Assignment #A: 图论：遍历，树算及栈

Updated 2018 GMT+8 Apr 21, 2024

2024 spring, Complied by 钟明衡 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。

编程环境

操作系统：Windows_NT x64 10.0.19045

Python编程环境：Visual Studio Code 1.76.1

C/C++编程环境：Visual Studio Code 1.76.1

1. 题目

20743: 整人的提词本

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

思路：

括号会把里面的内容给反转，因此，将最外层括号中的内容用栈提取出来，用递归的方式反转并按相同规则处理后，放回原来的位置

需要注意，正序的括号起始是'('，反过来就是')'，为了方便，用两个字符串l、r表示，每叠加一层括号就交换l和r

代码

```
def read(s, l, r):
    word, count, stack = '', 0, []
    for a in s:
        if a == r:
            count -= 1
        if count:
```

```
            stack.append(a)
        if a == l:
            count += 1
        if stack and not count:
            word += read(stack[::-1], r, l)
            stack = []
        if count == 0 and a not in '()':
            word += a
    if stack:
        word += read(stack[::-1], r, l)
    return word


s = input()
print(read(s, '(', ')'))

```

代码运行截图

#44317195提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
def read(s, l, r):
    word, count, stack = '', 0, []
    for a in s:
        if a == r:
            count -= 1
        if count:
            stack.append(a)
        if a == l:
            count += 1
        if stack and not count:
            word += read(stack[::-1], r, l)
            stack = []
        if count == 0 and a not in '()':
            word += a
    if stack:
        word += read(stack[::-1], r, l)
    return word


s = input()
print(read(s, '(', ')'))
```

基本信息

#: 44317195
题目: 20743
提交人: 23n2300011505(12号娱乐选手)
内存: 3604kB
时间: 23ms
语言: Python3
提交时间: 2024-03-20 20:12:06

02255: 重建二叉树

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

思路:

不同遍历顺序只有root、left和right相对位置变化，前序的第一个必定为root，在中序中找到root，则可以得到left和right的前序和中序

把root放在后序的left和right后面，一直递归就可以了

代码

```
def build(a, b):
    if a:
        root = a[0]
        n = b.index(a[0])
        la = a[1:n+1]
        lb = b[:n]
        ra = a[n+1:]
        rb = b[n+1:]
        return build(la, lb)+build(ra, rb)+root
    else:
        return ''


while True:
    try:
        a, b = input().split()
    except EOFError:
        break
    print(build(a, b))

```

代码运行截图

#44045336提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
def build(a, b):
    if a:
        root = a[0]
        n = b.index(a[0])
        la = a[1:n+1]
        lb = b[:n]
        ra = a[n+1:]
        rb = b[n+1:]
        return build(la, lb)+build(ra, rb)+root
    else:
        return ''


while True:
    try:
        a, b = input().split()
    except EOFError:
        break
    print(build(a, b))
```

基本信息

#: 44045336
题目: 02255
提交人: 23n2300011505(12号娱乐选手)
内存: 3532kB
时间: 24ms
语言: Python3
提交时间: 2024-03-03 00:33:47

01426: Find The Multiple

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

要求用bfs实现

思路:

先把输入变成奇数，除掉的2的数量，在最后用'0'补全，然后用bfs从0开始逐位添加0或者1，查找可以整除的数

代码

```
def bfs(n):
    l = [0]
    s, e = 0, 1
    while s != e:
        for i in range(s, e):
            for j in (0, 1):
                x = l[i]*10+j
                if x:
                    if x % n:
                        l.append(x)
                    else:
                        return str(x)
        s, e = e, len(l)
    return ''


while (n := int(input())):
    c = 0
    while (n+1) % 2:
        n //= 2
        c += 1
    print(bfs(n)+'0'*c)

