Assignment #C: 五味杂陈

Updated 1148 GMT+8 Dec 10, 2024

2024 fall, Complied by <mark>同学的姓名、院系</mark>

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

1115. 取石子游戏

dfs, https://www.acwing.com/problem/content/description/1117/

思路：这道题一开始没有想到很好的思路，直到我看到底下的提示：若 a>=b，且 a/b>=2，则先手必赢（这是因为：假设 r 为 a 除以 b 的余数，则先手者有两个方案：可以剩下 r,b 或者 r+b,b；而这两个方案中后手都只有一个选择，并且这两种选法的结果一定是不同的，所以一定有一种先手是必赢的，因此先手有必胜策略）；
想清楚这一点的原理之后，就直接用 dfs 方法来做就行了~（用一个 global 变量 status 来记录每一局的执子者，一旦有一局出现 a 大于等于 2b 或者 a 等于 b 就立刻结束，并利用此时的 status 的奇偶性来判断是谁拿完的这一堆石子即可~）

代码：

```python

```

```
def game(a, b):
    global status
    status += 1
    c, d = max(a, b), min(a, b)
```

```
    if c >= 2*d or c == d:
        return
    else:
        game(c - d, d)

while True:
    a, b = map(int,input().split())
    if a == 0 and b == 0:
        break
    else:
        status = 0 #记录这是第几个人的回合（mod2 余几便是谁）
        game(a, b)
        if status % 2 == 1:
            print("win")
        else:
            print("lose")
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

25570: 洋葱

Matrices, http://cs101.openjudge.cn/practice/25570

思路：很简单的一道题：先建立一个含有 n 个 0 的列表 level，其中 level 的第 i 个数代表第 i+1 层之和；随后，再对矩阵的第 i,j 个数判断其属于第几层，并将其加到对应的层数之和上；最后，再找出 level 列表中的最大的数即可~

代码：

```python

```

```
n = int(input())
level = [0]*n #第 i 个数代表第 i+1 层之和
matrix = []
for i in range(n):
    line = list(map(int,input().split()))
    matrix.append(line)
for i in range(n):
    for j in range(n): #对第 i,j 个数判断其属于第几层
        k = min(i, j , n - 1 - i, n - 1 - j)
        level[k] += matrix[i][j]
print(max(level))
```

代码运行截图 ==（至少包含有"Accepted"）==

1526C1. Potions(Easy Version)

greedy, dp, data structures, brute force, *1500, https://codeforces.com/problemset/problem/1526/C1

思路：建立一个 1 乘 n 的 dp 列表，其中第 j 个数表示此刻吃 j 个药水所能够得到的最大生命值（-1 则代表不存在）；初始化 t=0，用于记录记录此时的最大的能吃的药水数（从 0 开始算起）；每当可以加入一个新的药水的时候 t 加 1。
每时每刻增加一个进入考虑范围的药水，并时刻更新这个 dp 列表；最后，考虑完所有药水后得到的 t 就是最终的答案！

代码：

```python

```

```
n = int(input())
potions = list(map(int,input().split()))
dp = [-1]*(n) #每个时刻的第 j 个数表示此刻吃 j 个药水所能够得到的最大生命值（-1 则代表不存在）
t = 0 #记录此时的最大的能吃的药水数（从 0 开始算起）
for i in range(n): #一个一个药水进行考虑
    if t == 0:
        if potions[i] >= 0: #如果还没吃过药水，且当下这个药水可以吃
            dp[0] = potions[i]
            t += 1 #对最大药水数加一
    elif t > 0:
        #先讨论最后一列（一定要倒序！这样才不会使前面的对后面的有影响）
        status = False
        if potions[i] + dp[t - 1] >= 0: #则说明可以增加一个最大药水数了
            dp[t] = potions[i] + dp[t - 1]
            status = True #对最大药水数加一~
        #再讨论中间的列
        for j in range(t - 1,0,-1):
            if potions[i] + dp[j - 1] > dp[j]: #则说明需要更新这个数了
                dp[j] = potions[i] + dp[j - 1]
        #最后再讨论第一列
        if potions[i] > dp[0]: #说明需要更新
            dp[0] = potions[i]
        #最后检查 t 是否需要加 1
        if status == True:
            t += 1

print(t) #全部执行完之后，t 就是答案啦！
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

22067: 快速堆猪

辅助栈，http://cs101.openjudge.cn/practice/22067/

思路：建立一个辅助栈，栈顶是每时每刻的最小值即可~
（注意，同一个最小值只能压入一次！否则如果每时每刻无论新加入的元素是否为最小值都更新这个最小值的话，可能会导致栈顶有一堆远古时候的某个最小值 a，一旦把那个元素给 pop 了，辅助栈的栈顶还是会剩一堆 a，会出问题！）

