Assignment #6: Recursion and DP

Updated 2201 GMT+8 Oct 29, 2024

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

姓名：李彦臻
学号：2300010821
学院：数学科学学院

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：首先，不难猜出最少移动次数可由递推公式 an+1=(2*an)+1 确定，而方法就是先用 an 次把前 n 个圆盘按顺序摞到 B 上（B 和 C 是等价的）；然后再用 1 步把最大的盘子放到 C 上；最后再用 an 步把 B 的 n 个圆盘按顺序摞到 C 上（A 和 B 是等价的）；
因此，an+1=(2*an)+1 次是可以达到的！
至于为什么 an+1 确实在此时达到最小，是因为：
在移动的过程中一定有一步是把最底下那个圆盘从 A 移动到 C；又因为规定了同一个柱子上大的不能在小的上面，所以进行在这一步之前，必须得让 A 只有那一个最大的圆盘、C 什么都没有、B 是从小到大的前 n 个圆盘！
因此，在进行这一步之前，根据归纳假设，至少进行了 an 步；而且，进行了这一步之后，根据归纳假设，也得至少进行 an 步；因此 an+1 至少得是 an+1+an=(2*an)+1！证毕。

具体写移动方法时，可以用 dp，先把 n-1 的步骤搞出来；
再把 n-1 时的 C 和 B 互相对调一下便是第一个大步骤；中间插入一步 A—>C；再把 n-1 时的 A 和 B 互相对调一下便是第二个大步骤，结束！
为了方便起见,可以把每个移动方式都用一个数字代表,例如把 AB/AC/BC/BA/CA/CB 命名为 1~6；在第一步中把 12、36、45 对调；第二步中把 14、23、56 对调即可~

代码：

```python

```

```
n=int(input())
print(2**n - 1)
dp=[[] for _ in range(n)]
dp[0]=[2]
for i in range(1,n):
    for num in dp[i-1]:#第一个步骤
        if num == 1:
            dp[i].append(2)
        elif num == 2:
            dp[i].append(1)
        elif num == 3:
            dp[i].append(6)
        elif num == 6:
            dp[i].append(3)
        elif num == 4:
            dp[i].append(5)
        elif num == 5:
            dp[i].append(4)
    dp[i].append(2)
    for num in dp[i-1]:#第二个步骤
        if num == 1:
            dp[i].append(4)
        elif num == 4:
            dp[i].append(1)
        elif num == 2:
            dp[i].append(3)
        elif num == 3:
            dp[i].append(2)
        elif num == 5:
            dp[i].append(6)
        elif num == 6:
            dp[i].append(5)

for num in dp[n-1]:#最后再把 dp[n-1]里的数字翻译成人话即可
    if num == 1:
        print("A->B")
    elif num == 2:
        print("A->C")
```

```
    elif num == 3:
        print("B->C")
    elif num == 4:
        print("B->A")
    elif num == 5:
        print("C->A")
    elif num == 6:
        print("C->B")
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

sy132: 全排列 I

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

思路：这道题只要知道 permutations 这个函数就很简单了~
（尤其是考虑到 perms 函数会自动按照字典序进行排序，所以顺序问题也不用担心了~）

代码：

```python

```
from itertools import permutations#只要知道 perms 这个函数就很简单了

n=int(input())
data=[i for i in range(1,n+1)]
perms=permutations(data)#perms 会自动按照字典序进行排序，所以不用担心顺序~
for i in perms:
    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：这道题的本质就是求最长（非严格）递减序列；使用 dp 即可~
具体方法就是从第一个数开始逐步添加数，每时每刻（对前 i 个数）记录下长度为 j 的序列的最大尾数（不存在则为 0）；
考虑第 i 行（前 i 个数）时，先检查前 i-1 个数的最长递减序列长度（t），随后，对每一列而言，分三种情况进行考虑：

对第一列，永远保留最大的那个数；
```

对第 2~t-1 列，比较 heights 里的第 i 个数与第 i-1 行里第 j-1 个数的大小，若<=前 i-1 个数里 j-1 长度最大的尾数，则可以拼接！接下来，只用保留‘heights 第 i 个数’和‘原本的 j 长度最大的尾数’中较大的即可~；而反之，则说明无法拼接，长度不变
对第 t 列，只需检查 heights 第 i 个数能否拼接到后面即可；

