矩阵连乘第 2 讲----林大陈宇 （内部训练使用）

如何计算 fibonacci 数列的前 N 项和？

大意求第 a 个到第 b 个之间所有 fibonacci 的数和。
这个有点小技巧在里面，可以自己推导一下：
F(3) = F(1) + F(2)
F(4) = F(2) + F(3) = 1 * F(1) + 2 * F(2)
F(5) = F(3) + F(4) = 2 * F(1) + 3 * F(2)
F(6) = F(4) + F(5) = 3 * F(1) + 5 * F(2)
F(7) = F(5) + F(6) = 5 * F(1) + 8 * F(2)
F(8) = F(6) + F(7) = 8 * F(1) + 13 * F(2)

S(3) = 2 * F(1) + 2 * F(2)
S(4) = 3 * F(1) + 4 * F(2)
S(5) = 5 * F(1) + 7 * F(2)
S(6) = 8 * F(1) + 12 *F(2)
S(7) = 13 *F(1) + 20 *F(2)
不难发现，
S(n) = F(n + 2) - F(2)
若求 a 到 b 个之间的和：如下
因此题目就转换为了求 F(b + 2) - F(a + 2 - 1)
具体参见工大 OJ 2060，acm.hit.edu.cn

As we know , the Fibonacci numbers are defined as follows:

$$F(n)=\begin{cases}1,(n=0,1)\\F(n-1)+F(n-2),(n>1)\end{cases}$$ ""

Given two numbers a and b , calculate $$S=\sum_{i=a}^{b}F(i)$$.""

Input

The input contains several test cases. Each test case consists of two non-negative integer numbers a and b (0 ≤ a ≤ b ≤1,000,000,000). Input is terminated by a = b = 0.

Output

For each test case, output S mod 1,000,000,000, since S may be quite large.