代码：

```python

```

```
stack = []
min_stack = [] #辅助栈，栈顶是每时每刻的最小值
#小心：最小值不要重复压入！这样可能会导致栈顶有一堆远古时候的某个最小值 a，
#一旦把那个元素给 pop 了，还是会剩一堆 a，会出问题！
#（因此，同一个最小值只能压入一次！）
```

```
while True:
    try:
        command = input().split()
        if command[0] == 'pop':
            if stack:
                a = stack.pop()
                if a == min_stack[-1]:
                    min_stack.pop()
        if command[0] == 'push':
            a = int(command[1])
            stack.append(a)
            if min_stack:
                if a <= min_stack[-1]: #只压入一次~
                    min_stack.append(a)
            else:
                min_stack.append(a)
        if command[0] == 'min':
            if min_stack:
                print(min_stack[-1])
    except EOFError:
        break
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

20106: 走山路

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

思路：使用经典的 dijkstra 方法即可~
（用 dist 列表记录从出发点到达每一点的最短距离，再用一个 heapq 记录当下每个到达的点的所需要的距离和他们的坐标，最后再使用 bfs 拓展当下到达的点并检查：1.是否已到达终点；如果是的话则把 status 记录为 True；2.此时的距离是否已经大于已有的 min，如果是的话则可以扔掉这种情况；最后，遍历完所有能够到达的点之后，检查 status 是否为 True，如果是的话则说明可以到达终点，则 print 结果即可！）

代码：

```python
```
```
import heapq

def walk(a, b, c, d):
    if mountains[a][b] == '#' or mountains[c][d] == '#':
        print("NO")
    else:
        dist = [[float('inf')]*n for _ in range(m)] #记录从出发点到达每一点的最短距离
        dist[a][b] = 0
        queue = [(0,a,b)] #记录当下每个到达的点的所需要的距离和他们的坐标
        status = False #检查是否能够到达终点

        while queue:
            cur_dis, x, y = heapq.heappop(queue) #将 queue 视为堆，并 pop 出第一个数（也就是距离）最小的点
            if x == c and y == d:
                status = True #如果已经到达终点，则记录 True 表示确实可以到达终点
                continue

            if cur_dis > dist[c][d]: #如果此时的距离已经大于已有的 min，则可以扔掉这种情况了
                continue

            directions = [(0,1),(0,-1),(1,0),(-1,0)]
            for p, q in directions:
                nx, ny = x + p, y + q
                if nx in range(m) and ny in range(n) and mountains[nx][ny] != '#':
                    now_dis = cur_dis + abs(int(mountains[x][y]) - int(mountains[nx][ny])) #到达这个新点的距离
```

                    if now_dis < dist[nx][ny]: #如果这是迄今为止到达这个点的最好的方法，则更新它
                        dist[nx][ny] = now_dis
                        heapq.heappush(queue,(now_dis, nx, ny)) #将这个新点加入heapq

        if status == True: #最后，检查是否能够到达终点
            print(dist[c][d])
        else:
            print("NO")

```
m, n, p = map(int,input().split())
mountains = []
for i in range(m):
    mountains.append(list(map(str,input().split())))
for i in range(p):
    a, b, c, d = map(int,input().split())
    walk(a, b, c, d)
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

04129: 变换的迷宫

bfs, http://cs101.openjudge.cn/practice/04129/

思路：用常规的 bfs 方法来做即可。
需要注意的是：不能保留所有的走的路径，这样会导致超时；但是也不能仅仅保留到到达每个格子的最优路径，因为这样可能会损失一些方法，导致这个迷宫根本走不出去。
解决方法是：保留每一个点在 modk 意义下的最优方法即可～

代码：

```python
```

```
from collections import deque

def walk(maze, a, b, c, d, r, l, k):
    visited = {(0, a, b)} #用 set 来记录记录到达每个点所需要的时间（只保留 modk 意义下最好的方法）
    directions = [(0,1),(0,-1),(1,0),(-1,0)]
    queue = deque([(0, a, b)]) #记录当下每个到达的点的到达时间和他们的坐标
    while queue:
        time, x, y = queue.popleft()
        for dx, dy in directions:
            nx, ny = x + dx, y + dy
            new_time = time + 1
            if nx in range(r) and ny in range(l) and (new_time % k, nx, ny) not in visited:
                if nx == c and ny == d: #到达终点则立刻 return
                    return new_time
                elif maze[nx][ny] != '#' or new_time % k == 0: #如果这个格子此时不是墙
                    queue.append((new_time, nx, ny))
                    visited.add((new_time % k, nx, ny))

    return 'Oop!' #如果 queue 里已经没有点了还没有 return，则说明无可行路径

n = int(input())
for i in range(n):
    r, l, k = map(int,input().split())
    maze = []
    for j in range(r):
        row = list(map(str,input()))
        if 'S' in row:
            a, b = j, row.index('S')
        if 'E' in row:
            c, d = j, row.index('E')
        maze.append(row)
```

```
print(walk(maze, a, b, c, d, r, l, k))
```

