Assignment #D: 十全十美

Updated 1254 GMT+8 Dec 17, 2024

2024 fall, Complied by 陈冠宇 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。

1. 题目

02692: 假币问题

brute force, http://cs101.openjudge.cn/practice/02692

思路：先记录每次测量的的结果，然后看了好几组测试数据找规律...（投机取巧写法

后来才发现穷举只有24种情况。。。白费我写那么久。不过我这个算法稍加修改应该可以处理任意数量硬币比较任意次的结果（只要答案存在）

代码：

```
def check_coin(xx):
    ans=[]
    if xx[0]=='none' and xx[1]=='none' and xx[2]=='none':
        return (-1,-1)
    for i in range(0,3):
        if xx[i]=='even':
            return (-1,-1)
    heavy_counter=0
    light_counter=0
    for i in range(0,3):

        if xx[i]=='heavy':
            heavy_counter+=1
        elif xx[i]=='light':
            light_counter+=1
    if heavy_counter==0 or light_counter==0:
        return(heavy_counter,light_counter)
    else:
        return(-1,-1)

def mygo(time,coins,t):
    if time[2]=='even':
```

```
        for i in range(0,4):
            coins[ord(time[0][i])-65][t]='even'
            coins[ord(time[1][i])-65][t]='even'
            used_coins[ord(time[0][i]) - 65] = True
            used_coins[ord(time[1][i]) - 65] = True
    if time[2]=='up':
        for i in range(0,4):
            coins[ord(time[0][i])-65][t]='heavy'
            coins[ord(time[1][i])-65][t]='light'
            used_coins[ord(time[0][i]) - 65] = True
            used_coins[ord(time[1][i]) - 65] = True
    elif time[2]=='down':
        for i in range(0,4):
            coins[ord(time[0][i]) - 65][t] = 'light'
            coins[ord(time[1][i]) - 65][t] = 'heavy'
            used_coins[ord(time[0][i]) - 65] = True
            used_coins[ord(time[1][i]) - 65] = True

n=int(input())
for xx in range(0,n):
    coins=[['none' for _ in range(3)] for _ in range(12)]
    used_coins=[False]*12
    t=0
    for i in range(0,3):
        mygo(input().split(),coins,t)
        t+=1
    #for i in range(0,12):
        #print(chr(i+65),':',coins[i])
    #for i in range(0,12):
        #print(check_coin(coins[i]))

    h,l=-1,-1
    hi=-1
    li=-1
    current_max_h = 0
    current_max_l = 0
    for i in range(0,12):
        tt=check_coin(coins[i])
        if tt!=(-1,-1):
            h,l=tt
        if h>current_max_h:
            current_max_h=h
            hi=i
        if l>current_max_l:
            current_max_l=l
            li=i
    #print(current_max_h,current_max_l)
    if current_max_h>current_max_l:
        print(str(chr(hi+65))+' is the counterfeit coin and it is heavy.')
    else:
        print(str(chr(li + 65)) + ' is the counterfeit coin and it is light.')
```