最后的最后，只需要检查 dp 第 n 行（前 n 个数）里最后一个非零数所在的位置即可~

代码：

```python
```

```
n=int(input())
heights=list(map(int,input().split()))
dp=[[0]*(n+1) for _ in range(n+1)]#初始化一个(n+1)x(n+1)的 dp 表格
#其中，表格第 i 行第 j 列代表前 i 个数中长度为 j 的序列的最大尾数
dp[1][1]=heights[0]

for i in range(2,n+1):#考虑第 i 行（前 i 个数）
    t=dp[i-1][1:].index(0) + 1#先检查前 i-1 个数的最长递减序列长度（就是 t）
    #小心！！对一个列表的一个切片进行检索时，检索出的结果是对小列表而言的 index，
    #而不是原列表的 index！！如果切片是 list[k:]，则一定要加上 k！！

    #对第一列，永远保留最大的那个数：
    if heights[i-1] >= dp[i-1][1]:
        dp[i][1]=heights[i-1]
    else:
        dp[i][1]=dp[i-1][1]

    #对第 2~t-1 列，比较 heights 里的第 i 个数与第 i-1 行里第 j-1 个数的大小
    for j in range(2,t):
        if heights[i-1] <= dp[i-1][j-1]:#若<=前 i-1 个数里 j-1 长度最大的尾数，则可以拼接！
            #接下来，只用保留‘heights 第 i 个数’和‘原本的 j 长度最大的尾数’中较大的即可~
            dp[i][j]=max(heights[i-1],dp[i-1][j])
        else:#否则，则说明无法拼接，长度不变
            dp[i][j]=dp[i-1][j]

    #对第 t 列，只需检查 heights 第 i 个数能否拼接到后面即可
    if heights[i-1] <= dp[i-1][t-1]:#只有小于‘长度为 t-1 的最大尾数’才能拼接
        dp[i][t]=heights[i-1]
    #else 则说明无法拼接，因此还是 0！
```

```
#最后，只需要检查 dp 第 n 行（前 n 个数）里最后一个非零数所在的位置即可
if dp[n][n] != 0:
    print(n)
else:
    t=dp[n][1:].index(0)
    print(t)
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

23421: 小偷背包

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

思路：这道题没有很直接的方法，还是得用 dp 思想：
具体而言，可以先对前 i 个背包记录背包重量为 j 时的最大金额；
注意，在考虑第 i+1 个货物（假设重量为 k，金额为 t）时，只需对每个 j 考虑第（i-1，j-k）个数+t 与第（i-1，j）个数谁大即可！
（这里，把货物一个一个放进来考虑的好处是可以避免重复考虑！！）

代码：

```python

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

for i in range(1,n+1):
    k,t=weight[i-1],price[i-1]
    if k > b:#如果第 i 个货物压根儿就无法容纳，就不用管他
        for j in range(b+1):
            dp[i][j]=dp[i-1][j]
    else:#如果可以容纳第 i 个货物（设其重量为 k，金额为 t），则前 k-1 列不变
        for j in range(k):
            dp[i][j]=dp[i-1][j]
        for j in range(k,b+1):#对于第（k~b 中的）j 列，
            #对‘上一行的第 j-k 列的数+t’和‘上一行的第 j 列的数’进行比较即可！
            dp[i][j]=max(dp[i-1][j],dp[i-1][j-k] + t)

print(dp[n][b])#最后，输出前 n 个物品重量为 k 时的最大金额即可~
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

02754: 八皇后

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

思路：这道题目比较简单，直接将每个皇后所在的列视为1~8的一个排列（因为皇后不可能同列），然后用perms函数把所有的排列按顺序考虑一遍（因为perms自带字典序，所以相当于帮我们自动排了一次序~）；对于每个排列而言，因为不同行、列已经满足，只需要不在一个斜线上即可！要做到这一点，只需要让ai与ai+j的值不正好相差j就可以了~

代码：

```python

```
```
from itertools import permutations

position=[i for i in range(1,9)]
perms=permutations(position)
#将每个皇后所在的列视为1~8的一个排列（因为皇后不可能同列）

result=[]
for sequence in perms:
    status=True
    #对于每一个皇后的排列a1~a8，不同行、列已经满足，只需要不在一个斜线上即可
    #要做到这一点，只需要让ai与ai+j的值不正好相差j就可以了~
    for i in range(0,7):
        for j in range(i+1,8):
            if abs(sequence[i]-sequence[j]) == j-i:
                status=False
    if status == True:
        result.append("".join(map(str,sequence)))

n=int(input())
for i in range(n):
    m=int(input())
    print(result[m-1])
```

代码运行截图 <mark>（至少包含有"Accepted"）</mark>

189A. Cut Ribbon

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

思路：这题本质上不难想，直接 dp 就好了：
把 dp 表格初始化为 n+1 个 0，其中第 i 个数是用 abc 表示 i 的最少需要用到的个数（0 是无法表示）；这样一来，第 i 个数就是第 i-a 个数+1（第 i-a 个数不能为 0，下同）、第 i-b 个数+1、第 i-c 个数+1，这三个数中的最小值。
其中，有一个细节需要注意：为了让 dp[a]/dp[b]/dp[c]不等于 0，因此上方要求第 i-a/i-b/i-c 个数不能为 0 时，如果 i-a 就是 0，则也可以加 1！（bc 同理）

