

Assignment #B: 图论和树算

Updated 1709 GMT+8 Apr 28, 2024

2024 spring, Compiled by 陈奕好 工学院

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。

编程环境

(请改为同学的操作系统、编程环境等)

操作系统：macOS Sonoma 14.4 (23E214)

Python编程环境：PyCharm 2023.3.1 (Professional Edition)

1. 题目

28170: 算鹰

dfs, <http://cs101.openjudge.cn/practice/28170/>

思路：dfs的模版题

代码

```
1 board = [list(map(str, input())) for _ in range(10)]
2 visited = [[0] * 10 for _ in range(10)]
3 dx = [0, 0, 1, -1]
4 dy = [1, -1, 0, 0]
5 cnt = 0
6 # print(*board, sep='\n')
7 # ['- ', '- ', '- ', '. ', '- ', '- ', '. ', '- ', '. ', '. '
8 # ['- ', '. ', '. ', '- ', '. ', '- ', '. ', '. ', '. ', '. '
9 # ['. ', '. ', '. ', '- ', '- ', '. ', '. ', '. ', '. ', '- ']
```

```

10 # ['-','-', '-', '-', '-', '-', '-', '-', '-', '-', '-']
11 # ['-','-', '-', '-', '-', '-', '-', '-', '-', '-', '-']
12 # ['-','.', '-', '-', '-', '-', '-', '-', '-', '-', '-']
13 # ['.','.','.','.','-','-','-','-','-','-','-']
14 # ['-','.', '-', '-', '-', '-', '-', '-', '-', '-', '-']
15 # ['-','.', '-', '-', '-', '-', '-', '-', '-', '-', '-']
16 # ['.','.','.','.','.','.','-','-','-','-','-']
17
18
19 def dfs(chessboard, x, y, visited):
20     for i in range(4):
21         new_x = x + dx[i]
22         new_y = y + dy[i]
23         if 0 <= new_x < 10 and 0 <= new_y < 10 and chessboard[new_x][new_y] == '.'
and visited[new_x][new_y] == 0:
24             visited[new_x][new_y] = 1
25             dfs(chessboard, new_x, new_y, visited)
26
27
28 for i in range(10):
29     for j in range(10):
30         if board[i][j] == '.' and visited[i][j] == 0:
31             visited[i][j] = 1
32             dfs(board, i, j, visited)
33             cnt += 1
34
35 print(cnt)
36
37

```

代码运行截图 (至少包含有"Accepted")

状态: Accepted

源代码

基本信息

#: 44878936

题目: 28170

提交人: 23n2300011030(陈奕好)

内存: 3656kB

时间: 19ms

语言: Python3

提交时间: 2024-05-06 14:21:27

```
board = [list(map(str, input())) for _ in range(10)]
visited = [[0] * 10 for _ in range(10)]
dx = [0, 0, 1, -1]
dy = [1, -1, 0, 0]
cnt = 0
# print(*board, sep='\n')
# ['-',' ',' ',' ',' ',' ',' ',' ',' ',' ']
# ['-',' ',' ',' ',' ',' ',' ',' ',' ',' ']
# ['.',' ',' ',' ',' ',' ',' ',' ',' ',' ']
# ['-',' ',' ',' ',' ',' ',' ',' ',' ',' ']
# ['-',' ',' ',' ',' ',' ',' ',' ',' ',' ']
# ['-',' ',' ',' ',' ',' ',' ',' ',' ',' ']
# ['.',' ',' ',' ',' ',' ',' ',' ',' ',' ']
# ['-',' ',' ',' ',' ',' ',' ',' ',' ',' ']
# ['-',' ',' ',' ',' ',' ',' ',' ',' ',' ']
# ['.',' ',' ',' ',' ',' ',' ',' ',' ',' ']

def dfs(chessboard, x, y, visited):
    for i in range(4):
        new_x = x + dx[i]
        new_y = y + dy[i]
        if 0 <= new_x < 10 and 0 <= new_y < 10 and chessboard[new_x][new_y] == '.' and visited[new_x][new_y] == 0:
            visited[new_x][new_y] = 1
            dfs(chessboard, new_x, new_y, visited)

for i in range(10):
    for j in range(10):
        if board[i][j] == '.' and visited[i][j] == 0:
            visited[i][j] = 1
            dfs(board, i, j, visited)
```