Sample Input

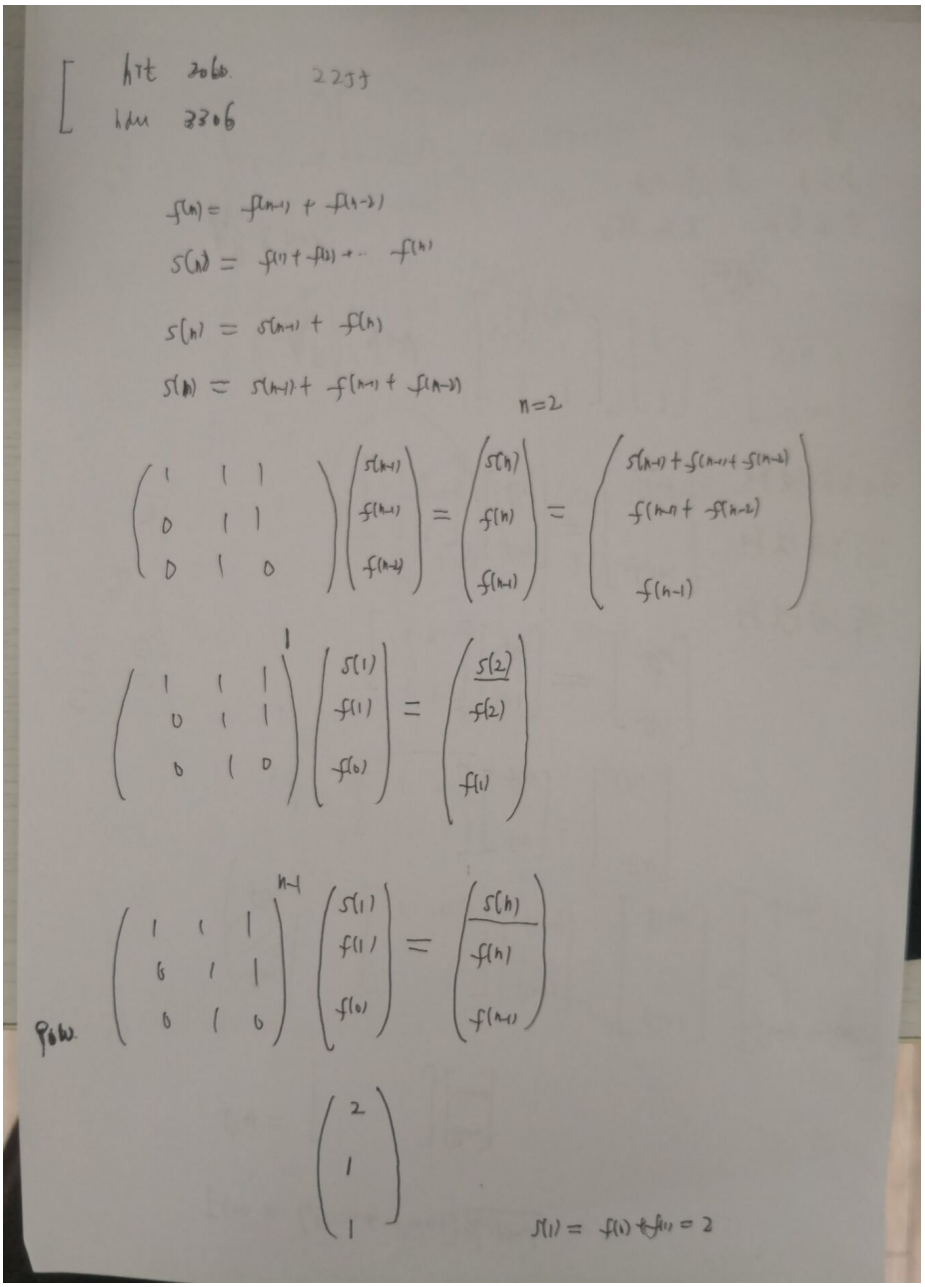
```
1 1
3 5
```

```
10 1000
0 0
```

Sample Output

```
1
16
496035733
```

其实可以直接推公式就行，不用背公式的？下为本题推导：林大陈宇

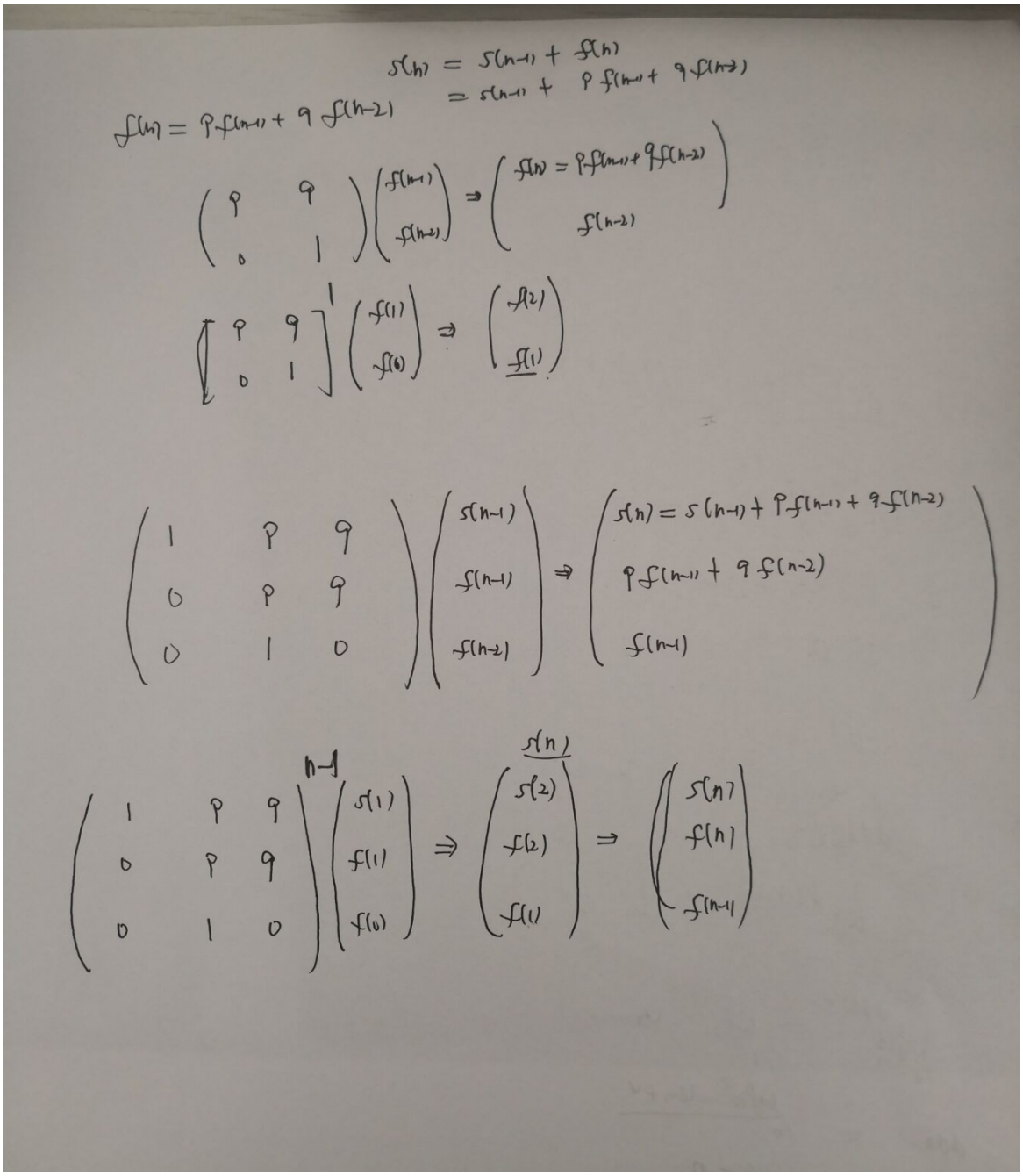


如果不是 fibonacci 数列，如何计算前 N 项和？

|o 1| |f(o)|
|q p| |f(1)|

这个可以利用矩阵幂次求得任何一个 F(n),可惜这里是求和,
那么令 A=前面那个 p,q 的矩阵
则 S(n) = f(0) + f(1) + ... + f(n) = f(0) + (A + A^2 + ... + A^n)*F 所得矩阵的右上角的值
那么如何求得 A + A^2 + ... + A^n 呢
可以继续构造如下的分块矩阵，其中 I 是单位矩阵
|A I|
|0 I|
令 R 等于上面的矩阵，则
R^2 = |A^2　A*I + I|
　　|0　 I　 |
R^3 = |A^3　A^2 * I + A * I + I|
　　| 0　 I　　　|
可以发现右上角即为 I + A + A^2 + ... + A^n，多个 I 后面给减掉就可以了
这样我们同样可以再次利用矩阵幂次求得 R^n
则 S(n) = ((R^(n+1).m12 - I) * F).a12 + f(0)
可以很完美的解决这个问题.

当然还是可以像我这样自己万能推导：下为推导过程：



见工大 OJ 2255　acm.hit.edu.cn

Maybe ACMers of HIT are always fond of fibonacci numbers, because it is so beautiful. Don't you think so? At the same time, *fishcanfly* always likes to change and this time he thinks about the following series of numbers which you can guess is derived from the definition of fibonacci number.

The definition of fibonacci number is:

f(0) = 0, f(1) = 1, and for n>=2, f(n) = f(n - 1) + f(n - 2)

We define the new series of numbers as below:

*f(0) = a, f(1) = b, and for n>=2, f(n) = p*f(n - 1) + q*f(n - 2)*,where *p* and *q* are integers.

Just like the last time, we are interested in the sum of this series from the *s*-th element to the *e*-th element, that is, to calculate $S = \sum_{i=s}^{e} f(i)$."""""