代码：

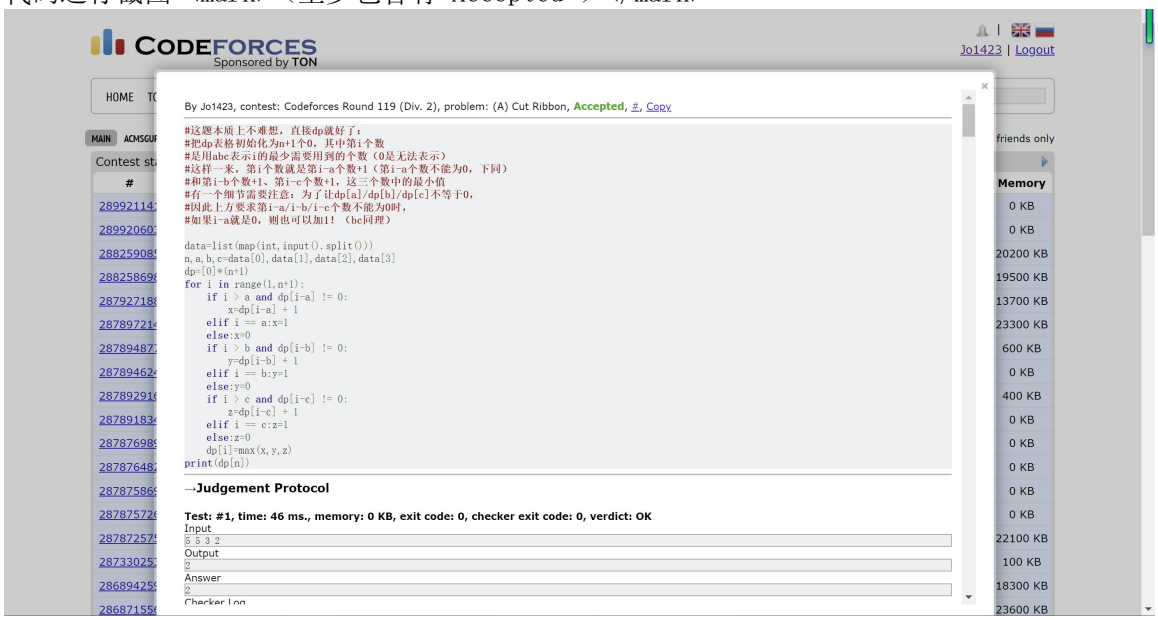
```python

```

```
data=list(map(int,input().split()))
n,a,b,c=data[0],data[1],data[2],data[3]
dp=[0]*(n+1)
for i in range(1,n+1):
    if i > a and dp[i-a] != 0:
        x=dp[i-a] + 1
    elif i == a:x=1
```

```
    else:x=0
    if i > b and dp[i-b] != 0:
        y=dp[i-b] + 1
    elif i == b:y=1
    else:y=0
    if i > c and dp[i-c] != 0:
        z=dp[i-c] + 1
    elif i == c:z=1
    else:z=0
    dp[i]=max(x,y,z)
print(dp[n])
```

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>

这次作业主要是 dp 和 permutations 题，虽然这两个知识点我之前都有所了解，但是因为最近在忙其他科目的期中考试，没花太多时间在计概上、也没有跟进每日选做，所以手有点生疏；但好在这些方法只要学会了，还是不太容易忘的，所以通过做这几道作业题也差不多把手感找回来了；与此同时，我也通过学习其他同学的 ac 代码，对一些思想和方法有了新的认知~

对于作业里每道题目的一些比较深入的思考，我主要写在了每道题的思路那里，我在总结这里就主要提一些我印象比较深刻的地方吧：

对于拦截导弹那道题，不难看出，本质就是求最长（非严格）递减序列；而我之前做过一道几乎一模一样的题（求最长递增子列），我知道把那道题的 dp 代码直接几乎原封不动的搬过来即可做出来，但是因为上次做那道题的时候是第一次接触 dp，使用起来很不熟练，钻研了好久才搞明白原理，因此，这次为了加深记忆，并且强化自己写 dp 代码的能力，我完全凭自己的推理新写了一遍，效果相当不错；在写代码的过程中我也进一步深入的理解了 dp 的本质，下次也更有信心做出类似的题目了~

不过，这道题我在一个地方卡了一会，因为这行代码：t=dp[i-1][1:].index(0) + 1（检查前 i-1 个数的最长递减序列长度 t），我没有加这个 1，导致出了比较严重的 bug。。通过这里的 bug，我学到了：对一个列表的一个切片进行检索时，检索出的结果是对小列表而言的 index，而不是原列表的 index！！比如，如果切片是 list[k:]，则一定要加上 k！！下次引以为戒~

对于小偷装货物那道题，我一开始的思路是：背包容量允许的情况，应尽量往里面装性价比高的物品，比如，如果直接把物品按性价比（price/weight）排序，并且先往里面扔靠前的物品，靠前的物品扔不进去了再扔后面的物品；但是，这个方法实际上是不对的！！！（反例：背包容量为 10kg，物品 A 为 5kg、5 块钱；B 为 10kg、6 块钱）因此，这道题并没有什么偷懒的方法，就只能老老实实用 dp 来做了~

这一段时间忙完期中 season 之后，我会重新按照进度做每日的选做题的；做这次作业虽然没花很长时间，但手感明显比较生疏，因此，每日的训练还是很有必要的~加油！