代码运行截图 （至少包含有"Accepted"）

01088: 滑雪

dp, dfs similar, http://cs101.openjudge.cn/practice/01088

思路：

先试了一下暴力对每个点进行bfs，如果走到已标记过的位置，说明后面的路不用再走了，可以直接计算出这条路“到尾”的长度。这样550ms左右能ac

后来想着拿最小堆优化一下，从最低点开始往上找，能压缩在65ms

在

代码：

```
from collections import deque
import heapq
def bfs(start_x,start_y,chizu,crychic,r,c):
    qq=deque([(1,start_x,start_y)])
    directions=[(1,0),(0,1),(-1,0),(0,-1)]
    max_length=0
    while qq:
        current_length,x,y=qq.pop()
        max_length = max(current_length,max_length)
        for dx,dy in directions:
            nx,ny=x+dx,y+dy
            if 0<=nx<r and 0<=ny<c and chizu[x][y]>chizu[nx][ny] and crychic[nx]
[ny]==0:
                qq.append((current_length+1,nx,ny))
            elif 0<=nx<r and 0<=ny<c and chizu[x][y]>chizu[nx][ny] and
crychic[nx][ny]!=0:
                max_length=max(current_length+crychic[nx][ny],max_length)
    return max_length

r,c=map(int,input().split())
chizu=[]
for i in range(r):
    chizu.append(list(map(int,input().split())))
crychic=[[0 for _ in range(c)] for _ in range(r)]

answer=0
pq=[]

for i in range(r):
    for j in range(c):
        heapq.heappush(pq,(chizu[i][j],i,j))
for i in range(r*c):
    m,x,y=heapq.heappop(pq)
    crychic[x][y]=bfs(x,y,chizu,crychic,r,c)
    answer=max(answer,crychic[x][y])
print(answer)
```

代码运行截图 ==（至少包含有"Accepted"）==

#47820711提交状态

查看 提交 统计 提问

状态: Accepted

源代码

```
from collections import deque
import heapq
def bfs(start_x,start_y,chizu,crychic,r,c):
    qq=deque([(1,start_x,start_y)])
    directions=[(1,0),(0,1),(-1,0),(0,-1)]
    max_length=0
    while qq:
        current_length,x,y=qq.pop()
        max_length = max(current_length,max_length)
        for dx,dy in directions:
            nx,ny=x+dx,y+dy
            if 0<=nx<r and 0<=ny<c and chizu[x][y]>chizu[nx][ny] and cry
                qq.append((current_length+1,nx,ny))
            elif 0<=nx<r and 0<=ny<c and chizu[x][y]>chizu[nx][ny] and
                max_length=max(current_length+crychic[nx][ny],max_length
    return max_length

r,c=map(int,input().split())
chizu=[]
for i in range(r):
    chizu.append(list(map(int,input().split())))
crychic=[[0 for _ in range(c)] for _ in range(r)]

answer=0
pq=[]

for i in range(r):
    for j in range(c):
        heapq.heappush(pq,(chizu[i][j],i,j))
for i in range(r*c):
    m,x,y=heapq.heappop(pq)
    crychic[x][y]=bfs(x,y,chizu,crychic,r,c)
    answer=max(answer,crychic[x][y])
print(answer)
```

基本信息

#: 47820711
题目: 01088
提交人: 陈冠宇(24n2400011004)
内存: 4752kB
时间: 67ms
语言: Python3
提交时间: 2024-12-18 16:58:17

25572: 螃蟹采蘑菇

bfs, dfs, http://cs101.openjudge.cn/practice/25572/

思路:

追踪第一个发现的身体格子就行，注意一下第二个身体格子不要越界/碰到障碍物，并且寻找到终点的时候需要用到第二个格子

代码:

```
from collections import deque
def lying_crab(x,y,chizu,visited,n):
    qq=deque([(x,y)])
    visited[x][y]=True
    dirs=[(1,0),(-1,0),(0,1),(0,-1)]
    while qq:
        x,y=qq.popleft()
        #print(x,y)
        for dx,dy in dirs:
            nx,ny=x+dx,y+dy
            if 0<=nx<n and 0<=ny<n and 0<ny+1<n and not visited[nx][ny] and
chizu[nx][ny]!=1 and chizu[nx][ny+1]!=1:
                qq.append((nx,ny))
```

```
                visited[nx][ny]=True
                if chizu[nx][ny]==9 or chizu[nx][ny+1]==9:
                    return 'yes'
    return 'no'

def standing_crab(x,y,chizu,visited,n):
    qq=deque([(x,y)])
    visited[x][y]=True
    dirs=[(1,0),(-1,0),(0,1),(0,-1)]
    while qq:
        x,y=qq.popleft()
        #print(x,y)
        for dx,dy in dirs:
            nx,ny=x+dx,y+dy
            if 0<=nx<n and 0<=ny<n and 0<nx+1<n and not visited[nx][ny] and
chizu[nx][ny]!=1 and chizu[nx+1][ny]!=1:
                qq.append((nx,ny))
                visited[nx][ny]=True
                if chizu[nx][ny]==9 or chizu[nx+1][ny]==9:
                    return 'yes'
    return 'no'

n=int(input())
chizu=[]
for i in range(n):
    chizu.append(list(map(int,input().split())))
visited=[[False]*n for _ in range(n)]

w=False
for i in range(n):
    for j in range(n):
        if chizu[i][j]==5:
            start_x,start_y=i,j
            if i+1<n and chizu[i+1][j]==5:
                print(standing_crab(start_x,start_y,chizu,visited,n))
            else:
                print(lying_crab(start_x,start_y,chizu,visited,n))
            w=True
            break
    if w:
        break
```