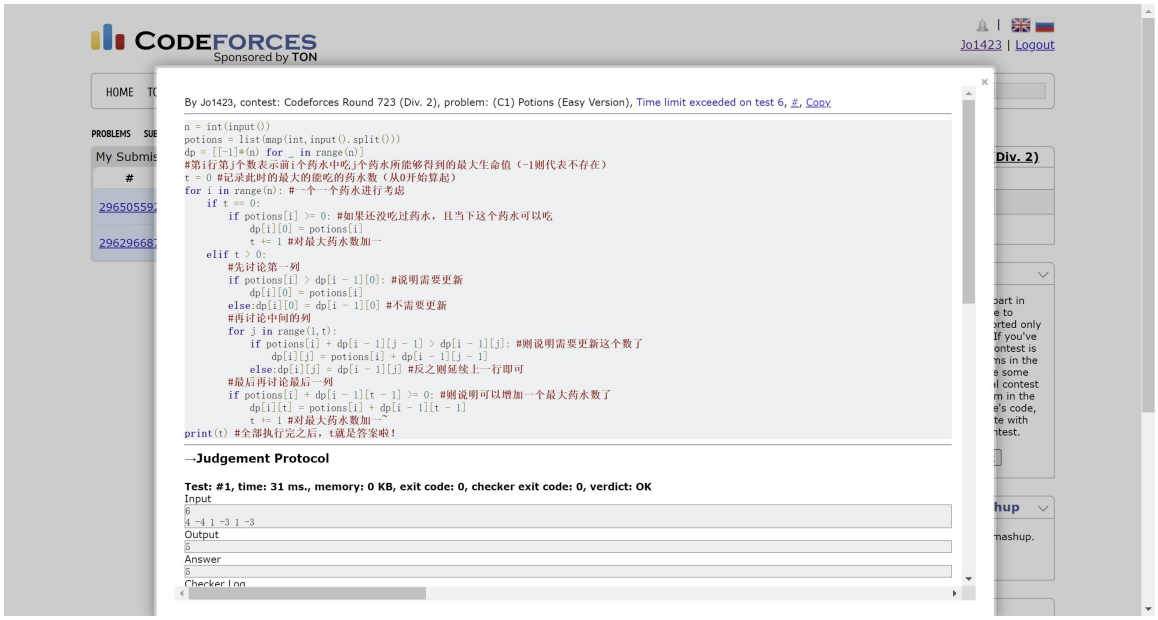
代码运行截图 <mark>（至少包含有"Accepted"）</mark>

2. 学习总结和收获

<mark>如果作业题目简单，有否额外练习题目，比如：OJ“计概 2024fall 每日选做”、CF、LeetCode、洛谷等网站题目。</mark>

这次作业的总体难度感觉还是比较简单的，至少感觉比上次简单多了，不至于每道题都要做好久才能勉强做出来或者根本做不出来。。

其中第三题有一个神奇的地方是，我一开始用的是一个 nxn 的 dp 表来做的这道题，思路就是常规的用第 i 行第 j 个数表示前 i 个药水中吃 j 个药水所能够得到的最大生命值；但是，明明复杂度是 on^2，n 也小于 2000，却会超时。。（如下图）



后来，思来想去，我还是没想到一个好的方法，于是直接一不做二不休直接把这个列表变成了 1xn 成的 dp 列表，方法完全没变，时间复杂度还是 on^2，结果就过了，笑死，不知道为什么（复杂度和方法都完全一样，为什么会一个 TLE 一个 AC 呢？）

第五题也蛮有意思的，虽然我做出来这道题并没有花多久，方法也是我自己想出来的；但是做完之后我才意外的发现，自己使用的方法其实就是自己之前并没有学过的大名鼎鼎的 Dijkstra 方法！（完全靠自己想出来一个有别人名字的方法还是蛮有成就感的哈哈哈）。不过有一说一，其实感觉这个方法本质上就是 BFS，代码思路没有什么区别，只不过往里面融入了 heapq 方法来减少时间复杂度而已。。

第六题做起来倒是出乎意料的轻松。我一开始使用的是最基础的 BFS 方法（就是直接保留所有的路径，看最终能不能走出去），但是这样可惜会 TLE。后来我又试图仅仅保留走到每个格子的最优路径，但是通过对已给案例的测试，发现这样会导致迷宫根本走不出去，因为会损失很多方法。所以经过一些思考，我就发现需要保留且仅保留走到每个格子的在 modk 意义下的 k 个最优路径，这样既不会损失已有的方法，又可以大幅减少时间复杂度。然后不出所料，就直接 AC 了。总的来讲，这道题并不算简单，但对于第六题这个位置而言确实不算难；希望自己能继续加油，机考考出好成绩~