```

代码运行截图

#44749934提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
def bfs(n):
    l = [0]
    s, e = 0, 1
    while s != e:
        for i in range(s, e):
            for j in (0, 1):
                x = l[i]*10+j
                if x:
                    if x % n:
                        l.append(x)
                    else:
                        return str(x)
        s, e = e, len(l)
    return ''


while (n := int(input())):
    c = 0
    while (n+1) % 2:
        n //= 2
        c += 1
    print(bfs(n)+'0'*c)
```

基本信息

#: 44749934
题目: 01426
提交人: 23n2300011505(12号娱乐选手)
内存: 18212kB
时间: 334ms
语言: Python3
提交时间: 2024-04-22 11:32:35

04115: 鸣人和佐助

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

思路:

用一个特殊的bfs，除了记录位置，还要记录剩下的查克拉数量，某个位置能走的前提是，这次走到这个位置所剩下的查克拉，要比上一次来的时候多（初始为-1），其他的和正常bfs相同

代码

```
m, n, t = map(int, input().split())
M = [input() for _ in range(m)]
x = y = f = 0
for i in range(m):
    for j in range(n):
        if M[i][j] == '@':
            x, y = i, j
            f = 1
            break
    if f:
        break
s, e = 0, 1
l = [(x, y, t)]
g = [[-1]*n for i in range(m)]
g[x][y] = t
c = 0
dx, dy = [1, 0, -1, 0], [0, 1, 0, -1]
while s != e:
```

```
    c += 1
    for i in range(s, e):
        for j in range(4):
            x, y, k = l[i][0]+dx[j], l[i][1]+dy[j], l[i][2]
            if x in (-1, m) or y in (-1, n):
                continue
            if M[x][y] == '+':
                print(c)
                exit()
            elif M[x][y] == '#':
                if k-1 > g[x][y]:
                    g[x][y] = k-1
                    l.append((x, y, k-1))
            elif M[x][y] == '*':
                if k > g[x][y]:
                    g[x][y] = k
                    l.append((x, y, k))
    s, e = e, len(l)
print(-1)

```

代码运行截图

#44750714提交状态 查看 提交 统计 提问

状态: Accepted

基本信息
#: 44750714
题目: 04115
提交人: 23n2300011505(12号娱乐选手)
内存: 5716kB
时间: 115ms
语言: Python3
提交时间: 2024-04-22 13:13:48

源代码

```
m, n, t = map(int, input().split())
M = [input() for _ in range(m)]
x = y = f = 0
for i in range(m):
    for j in range(n):
        if M[i][j] == '@':
            x, y = i, j
            f = 1
            break
    if f:
        break
s, e = 0, 1
l = [(x, y, t)]
g = [[-1]*n for i in range(m)]
g[x][y] = t
c = 0
dx, dy = [1, 0, -1, 0], [0, 1, 0, -1]
while s != e:
    c += 1
    for i in range(s, e):
        for j in range(4):
            x, y, k = l[i][0]+dx[j], l[i][1]+dy[j], l[i][2]
            if x in (-1, m) or y in (-1, n):
                continue
            if M[x][y] == '+':
                print(c)
                exit()
            elif M[x][y] == '#':
                if k-1 > g[x][y]:
                    g[x][y] = k-1
                    l.append((x, y, k-1))
            elif M[x][y] == '*':
                if k > g[x][y]:
                    g[x][y] = k
                    l.append((x, y, k))
    s, e = e, len(l)
print(-1)
```

20106: 走山路

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

思路1:

很久以前用特殊的bfs过了，当时的思路是，走过的地方还可以继续走，只要走到那里消耗的体力比之前要低

代码

```
dx, dy = [1, 0, -1, 0], [0, 1, 0, -1]


def bfs(x, y, a, b):
    global M, ans
    lx, ly = [x], [y]
    start, end = 0, 0
    while end != len(lx):
        start = end
```

