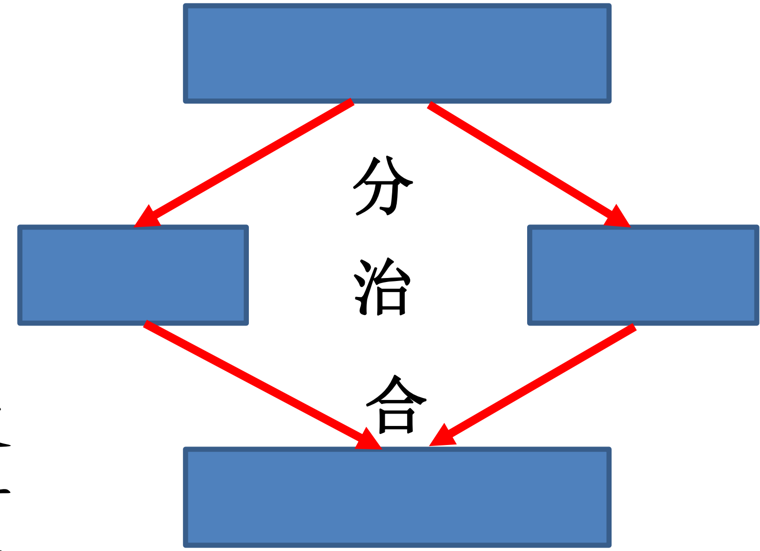
倍 增 - 快速幂

分治与倍增

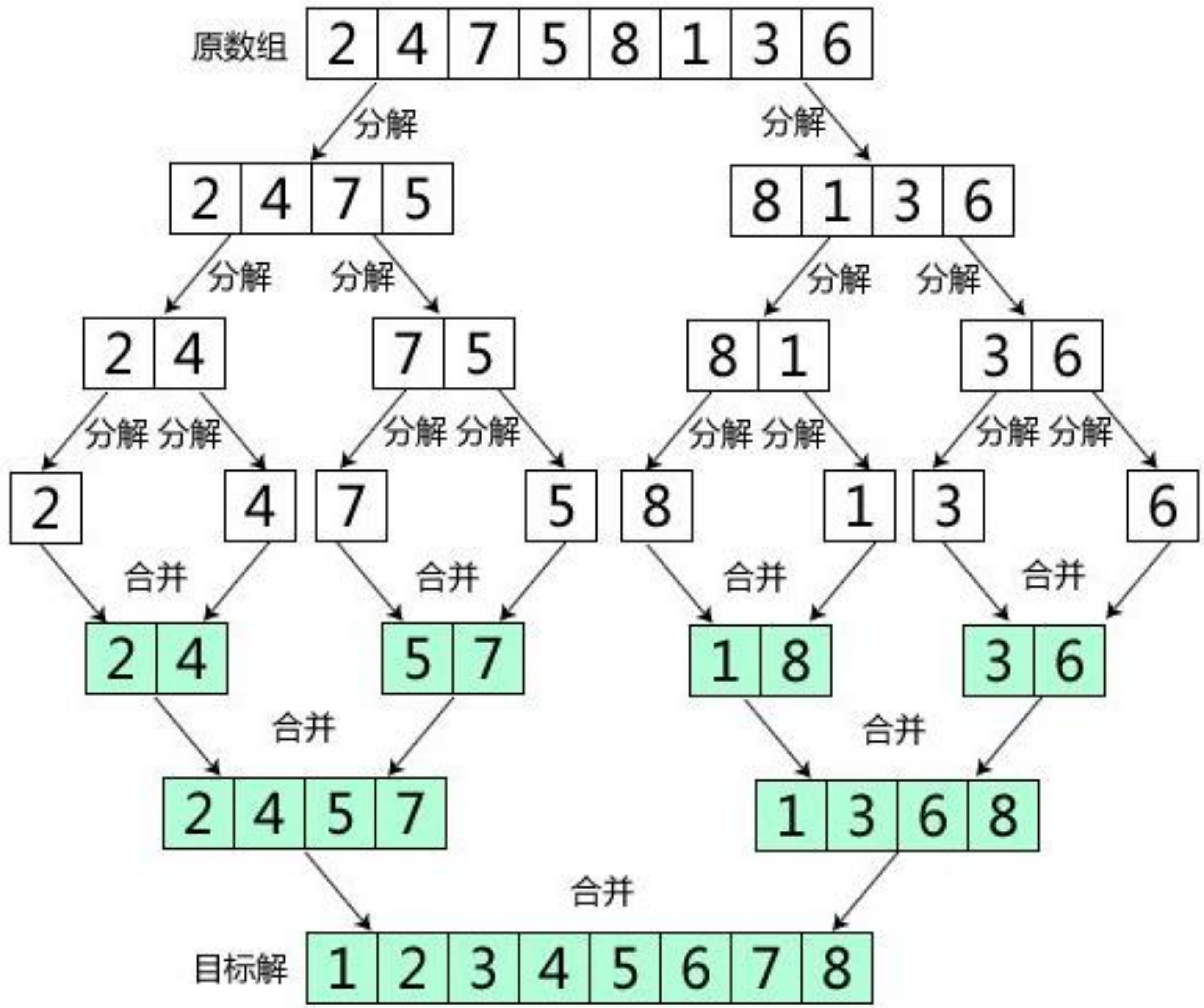


- **分治：**

“分而治之”，就是把一个复杂的问题分成两个或更多的相同或相似的子问题，再把子问题分成更小的子问题……直到最后子问题可以简单的直接求解，原问题的解即子问题的解的合并。

- 分治主要分三步走：分、治、合。
- 分：把问题分成n(一般为2)个子问题；
- 治：解决子问题（继续分下去，或是规模小到一定程度后直接求解）；