Great!Let's go!

Input

The first line of the input file contains a single integer *t* (*1 <= t <= 30*), the number of test cases, followed by the input data for each test case.

Each test case contains *6* integers *a,b,p,q,s,e* as concerned above. We know that *-1000 <= a,b <= 1000,-10 <= p,q <= 10* and *0 <= s <= e <= 2147483647*.

Output

One line for each test case, containing a single interger denoting *S MOD (10^7)* in the range *[0,10^7)* and the leading zeros should not be printed.

Sample Input

```
2
0 1 1 -1 0 3
0 1 1 1 2 3
```

Sample Output

```
2
3
```

Hint: You should not use int/long when it comes to an integer bigger than *2147483647*

构造矩阵：通过 HDU3306 来讲

Another kind of Fibonacci

Time Limit: 3000/1000 MS (Java/Others) Memory Limit: 65536/65536 K (Java/Others)

Total Submission(s): 802 Accepted Submission(s): 326

Problem Description

As we all known , the Fibonacci series : F(0) = 1, F(1) = 1, F(N) = F(N - 1) + F(N - 2) (N >= 2).Now we define another kind of Fibonacci : A(0) = 1 , A(1) = 1 , A(N) = X * A(N - 1) + Y * A(N - 2) (N >= 2).And we want to Calculate S(N) , S(N) = $A(0)^2$ +$A(1)^2$+……+$A(n)^2$.

Input

There are several test cases.

Each test case will contain three integers , N, X , Y .

N : 2<= N <= 2^{31} – 1

X : 2<= X <= 2^{31}– 1

Y : 2<= Y <= 2^{31} – 1

Output

For each test case , output the answer of S(n).If the answer is too big , divide it by 10007 and give me the reminder.