```
        end = len(lx)
        for i in range(start, end):
            for j in range(4):
                newx, newy = lx[i]+dx[j], ly[i]+dy[j]
                if M[newx][newy] != '#':
                    newans = ans[lx[i]][ly[i]] + \
                        abs(int(M[lx[i]][ly[i]])-int(M[newx][newy]))
                    if ans[newx][newy] == -1 or newans < ans[newx][newy]:
                        ans[newx][newy] = newans
                        lx.append(newx)
                        ly.append(newy)
    return


def intt(s):
    return int(s)+1


m, n, p = map(int, input().split())
M = [['#']*(n+2) for i in range(m+2)]
for i in range(m):
    M[i+1] = ['#']+input().split()+['#']
for _ in range(p):
    x, y, a, b = map(intt, input().split())
    if M[x][y] == '#' or M[a][b] == '#':
        print('NO')
    else:
        ans = [[-1]*(n+2) for i in range(m+2)]
        ans[x][y] = 0
        bfs(x, y, a, b)
        if ans[a][b] == -1:
            print('NO')
        else:
            print(ans[a][b])

```

代码运行截图

#43224424提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
dx, dy = [1, 0, -1, 0], [0, 1, 0, -1]


def bfs(x, y, a, b):
    global M, ans
    lx, ly = [x], [y]
    start, end = 0, 0
    while end != len(lx):
        start = end
        end = len(lx)
        for i in range(start, end):
            for j in range(4):
                newx, newy = lx[i]+dx[j], ly[i]+dy[j]
                if M[newx][newy] != '#':
                    newans = ans[lx[i]][ly[i]] + \
                        abs(int(M[lx[i]][ly[i]])-int(M[newx][newy]))
                    if ans[newx][newy] == -1 or newans < ans[newx][newy
                        ans[newx][newy] = newans
                        lx.append(newx)
                        ly.append(newy)
    return


def intt(s):
    return int(s)+1


m, n, p = map(int, input().split())
M = [['#']*(n+2) for i in range(m+2)]
for i in range(m):
    M[i+1] = ['#']+input().split()+['#']
for _ in range(p):
    x, y, a, b = map(intt, input().split())
    if M[x][y] == '#' or M[a][b] == '#':
        print('NO')
    else:
        ans = [[-1]*(n+2) for i in range(m+2)]
        ans[x][y] = 0
        bfs(x, y, a, b)
        if ans[a][b] == -1:
            print('NO')
        else:
            print(ans[a][b])
```

基本信息

#: 43224424
题目: 20106
提交人: 23n2300011505(12号娱乐选手)
内存: 3888kB
时间: 1023ms
语言: Python3
提交时间: 2023-12-19 15:56:37

思路2:

用Dijkstra，每次都走能走到的位置中消耗最低的位置，为了防止超时，如果一个位置不能往别处走了，就删掉

代码

```
dx, dy = [1, 0, -1, 0], [0, 1, 0, -1]


def bfs(l, a, b):
    global M, ans
    while True:
        N = set()
        S = set()
        c = -1
        for x, y in l:
```

```
            f = 1
            for k in range(4):
                nx, ny = x+dx[k], y+dy[k]
                if M[nx][ny] != '#' and ans[nx][ny] == -1:
                    f = 0
                    newans = ans[x][y] + abs(int(M[x][y])-int(M[nx][ny]))
                    if c == -1 or newans < c:
                        c = newans
                        N = set()
                    if newans == c:
                        N.add((nx, ny))
            if f:
                S.add((x, y))
        if N:
            l |= N
            l -= S
            for Nx, Ny in N:
                ans[Nx][Ny] = c
                if Nx == a and Ny == b:
                    return
        else:
            return


def intt(s):
    return int(s)+1


m, n, p = map(int, input().split())
l = ['#']
M = [l*(n+2)]+[l+input().split()+l for _ in range(m)]+[l*(n+2)]
for _ in range(p):
    x, y, a, b = map(intt, input().split())
    ans = [[-1]*(n+2) for i in range(m+2)]
    ans[x][y] = 0
    if M[x][y] == '#' or M[a][b] == '#':
        print('NO')
        continue
    bfs({(x, y)}, a, b)
    if ans[a][b] == -1:
        print('NO')
    else:
        print(ans[a][b])

```

代码运行截图

#44749705提交状态 查看 提交 统计 提问

状态: Accepted

基本信息

#: 44749705
题目: 20106
提交人: 23n2300011505(12号娱乐选手)
内存: 3784kB
时间: 969ms
语言: Python3
提交时间: 2024-04-22 11:11:40