- 合：将两个子问题合并，求出原问题的解。
- 归并排序，快速排序

归并排序



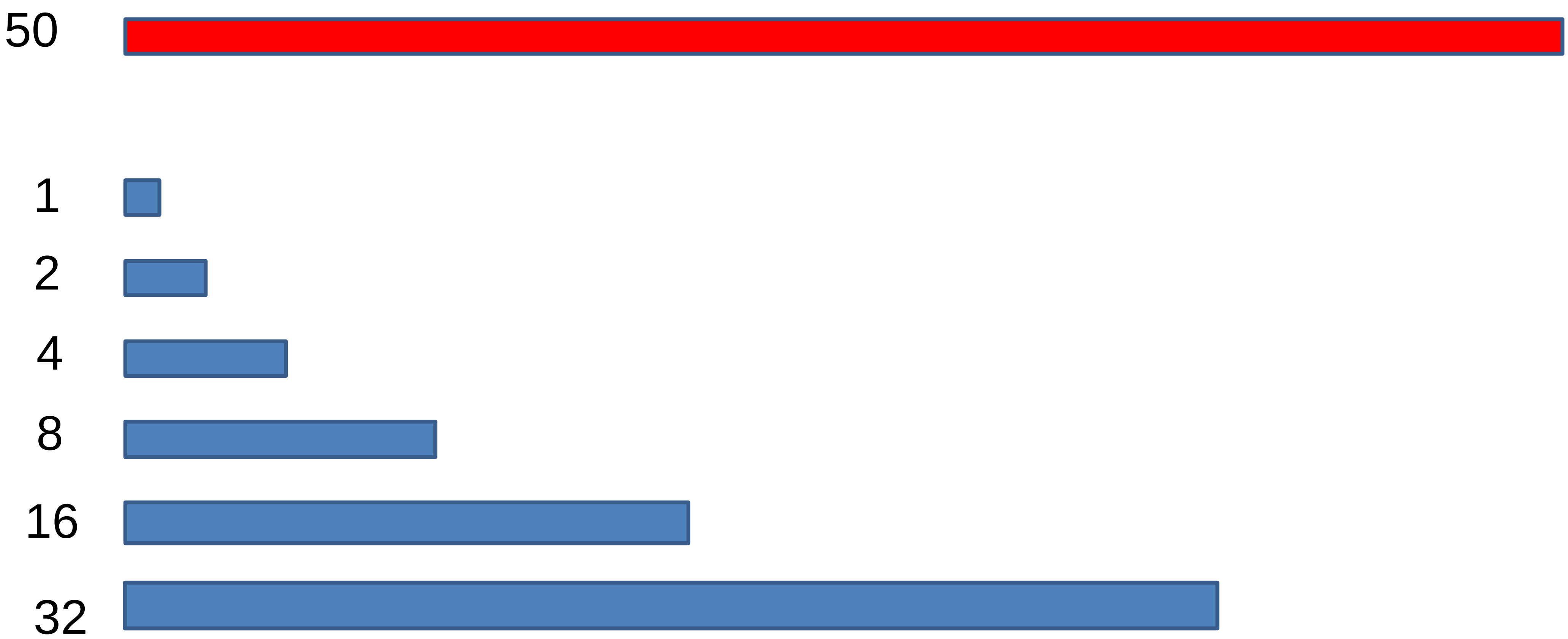
原数组
2 4 7 5 8 1 3 6
分解
分解
2 4 7 5
8 1 3 6
分解 分解
分解 分解
2 4
7 5
8 1
3 6
分解 分解
分解 分解
分解 分解
分解 分解
2
4
7
5
8
1
3
6
合并
合并
合并
合并
2 4
5 7
1 8
3 6
合并
合并
2 4 5 7
1 3 6 8
合并
目标解
1 2 3 4 5 6 7 8

