数论

0.写在前面

这部分不需要了解得很深入，因为难度过高，出题不多。

当然想学的话学一些最好。

不讲多项式，也 **十分不推荐** 学习多项式全家桶。

下面列的都是必须会的，否则非数论题可能都做不了，无特殊说明下文中的数（以及函数的定义域）**均为整数**。

素数集合记为 P 或 $Prime$.

最大公因数记为 (a,b) 或 $\gcd(a,b)$。

$[expr]$ 表示 $expr$ 成立时取 1，否则取 0。（相当于 C++ 中的 `(expr)?1:0`）

I.同余与整除

- 带余除法：就是小学最早学的那个除法。设 $p=r*s+m$，称 r 是 p 除以 s 的商，m 为 p 除以 s 的余数。其中 $0\le m\le s$.
- 同余：两数除以 p 余数相同，称两数 **模 p 意义下同余**，记作 $a\equiv b\pmod p$，且 a,b 可以为负数。

  一般取余运算记作 $a \bmod p$（数学中不写取余运算）。平时直接写成 C++ 的运算 $a\%b$ 也行。（不正规）

  一个计算式：$x \bmod p=x-p\cdot\lfloor\frac{x}{p}\rfloor$.
- 同余运算性质：加、减、乘运算结果与对应余数的运算结果同余。即：
  - $(a\pm b)\bmod p=(a\bmod p)\pm(b\bmod p)$
  - $(a\cdot b)\bmod p=(a\bmod p)\cdot(b\bmod p)$
  - $(A\bmod p)^k=(A^k)\bmod p$

  没有除法!

tip1.当题目中要求对某数取余时，注意取余的目的，有可能是单纯为了避免溢出，也有可能与复杂度或所需算法有关。

tip2.常见质数：$19260817, 998244353, 1e9+7, 1e9+9$，后两个常用作双模 hash 模数。

常见非质数：$998244853, 1e7+7, 1e6+7$

- 费马小定理与欧拉公式

  下式的 $\phi(x)$ 为欧拉函数，下面会讲；费马小定理是欧拉公式的特例。

  - 欧拉定理：$a^k \equiv a^{k \bmod \phi(m)} \pmod m, 其中 (a,m)=1$.
  - 费马小定理：$a^k \equiv a^{k \bmod (p-1)} \pmod p, 其中 p \in Prime$.
  - 欧拉降幂公式: $a^k \equiv a^{k \bmod \phi(m)+\phi(m)} \pmod m, 允许 (a,m) \neq 1$.

    此式可解决指数过大，快速幂会超时的求指数问题，或一些指数较特殊的问题。

- 整除：当 a 除以 p 余数为 0 时，称 p 整除 a，记作 $p|a$ （注意左右顺序）。

II.素数筛、积性函数

素数筛

埃氏筛（埃拉托斯特尼筛法）：找到一个素数，就删去所有其倍数。

复杂度：$\sum\limits_{p \le n} \frac{n}{p} (p \in Prime)$ 即 $O(n \log \log n)$

复杂度证明需要更高级的知识，此处略。

```
bool notp[N];
void init(){
    notp[1]=1;
    for(int i=1;i<=n;++i)
        if(not notp[i]){
            int lim=n/i;
            for(int j=2;j<=lim;++j)
                notp[i*j]=1;
        }
}
```

欧拉筛：埃氏筛慢的原因为部分数被多次删除，如 6 被 $2,3$ 各删除一次，考虑改善该情况。我们令一个数只被其 **最小质因数** 删去。具体地，从 2 到 $\frac{n}{i}$ 枚举素数 j 时，我们只枚举到第一个满足 $j|i$ 的 j 就停止，这样可以保证 j 是 $i*j$ 的最小质因数。

复杂度：$O(n)$

```
bool isp[N];
vector<int> p;
void init(){
    memset(isp,1,sizeof(isp));
    isp[1]=0;
    for(int i=2;i<=n;++i){
        if(isp[i])p.push_back(i);
        for(auto x:p){
            if(i*x>n)break;
            isp[i*x]=0;
            if(!i%x)break;
        }
    }
}
```

积性函数

积性函数：满足 $f(a\cdot b)=f(a)\cdot f(b)$,其中$(a,b)=1$ 的函数。

完全积性函数：满足 $f(a\cdot b)=f(a)\cdot f(b)$,其中 a,b 任意 的函数。

常见的积性函数：欧拉函数$\phi(x)$，莫比乌斯函数 $\mu(x)$.

