# 卡特兰数

**一.概述**

**卡特兰数是一种经典的组合数，经常出现在各种计算中，其前几项为 :**

**1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, 2674440, 9694845…**

**二.常见公式**

**三.经典例题**

**公式3**

**例1 P1641 [SCOI2010]生成字符串**

**题目描述**

lxhgww 最近接到了一个生成字符串的任务，任务需要他把 n 个 1 和 m 个 0 组成字符串，但是任务还要求在组成的字符串中，在任意的前 k 个字符中，1 的个数不能少于 0 的个数。现在 lxhgww 想要知道满足要求的字符串共有多少个，聪明的程序员们，你们能帮助他吗？

**输入格式**

输入数据只有一行，包括 2 个数字 n 和 m。

**输出格式**

输出数据是一行，包括 1 个数字，表示满足要求的字符串数目，这个数可能会很大，只需输出这个数除以 20100403 的余数

**输入输出样例**

**输入#1**

2 2

**输入#1**

2

**思路**

**将1和0的操作转换到坐标系中，假设有1则（x+1，y+1），有0则（x+1,y-1），我们发现终点为（n+m,n-m），若存在前k字母中0个数大于1，则在坐标系中曲线经过y=-1，将终点关于y=-1对称，则终点对称点为（n+m，m-n-2），于是就有n-m+1步的向上变成向下，即一共有n+1步向下，所有情况有种可能，不合法的有种可能，答案相减即可，另一种对称方法是将起点关于y=-1对称，则起点变为（0，-2），到达（n+m，n-m）,需要将一步向下变为向上，则不合法有种可能**

**代码**

1. #include<bits/stdc++.h>
2. #define int long long
3. **using** **namespace** std;
4. **const** **int** mod=20100403;
5. **int** qpow(**int** a,**int** n,**int** mod)
6. {
7. **int** res=1;
8. **while**(n)
9. {
10. **if**(n&1) res=res\*a%mod;
11. a=a\*a%mod;
12. n>>=1;
13. }
14. **return** res;
15. }
16. **int** C(**int** a,**int** b,**int** mod)
17. {
18. **int** res=1;
19. **for**(**int** i=1,j=a;i<=b;i++,j--)
20. {
21. res\*=j;
22. res%=mod;
23. res\*=qpow(i,mod-2,mod);
24. res%=mod;
25. }
26. **return** res;
27. }
28. **signed** main()
29. {
30. **int** n,m;
31. cin>>n>>m;
32. cout<<((C(n+m,m,mod)-C(n+m,m-1,mod))%mod+mod)%mod<<endl;
33. **return** 0;
34. }