02754: 八皇后

dfs, <http://cs101.openjudge.cn/practice/02754/>

思路: 换用bfs, 从835ms下降至35ms

代码

```
1 from collections import deque
2 n = int(input())
3 ans = []
4
5
6 def isVaild(string, x):
7     for i in string:
8         if abs(int(i)-int(x)) == abs(string.index(i)-len(string)):
9             return False
10    return True
11
12
13 def bfs(size):
```

```

14     queue = deque()
15     for i in range(1, size+1):
16         queue.append(str(i))
17
18     while queue:
19         string = queue.popleft()
20         if len(string) == size:
21             ans.append(string)
22         else:
23             for i in range(1, size+1):
24                 if str(i) not in string and isVaild(string, str(i)):
25                     queue.append(string+str(i))
26
27
28 bfs(8)
29 for i in range(n):
30     print(ans[int(input()) - 1])
31
32

```

代码运行截图 (至少包含有"Accepted")

状态: **Accepted**

源代码

```

from collections import deque
n = int(input())
ans = []

def isVaild(string, x):
    for i in string:
        if abs(int(i)-int(x)) == abs(string.index(i)-len(string)):
            return False
    return True

def bfs(size):
    queue = deque()
    for i in range(1, size+1):
        queue.append(str(i))

    while queue:
        string = queue.popleft()
        if len(string) == size:
            ans.append(string)
        else:
            for i in range(1, size+1):
                if str(i) not in string and isVaild(string, str(i)):
                    queue.append(string+str(i))

bfs(8)
for i in range(n):
    print(ans[int(input()) - 1])

```

基本信息

#: 44879044
 题目: 02754
 提交人: 23n2300011030(陈奕好)
 内存: 3632kB
 时间: 35ms
 语言: Python3
 提交时间: 2024-05-06 14:41:09

bfs, <http://cs101.openjudge.cn/practice/03151/>

思路:

代码

```
1  from collections import deque
2
3  def bfs(A, B, C):
4      queue = deque([((0, 0), [])])
5      visited = set([(0, 0)])
6      while queue:
7          (a, b), path = queue.popleft()
8          if a == C or b == C:
9              return path
10         states = [(A, b), path + ['FILL(1)']],
11                 ((a, B), path + ['FILL(2)']],
12                 ((0, b), path + ['DROP(1)']],
13                 ((a, 0), path + ['DROP(2)']],
14                 ((a-min(a, B-b), b+min(a, B-b)), path + ['POUR(1,2)']],
15                 ((a+min(b, A-a), b-min(b, A-a)), path + ['POUR(2,1)'])
16         for state, new_path in states:
17             if state not in visited:
18                 queue.append((state, new_path))
19                 visited.add(state)
20     return None
21
22 def solve(A, B, C):
23     if A < C and B < C:
24         return "impossible"
25     path = bfs(A, B, C)
26     if path is None:
27         return "impossible"
28     return f"{len(path)}\n" + "\n".join(path)
29
30 A, B, C = map(int, input().split())
31 print(solve(A, B, C))
32
33
```

代码运行截图 (AC代码截图, 至少包含有"Accepted")

状态: Accepted

源代码

```
from collections import deque

def bfs(A, B, C):
    queue = deque([(0, 0), []])
    visited = {(0, 0)}
    while queue:
        (a, b), path = queue.popleft()
        if a == C or b == C:
            return path
        states = [(A, b), path + ['FILL(1)']],
                ((a, B), path + ['FILL(2)']],
                ((0, b), path + ['DROP(1)']],
                ((a, 0), path + ['DROP(2)']],
                ((a-min(a, B-b), b+min(a, B-b)), path + ['POUR(1,2)']],
                ((a+min(b, A-a), b-min(b, A-a)), path + ['POUR(2,1)']]
        for state, new_path in states:
            if state not in visited:
                queue.append((state, new_path))
                visited.add(state)
    return None

def solve(A, B, C):
    if A < C and B < C:
        return "impossible"
    path = bfs(A, B, C)
    if path is None:
        return "impossible"
    return f"({path[0]})\n" + "\n".join(path[1:])
```