代码运行截图 （至少包含有"Accepted"）

#47821709提交状态 查看 提交 统计 提问

状态: Accepted

基本信息

#: 47821709
题目: 25572
提交人: 陈冠宇(24n2400011004)
内存: 3784kB
时间: 26ms
语言: Python3
提交时间: 2024-12-18 17:23:35

源代码

```
from collections import deque
def lying_crab(x,y,chizu,visited,n):
    qq=deque([(x,y)])
    visited[x][y]=True
    dirs=[(1,0),(-1,0),(0,1),(0,-1)]
    while qq:
        x,y=qq.popleft()
        #print(x,y)
        for dx,dy in dirs:
            nx,ny=x+dx,y+dy
            if 0<=nx<n and 0<=ny<n and 0<ny+1<n and not visited[nx][ny]
                qq.append((nx,ny))
                visited[nx][ny]=True
                if chizu[nx][ny]==9 or chizu[nx][ny+1]==9:
                    return 'yes'
    return 'no'

def standing_crab(x,y,chizu,visited,n):
    qq=deque([(x,y)])
    visited[x][y]=True
    dirs=[(1,0),(-1,0),(0,1),(0,-1)]
    while qq:
        x,y=qq.popleft()
        #print(x,y)
        for dx,dy in dirs:
            nx,ny=x+dx,y+dy
            if 0<=nx<n and 0<=ny<n and 0<nx+1<n and not visited[nx][ny]
                qq.append((nx,ny))
                visited[nx][ny]=True
                if chizu[nx][ny]==9 or chizu[nx+1][ny]==9:
                    return 'yes'
    return 'no'

n=int(input())
chizu=[]
for i in range(n):
    chizu.append(list(map(int,input().split())))
visited=[[False]*n for _ in range(n)]

w=False
for i in range(n):
    for j in range(n):
        if chizu[i][j]==5:
            start_x,start_y=i,j
            if i+1<n and chizu[i+1][j]==5:
                print(standing_crab(start_x,start_y,chizu,visited,n))
            else:
                print(lying_crab(start_x,start_y,chizu,visited,n))
            w=True
            break
    if w:
        break
```

27373: 最大整数

dp, http://cs101.openjudge.cn/practice/27373/

思路:

其他部分基本自己写出来了，但是一开始没想通第一步冒泡的充分性，导致动态规划的过程中难以判断第i个数加入之后，如何与前i-1个数字组合出最大数。看了题解才知道，只要前面有这样的类冒泡排序，就只要将字符串直接按顺序相加就能保证得到的数字最大。

代码:

```
def f(string):
    if string=='':
        return 0
    else:
        return int(string)
```

```
m=int(input())
n=int(input())
num_list=input().split()

for i in range(n):
    for j in range(n-i-1):
        if num_list[j]+num_list[j+1]>num_list[j+1]+num_list[j]:
            num_list[j],num_list[j+1]=num_list[j+1],num_list[j]

weight=[]
for i in range(len(num_list)):
    weight.append(len(num_list[i]))



dp=[['' for _ in range(m+1)] for _ in range(n+1)]
used=[[] for _ in range(n+1)]
for i in range(0,m+1):
    dp[0][i]=''

for i in range(1,n+1):
    for j in range(1,m+1):
        if weight[i-1]<=j:
            dp[i][j] = str(max(f(dp[i - 1][j]), int(num_list[i - 1] + dp[i - 1][j
- weight[i - 1]])))
        else:
            dp[i][j] = dp[i-1][j]
print(dp[n][m])
```