◆二分是一种特殊的分治

- 一类问题，我们很难直接从正面求解，但它的答案范围有边界，即具有有界性，且（至少在一定范围内）具有单调性（顶多局部不变），我们就可以拓展一下分治思想，从答案这边二分，不断逼近正解，直到找到正解（整数）或逼近到精度内（实数）。
- 2010关押罪犯，二分答案+染色；
- 2011 聪明的质检员，二分答案；
- 2012 借教室，二分答案+前缀和；
  疫情控制，二分答案+图论；
- 2015 河中跳房子。
- 所以，二分答案也是NOIP中一个常考知识点，需要读者认真思考，辨别二分答案特征，用二分答案更好地解决问题。

◆倍增（成倍的增长，通常是2倍）

- 所谓倍增，就是把一个数据规模为n的问题分解成若干个2^{a_i}的和，预处理数据范围内所有2^{a_i}的情况，再将这些规模为2^{a_i}的问题通过一定的方法合并，得出原问题的解。
- 分治是把整个问题分成几个互不重复的子问题，合并求解；
- 倍增是找互为倍数关系的子问题之间的联系，再合并求解。
- 可以认为，倍增也体现了分治思想。
- 适用题型：问题规模大，且成倍数据规模的问题之间存在简单的递推关系，可以轻易地由小范围求出大范围。


50
1
2
4
8
16
32

- 每次根据已经得到的信息，将考虑的范围扩大一倍，从而加速操作。
- 使用范围：
- 1. 每次的变化规则相同
- 2. 变化规则必须满足结合律

常见倍增：

- 快速幂
- ST表求RMQ
- LCA

一 快速幂运算

◆快速求幂：

- 整数快速幂：a是正整数：a^n
- 矩阵快速幂：A是矩阵：A^n

整数快速幂

已知a，n，p是正整数：求a^n mod p的值问题。

1<=a,p<=10000，n<=10^9（有时候n<10^{15})

模运算:

- $(a + b) \% p = (a \% p + b \% p) \% p = (a\%p+b)\%p$
- $(a * b) \% p = (a \% p * b \% p) \% p = (a\%p*b)\%p$
- $(a - b) \% p = (a \% p - b \% p) \% p == (a \% p - b \% p + p) \% p$
- $a^n \bmod p = (\ldots((((a \bmod p)*a) \bmod p)*a) \bmod p \ldots *a) \bmod p$

```
s=1;
for(int i=0;i<n;i++)s=(s*a)%p;
```

