代码

```
from collections import deque

def find_multiple(n):
    if n == 1:
        return "1"

    queue = deque([(1, "1")])
    visited = set()
    while queue:
        curr_rem, curr_m = queue.popleft()
        if curr_rem == 0:
            return curr_m
        next_rem1 = (curr_rem * 10) % n
        next_m1 = curr_m + "0"
```

```
        next_rem2 = (next_rem1 + 1) % n
        next_m2 = curr_m + "1"
        if next_rem1 not in visited:
            visited.add(next_rem1)
            queue.append((next_rem1, next_m1))
        if next_rem2 not in visited:
            visited.add(next_rem2)
            queue.append((next_rem2, next_m2))

while True:
    n = int(input())
    if n == 0:
        break
    result = find_multiple(n)
    print(result)
```

状态: Accepted

源代码

```
from collections import deque

def find_multiple(n):
    if n == 1:
        return "1"

    queue = deque([(1, "1")])
    visited = set()
    while queue:
        curr_rem, curr_m = queue.popleft()
        if curr_rem == 0:
            return curr_m
        next_rem1 = (curr_rem * 10) % n
        next_m1 = curr_m + "0"
        next_rem2 = (next_rem1 + 1) % n
        next_m2 = curr_m + "1"
        if next_rem1 not in visited:
            visited.add(next_rem1)
            queue.append((next_rem1, next_m1))
        if next_rem2 not in visited:
            visited.add(next_rem2)
            queue.append((next_rem2, next_m2))

while True:
    n = int(input())
    if n == 0:
        break
```

基本信息

#: 44828043
题目: 01426
提交人: 23n2300011538
内存: 3564kB
时间: 38ms
语言: Python3
提交时间: 2024-04-29 04:25:39

04115: 鸣人和佐助

bfs, http://cs101.openjudge.cn/practice/04115/

github上搬的，我写的就很奇怪老少一点

代码

```
from collections import deque
```

```python
class Node:
    def __init__(self, x, y, tools, steps):
        self.x = x
        self.y = y
        self.tools = tools
        self.steps = steps


M, N, T = map(int, input().split())
maze = [list(input()) for _ in range(M)]
visit = [[[0]*(T+1) for _ in range(N)] for _ in range(M)]
directions = [[-1, 0], [1, 0], [0, -1], [0, 1]]
start = end = None
flag = 0
for i in range(M):
    for j in range(N):
        if maze[i][j] == '@':
            start = Node(i, j, T, 0)
            visit[i][j][T] = 1
        if maze[i][j] == '+':
            end = (i, j)
            maze[i][j] = '*'

queue = deque([start])
while queue:
    node = queue.popleft()
    if (node.x, node.y) == end:
        print(node.steps)
        flag = 1
        break
    for direction in directions:
        nx, ny = node.x+direction[0], node.y+direction[1]
        if 0 <= nx < M and 0 <= ny < N:
            if maze[nx][ny] == '*':
                if not visit[nx][ny][node.tools]:
                    queue.append(Node(nx, ny, node.tools, node.steps+1))
                    visit[nx][ny][node.tools] = 1
            elif maze[nx][ny] == '#':
                if node.tools > 0 and not visit[nx][ny][node.tools-1]:
                    queue.append(Node(nx, ny, node.tools-1, node.steps+1))
                    visit[nx][ny][node.tools-1] = 1

if not flag:
    print("-1")
```

状态: Accepted

基本信息

#: 44830823
题目: 04115
提交人: 23n2300011
内存: 7252kB
时间: 115ms
语言: Python3
提交时间: 2024-04-29

源代码

```
from collections import deque


class Node:
    def __init__(self, x, y, tools, steps):
        self.x = x
        self.y = y
        self.tools = tools
        self.steps = steps


M, N, T = map(int, input().split())
maze = [list(input()) for _ in range(M)]
visit = [[[0]*(T+1) for _ in range(N)] for _ in range(M)]
directions = [[-1, 0], [1, 0], [0, -1], [0, 1]]
start = end = None
flag = 0
for i in range(M):
    for j in range(N):
        if maze[i][j] == '@':
            start = Node(i, j, T, 0)
            visit[i][j][T] = 1
        if maze[i][j] == '+':
            end = (i, j)
            maze[i][j] = '*'
```