代码运行截图（至少包含有"Accepted"）

#47845977提交状态 查看 提交 统计 提问

状态: Accepted

源代码

```
def f(string):
    if string=='':
        return 0
    else:
        return int(string)

m=int(input())
n=int(input())
num_list=input().split()

for i in range(n):
    for j in range(n-i-1):
        if num_list[j]+num_list[j+1]>num_list[j+1]+num_list[j]:
            num_list[j],num_list[j+1]=num_list[j+1],num_list[j]

weight=[]
for i in range(len(num_list)):
    weight.append(len(num_list[i]))



dp=[['' for _ in range(m+1)] for _ in range(n+1)]
used=[[] for _ in range(n+1)]
for i in range(0,m+1):
    dp[0][i]=''

for i in range(1,n+1):
    for j in range(1,m+1):
        if weight[i-1]<=j:
            dp[i][j] = str(max(f(dp[i - 1][j]), int(num_list[i - 1] + dp
        else:
            dp[i][j] = dp[i-1][j]
print(dp[n][m])
```

基本信息

#:	47845977
题目:	27373
提交人:	陈冠宇(24n2400011004)
内存:	31596kB
时间:	617ms
语言:	Python3
提交时间:	2024-12-19 19:01:38

02811: 熄灯问题

brute force, http://cs101.openjudge.cn/practice/02811

思路:

一开始想着用纯数学奇偶性分析的方法控制在O(mn)内完成，死活算不出来，最后看了题解才发现大家都只能暴力枚举（除了那个线性代数的做法，虽然我学了一学期线代但是根本没法像这位数院同学这样完美运用）

代码:

```
import copy
def press(x,y,lights):
    if lights[x][y]==1:
        lights[x][y]=0
    else:
        lights[x][y]=1
    dirs=[(1,0),(-1,0),(0,1),(0,-1)]
    for dx,dy in dirs:
        if 0<=x+dx<5 and 0<=y+dy<6:
            if lights[x+dx][y+dy]==0:
                lights[x+dx][y+dy]=1
            else:
                lights[x+dx][y+dy]=0

first_column=[]
```

```
o=[0,0,0,0,0]
for i in range(0,2):
    for j in range(0,2):
        for k in range(0,2):
            for l in range(0,2):
                for m in range(0,2):
                    xo=[i,j,k,l,m]
                    first_column.append(xo)
lights=[]
bottoms=[[0]*6 for _ in range(5)]
for i in range(5):
    lights.append(list(map(int,input().split())))

for t in first_column:
    this_lights=copy.deepcopy(lights)
    this_bottoms= copy.deepcopy(bottoms)
    for i in range(5):
        if t[i]==1:
            press(i,0,this_lights)
            this_bottoms[i][0]=1
    for j in range(1,6):
        for i in range(5):
            if this_lights[i][j-1]==1:
                press(i,j,this_lights)
                this_bottoms[i][j]=1
    if this_lights==[[0]*6 for _ in range(5)]:
        for i in range(5):
            for j in range(6):
                print(this_bottoms[i][j],end=' ')
            print()
        break
```

代码运行截图（至少包含有"Accepted"）

#47848018提交状态 查看 提交 统计 提问

状态: Accepted

基本信息

#: 47848018
题目: 02811
提交人: 陈冠宇(24n2400011004)
内存: 3744kB
时间: 25ms
语言: Python3
提交时间: 2024-12-19 20:23:51