Sample Input

```
2 1 1
3 2 3
```

Sample Output

```
6
196
```

下为解题报告：林大陈宇

该题为矩阵连乘问题，主要考虑以下公式：

f(n)=x*f(n-1)+y*f(n-2)

先看 s(n)=s(n-1)+f(n)^2;

右边出现了 n，不可以的，只能出现 n-1 或 n-2 //这句话有误

所以：s(n-1)=s(n-2)+f(n-1)^2; 右边有 2 项，s(n-2)和 f(n-1)^2 必须要有了

S(n-2)----------→s(n-1)

F(n-1)^2------→ f(n)^2=[xf(n-1)+yf(n-2)]^2=出现了 f(n-2)^2 和 2xyf(n-1)f(n-2),所以共 4 项

$$\begin{bmatrix} \\ \\ \\ \end{bmatrix}\begin{bmatrix} s(n-2) \\ f^2(n-1) \\ f^2(n-2) \\ f(n-1)f(n-2) \end{bmatrix}=\begin{bmatrix} s(n-1) \\ f^2(n) \\ f^2(n-1) \\ f(n)f(n-1) \end{bmatrix}=\begin{bmatrix} s(n-2)+f^2(n-1) \\ x^2f^2(n-1)+2xyf(n-1)f(n-2)+y^2f^2(n-2) \\ f^2(n-1) \\ xf^2(n-1)+yf(n-1)f(n-2) \end{bmatrix}$$

最左边是 1 个空的矩阵(4*4)，也是我们要构造的矩阵（只要学过矩阵的人，就能把它算出来了吧）

$$A=\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & x^2 & y^2 & 2xy \\ 0 & 1 & 0 & 0 \\ 0 & x & 0 & y \end{bmatrix}$$

然后代入矩阵连乘幂的模板算下就行了（该模板要会背着敲）

```
#include <iostream>
#include <stdio.h>
#include <cstring>
#define Mod 10007
using namespace std;
const int MAX = 4;

typedef    struct{
              int    m[MAX][MAX];
}    Matrix;
Matrix P={1,1,0,0,
              0,0,0,0,
              0,1,0,0,
              0,0,0,0};
Matrix I={1,0,0,0,
              0,1,0,0,
              0,0,1,0,
              0,0,0,1};
Matrix matrixmul(Matrix a,Matrix b) //矩阵乘法
{
          int i,j,k;
          Matrix c;
          for (i = 0 ; i < MAX; i++)
                for (j = 0; j < MAX;j++)
                   {
                         c.m[i][j] = 0;
                         for (k = 0; k < MAX; k++)
```

```
                    c.m[i][j] += (a.m[i][k]* b.m[k][j])%Mod;
                c.m[i][j] %= Mod;
            }
        return c;
}

Matrix quickpow(long long n)
{
    Matrix m = P, b = I;
            while (n >= 1)
     {
            if (n & 1)
                b = matrixmul(b,m);
            n = n >> 1;
            m = matrixmul(m,m);
     }
     return b;
}

int main()
{
    int n,x,y,sum;
    Matrix b;
/    while(cin>>n>>x>>y)
    {
      sum=0;
      x=x%Mod;
      y=y%Mod;
      P.m[1][1]=(x*x)%Mod;P.m[1][2]=(y*y)%Mod;P.m[1][3]=(2*x*y)%Mod;
      P.m[3][1]=x;P.m[3][3]=y;
        b=quickpow(n);
        for(int i=0;i<4;i++)
        sum+=b.m[0][i]%Mod;
        cout<<sum%Mod<<endl;

    }
        return 0;
}
```

注意： 注意每一步都要取模，别怕麻烦；

对于 x 和 y 要是取负数的话（本题没有），则要(data%mod+mod)%mod;

正的就直接 data%mod 就行；

```
typedef   struct{
   int   m[MAX][MAX];
```

} Matrix;