◆位运算：&，|，>>，<<

- n & 1:取n的二进制最末位;
- n >> 1:右移1位，相当于去掉n的二进制最末位。
- n>>1相当于n/2;
- n<<1相当于n*2;
- if(n%2==1)可以写成if((n&1)==1)或if(n&1)
- 位运算比+-*/速度快

- 朴素的算法求$a^n \% p$：O(n)
- s=1;
- for (int i=0;i<n;i++) s=(s*a)%p;

◆减少乘法次数的快速方法：

- 求a^{19}
- a
- $a*a=a^2$
- $a^2*a^2=a^4$
- $a^4*a^4=a^8$
- $a^8*a^8=a^{16}$
- $a^{19}=a*a^2*a^{16}$
- 6次乘法

◆方法1：分治思想

- a^100=(a50)^2
- a^101=(a^50)*a
- 求a^n的值:
- s=a^(n/2);
- s=s*s;
- if(n%2==1)s=s*a;
- return s;

◆//分治（递归实现）： $a^n \bmod p$

```
int fast2(int a,int n,int p){
    if(n==1)return a%p;
    int s=fast2(a,n/2,p);
    s=(s*s)%p;
    if(n%2==1)s=(s*a)%p;
    return s;
}
```

2. 倍增思想：$a^n \bmod p$

- 求a^{19}的快速实现：
- （1）把19 转化为二进制
- $(19)_{10}=(10011)_2$，
- 即$19=2^4+2^1+2^0=16+2+1$；
- 转化二进制的同时依次求出a^2, a^4, a^8, a^{16}。
- （2）求a^{19}转为求$a^{2^4+2^1+2^0}=a^{1+2+16}=a^{16}*a^2*a^1$。

对于a^n

- 把n按二进制展开,其中b_i（0<=i<=k）为0或1

$$n = b_k 2^k + b_{k-1} 2^{k-1} + \cdots + b_1 2^1 + b_0$$

$$a^n = \mathrm{a}^{b_k 2^k + b_{k-1} 2^{k-1} + \cdots + b_1 2^1 + b_0}$$
$$= \mathrm{a}^{b_k 2^k} * \mathrm{a}^{b_{k-1} 2^{k-1}} * \cdots * \mathrm{a}^{b_1 2^1} * \mathrm{a}^{b_0}$$

b_i=0的情况不用处理，只处理b_i=1的情况即可

$$\mathrm{a}^{2^i} = \left(\mathrm{a}^{2^{i-1}}\right)^2$$

```
//a^n%p
s=1;
while(n>0){
    if(n%2==1)s=(s*a)%p;
    a=(a*a)%p;
    n=n/2;
}
```

◆倍增实现：$a^n \bmod p$

```
int fast(int a,int n,int p){
    int s=1%p;
    while(n){
        if(n&1)s=(s*a)%p;
        a=(a*a)%p;
        n=n>>1;
    }
    return s;
}
```

◆训练1：P1226 【模板】快速幂||取余运算

◆输入b，p，k的值，求b^p mod k的值。

◆其中b，p，k*k为长整型数。

◆训练2：洛谷：AT1357 n^p mod m

- **题意翻译**
- 求 n^p mod m 的值
- **输入**
- 一行，为整数 n,m,p(注意顺序)
- **输出格式**
- 一行，为 n^p mod m的值
- **数据说明**
- 对于100%的数据 1≤n,m≤10^9;1≤p≤10^14

- 注意两种范围：
- 1.$n<10^9$,$a,p<10000$.
- 2.$a,p,n<10^9$.

