Assignment #6: Recursion and DP

Updated 2201 GMT+8 Oct 29, 2024

2024 fall, Complied by 胡杨 元培学院

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。

3）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。

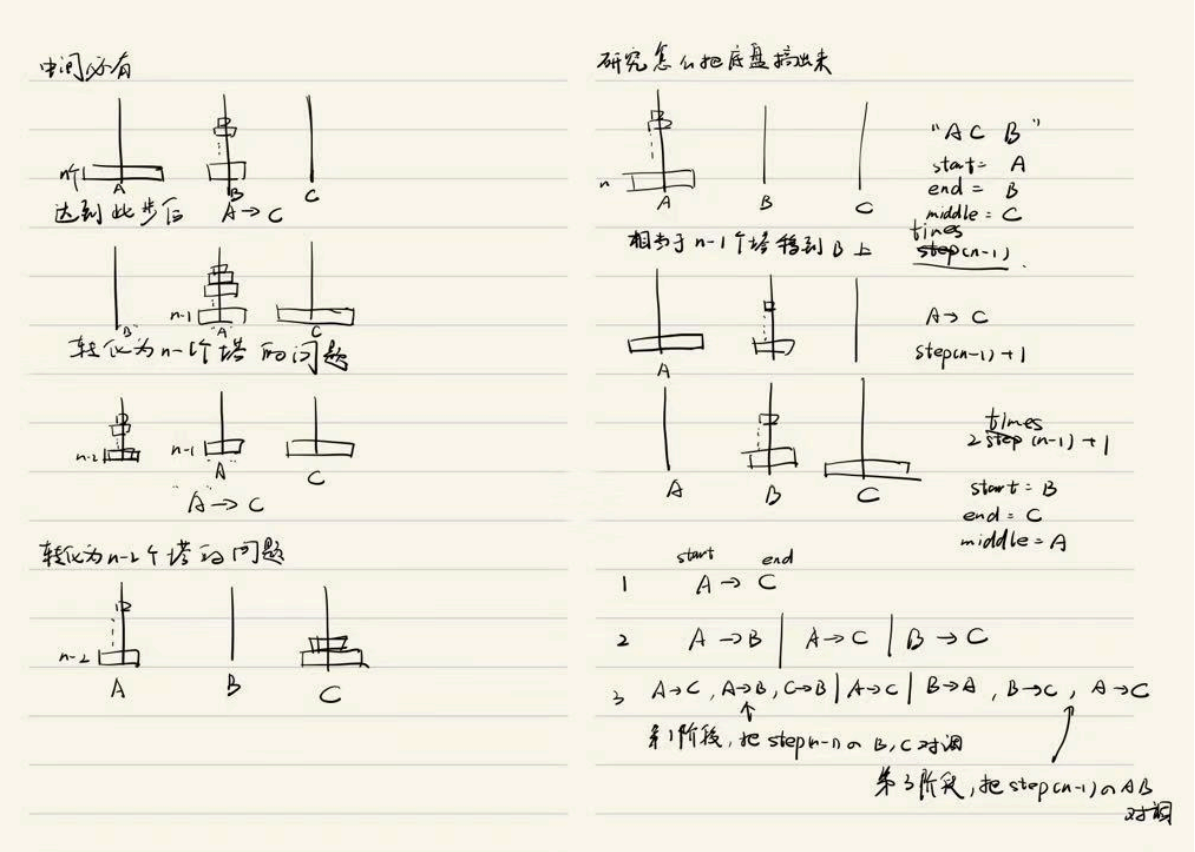
4）如果不能在截止前提交作业，请写明原因。

1. 题目

sy119: 汉诺塔

recursion, https://sunnywhy.com/sfbj/4/3/119

思路：发自我的草稿



耗时：1.5h

代码：

```
#汉洛塔
def times(n):
    if n==1:
        return 1
    else:
        return 2*times(n-1)+1

def steps(n):
    if n==1:
        return "A->C"
    else:
        temp_char='X'
        step1=steps(n-
1).replace('B',temp_char).replace('C','B').replace(temp_char,'C')
        step2='A->C'
        step3=steps(n-
1).replace('B',temp_char).replace('A','B').replace(temp_char,'A')
        return step1+'\n'+step2+'\n'+step3

n=int(input())
print(times(n))
print(steps(n))
```