该结构体的定义要注意，若将 int 类型换成 long long ，则运行时间会翻倍，可能会 TLE,注意哦

---------- 这 题 最 标 准 的 推 法 是 ：

$$f(n)=xf(n-1)+yf(n-2)$$

$$s(n)=f(0)^2+f(1)^2+f(2)^2+....+f(n)^2=s(n-1)+f(n)^2$$

$$s(n)=s(n-1)+f(n)^2=s(n-1)+x^2f^2(n-1)+2xyf(n-1)f(n-2)+y^2f^2(n-2)$$

$$\begin{pmatrix} & & & \end{pmatrix}\begin{pmatrix} s(n-1) \\ f^2(n-1) \\ f^2(n-2) \\ f(n-1)f(n-2) \end{pmatrix}\Rightarrow\begin{pmatrix} s(n) \\ f^2(n) \\ f^2(n-1) \\ f(n)f(n-1) \end{pmatrix}$$

$$\begin{pmatrix} & & & \end{pmatrix}\begin{pmatrix} s(n-1) \\ f^2(n-1) \\ f^2(n-2) \\ f(n-1)f(n-2) \end{pmatrix}\Rightarrow\begin{pmatrix} s(n-1)+x^2f^2(n-1)+2xyf(n-1)f(n-2)+y^2f(n-2) \\ x^2f^2(n-1)+2xyf(n-1)f(n-2)+y^2f(n-2) \\ f^2(n-1) \\ f^2(n-1)x+yf(n-1)f(n-2) \end{pmatrix}$$

左边的矩阵如下：

$$\begin{pmatrix} 1 & x^2 & y^2 & 2xy \\ 0 & x^2 & y^2 & 2xy \\ 0 & 1 & 0 & 0 \\ 0 & x^2 & 0 & y \end{pmatrix}$$

有兴趣的同学可以接着看，很有收获的，下面的方法和我的类似，我是把够造矩阵 A 放在左边，它是放在中间的，请用我的方法推 1 下，我给答案了。

构造常系数矩阵！

（一）Fibonacci 数列 f[n]=f[n-1]+f[n-2],f[1]=f[2]=1 的第 n 项的快速求法（不考虑高精度）.

解法：

考虑 1×2 的矩阵【f[n-2],f[n-1]】。根据 fibonacci 数列的递推关系，我们希望通过乘以一个 2×2 的矩阵，得到矩阵【f[n-1],f[n]】=【f[n-1],f[n-1]+f[n-2]】

很容易构造出这个 2×2 矩阵 A，即：

0　1

1　1

所以，有【f[1],f[2]】×A＝【f[2],f[3]】

又因为矩阵乘法满足结合律，故有：

【f[1],f[2]】×A n-1=【f[n],f[n+1]】

这个矩阵的第一个元素即为所求。

至于如何快速求出 A n-1，相信大家都会，即递归地：n 为偶数时，An=(A n/2)2；n 为奇数时，An=(A n/2)2*A。

问题（一）解决。

我的方法：$\begin{bmatrix} \end{bmatrix}\begin{bmatrix} f(n-2) \\ f(n-1) \end{bmatrix}=\begin{bmatrix} f(n-1) \\ f(n) \end{bmatrix}=\begin{bmatrix} f(n-1) \\ f(n-1)+f(n-2) \end{bmatrix}$　（a）

还是填空：

$A=\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$　还没会吗？ (a)式中左边的空矩阵就是让你填的 A

（二）数列 f[n]=f[n-1]+f[n-2]+1,f[1]=f[2]=1 的第 n 项的快速求法（不考虑高精度）.

解法：

仿照前例，考虑 1×3 的矩阵【f[n-2],f[n-1],1】，希望求得某 3×3 的矩阵 A，使得此 1×3 的矩阵乘以 A 得到矩阵：【f[n-1],f[n],1】＝【f[n-1],f[n-1]+f[n-2]+1,1】

容易构造出这个 3×3 的矩阵 A，即：

0　1　0

1　1　0

0　1　1

问题（二）解决。

我的推法：

$$\begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}\begin{bmatrix} f(n-2) \\ f(n-1) \\ 1 \end{bmatrix}=\begin{bmatrix} f(n-1) \\ f(n) \\ 1 \end{bmatrix}=\begin{bmatrix} f(n-1) \\ f(n-1)+f(n-2) \\ 1 \end{bmatrix}$$

A 是 3*3 的矩阵，填空吧

$$A=\begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

（三）数列 f[n]=f[n-1]+f[n-2]+n+1,f[1]=f[2]=1 的第 n 项的快速求法（不考虑高精度）.