- 训练3：转圈游戏p1965
- （NOIP2013 day1第一题）

- 第1轮后：$(x+m) \bmod n$
- 第2轮后：$((x+m) \bmod n)+m) \bmod n = (x+2m) \bmod n$
- …
- 第i轮后：$(x+i*m) \bmod n$
- 所以10^k轮后x号的位置是$(x+10^k*m) \bmod n$
- 当$0<k<10^9$时，求$(x+10^k*m) \bmod n$，事实上：
- $(x+10^k*m) \bmod n = (x+((10^k \bmod n)*m) \bmod n) \bmod n$
- 问题的关键就是求$10^k \bmod n$，显然用到快速幂求模。

应用扩展：

- **输入x，n，p**
- **求$(x+x^2+x^3+\ldots+x^n) \bmod p$的值。**
- x,p<10000
- $n<=10^9$。

输入：3 1000000000 10007
输出：6215

输入：3 999999999 10007
输出：8742

- 由于n比较大，不能逐项求幂然后相加。
- 采取分治（二分）的方法：
- 当n为偶数时：
- $x+x^2+x^3+\cdots+x^n$
- $=(x+x^2+\cdots+x^{n/2})+x^{n/2}(x+x^2+\cdots+x^{n/2})$
- 当n为奇数时：
- $x+x^2+x^3+\cdots+x^n$
- $=(x+x^2+\cdots+x^{n/2})+x^{n/2}(x+x^2+\cdots+x^{n/2})+x^n$
- 或者$=(x+x^2+\cdots+x^{n/2})+x^{n/2+1}(1+x+x^2+\cdots+x^{n/2})$

```
int sum(int x,int n){//x+x^2+..+x^n
	if(n==1)return x%p;
	int s=sum(x,n/2)%p;
	s=(s+s*fast(x,n/2,p))%p;
	if(n&1)s=(s+fast(x,n,p))%p;
	return s;
}
```

矩阵快速幂

- 将整数快速幂中的整数a换成矩阵A，求A^n
- 快速幂运算依然能进行。

$$A_{n*n}=\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \qquad B_{n*n}=\begin{pmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{nn} \end{pmatrix}$$

$$A_{n*n}+B_{n*n}=C_{n*n}=\begin{pmatrix} c_{11} & \cdots & c_{1n} \\ \vdots & \ddots & \vdots \\ c_{nn} & \cdots & c_{nn} \end{pmatrix}$$

其中：$c_{ij=}a_{ij+}b_{ij}$

$$A_{n*m} = \begin{pmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nm} \end{pmatrix} \qquad B_{m*p} = \begin{pmatrix} b_{11} & \cdots & b_{1p} \\ \vdots & \ddots & \vdots \\ b_{m1} & \cdots & b_{mp} \end{pmatrix}$$

$$A_{n*m} * B_{m*p} = C_{n*p} \begin{pmatrix} c_{11} & \cdots & c_{1p} \\ \vdots & \ddots & \vdots \\ c_{n1} & \cdots & c_{np} \end{pmatrix}$$

$$c_{ij} = a_{i1} * b_{1j} + a_{i2} * b_{2j} + \cdots + a_{im} * b_{mj} = \sum_{k=1}^{m} a_{ik} * b_{kj}$$

$$\begin{bmatrix} 1 & 0 & 2 \\ -1 & 3 & 1 \end{bmatrix} \times \begin{bmatrix} 3 & 1 \\ 2 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} (1\times3+0\times2+2\times1) & (1\times1+0\times1+2\times0) \\ (-1\times3+3\times2+1\times1) & (-1\times1+3\times1+1\times0) \end{bmatrix} = \begin{bmatrix} 5 & 1 \\ 4 & 2 \end{bmatrix}$$

单位矩阵E：E*A=A

$$E=\begin{bmatrix}1 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & 1\end{bmatrix}$$