**例2 P1044 [NOIP2003 普及组] 栈**

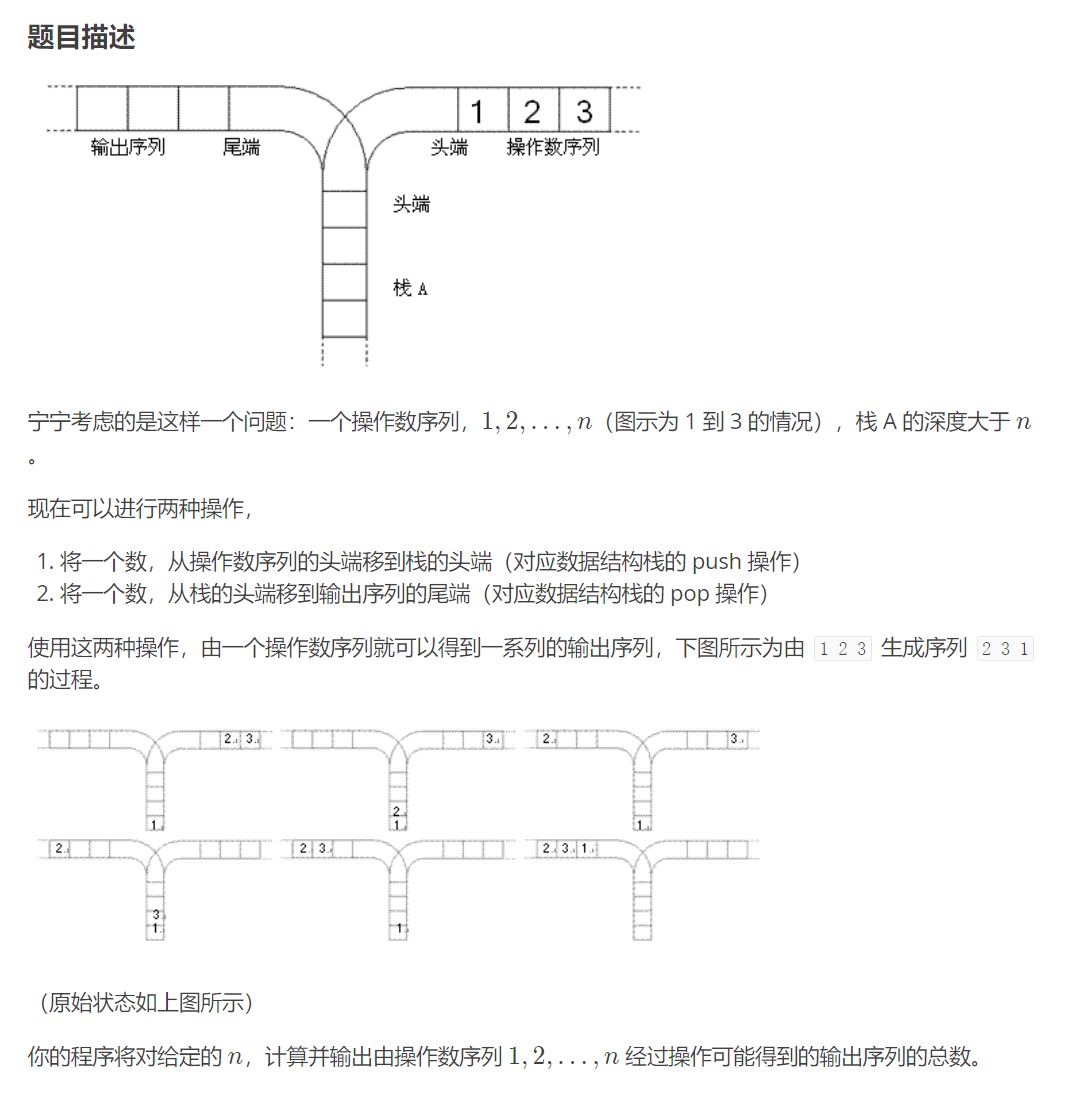
**题目背景**

栈是计算机中经典的数据结构，简单的说，栈就是限制在一端进行插入删除操作的线性表。

栈有两种最重要的操作，即 pop（从栈顶弹出一个元素）和 push（将一个元素进栈）。

栈的重要性不言自明，任何一门数据结构的课程都会介绍栈。宁宁同学在复习栈的基本概念时，想到了一个书上没有讲过的问题，而他自己无法给出答案，所以需要你的帮忙。