源代码

```
import copy
def press(x,y,lights):
    if lights[x][y]==1:
        lights[x][y]=0
    else:
        lights[x][y]=1
    dirs=[(1,0),(-1,0),(0,1),(0,-1)]
    for dx,dy in dirs:
        if 0<=x+dx<5 and 0<=y+dy<6:
            if lights[x+dx][y+dy]==0:
                lights[x+dx][y+dy]=1
            else:
                lights[x+dx][y+dy]=0

first_column=[]

o=[0,0,0,0,0]
for i in range(0,2):
    for j in range(0,2):
        for k in range(0,2):
            for l in range(0,2):
                for m in range(0,2):
                    xo=[i,j,k,l,m]
                    first_column.append(xo)
lights=[]
bottoms=[[0]*6 for _ in range(5)]
for i in range(5):
    lights.append(list(map(int,input().split())))

for t in first_column:
    this_lights=copy.deepcopy(lights)
    this_bottoms= copy.deepcopy(bottoms)
    for i in range(5):
        if t[i]==1:
            press(i,0,this_lights)
            this_bottoms[i][0]=1
    for j in range(1,6):
        for i in range(5):
            if this_lights[i][j-1]==1:
                press(i,j,this_lights)
                this_bottoms[i][j]=1
    if this_lights==[[0]*6 for _ in range(5)]:
        for i in range(5):
            for j in range(6):
                print(this_bottoms[i][j],end=' ')
            print()
        break
```


08210: 河中跳房子

binary search, greedy, http://cs101.openjudge.cn/practice/08210/

思路:

写了半天贪心也想不到居然是二分搜索暴力查找。。。

代码:

```
def can_achieve_distance(rocks, L, M, distance):
    prev = 0  # 上一个保留的岩石位置（初始为起点）
    removed = 0  # 移除的岩石数量

    for rock in rocks:
        if rock - prev < distance:
            removed += 1  # 当前岩石不能满足最小跳跃距离，移除
            if removed > M:
                return False  # 移除岩石超过限制，返回不可行
```

```
        else:
            prev = rock  # 更新上一个保留的岩石位置

    # 检查最后一个跳跃（到终点）
    if L - prev < distance:
        removed += 1

    return removed <= M


def max_min_distance(L, N, M, rocks):
    rocks.sort()  # 按距离排序
    left, right = 1, L  # 最小可能的距离是1，最大是L
    result = 0

    while left <= right:
        mid = (left + right) // 2
        if can_achieve_distance(rocks, L, M, mid):
            result = mid  # 更新答案
            left = mid + 1  # 尝试更大的最小跳跃距离
        else:
            right = mid - 1  # 缩小跳跃距离

    return result


# 输入处理
L, N, M = map(int, input().split())
rocks = [int(input()) for _ in range(N)]

# 输出结果
print(max_min_distance(L, N, M, rocks))
```

代码运行截图 （至少包含有"Accepted"）

#47875548提交状态 查看 提交 统计 提问

状态: Accepted

基本信息

#: 47875548
题目: 08210
提交人: 陈冠宇(24n2400011004)
内存: 5596kB
时间: 180ms
语言: Python3
提交时间: 2024-12-21 12:03:01

源代码

```python
def can_achieve_distance(rocks, L, M, distance):
    prev = 0  # 上一个保留的岩石位置（初始为起点）
    removed = 0  # 移除的岩石数量

    for rock in rocks:
        if rock - prev < distance:
            removed += 1  # 当前岩石不能满足最小跳跃距离，移除
            if removed > M:
                return False  # 移除岩石超过限制，返回不可行
        else:
            prev = rock  # 更新上一个保留的岩石位置

    # 检查最后一个跳跃（到终点）
    if L - prev < distance:
        removed += 1

    return removed <= M


def max_min_distance(L, N, M, rocks):
    rocks.sort()  # 按距离排序
    left, right = 1, L  # 最小可能的距离是1，最大是L
    result = 0

    while left <= right:
        mid = (left + right) // 2
        if can_achieve_distance(rocks, L, M, mid):
            result = mid  # 更新答案
            left = mid + 1  # 尝试更大的最小跳跃距离
        else:
            right = mid - 1  # 缩小跳跃距离

    return result


# 输入处理
L, N, M = map(int, input().split())
rocks = [int(input()) for _ in range(N)]

# 输出结果
print(max_min_distance(L, N, M, rocks))
```


English 帮助 关于

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“计概2024fall每日选做”、CF、LeetCode、洛谷等网站题目。

只会写模板题。。。经常错在第一步的大方向选择

每日选座还有一万题需要补，感觉机考前途一片黑暗，剩下一周估计是没救了