欧拉筛将多数合数 x 拆分为 $p*q$，且 p 为其最小质因数，显然有 $(p,q)=1$ 或 p，我们可以借助这一性质计算 1 至 n 中每个数的函数值。

代码如下，我们只需素数的函数值和 $(p,q)=q$ 时 $f(p\cdot q)$ 的计算式即可。

```cpp
bool isp[N];
int f[N];
vector<int> p;
void init(){
    memset(isp,1,sizeof(isp));
    isp[1]=0;
    for(int i=2;i<=n;++i){
        if(isp[i]){
            p.push_back(i);
            f[i]=f_1(i); //i为素数，需要公式，如phi(i)=i-1
        }
        for(auto x:p){
            if(i*x>n)break;
            isp[i*x]=0;
            if(!(i%x)){
                f[i]=f_2(i*x);//(i,x)=x，需要公式，如phi(i*x)=x*phi(i)
                break;
            }
            f[i*x]=f[i]*f[x];//(i,x)=1,利用积性函数性质计算
        }
    }
}
```

III.算数基本定理、GCD、逆元、同余方程

算数基本定理

算数基本定理：$x = p_1^{k_1} \cdot p_2^{k_2} \cdot \ ... \ \cdot p_n^{k_n}$，其中 $p_i \in Prime$

即一个整数可以分解为若干质数之积。$O(\sqrt{n})$ 的分解应该都会写。

这一般不是一个直接考察的知识点，但是常见的思考问题方式。

eg. 你们可能做过的比赛题

GCD

更相减损，辗转相除，大伙应该都会。

高精度数据的 GCD应当使用更相减损，因为你大概不会大数求余。~~我也不会~~

批量GCD也可以考虑使用算数基本定理，不过不常见，不要求掌握。

求解二元一次方程（扩展欧几里得算法）

求形如 $ax + by = c$ 的关于 x, y 的一元二次方程的整数解。

首先有 $(a,b)|(ax+by)$，即要求 $(a,b)|c$，不满足则无解。

先考虑求解 $ax+by=(a,b)$，若已知 $bx_0+(a\ mod\ b)y_0=(b,a\ mod\ b)$，则有 $bx_0+(a-\lfloor\frac{a}{b}\rfloor\cdot b)y_0=(a,b)$.

即 $ay_0+b\cdot(x_0-\lfloor\frac{a}{b}\rfloor y_0)=(a,b)$.

对应项相等，即 $x=y_0$，$y=x_0-\lfloor\frac{a}{b}\rfloor y_0$ 是方程 $ax+by=(a,b)$ 的一组解。

故递归求解 $bx_0+(a\ mod\ b)y_0=(b,a\ mod\ b)$ 即可，与辗转相除法过程相似（称作 exGCD），故复杂度：$O(\log n)$

解出 $ax+by=(a,b)$ 后，有 $a(x\cdot\frac{c}{(a,b)})+b(y\cdot\frac{c}{(a,b)})=c$，即得原方程 $ax+by=c$ 的解。

```
int x,y
void exgcd(int a,int b){
    if(b==0){
        x=1,y=0;
        return;
    }
    exgcd(b,a%b);
    int tmp=x;
    x=y;
    y=tmp-(a/b)*y;
}
```

板子

乘法逆元

逆元：若 $a\cdot r\equiv 1\ (mod\ p)$，称 r 为 a 的逆元（又称数论倒数），记作 a^{-1}。

因为除法不能保证同余，我们可将除法改为乘逆元。

求逆元的方法：

- 费马小定理：$a^k\equiv a^{k\ mod\ (p-1)}$ 即 $a^{p-2}\cdot a\equiv a^{p-1}\equiv 1\ (mod\ p)$，即 p 为质数时 a 的逆元为 a^{p-2}
- 扩展欧几里得算法：$a\cdot r\equiv 1\ (mod\ p)$ 即 $a\cdot r+k\cdot p=1$，利用 exgcd 求出 r 即可，适用于 p 非质数的情况。
- 递推求逆：$a^{-1}\equiv-\lfloor\frac{p}{a}\rfloor\cdot(p\ mod\ a)^{-1},(mod\ p)$，递推即可。