**题目描述**

**输入格式**

输入文件只含一个整数

**输出格式**

输出文件只有一行，即可能输出序列的总数目。

**输入输出样例#1**

**输入#1**

3

**输出#1**

5

**思路**

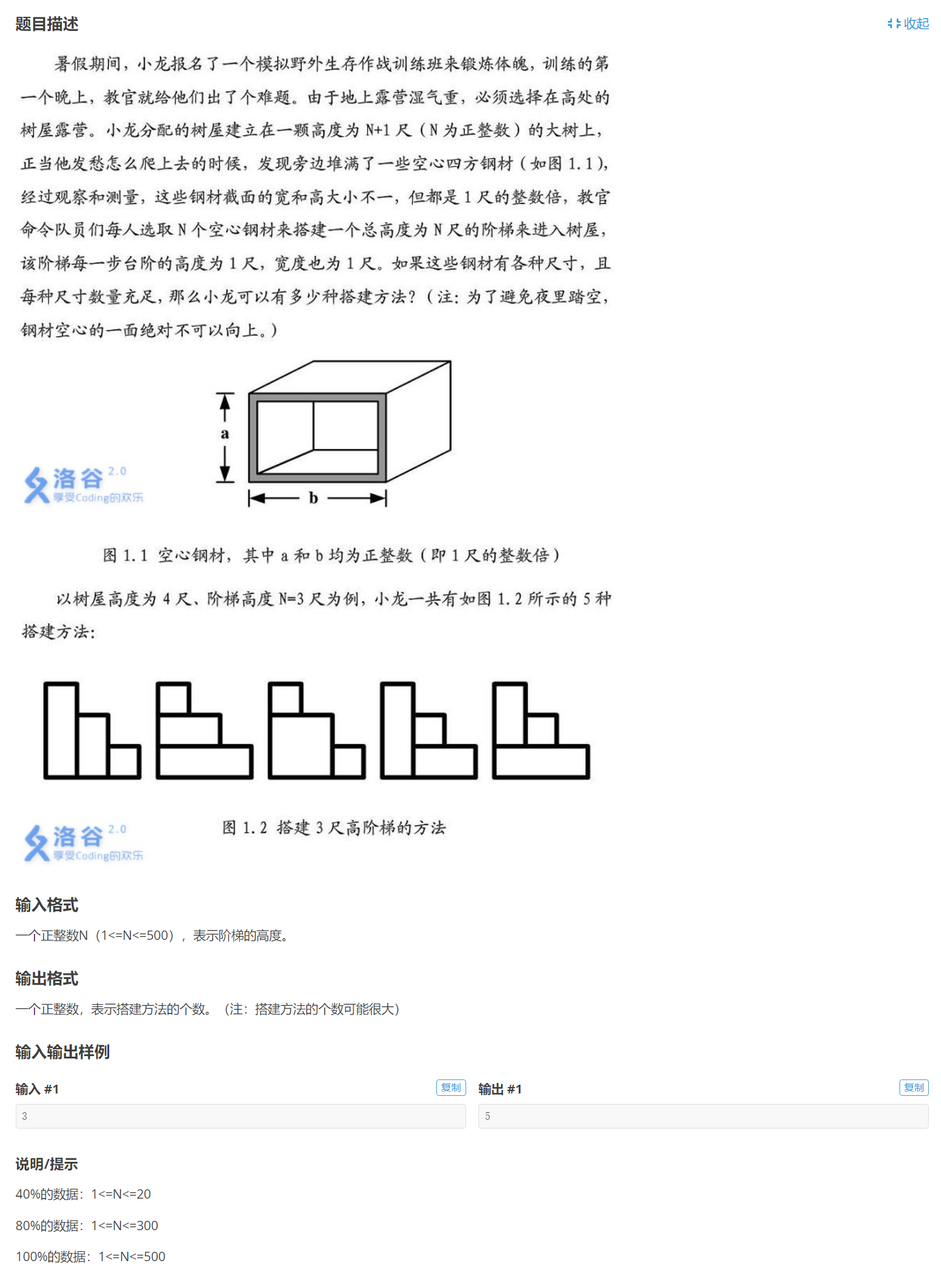
**将出入栈投射到坐标轴中，入栈为(x+1,y+1)，出栈为(x+1,y-1)，则入栈数始终大于出栈数，保证坐标曲线不经过y=-1，同例1**