解法：

仿照前例，考虑 1×4 的矩阵【f[n-2],f[n-1],n,1】，希望求得某 4×4 的矩阵 A，使得此 1×4 的矩阵乘以 A 得到矩阵：

【f[n-1],f[n],n+1,1】＝【f[n-1],f[n-1]+f[n-2]+n+1,n+1,1】

容易构造出这个 4×4 的矩阵 A，即：

0 1 0 0

1 1 0 0

0 1 1 0

0 1 1 1

问题（三）解决……

我自己的推法

$$\begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \end{bmatrix}\begin{bmatrix} f(n-2) \\ f(n-1) \\ n \\ 1 \end{bmatrix}=\begin{bmatrix} f(n-1) \\ f(n) \\ n+1 \\ 1 \end{bmatrix}=\begin{bmatrix} f(n-1) \\ f(n-1)+f(n-2)+n+1 \\ n+1 \\ 1 \end{bmatrix}$$

接着填空：

$$A=\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

四）数列 f[n]=f[n-1]+f[n-2],f[1]=f[2]=1 的前 n 项和 s[n]的快速求法（不考虑高精度）.

解法：

虽然我们有 S[n]=F[n+2]-1，但本文不考虑此方法，我们想要得到更一般的方法。

考虑（一）的矩阵 A，容易发现我们要求【f[1],f[2]】×（A+A2+A3+…+AN-1）。很多人使用一种很数学的方法构造一个 2r*2r（r 是 A 的阶数，这里为 2）的矩阵来计算，这种方法比较麻烦且很慢，这里不再介绍。下面考虑一种新方法。

仿照之前的思路，考虑 1×3 的矩阵【f[n-2],f[n-1],s[n-2]】，我们希望通过乘以一个 3×3 的矩阵 A，得到 1×3 的矩阵：

【f[n-1],f[n],s[n-1]】=【f[n-1],f[n-1]+f[n-2],s[n-2]+f[n-1]】

容易得到这个 3×3 的矩阵是：

0 1 0

1 1 1

0 0 1

然后……容易发现，这种方法的矩阵规模是(r+1)*(r+1)，比之前流行的方法好得多。
我用分块矩阵做过这题，但该方法的确应该推广，是主流的思想。
我的推法：

$$\begin{bmatrix} & & \\ & & \\ & & \end{bmatrix}\begin{bmatrix} f(n-2) \\ f(n-1) \\ s(n-2) \end{bmatrix}=\begin{bmatrix} f(n-1) \\ f(n) \\ s(n-1) \end{bmatrix}=\begin{bmatrix} f(n-1) \\ f(n-1)+f(n-2) \\ s(n-2)+f(n-1) \end{bmatrix}$$，所以矩阵见下，填空

$$A=\begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$为所求

也可以这样推：

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} s(n-1) \\ f(n-1) \\ f(n-2) \end{bmatrix}=\begin{bmatrix} s(n) \\ f(n) \\ f(n-1) \end{bmatrix}=\begin{bmatrix} s(n-1)+f(n-1)+f(n-2) \\ f(n-1)+f(n-2) \\ f(n-1) \end{bmatrix}$$

（五）数列 f[n]=f[n-1]+f[n-2]+n+1,f[1]=f[2]=1 的前 n 项和 s[n]的快速求法（不考虑高精度）.

解法：

结合（三）（四），容易想到……

考虑 1×5 的矩阵【f[n-2],f[n-1],s[n-2],n,1】，

我们需要找到一个 5×5 的矩阵 A，使得它乘以 A 得到如下 1×5 的矩阵：

【f[n-1],f[n],s[n-1],n+1,1】

=【f[n-1], f[n-1]+f[n-2]+n+1,s[n-2]+f[n-1],n+1,1】

容易构造出 A 为：

0 1 0 0 0

1 1 1 0 0

0 0 1 0 0

0 1 0 1 0

0 1 0 1 1

然后……问题解决。

我的推法：

$$\begin{bmatrix} \end{bmatrix}\begin{bmatrix} f(n-2) \\ f(n-1) \\ S(n-2) \\ n \\ 1 \end{bmatrix}=\begin{bmatrix} f(n-1) \\ f(n) \\ s(n-1) \\ n+1 \\ 1 \end{bmatrix}=\begin{bmatrix} f(n-1) \\ f(n-1)+f(n-2)+n+1 \\ s(n-2)+f(n-1) \\ n+1 \\ 1 \end{bmatrix}$$，填空

$$A=\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$为构造的矩阵。

也可以这样推：

$$\begin{bmatrix} \end{bmatrix}\begin{bmatrix} \mathrm{s}(n-1) \\ f(n-2) \\ f(n-1) \\ n \\ 1 \end{bmatrix}=\begin{bmatrix} \mathrm{s}(n) \\ f(n-1) \\ \mathrm{f}(n) \\ n+1 \\ 1 \end{bmatrix}=\begin{bmatrix} s(n-1)+f(n-1)+f(n-2)+n+1 \\ f(n-1) \\ \mathrm{f}(n-2)+f(n-1)+\mathrm{n}+1 \\ n+1 \\ 1 \end{bmatrix}$$