- **例4：P3390【模板】矩阵快速幂**
- **已知A_{n*n}求,求A^{k}mod** 10^9+7；
- n<=100, k<=10^12, |矩阵元素|<=1000。
- 输入:
- 第一行：n,k
- 以下是n行n列的矩阵A。
- 样例输入：
- 2 10 10000
- 1 1
- 1 0
- 样例输出：
- 89 55
- 55 34

```cpp
#define LL long long
#define MD 1000000007;
using namespace std;
struct Matrix{
    LL a[101][101];
};
Matrix A;
int n;
LL k;
Matrix operator *(const Matrix &A,const Matrix &B){
    Matrix C;
    memset(C.a,0,sizeof(C.a));
    for(int i=1;i<=n;i++)
        for(int j=1;j<=n;j++)
            for(int k=1;k<=n;k++)
                C.a[i][j]=(C.a[i][j]+A.a[i][k]*B.a[k][j])%MD;
    return C;
}
```

```
Matrix fast(Matrix A,LL k){
	Matrix S=A;
	k--;
	while(k){
		if(k&1) S=S*A;
		A=A*A;
		k=k>>1;
	}
	return S;
}
```

```cpp
int main(){
	cin>>n>>k;
	for(int i=1;i<=n;i++)
		for(int j=1;j<=n;j++)cin>>A.a[i][j];
	Matrix ans=fast(A,k);
	for(int i=1;i<=n;i++){
		for(int j=1;j<n;j++)cout<<(ans.a[i][j])<<" ";
		cout<<(ans.a[i][n])<<endl;
	}
	return 0;
}
```

5.Matrix Power Series（poj3233）

- 给定一个n*n的矩阵A以及一个正整数k。
- 计算 $S = A^1 + A^2 + A^3 + \cdots + A^k$
- 输入：n,k,m
- n*n的矩阵。
- n $(n \le 30)$
- k $(k \le 10^9)$
- m $(m < 10^4)$ 模

输入：
2 2 4
0 1
1 1

输入：
1 2
2 3

6. Fibonacci 第 n 项

- **题目背景**
- 大家都知道，斐波那契数列是满足如下性质的一个数列：
- • $f(1) = 1$
- • $f(2) = 1$
- • $f(n) = f(n-1) + f(n-2)$ ($n \ge 2$ 且 n 为整数)
- **题目描述**
- 请你求出 $f(n) \bmod 1000000007$ 的值。

$f(1)=f(2)=1$; $f(n)=f(n-1)+f(n-2)$

$$[f(n),f(n-1)]=[f(n-1),f(n-2)]*\begin{bmatrix}1 & 1\\1 & 0\end{bmatrix}$$

$$[f(n),f(n-1)]=[f(2),f(1)]*\begin{bmatrix}1 & 1\\1 & 0\end{bmatrix}^{n-2}$$

$$[f(n),f(n-1)]=[1,1]*\begin{bmatrix}1 & 1\\1 & 0\end{bmatrix}^{n-2}$$

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

求出$S=A^{n-2}$

$f(n)=1*S[1][1]+1*S[2][1]$

- **7. P1939 【模板】矩阵加速（数列）**
- f[1]=f[2]=f[3]=1
- f[i]=f[i-3]+f[i-1] (i>3)
- 求a数列的第n项对1000000007(10^9+7)取余的值。

- $[f(i),f(i-1),f(i-2)]$
- $=[f(i-1),f(i-2),f(i-3)]*\begin{bmatrix}1 & 1 & 0\\0 & 0 & 1\\1 & 0 & 0\end{bmatrix}$
- ...
- $=[f(3),f(2),f(1)]*\begin{bmatrix}1 & 1 & 0\\0 & 0 & 1\\1 & 0 & 0\end{bmatrix}^{i-3}$

$=[1,1,1]*\begin{bmatrix}1 & 1 & 0\\0 & 0 & 1\\1 & 0 & 0\end{bmatrix}^{i-3}$

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

$[f(n),f(n-1),f(n-2)]=[1,1,1]*A^{n-3}$

$S=[1,1,1]*A^{n-3}$

f(n)=S.a[1][1]+S.a[2][1]+S.a[3][1]

◆矩阵加速算法求k阶齐次递推数列：

- 已知k阶齐次递推数列f(i)的通向公式
- f(i)=a1*f(i-1)+a2*f(i-2)+..+ak*f(i-k)　(i>=k)
- 初始值f(1),f(2),..,f(k)。

构造状态矩阵和转移矩阵（必须是常数）

- [f(i),f(i-1),..,f(i-k+1)]=[f(i-1),..,f(i-k)]*A

$$A=\begin{bmatrix} \mathrm{a}1 & 1 & 0 & . & . & 0 \\ a2 & 0 & 1 & . & . & 0 \\ a3 & 0 & 0 & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & 1 \\ ak & 0 & 0 & . & . & 0 \end{bmatrix}$$