**代码**

1. #include<bits/stdc++.h>
2. **using** **namespace** std;
3. **int** qpow(**int** a,**int** n)
4. {
5. **int** res=1;
6. **while**(n)
7. {
8. **if**(n&1) res=res\*a;
9. a=a\*a;
10. n>>=1;
11. }
12. **return** res;
13. }
14. **const** **int** N=50;
15. **int** c[N][N];
16. **void** init(**int** n)
17. {
18. **for**(**int** i=0;i<=n;i++)
19. {
20. **for**(**int** j=0;j<=i;j++)
21. {
22. **if**(!j) c[i][j]=1;
23. **else** c[i][j]=c[i-1][j-1]+c[i-1][j];
24. }
25. }
26. }
27. **int** main()
28. {
29. **int** n;
30. cin>>n;
31. init(n\*2+1);
32. cout<<c[2\*n][n]-c[2\*n][n-1]<<endl;
33. **return** 0;
34. }

**公式1**

**例1 P2532 [AHOI2012]树屋阶梯**

****

**思路**

**我们固定左下角的方块，假设变成为i，则方块上方需组成高度i-1的台阶，方块右边需组成高度为n-i-1的，于是就可以用公式1解决，注意本题需要用到高精度**

**代码**

1. #include<iostream>
2. #include<cstring>
3. **const** **int** N=505,L=500;
4. std::string f[N];
5. **int** na[L],nb[L],nc[L];
6. std::string mul(std::string a,std::string b)//高精度乘法a,b,均为非负整数
7. {
8. std::string s;
9. **int** La=a.size(),Lb=b.size();//na存储被乘数，nb存储乘数，nc存储积
10. memset(na,0,**sizeof**(na));//将na,nb,nc都置为0
11. memset(nb,0,**sizeof**(nb));
12. memset(nc,0,**sizeof**(nc));
13. **for**(**register** **int** i=La-1;i>=0;i--) na[La-i]=a[i]-'0';//将字符串表示的大整形数转成i整形数组表示的大整形数
14. **for**(**register** **int** i=Lb-1;i>=0;i--) nb[Lb-i]=b[i]-'0';
15. **for**(**register** **int** i=1;i<=La;i++)
16. **for**(**register** **int** j=1;j<=Lb;j++)
17. nc[i+j-1]+=na[i]\*nb[j];//a的第i位乘以b的第j位为积的第i+j-1位（先不考虑进位）
18. **for**(**register** **int** i=1;i<=La+Lb;i++)
19. nc[i+1]+=nc[i]/10,nc[i]%=10;//统一处理进位
20. **if**(nc[La+Lb]) s+=nc[La+Lb]+'0';//判断第i+j位上的数字是不是0
21. **for**(**register** **int** i=La+Lb-1;i>=1;i--)
22. s+=nc[i]+'0';//将整形数组转成字符串
23. **return** s;
24. }
25. std::string add(std::string a,std::string b)//只限两个非负整数相加
26. {
27. std::string ans;
28. memset(na,0,**sizeof**(na));
29. memset(nb,0,**sizeof**(nb));
30. **int** la=a.size(),lb=b.size();
31. **for**(**register** **int** i=0;i<la;i++) na[la-1-i]=a[i]-'0';
32. **for**(**register** **int** i=0;i<lb;i++) nb[lb-1-i]=b[i]-'0';
33. **int** lmax=la>lb?la:lb;
34. **for**(**register** **int** i=0;i<lmax;i++) na[i]+=nb[i],na[i+1]+=na[i]/10,na[i]%=10;
35. **if**(na[lmax]) lmax++;
36. **for**(**register** **int** i=lmax-1;i>=0;i--) ans+=na[i]+'0';
37. **return** ans;
38. }
39. **void** init(**int** n)
40. {
41. f[0]="1";
42. f[1]="1";
43. f[2]="2";
44. **for**(**int** i=3;i<=n;i++)
45. {
46. **for**(**int** j=0;j<i;j++)
47. {
48. f[i]=add(mul(f[j],f[i-j-1]),f[i]);
49. //cout<<i<<" "<<f[i]<<endl;
50. }
51. }
52. }
53. **int** main()
54. {
55. **int** n;
56. scanf("%d",&n);
57. init(n);
58. std::cout<<f[n];
59. **return** 0;
60. }