$$A=\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

下面的内容来源于农夫三拳的帖子
题目：
1. pku3070 Fibonacci
http://acm.pku.edu.cn/JudgeOnline/problem?id=3070
这个题目按照题意做就可以轻松的搞定了.问题关键是给出了我们另外一种求解 Fibonacci 的方法。利用矩阵乘法！
那为什么它就比递推式和通项公式要好呢,原因是因为 Fibonacci 的增长速度非常快,通常普通解法没有办法求得
后面的大数,所以题目通常要求取模,而如果用通项,肯定不够精确了,用递推公式又肯定会超时(比如要你求第 1000000000 个
Fibonacci 数模 10000000 的最终结果),这个时候转换为求矩阵的幂次,可以在 logn 的时间内搞定.
我比较喜欢下面这个形式
|0 1|　|f(0)|　　|f(1)|
　　===>
|1 1|　|f(1)|　　|f(0)+f(1)|　后面这个也就是 f(2)了
如果令 A 为那个 0,1 矩阵,可以发现
f(n) = (A^n).a12 * f(1) (a12 为右上角的元素值)
特别的 ，
A^0 = I (I 是单位矩阵)

2. hdu1021 Fibonacci Again
http://acm.hdu.edu.cn/showproblem.php?pid=1021
通过一个模 3 障眼法，其实也可以使用上述的矩阵幂次进行求解.不过看题目的数据量,貌似直接用推倒也可以搞定

3. hdu1568
http://acm.hdu.edu.cn/showproblem.php?pid=1568
输出一个 fibonacci 的前 4 个字母
这个要使用 fibonacci 的通项公式了
((1+√5)/2)^n - ((1-√5)/2)^ n

　　√5
可以发现后面的((1-√5)/2)^ n 当 n 很大的时候数值非常小，可以忽略，不会对前 4 位造成影响。
因此转变为了
((1+√5)/2)^n

√5
这个式子随着 n 的增长也是增长很快，但是我们可以发现，题目要求的是前 4 位，因此我们发现将长度固定在某个范围
之内(即如果数大了，就除一下)，这样的话就可以取到前 4 位了.

4. hdu1250 Hat's Fibonacci
http://acm.hdu.edu.cn/showproblem.php?pid=1250
我用的赤裸裸的高精度算的把结果存起来了,大汗.内存占了 18M. Statistic 里面的貌似都很少的样子.
后来我用数组去循环递推做的,内存自然就下降下来了.这个题目貌似也可以构造矩阵,只不过要在计算
里面处理好高精度.

5. hit1214 Fibonacci Coding
http://acm.hit.edu.cn/ojs/show.php?Proid=1214&Contestid=0
题目大意是去除相邻的 1,改变成前面一个.
我是直接模拟的,从高位开始判断,如果相邻位为 1 且前面一个为 0,则变化,直至没有出现该情况为止
要注意的是,如果使用 string,且在高位前产生进位的时候,要更新 len,如果你没有直接调用 size 的话.

6. hit1533 Fibonacci Numbers
http://acm.hit.edu.cn/ojs/show.php?Proid=1533&Contestid=0
高精度,与 hdu1350 类似

7. hit1864 Fibonacci
http://acm.hit.edu.cn/ojs/show.php?Proid=1864&Contestid=0
这个是求第 k 个 finbonacci 数的位数,求位数,用大数就太不明智了.
这里我同样的使用了通项的, 当 n 比较大的时候,利用变化过的通项
((1+√5)/2)^n

√5
那么求位数就用 log10(F) + 1
再利用 log10(a*b) = log10(a) + log10(b), log10(a/b) = log10(a) - log10(b)

8. hit2060 Fibonacci Problem Again
http://acm.hit.edu.cn/ojs/show.php?Proid=2060&Contestid=0
大意求第 a 个到第 b 个之间所有 fibonacci 的数和。
这个有点小技巧在里面，可以自己推导一下：
F(3) = F(1) + F(2)
F(4) = F(2) + F(3) = 1 * F(1) + 2 * F(2)
F(5) = F(3) + F(4) = 2 * F(1) + 3 * F(2)
F(6) = F(4) + F(5) = 3 * F(1) + 5 * F(2)
F(7) = F(5) + F(6) = 5 * F(1) + 8 * F(2)
F(8) = F(6) + F(7) = 8 * F(1) + 13 * F(2)