代码运行截图 （至少包含有"Accepted"）

sy132: 全排列I

recursion, https://sunnywhy.com/sfbj/4/3/132

思路：其实第一个想法是直接把所有情况列出来因为最多就8个数XD

我们需要一个临时的空列表记录当前进行的排列，固定一个数在开头，然后问题即转化为n-1个数全排列的问题。全部排完之后，把临时列表清空，把当时固定的那个数的状态改成没用过，然后继续固定下一个数进行排列

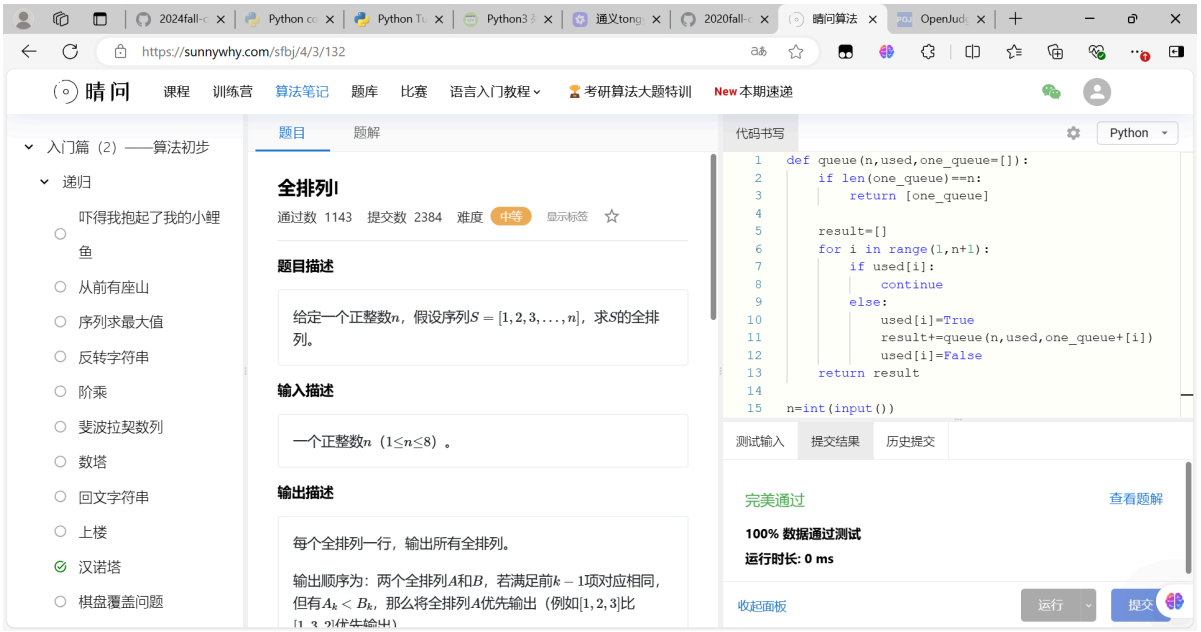
耗时：3h

会在纸上模拟但是不知道怎么转化成代码语言，看了答案，又结合了图解，但好像依然似懂非懂

代码：

```
def queue(n,used,one_queue=[]):
    if len(one_queue)==n:
        return [one_queue]

    result=[]
    for i in range(1,n+1):
        if used[i]:
            continue
        else:
            used[i]=True
            result+=queue(n,used,one_queue+[i])
            used[i]=False
    return result

n=int(input())
used=[False]*(n+1)
result=queue(n,used)
for i in result:
    print(' '.join(map(str,i)))
```

代码运行截图 （至少包含有"Accepted"）



02945: 拦截导弹

dp, http://cs101.openjudge.cn/2024fallroutine/02945

思路：一共有k枚导弹，假设最后以拦截第i枚导弹结束，此时能拦截的最多导弹数目记为dp[i]，则此时有两种情况：第一，第i枚导弹前面的导弹高度都比i的高度小，那么只能拦截它自己，dp[i]=1。第二，前面第j枚导弹的高度大于等于i的高度，此时可以选择在dp[j]之后再拦截i导弹，或者选择维持当前dp[i]的策略，取决于dp[i]=max(dp[j]+1,dp[i])