- 阶乘求逆：计算组合数常用。求 $1!$ 到 $n!$ 的逆元：先求 $n!$ 的逆元，则 $(i!)^{-1}=((i+1)!)^{-1}*(i+1)$

单独求逆板子

连续求逆板子

同余方程

指形如 $f(x)\equiv r\pmod p$ 的关于 x 的方程。

下面列出两种较简单的方程。

- 线性同余方程 $kx\equiv r\pmod p$

  $r=1$ 其实就是我们求逆元时的情况。化成 $k\cdot x+m\cdot p=r$ 求解即可。

- 指数同余方程 $k^x\equiv r\pmod p$，$p\in Prime$

  使用 BSGS 算法。

  由费马小定理 $x\in[0,p)$，设 $x=ta-b$，t 为常数，$b\in[0,t)$，$k^t=m$，则 $k^{ta-b}\equiv r\pmod p$，即$m^a\equiv r^b\pmod p$.

  注意到右侧 r^b 有 t 个取值，左侧 m^a 有不超过 $\lceil\frac{p}{t}\rceil$ 个取值，问题变为在两个序列中查找相同值。

  可用 hash 实现，期望复杂度 $O(\max(t,\lceil\frac{p}{t}\rceil))$；或用 set 实现，复杂度 $O(\max(t,\lceil\frac{p}{t}\rceil)\log t)$

  取 $t=\lfloor\sqrt{p}\rfloor$ 获得最优复杂度 $O(\sqrt{p})$

IV.欧拉函数、莫比乌斯函数、数论分块

欧拉函数

$\phi(x)$ 表示 $1-x$ 中与 x 互质的数的个数。

不容易证明，$\phi(x)$ 是积性函数，所以证明略。

显然 $\phi(p)=p-1\ ,p\in Prime$，$\phi(p^n)=p^{n-1}\cdot(p-1)$

由积性，$\phi(x)=\prod\phi(p_i^{k_i})=x\prod(1-\frac{1}{p_i})$，其中 $x=p_1^{k_1}\cdot p_2^{k_2}\cdot\ \dots\ \cdot p_n^{k_n}$。

欧拉反演：$x=\sum\limits_{k|x}\phi(k)$

eg. 求$\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n}[\gcd(i,j)=1]$.(P2158)

根据欧拉函数的定义：$\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n}[\gcd(i,j)=1]=2\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{i-1}[\gcd(i,j)=1]+1=2\sum\limits_{i=1}^{n}\phi(i)+1$

使用欧拉筛可以 $O(n)$ 计算 $\sum\limits_{i=1}^{n}\phi(i)$

莫比乌斯函数

令 $x=p_1^{k_1}\cdot p_2^{k_2}\cdot\ ...\ \cdot p_n^{k_n}$，其中 $p_i\in Prime$

$\exists k_i>1$ 时 $\mu(x)=0$，否则 $\mu(x)=(-1)^n$，特殊地 $\mu(1)=1$。

容易证明，$\mu(x)$ 是积性函数，所以证明略。

莫比乌斯反演：$[x=1]=\sum\limits_{k|x}\mu(k)$.

数论分块

求 $\sum\limits_{i=1}^{m}\lfloor\frac{n}{i}\rfloor f(i)$，其中 $\sum\limits_{i=1}^{n}f(i)=S(n)$，$S(n)$ 均已知。