基本信息

#: 44883430

题目: 03151

提交人: 23n2300011030(陈奕好)

内存: 3676kB

时间: 22ms

语言: Python3

提交时间: 2024-05-06 21:44:12

05907: 二叉树的操作

<http://cs101.openjudge.cn/practice/05907/>

思路: 用字典模拟树

代码

```
1  for turn in range(int(input())):
2      n, m = map(int, input().split())
3      nodes = {i: (-1, -1) for i in range(n)}
4
5      for node in range(n):
6          X, Y, Z = map(int, input().split())
7          nodes[X] = (Y, Z)
8
9      for operation in range(m):
10         op = list(map(int, input().split()))
11         if op[0] == 1:
12             x, y = op[1], op[2]
13             for node in nodes:
14                 if nodes[node][0] == x:
15                     nodes[node] = (y, nodes[node][1])
```

```

16         elif nodes[node][0] == y:
17             nodes[node] = (x, nodes[node][1])
18
19         if nodes[node][1] == x:
20             nodes[node] = (nodes[node][0], y)
21         elif nodes[node][1] == y:
22             nodes[node] = (nodes[node][0], x)
23
24     elif op[0] == 2:
25         x = op[1]
26         while nodes[x][0] != -1:
27             x = nodes[x][0]
28         print(x)
29

```

代码运行截图 (AC代码截图, 至少包含有"Accepted")

#44883630提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```

for turn in range(int(input())):
    n, m = map(int, input().split())
    nodes = {i: (-1, -1) for i in range(n)}

    for node in range(n):
        X, Y, Z = map(int, input().split())
        nodes[X] = (Y, Z)

    for operation in range(m):
        op = list(map(int, input().split()))
        if op[0] == 1:
            x, y = op[1], op[2]
            for node in nodes:
                if nodes[node][0] == x:
                    nodes[node] = (y, nodes[node][1])
                elif nodes[node][0] == y:
                    nodes[node] = (x, nodes[node][1])

                if nodes[node][1] == x:
                    nodes[node] = (nodes[node][0], y)
                elif nodes[node][1] == y:
                    nodes[node] = (nodes[node][0], x)

        elif op[0] == 2:
            x = op[1]
            while nodes[x][0] != -1:
                x = nodes[x][0]
            print(x)

```

基本信息

#: 44883630
 题目: 05907
 提交人: 23n2300011030(陈奕好)
 内存: 3728kB
 时间: 288ms
 语言: Python3
 提交时间: 2024-05-06 22:05:26

18250: 冰阔落 I

Disjoint set, <http://cs101.openjudge.cn/practice/18250/>

思路：优化了union，使其成为了判断条件

代码

```
1 class DisjSet:
2     def __init__(self, n):
3         self.rank = [1] * (n+1)
4         self.parent = [(i) for i in range(n+1)]
5
6     def find(self, x):
7         if self.parent[x] != x:
8             self.parent[x] = self.find(self.parent[x])
9         return self.parent[x]
10
11    def union(self, x, y):
12        x_root, y_root = self.find(x), self.find(y)
13        if x_root != y_root:
14            self.parent[y_root] = x_root
15            self.rank[x_root] += 1
16            return False
17        return True
18
19
20 while True:
21     try:
22         n, m = map(int, input().split())
23         ds = DisjSet(n)
24         for i in range(m):
25             x, y = map(int, input().split())
26             if ds.union(x, y):
27                 print("Yes")
28             else:
29                 print("No")
30
31         ds.parent[1:] = [ds.find(i) for i in ds.parent[1:]]
32         print(len(set(ds.parent[1:])))
33         print(" ".join(str(i) for i in sorted(list(set(ds.parent[1:])))
34
35     except EOFError:
36         break
37
```

代码运行截图 (AC代码截图，至少包含有"Accepted")

状态: Accepted

源代码

