**第5题 母牛的故事**

我们都知道递推（动态规划）是递归（搜索）的反向操作，本题虽然注明“【递归】”，但同样可以用递推方式解决本题。

由于本题很多题解都详细讲解了递归的方式，那么这篇题解将不再赘述递归的方式。

我们试着考虑一下从递推的方式解决本题，最重要的是什么？  
对于一个动态规划问题，最重要也是最难的部分在于“状态转移方程”。为了推导出这个状态转移方程，我们不妨在Excel中制作一张表格，用于记录每年的母牛的数量，这里我们列举出前九年的母牛数量：https://blog.dotcpp.com/image_editor_upload/20191012031617_24550.png

我们可以很轻易的看出从第四年开始，每年的母牛数量都等于前一年的母牛数量加上三年前的母牛数量即状态转移方程为：cows[i] = cows[i-1] + cows[i-3];

到此为止，我们已经解决了最难的部分，但是还有一些小问题，比如：我们前三年的能不能用状态转移方程表示出来。

其实我们可以直接初始化为1，2，3，但是为了整体简短些，我们换一种方式，我们可以将整个数组向后推三个位置，即前面流出三个位置用于保证cows[i-3]不越界，附初始化函数代码：

1. int cows[70];
2. void init ()
3. {
4. for (int i = 0;i < 70;i ++){
5. cows[i] = 1;//全部赋为1，便于前三年母牛数量的增加
6. }
7. for (int i = 5;i < 70;i ++){
8. cows[i] = cows[i-1] + cows[i-3];//状态转移方程
9. }
10. }

本题对于cows开55个空间就足够。  
由于我们将整个数据向后推移了三个空间，当我们输出答案时，也应该向后移三个空间。附AC代码：

1. #include <iostream>
2. #include <cstdlib>
3. #include <cstdio>
4. #include <cstring>
5. #include <cmath>
6. #include <algorithm>
7. //有很多头文件其实没有用到，由于个人习惯就写上了六个，至于为什么不用<bits/stdc++.h>是因为早些版本的C++并没有整个头文件，就没有使用。
8. int cows[70];
9. void init ()
10. {
11. for (int i = 0;i < 70;i ++){
12. cows[i] = 1;
13. }
14. for (int i = 5;i < 70;i ++){
15. cows[i] = cows[i-1] + cows[i-3];
16. }
17. }
18. int main ()
19. {
20. int n;
21. init ();
22. while (scanf ("%d",&n) != EOF){//使用scanf的原因是scanf较cin快一点，printf同理。
23. if (n != 0){
24. printf ("%d\n",cows[n+3]);//我们初始化将所有数据顺移三个单位，这里为保证结构正确也顺移三个单位。
25. }
26. }
27. return 0;
28. }

**第6 题 : 最长不下降子序列的长度**

#include<iostream>

#include<algorithm>

using namespace std;

int a[1001],n;

int b[1001]={0},k=0;

int main()

{

cin>>n;

for(int i=1;i<=n;i++)

cin>>a[i];

for(int i=1;i<=n;i++)

{

if(a[i]>b[k-1])

{//如果a[i]大于b数组中的最后一个数 b[k]=a[i]且b数组长度+1

b[k++]=a[i];

}

else

{

int r=upper\_bound(b,b+k,a[i])-b;//二分查找在b中第一个大于a[i]的位置

b[r]=a[i];

}

}

cout<<k<<endl;

return 0;

}