- [f(i),f(i-1),..,f(i-k+1)]=[f(k),..,f(1)]*A^{i-k}
- S=A^{i-k}
- f(i)=f(k)*S.a[1][1]+f(k-1)*S.a[2][1]+..+f(1)*S.a[k][1]

◆提升：矩阵加速求和

1.求s(n)=x+x^2+..+x^n的矩阵加速算法

a(n)=x^n
a(n)=x*a(n-1)
s(n)=s(n-1)+a(n)

[s(n),a(n+1)]
=[s(n-1),a(n)]* $\begin{bmatrix} 1 & 0 \\ 1 & x \end{bmatrix}$

=[s(1),a(2)]* $\begin{bmatrix} 1 & 0 \\ 1 & x \end{bmatrix}^{n-1}$

8. 1643：【例 3】Fibonacci 前 n 项和

- f(n)=f(n-1)+f(n-2)
- s(n)=f(1)+..+f(n)
- s(n)=s(n-1)+f(n)
- 构造状态矩阵：
- [s(n),f(n),f(n-1)]
- =[s(n-1),f(n-1),f(n-2)]* $\begin{vmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \end{vmatrix}$
- =[s(2),f(2),f(1)] $\begin{vmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \end{vmatrix}^{n-2}$

◆或者更好：

- $f(n)=f(n-1)+f(n-2)$
- $s(n)=f(1)+..+f(n)$
- $s(n)=s(n-1)+f(n)$
- 构造状态矩阵：
- $[s(n),f(n+1),f(n)]$
- $=[s(n-1),f(n),f(n-1)]*\begin{vmatrix}1 & 0 & 0\\1 & 1 & 1\\0 & 1 & 0\end{vmatrix}$
- $=[s(1),f(2),f(1)]\begin{vmatrix}1 & 0 & 0\\1 & 1 & 1\\0 & 1 & 0\end{vmatrix}^{n-1}$

◆9.P5175 数列

题目描述

一个数列a_n，已知a_1及a_2两项。

数列a_n满足递推式$a_n = x \times a_{n-1} + y \times a_{n-2} (n \geq 3)$.

求

$$\sum_{i=1}^{n} a_i^2.$$

由于答案可能过大，对$10^9 + 7$取模。

> 对于前20%的数据，保证$x = y = 1$。
>
> 对于100%的数据，$T = 30000, 1 \leq n \leq 10^{18}, 1 \leq a_1, a_2, x, y \leq 10^9$。

$$a_n = x a_{n-1} + y a_{n-2}$$

$$a_n^2 = x^2 a_{n-1}^2 + 2xy\, a_{n-1} a_{n-2} + y^2 a_{n-2}^2$$

$$S_n = S_{n-1} + a_n^2$$

$$\begin{bmatrix} S_n & a_{n+1}^2 & a_n^2 & a_n a_{n+1} \end{bmatrix}$$

$$= \begin{bmatrix} S_{n-1} & a_n^2 & a_{n-1}^2 & a_{n-1} a_n \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & x^2 & 1 & x \\ 0 & y^2 & 0 & 0 \\ 0 & 2xy & 0 & y \end{bmatrix}$$

◆10.yt1644：【例 4】佳佳的 Fibonacci

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

$S(n)=\cancel{S(n-1)+f(n)}\ f(1)+f(2)+\cdots f(n)$

$t(n)=f(1)+2f(2)+\cdots+nf(n)$.

令 $y(n)=nS(n)-t(n)=(n-1)f(1)+(n-2)f(2)+\cdots f(n-1)$

有：$y(n+1)=y(n)+S(n)$.

求出 $y(n)$ 即可. $t(n)=nS(n)-y(n)$.

$$[y(n), S(n), f(n+1), f(n)]=[y(n-1), S(n-1), f(n), f(n-1)]\begin{bmatrix}1&0&0&0\\1&1&0&1\\0&1&1&0\\0&0&1&0\end{bmatrix}$$