```
class DisjSet:
    def __init__(self, n):
        self.rank = [1] * (n+1)
        self.parent = [(i) for i in range(n+1)]

    def find(self, x):
        if self.parent[x] != x:
            self.parent[x] = self.find(self.parent[x])
        return self.parent[x]

    def union(self, x, y):
        x_root, y_root = self.find(x), self.find(y)
        if x_root != y_root:
            self.parent[y_root] = x_root
            self.rank[x_root] += 1
            return False
        return True

while True:
    try:
        n, m = map(int, input().split())
        ds = DisjSet(n)
        for i in range(m):
            x, y = map(int, input().split())
            if ds.union(x, y):
                print("Yes")
```

基本信息

#: 44679615
题目: 18250
提交人: 23n2300011030(陈奕好)
内存: 6804kB
时间: 386ms
语言: Python3
提交时间: 2024-04-16 23:06:35

05443: 兔子与樱花

<http://cs101.openjudge.cn/practice/05443/>

思路: dijkstra

代码

```
1 import heapq
2
3 def dijkstra(graph, start):
4     distances = {node: (float('infinity'), []) for node in graph}
5     distances[start] = (0, [start])
6     queue = [(0, start, [start])]
7     while queue:
8         current_distance, current_node, path = heapq.heappop(queue)
9         if current_distance > distances[current_node][0]:
10             continue
11         for neighbor, weight in graph[current_node].items():
12             distance = current_distance + weight
13             if distance < distances[neighbor][0]:
14                 distances[neighbor] = (distance, path + [neighbor])
15                 heapq.heappush(queue, (distance, neighbor, path + [neighbor]))
16     return distances
```

```

17
18 P = int(input())
19 places = {input(): i for i in range(P)}
20 graph = {i: {} for i in range(P)}
21
22 Q = int(input())
23 for _ in range(Q):
24     place1, place2, distance = input().split()
25     distance = int(distance)
26     graph[places[place1]][places[place2]] = distance
27     graph[places[place2]][places[place1]] = distance
28
29 R = int(input())
30 for _ in range(R):
31     start, end = input().split()
32     distances = dijkstra(graph, places[start])
33     path = distances[places[end]][1]
34     result = ""
35     for i in range(len(path) - 1):
36         result += f"{list(places.keys())[list(places.values()).index(path[i])]}->"
37         result += list(places.keys())[list(places.values()).index(path[-1])]
38     print(result)
39

```

代码运行截图 (AC代码截图, 至少包含有"Accepted")

#44884256提交状态

[查看](#) [提交](#) [统计](#) [提问](#)

状态: **Accepted**

源代码

```

import heapq

def dijkstra(graph, start):
    distances = {node: (float('infinity'), []) for node in graph}
    distances[start] = (0, [start])
    queue = [(0, start, [start])]
    while queue:
        current_distance, current_node, path = heapq.heappop(queue)
        if current_distance > distances[current_node][0]:
            continue
        for neighbor, weight in graph[current_node].items():
            distance = current_distance + weight
            if distance < distances[neighbor][0]:
                distances[neighbor] = (distance, path + [neighbor])
                heapq.heappush(queue, (distance, neighbor, path + [neighbor]))
    return distances

P = int(input())
places = {input(): i for i in range(P)}
graph = {i: {} for i in range(P)}

Q = int(input())
for _ in range(Q):
    place1, place2, distance = input().split()
    distance = int(distance)
    graph[places[place1]][places[place2]] = distance
    graph[places[place2]][places[place1]] = distance

```

基本信息

#: 44884256
 题目: 05443
 提交人: 23n2300011030(陈奕好)
 内存: 3700kB
 时间: 20ms
 语言: Python3
 提交时间: 2024-05-06 23:21:26

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。

五一鸽了😓，不知道该怎么活过期末。

太焦虑了，但题目又很好玩，图的知识学的又不是太懂，做的时候很快乐，但真的不知道怎么挤出时间了。。。