S(3) = 2 * F(1) + 2 * F(2)
S(4) = 3 * F(1) + 4 * F(2)
S(5) = 5 * F(1) + 7 * F(2)
S(6) = 8 * F(1) + 12 *F(2)
S(7) = 13 *F(1) + 20 *F(2)

不难发现，
S(n) = F(n + 2) - F(2)
因此题目就转换为了求 F(b + 2) - F(a + 2 - 1)
而又是大数取模，考虑使用矩阵求幂

9. hit2065 Fibonacci Number
http://acm.hit.edu.cn/ojs/show.php?Proid=2065&Contestid=0
赤裸裸的 Fibonacci，最基本的那种

10. hit2204 Fibonacci Numbers
http://acm.hit.edu.cn/ojs/show.php?Proid=2204&Contestid=0
这个题目相当于求第 k 个 Fibonacci 的前 4 位和后 4 位，那么前面的题目
综合起来，就是前 4 位用通项，后 4 位用矩阵求模。
就不难了。要注意的是，前面求前 4 位的时候我用的 O(n)的做法，过了。
这边如果仍然采用 O(n)的会 TLE，必须采用 O(logn),求幂次的方法才能过.

11. hit2255 Not Fibonacci
http://acm.hit.edu.cn/ojs/show.php?Proid=2255&Contestid=0
这个题目是个 BT 题,交了我 n 次,后来发现是 s=0 的时候负数情况没有处理,汗.
粗看一下，这个题目和 hit2060 一样，不过这里好像都泛化了。这里面的规律
可没有上面那题那么好找了。wy 他们貌似直接推导 S 的公式，都是采用的什么
等比数列弄的,我和 felica 讨论了一下，发现了一个好的解法。通过构造分块
矩阵，可以很好的解决这个问题。
首先构造
|0 1| |f(0)|
|q p| |f(1)|
这个可以利用矩阵幂次求得任何一个 F(n),可惜这里是求和,
那么令 A=前面那个 p,q 的矩阵
则 S(n) = f(0) + f(1) + ... + f(n) = f(0) + (A + A^2 + ... + A^n)*F 所得矩阵的右上角的值
那么如何求得 A + A^2 + ... + A^n 呢
可以继续构造如下的分块矩阵，其中 I 是单位矩阵
|A I|
|0 I|
令 R 等于上面的矩阵，则
R^2 = |A^2 A*I + I|
  |0 I |
R^3 = |A^3 A^2 * I + A * I + I|
  | 0 I |
可以发现右上角即为 I + A + A^2 + ... + A^n，多个 I 后面给减掉就可以了
这样我们同样可以再次利用矩阵幂次求得 R^n
则 S(n) = ((R^(n+1).m12 - I) * F).a12 + f(0)
可以很完美的解决这个问题.

12. hnu10072 Fibonacci Number
http://acm.hnu.cn:8080/online/?action=problem&type=show&id=10072&courseid=0
与 hit2065 一样

13. tju1208 Fibonacci Numbers
http://acm.tju.edu.cn/toj/showp1208.html
与 hit1533 一样

14. zju2060 Fibonacci Again
http://acm.zju.edu.cn/show_problem.php?pid=2060
与 hdu1021 一样

15. zju1828 Fibonacci Numbers
http://acm.zju.edu.cn/show_problem.php?pid=1828
与 hit1533 一样

16. zju2672 Fibonacci Subsequence
http://acm.zju.edu.cn/show_problem.php?pid=2672

又是一道 bt 题,时间卡的非常之紧.
这个题目我使用的 dp+hash.
记 d(i,j)为以第 i 个数作为结尾,前一个数是第 j 个的 fibonacci 串的最大长度
则 d(i,j) = max{d(j, i - j)} + 1,
注意查找 data[i] - data[j]的时候利用 hash 查找到的应该是 j 小的所有当中最大下标的那个。
因此需要在每次循环结束的时候更新已存在的数的 hash。

17. hzu1060 Fibonacci 数列
http://acm.fzu.edu.cn/problem.php?pid=1060
与 hit1533 相同

18. fjnu1158 Fibonacci 数列
http://acm.fjnu.edu.cn/show?problem_id=1158
与 hit1533 相同

19. zjut Fibonacci 数
http://acm.zjut.edu.cn/ShowProblem.aspx?ShowID=1029
与 hit2065 一样,题目描述稍有不一样