20106: 走山路

Dijkstra, http://cs101.openjudge.cn/practice/20106/

代码

```
import heapq


def walk_bu_liao_yi_dian():
    xs, ys, xe, ye = map(int, input().split())
    if matrix[xs][ys] == "#" or matrix[xe][ye] == "#":
        return 'NO'
    directions = [[-1, 0], [1, 0], [0, 1], [0, -1]]
    queue = []
    heapq.heappush(queue, [0, xs, ys])
    visited = set()
    visited.add((xs, ys, -1))
    answer = []
    while queue:
        step, x, y = map(int, heapq.heappop(queue))
        if x == xe and y == ye:
            answer.append(step)
        for i in range(4):
            dx, dy = x + directions[i][0], y + directions[i][1]
            if 0 <= dx < m and 0 <= dy < n and matrix[dx][dy] != '#' and (dx, dy,
i) not in visited:
                heapq.heappush(queue, [step + abs(int(matrix[dx][dy]) -
int(matrix[x][y])), dx, dy])
                # print([step + abs(int(matrix[dx][dy]) - int(matrix[x][y])), dx,
dy])
                visited.add((dx, dy, i))
```

```
    return min(answer) if answer else 'NO'


m, n, p = map(int, input().split())
matrix = [list(map(str, input().split())) for i in range(m)]
for _ in range(p):
    print(walk_bu_liao_yi_dian())
```

状态: Accepted

源代码

```
import heapq


def walk_bu_liao_yi_dian():
    xs, ys, xe, ye = map(int, input().split())
    if matrix[xs][ys] == "#" or matrix[xe][ye] == "#":
        return 'NO'
    directions = [[-1, 0], [1, 0], [0, 1], [0, -1]]
    queue = []
    heapq.heappush(queue, [0, xs, ys])
    visited = set()
    visited.add((xs, ys, -1))
    answer = []
    while queue:
        step, x, y = map(int, heapq.heappop(queue))
        if x == xe and y == ye:
            answer.append(step)
        for i in range(4):
            dx, dy = x + directions[i][0], y + directions[i][1]
            if 0 <= dx < m and 0 <= dy < n and matrix[dx][dy] != '#' and
                heapq.heappush(queue, [step + abs(int(matrix[dx][dy]) -
                # print([step + abs(int(matrix[dx][dy]) - int(matrix[x]
                visited.add((dx, dy, i))
    return min(answer) if answer else 'NO'


m, n, p = map(int, input().split())
matrix = [list(map(str, input().split())) for i in range(m)]
for _ in range(p):
    print(walk_bu_liao_yi_dian())
```

基本信息

#:	43249542
题目:	20106
提交人:	23n2300011538
内存:	4736kB
时间:	1458ms
语言:	Python3
提交时间:	2023-12-20 14:56:08

05442: 兔子与星空

Prim, http://cs101.openjudge.cn/practice/05442/

就是贪心啊

代码

```
import heapq

n = int(input())
edges_info = [input() for _ in range(n - 1)]

graph = {chr(ord('A') + i): [] for i in range(n)}

for i in range(n - 1):
    parts = edges_info[i].split()
    star = parts[0]
    num_edges = int(parts[1])
    edges = parts[2:]
```

```
    for j in range(num_edges):
        neighbor, weight = edges[2*j], int(edges[2*j + 1])
        graph[star].append((neighbor, weight))
        graph[neighbor].append((star, weight))

visited = set()
start_node = list(graph.keys())[0]
min_heap = [(0, start_node)]
total_weight = 0

while min_heap:
    weight, node = heapq.heappop(min_heap)
    if node not in visited:
        visited.add(node)
        total_weight += weight
        for neighbor, edge_weight in graph[node]:
            if neighbor not in visited:
                heapq.heappush(min_heap, (edge_weight, neighbor))

print(total_weight)
```

状态: Accepted

源代码

```
import heapq

n = int(input())
edges_info = [input() for _ in range(n - 1)]

graph = {chr(ord('A') + i): [] for i in range(n)}

for i in range(n - 1):
    parts = edges_info[i].split()
    star = parts[0]
    num_edges = int(parts[1])
    edges = parts[2:]

    for j in range(num_edges):
        neighbor, weight = edges[2*j], int(edges[2*j + 1])
        graph[star].append((neighbor, weight))
        graph[neighbor].append((star, weight))

visited = set()
start_node = list(graph.keys())[0]
min_heap = [(0, start_node)]
total_weight = 0

while min_heap:
    weight, node = heapq.heappop(min_heap)
```

基本信息

#:	44831482
题目:	05442
提交人:	23n2300011538
内存:	3668kB
时间:	21ms
语言:	Python3
提交时间:	2024-04-29 17:33:44

2. 学习总结和收获

最近事情真的很多，各种论文、pre，还有一个实验比赛要搞，一直没有时间练习，感觉确实手生了许多，码力下降严重