源代码

```
dx, dy = [1, 0, -1, 0], [0, 1, 0, -1]


def bfs(l, a, b):
    global M, ans
    while True:
        N = set()
        S = set()
        c = -1
        for x, y in l:
            f = 1
            for k in range(4):
                nx, ny = x+dx[k], y+dy[k]
                if M[nx][ny] != '#' and ans[nx][ny] == -1:
                    f = 0
                    newans = ans[x][y] + abs(int(M[x][y])-int(M[nx][ny]
                    if c == -1 or newans < c:
                        c = newans
                        N = set()
                    if newans == c:
                        N.add((nx, ny))
            if f:
                S.add((x, y))
        if N:
            l |= N
            l -= S
            for Nx, Ny in N:
                ans[Nx][Ny] = c
                if Nx == a and Ny == b:
                    return
        else:
            return


def intt(s):
    return int(s)+1


m, n, p = map(int, input().split())
l = ['#']
M = [l*(n+2)]+[l+input().split()+l for _ in range(m)]+[l*(n+2)]
for _ in range(p):
    x, y, a, b = map(intt, input().split())
    ans = [[-1]*(n+2) for i in range(m+2)]
    ans[x][y] = 0
    if M[x][y] == '#' or M[a][b] == '#':
        print('NO')
        continue
    bfs({(x, y)}, a, b)
    if ans[a][b] == -1:
        print('NO')
    else:
        print(ans[a][b])
```

05442: 兔子与星空

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

思路:

最小生成树，按权重从小到大的顺序，判断一条边是否有必要添加（利用并查集）

代码

```python
d, p = [], {}


def F(x):
    if p[x] != x:
        p[x] = F(p[x])
    return p[x]


for _ in range(int(input())-1):
    s = input().split()
    a = s[0]
    p[a] = a
    for i in range(int(s[1])):
        d.append((a, s[2+2*i], int(s[3+2*i])))
        p[s[2+2*i]] = s[2+2*i]
d.sort(key=lambda x: x[2])
ans = 0
for u, v, x in d:
    pu, pv = F(u), F(v)
    if pu != pv:
        p[pu] = pv
        ans += x
print(ans)

```

代码运行截图

#44753899提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```python
d, p = [], {}


def F(x):
    if p[x] != x:
        p[x] = F(p[x])
    return p[x]


for _ in range(int(input())-1):
    s = input().split()
    a = s[0]
    p[a] = a
    for i in range(int(s[1])):
        d.append((a, s[2+2*i], int(s[3+2*i])))
        p[s[2+2*i]] = s[2+2*i]
d.sort(key=lambda x: x[2])
ans = 0
for u, v, x in d:
    pu, pv = F(u), F(v)
    if pu != pv:
        p[pu] = pv
        ans += x
print(ans)
```

基本信息

#: 44753899
题目: 05442
提交人: 23n2300011505(12号娱乐选手)
内存: 3660kB
时间: 27ms
语言: Python3
提交时间: 2024-04-22 19:12:07

2. 学习总结和收获

通过作业题复习了栈和遍历二叉树，以及特殊条件的bfs

这一类特殊bfs的一般思路可以概括为：不能“两败俱伤”，即，如果将要走到一个走过的节点时，耗费的能量比之前到这里时还多，就不能走，因为这是必定亏损的。要实现这个，可以在记录走过位置的表中，记录上次走到这个位置所耗费的能量，下次走到一个走过的点，如果更省能量，就允许走这一步，走完后记得更新

这种bfs和Dijstra可以解决的问题挺像的，不过后者的思路是，挑最节约的下一步去走，每个位置只会走一次，根据实际情况选择用哪种

最小生成树的思路很好理解，感觉这种算法很实用，比如城市之间铺设电缆

许多问题说到底就是贪心，怎么多贪的法子，雅称“算法”，其中主流的两种方法是“稳赚”和“不亏”，根据这两个，可以理解很多算法，比如Dijkstra每次只走权重最小的边就是“稳赚”，bfs不允许步数和消耗同时增大就是“不亏”

总是说“贪心”就好像程序员是什么坏人一样，还是“算法”比较好听