耗时：10min

前置学习了斐波那契数列、数字三角形、最大上升子序列。拦截导弹的思路基本上和最大上升子序列一模一样，所以做得很快。

代码：

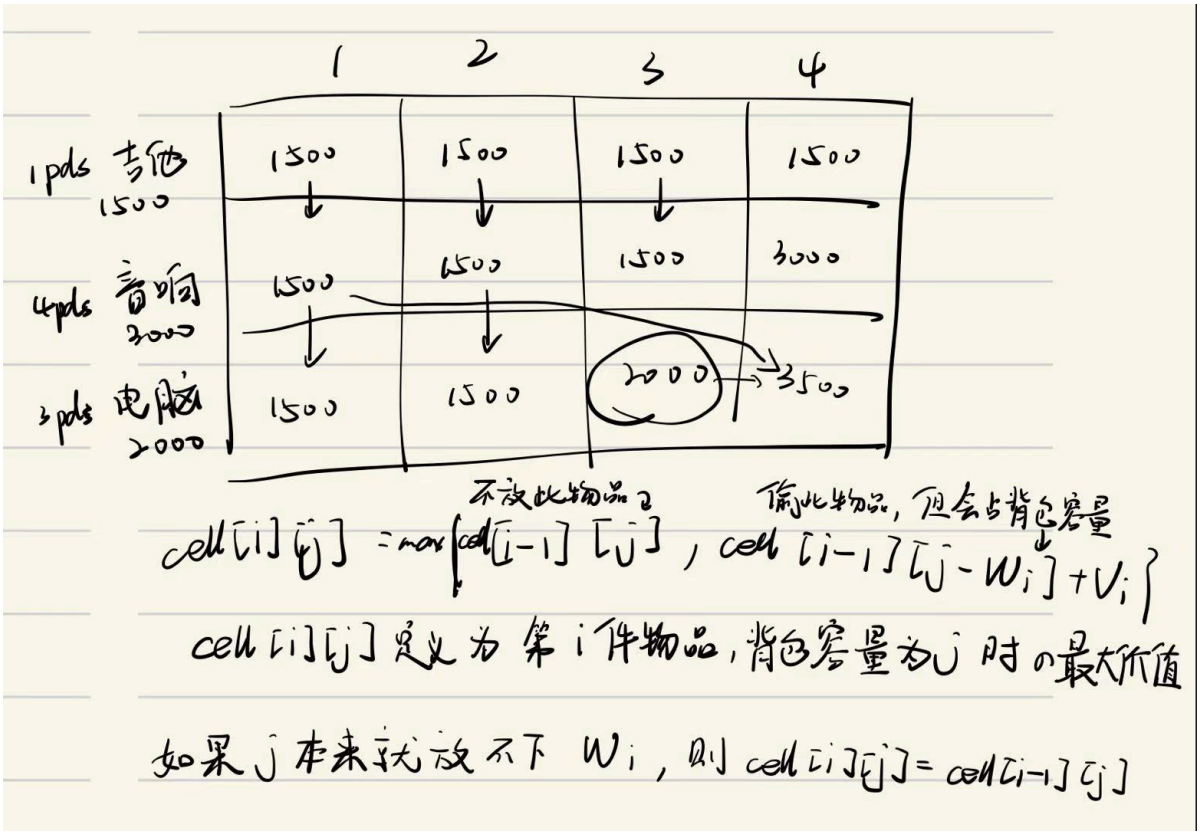
```
#拦截导弹

k=int(input())
a=list(map(int,input().split()))
dp=[1]*k

for i in range(k):
    for j in range(i):
        if a[j]>=a[i]:
            dp[i]=max(dp[i],dp[j]+1)

print(max(dp))
```

代码运行截图 （至少包含有"Accepted"）

23421: 小偷背包

dp, http://cs101.openjudge.cn/practice/23421

思路：发自我的草稿



耗时：1h

先学习了讲义上算法图解的思路之后完成

代码：

```
#小偷背包问题

n,b=map(int,input().split())
price=list(map(int,input().split()))
weight=list(map(int,input().split()))
dp=[[0]*(b+1) for _ in range(n+1)]

def value(n,b,price,weight,dp):
    for i in range(1,n+1):
        for j in range(1,b+1):
            if weight[i-1]>j:
                dp[i][j]=dp[i-1][j]
            else:
                dp[i][j]=max(dp[i-1][j],dp[i-1][j-weight[i-1]]+price[i-1])

    return dp[n][b]

print(value(n,b,price,weight,dp))
```

