## 楼梯问题

【题目描述】楼梯问题（Staircases）URAL 1017

有一个用数目为N的砖块堆起来的楼梯，楼梯的每层严格由不同个数的砖块按照由大到小的次序排列。在排列中，不允许各层有相同的高度，每个楼梯至少有两层，每层至少有一块。

图4.7给出N＝11和N＝5的时的摆法：

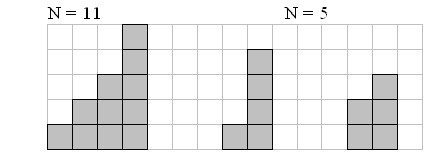


图4.7

你的任务是写一个程序，输入砖块数N，打印出共有多少种不同的摆法。

【输入格式】

砖块数N（3≤N≤500）。

【输出格式】一个整数，表示共有多少种不同的摆法。

【输入样例】

5

【输出样例】

2

### 算法1★

设F[i][j]表示把i划分后，最大的数为j（j＜i）时的方案数，则问题转化为剩余的数即i－j划分后，最大的数小于j时的方案数，动态转移方程为：

F[i][j] ＝ 

参考程序如下所示。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | //楼梯问题 — 算法1  #include <bits/stdc++.h>  **using** **namespace** std**;**  long long f**[**501**][**501**];**  int main**()**  **{**  int n**;**  long long sum**=**0**;**  scanf**(**"%d"**,&**n**);**  **for(**int i**=**1**;**i**<=**n**;**i**++)**  f**[**i**][**i**]=**1**;** //把i划分，最大数为i，方案数显然为1  **for(**int i**=**1**;**i**<=**n**;**i**++)** 　 //枚举数n  **for(**int j**=**1**;**j**<**i**;**j**++)** //枚举j  **for(**int k**=**1**;**k**<=**min**(**i**-**j**,**j**-**1**);**k**++)**  **if(**i**-**j**>=**k**)**  f**[**i**][**j**]+=**f**[**i**-**j**][**k**];**  **for(**int i**=**1**;**i**<**n**;**i**++)**  sum**+=**f**[**n**][**i**];**  printf**(**"%lld\n"**,**sum**);**  **return** 0**;**  **}** |

### 算法2★

设f[n][k]表示将n个砖头分为若干列且其中最大列不超过k的方案数，则可能的情况有两种，一种是最大列数恰好为k，一种是最大数列小于k，则动态转移方程为：

f[n][k]＝f[n－k][k－1]＋f[n][k－1]

参考程序如下所示。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | //楼梯问题 — 算法2  #include <bits/stdc++.h>  using namespace std;  long long f[505][505];  int main()  {  int n;  scanf("%d",&n);  for(int i=0;i<=n;++i)  f[0][i]=1;  for(int i=1;i<n;++i)  {  for(int j=i;j<=n;++j)  f[j][i]=f[j-i][i-1]+f[j][i-1];  for(int j=1;j<i;++j)  f[j][i]=f[j][i-1];  }  printf("%lld\n",f[n][n-1]);  return 0;  } |

### 算法3★

设f[i][j]表示i块砖头中最多的那一行最多有j块的方案数，则动态转移方程为：

f[i][j]＝f[i－j][j－1]＋f[i－j][j]

此方法类似于方法2，只是动规算法2由列入手，动规算法3由行入手。如图12.2，当N＝5时，有两种情况，一种情况是去掉最底层一行后（此行砖头数必等于列数i），剩下各行的砖头数均小于i，因此仅剩下了i－1列，一种情况是去掉最底层一行后，剩下的各行的砖头数至少还有一行数等于i。

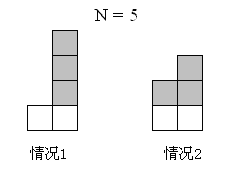


图12.2

边界条件为：f[1][1]＝1；

最后答案要从列数k从2到n（虽然n列、n－1列、n－2列可能是0）求和。

参考程序如下所示。

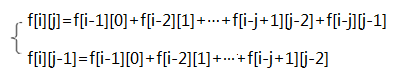
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | //楼梯问题 — 算法3  #include<bits/stdc++.h>  using namespace std;  long long f[505][505],ans;  int n;  int main()  {  cin>>n;  f[1][1]=1;  for(int i=2; i<=n; ++i)  for(int j=1; j<i; ++j)  f[i][j]=f[i-j][j]+f[i-j][j-1];  for(int k=2; k<n; ++k)  ans+=f[n][k];  cout<<ans<<endl;  return 0;  } |

### 算法4★

　　设f[i][j]表示将i分解且分解出的最大数字不超过j时分解的方案有多少。则动态转移方程为：

　　f[i][j]＝ 

　　这个方程时间复杂度是O（），我们用数学方法进行优化：



　　则f[i][j]－f[i][j－1]＝f[i－j][j－1]

　　即：f[i][j]＝f[i][j－1]＋f[i－j][j－1]

　　这里我们是将决策集合列出，并和以前得到的结果进行比较寻找相似的集合并直接引用以前的结果，达到优化决策的目的。幸运的是，我们发现原状态数组中就是保存的决策集合，但是有的时候并没有这么巧合，我们可以重新定义一个优化数组来保存决策的集合，并在每次求出某个状态后用常数的时间来维护这个数组，以便达到优化常数甚至O（1）的决策。

### 算法5★

　　其实该题的实质就是把一个整数N拆分为几部分，各部分的值互不相同。因此对于某一部分的数t（1≤t≤N）来讲，t或者不出现，或者只出现一次，因此我们完全可以用经典的0/1背包的模型解决该题。

整数N可以看成是背包的最大承重量，可装的各物品重量值分别为1~N－1，则根据优化的0/1背包方程可写出代码如下。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | //楼梯问题 — 算法5  #include <bits/stdc++.h>  using namespace std;  int main()  {  int n;  scanf("%d",&n);  long long f[1005]={1};  for(int i=1;i<n;++i)  for(int j=n;j>=i;--j)  f[j]+=f[j-i];  printf("%lld\n",f[n]);  return 0;  } |

### 算法6★

参照数的划分问题的动规算法3。

参考程序如下所示。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | //楼梯问题 — 算法6  #include <bits/stdc++.h>  using namespace std;  long long g[501][501];  int main()  {  int m;  scanf("%d",&m);  for(int i=1;i<=m;++i)  g[i-1][0]=1;  for(int i=1;i<m;++i)  for(int j=1;j<=m;++j)  if(j>=i)  g[i][j]=g[i-1][j]+g[i-1][j-i];  else g[i][j]=g[i-1][j];  printf("%lld\n",g[m-1][m]);  return 0;  } |

### 母函数算法★★

　　我们先来看下面两个例子：

　　【引例1】

　　有2个骰子掷出6点，共有多少方法？

　　我们可以设想把骰子出现的点数1，2，…，6和t，t2，…，t6对应起来，则第一个骰子可能出现的点数就与(t＋t2＋…＋t6)中的t的各次幂一一对应。

　　若有两个骰子，则（t＋t2＋…＋t6）×（t＋t2＋…＋t6）＝t2＋2t3＋3t4＋4t5＋5t6＋…中的t6的系数为5，显然是t1×t5＝t6，t2×t4＝t6，t3×t3＝t6，t4×t2＝t6，t5×t1＝t6，诸乘积都产生了t6这一方案数。

　　故两个骰子掷出6点的方法数等价于f(t)＝（t＋t2＋…＋t6）2 中t6的系数5。f(t)为母函数。这种对应把组合问题的加法法则和幂级数的t的乘幂的相加对应了起来。

母函数的思想实质就是把离散数列和幂级数一一对应起来，把离散数列间的相互结合关系对应成为幂级数间的运算关系，最后由幂级数形式来确定离散数列的构造。

　　【引例2】

　　有1克、2克、3克、4克的砝码各一枚，能称出哪几种重量？每种重量各有几种可能方案？

　　分析：我们假设x表示砝码，x的指数表示砝码的重量，这样：

　　1个1克的砝码可以用函数1＋x表示；

　　1个2克的砝码可以用函数1＋x2表示；

　　1个3克的砝码可以用函数1＋x3表示；

　　1个4克的砝码可以用函数1＋x4表示。

　　这四个式子是什么意思呢？我们拿1＋x2来说，前面已经说过，x表示砝码，x的指数表示重量，则1＋x2表示了两种情况：1表示质量为2的砝码取0个的情况，x2表示质量为2的砝码取1个的情况。“把组合问题的加法法则和幂级数的t的乘幂的相加对应起来”，明白了么？

　　几种砝码的组合可以称重的情况，可以用以上几个函数的乘积表示：

　　(1＋x)(1＋x2)(1＋x3)(1＋x4)

　　=(1＋x＋x2＋x3)(1＋x)(1＋x3＋x4＋x7)

　　=1＋x＋x2＋2x3＋2x4＋2x5＋2x6＋2x7＋x8＋x9＋x10

　　从上面的函数知道：可称出从1克到10克，系数便是方案数。

　　【母函数算法】

　　有了上面两个引例的基础，考虑到选取第i块都有两种情况：选和不选，我们可以写出楼梯问题的母函数即：

　　F（x）＝(1＋x)(1＋x2)(1＋x3)…(１＋xn）

　　F(x)的展开式中xn的系数就是各个分部量不同的整数拆分问题。其中要减掉1×xn（因为至少两列）的这种拆分情况，也就是最终结果的xn的系数减1即是答案。

参考程序如下所示。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | //楼梯问题 —— 母函数算法  #include <bits/stdc++.h>  using namespace std;  long long ans[510]={1,1}; //ans[i]存x^i的系数，起始为（1+x）,0和1的系数均为1  int main()  {  int N;  scanf("%d",&N);  for(int i=2;i<=N;i++) //依次从(1+x^2)、(1+x^3)……开始往里乘  for(int j=N;j>=0;j--) //更新乘进去后的所有可能系数  if(i+j<=N) //无需考虑超过N的系数  ans[i+j]+=ans[j];  printf("%lld\n",ans[N]-1);  return 0;  } |