优化空间复杂度的代码：

理解：此时即仅储存第*i*件物品前一件物品放入不同大小背包时的最大价值，即此时*CELL[B]==CELL[i-1][B]*。而由于此时遍历数组从后往前遍历，故*CELL[i-1][B-W_i]*不会被*CELL[i][B-W_i]*的值所覆盖。同时，遍历到能装该物品的背包即停止，放不下该物体的背包不用重复计算

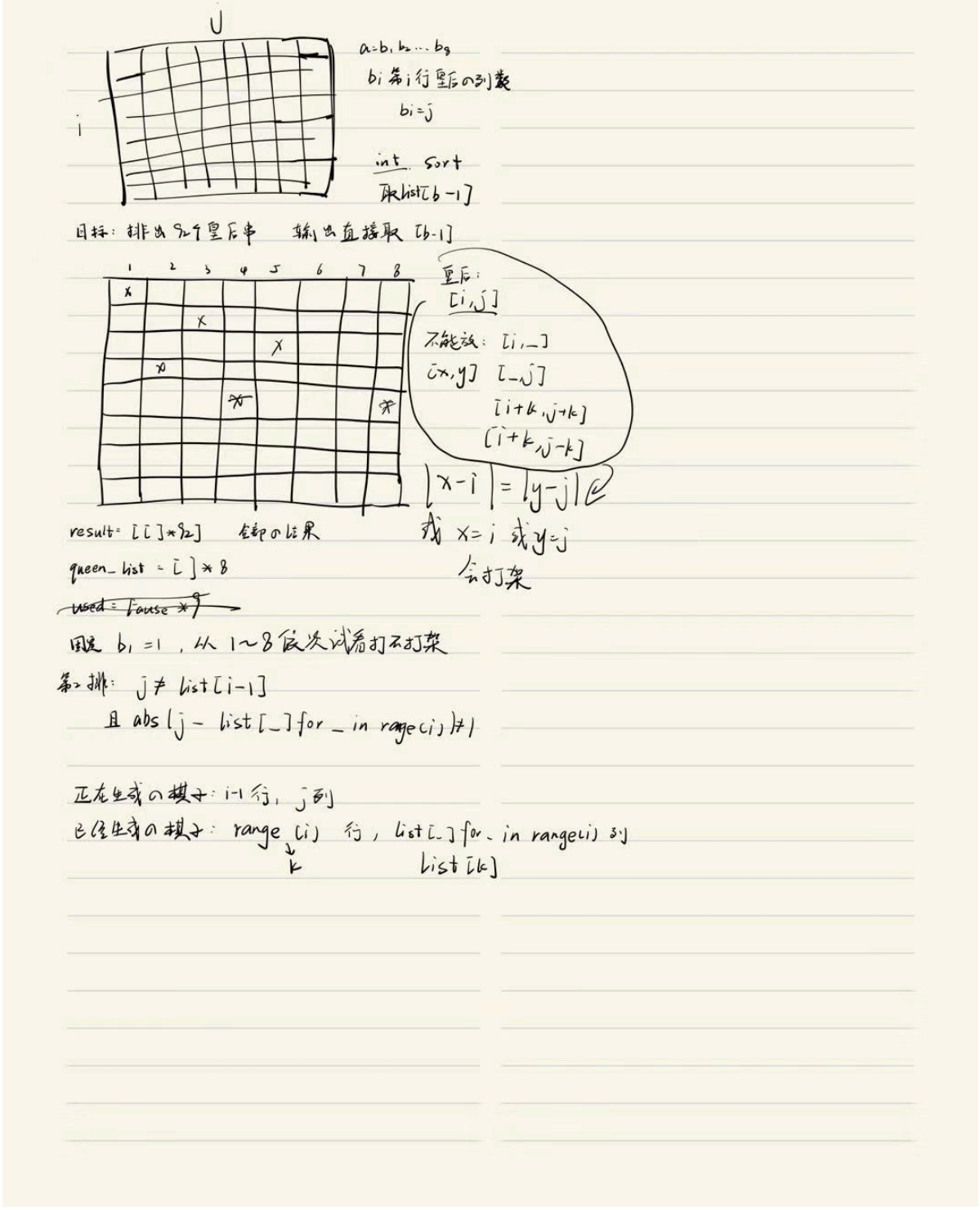
```
#优化空间复杂度的小偷背包

n,b=map(int,input().split())
price=list(map(int,input().split()))
weight=list(map(int,input().split()))
dp=[0]*(b+1)

for i in range(n):
    for j in range(b,weight[i]-1,-1):
        dp[j]=max(dp[j],dp[j-weight[i]]+price[i])

print(dp[b])
```

代码运行截图 （至少包含有"Accepted"）

02754: 八皇后

dfs and similar, http://cs101.openjudge.cn/practice/02754

思路：先确定第一排的棋子的位置，比如（1，1），然后第二排从第一个位置到第八个位置依次尝试看棋子会不会打架，不会打架则放下，会打架就右移一格。如果到某一行发现所有位置都会打架，回到前一行，把棋子继续往右移动。



耗时：一下午

题解看的是助教的讲解视频，思路和我的一致，题解文档里的没有看因为代码太多十分苦痛X(，感觉有讲解视频的话思路会清晰很多

代码：

```
#八皇后
#i是从0开始数的，j是从1开始数的

def queen(x):
    result = []
```

```
    queen_list = [0]*x

    def place(i):
        if i==x:
            result.append(queen_list.copy())
        else:
            for j in range(1,9):
                if not is_attack(i,j):
                    queen_list[i]=j
                    place(i+1)
                    #如果发现下一行每一个位置都无法放皇后，会跳出place(i+1)函数，来到本行
                    queen_list[i]=0

    def is_attack(i, j):
        for k in range(i):
            if j==int(queen_list[k]) or abs(i-k)==abs(j-int(queen_list[k])):
                return True
        return False

    place(0)

    return result

def get_queen_string(b):
    result=queen(8)
    queen_string=''.join(str(j) for j in result[b-1] )
    return queen_string

n=int(input())
for _ in range(n):
    b=int(input())
    print(get_queen_string(b))
```

代码运行截图 （至少包含有"Accepted"）

189A. Cut Ribbon

brute force, dp 1300 https://codeforces.com/problemset/problem/189/A

思路：相当于必须装满背包的背包问题。需要担心的就是装的个数最多，但是填不满背包的情况。可以把dp数组设成一维的，**dp[0]=0，其他n个都是负无穷**，此时dp数组中储存的数据如果没有填满背包将是负无穷，再怎么加也不会比正好填满背包的个数更多。最后print最后一个即可。

耗时：1h

不知道怎么保证最后装的个数最多的情况是把背包填满了的，参考了题解，设置dp=[0]+[float('-inf')]*n，好天才的想法

代码：

```
n,a,b,c=map(int,input().split())
l=[a,b,c]
dp=[0]+[float('-inf')]*n

for i in range(1,n+1):
    for j in range(3):
        if i>=l[j]:
            dp[i]=max(dp[i-l[j]]+1,dp[i])

print(dp[-1])
```

代码运行截图 （至少包含有"Accepted"）

289925336	Nov/04/2024 21:05 UTC+8	Itseventeen	189A - Cut Ribbon	Python 3	Accepted	77 ms	0 KB

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。

作业题目好难啊，好难啊，好难啊……多少有点边做边抄的感觉，汉洛塔是纯自己写的，拦截导弹和小偷背包是在学了讲义过后自己写的，全排列是看的答案，八皇后是自己想的思路但是有的地方不知道怎么用代码表示加bug太多参考了答案，cut ribbon是没想到可以设成负无穷。

递归和dp的问题都可以分解成相同结构的子问题，当前问题的解可以分解成子问题的解的组合，并且总是在函数内部调用它本身。所不同的是dp一定解决的是最优xxx的问题，而递归似乎更广泛一些。

感觉递归比dp还难理解qwq

把思路转化成代码语言仍然有困难，并且常常在细节上出现问题，或许我应该连一些模拟的题目来恶心亿下自己XD

每日选做前面落太多了，准备先跟dp试试，目前做了5道。等考完高数期中加油吧

以及感觉如果有逐行代码讲解的话看题解会好理解很多，也不会出现面对一大堆代码心态炸烈的情况，或许老师可以多提供一些题目的讲解吗QWQ