直接求复杂度肯定是 $O(n)$ 的，但我们发现 $\lfloor\frac{n}{i}\rfloor$ 是不增的，且存在大量重复的值，可以一次性计算数值相同一段的和。

eg. $n=12$ 时$\lfloor\frac{n}{i}\rfloor=12,6,4,3,2,2,1,1,1,1,1,1$，只有 6 个不同的值。

若已知左端点为 l，则容易得出右端点 $r=\lfloor\frac{n}{\lfloor\frac{n}{l}\rfloor}\rfloor$，区间和为 $(r-l+1)\cdot\lfloor\frac{n}{l}\rfloor$

```
void calc(int m,int n){
    for(int l=1,r;l<=m;l=r+1){
        if(n/l)r=min(n/(n/l),m);
        else r=m;
        //注意l>n时n/l为0，n/(n/l)会发生错误。
        ans+=(n/l)*(S(r)-S(l-1));
    }
}
```

复杂度分析：即求 $\lfloor \frac{n}{i} \rfloor$ 不同值的个数。

- $i \le \sqrt{n}$ 时：至多 $\sqrt{n}$ 个，因为 i 只有 $\sqrt{n}$ 个。
- $i > \sqrt{n}$ 时：$\lfloor \frac{n}{i} \rfloor \le \sqrt{n}$ ，至多 $\sqrt{n}$ 个取值。

因此复杂度为 $O(\sqrt{n})$

应用：

- 求 $\sum\limits_{i=1}^{m} n \bmod i$.(P2261)

  $\sum\limits_{i=1}^{m} n \bmod i = \sum\limits_{i=1}^{m} (n - \lfloor \frac{n}{i} \rfloor \cdot i) = nm - \sum\limits_{i=1}^{m} (\lfloor \frac{n}{i} \rfloor \cdot i)$

- 求 $\sum\limits_{i=1}^{m} \sum\limits_{j=1}^{n} [\gcd(i,j) = 1]$. (P2257)

  使用莫比乌斯反演：$\sum\limits_{i=1}^{m} \sum\limits_{j=1}^{n} [\gcd(i,j) = 1] = \sum\limits_{i=1}^{m} \sum\limits_{j=1}^{n} \sum\limits_{k \mid i, k \mid j} \mu(k) = \sum\limits_{i=1}^{m} \sum\limits_{k \mid i} \sum\limits_{j=1}^{\lfloor \frac{n}{k} \rfloor} \mu(k) = \sum\limits_{k=1}^{\min(m,n)} \lfloor \frac{m}{k} \rfloor \lfloor \frac{n}{k} \rfloor \mu(k)$

  这是二阶数论分块，复杂度不变。$\mu(k)$ 可使用欧拉筛求出。

V.矩阵相关

矩阵运算

矩阵运算满足结合律、分配律，不满足交换律。

- 矩阵加法：$A_{i,j} = B_{i,j} \cdot C_{i,j}$
- 矩阵乘法：$A_{i,j} = \sum\limits_{k=1}^{n} B_{i,k} \cdot C_{k,j}$
- 行列式求值

矩阵优化递推

- 二阶递推数列：$a_i = x a_{i-1} + y a_{i-2}$.

对此类数列，我们可写出转移矩阵：$T=\begin{bmatrix}x & y\\ 1 & 0\end{bmatrix}$，即 $\begin{bmatrix}a_{i-1} & a_{i-2}\end{bmatrix}\cdot\begin{bmatrix}x & 1\\ y & 0\end{bmatrix}=\begin{bmatrix}a_i & a_{i-1}\end{bmatrix}$.

则 $\begin{bmatrix}a_1 & a_0\end{bmatrix}\cdot T^k=\begin{bmatrix}a_{k+1} & a_k\end{bmatrix}$，只需求 T^k 即可知 a_k，而 T^k 可用快速幂求出。

由此可 $O(\log n)$ 求出 a_n .

如果递推式还含a_{i-3}，则称为三阶递推，方法不变。进一步地，m 阶递推 a_n 的复杂度为 $O(m^3\log n)$

最常见的二阶递推数列为斐波那契数列，这个数列性质很多，感兴趣的可自行研究。

  - 与矩阵无关的东西：若 a_n 对一个较小的数 p 取模 （如 5000 ），也可以找递推数列的循环节长度进行计算。

    a_i,a_{i+1} 取余后各有 p 种可能，故循环节长度不超过 p^2 ，复杂度不超过 $O(p^2)$

- 线段树维护数列

  CF718C：实现两种操作：区间 $[l,r]$ 间的 a_i 增加k；给定 l,r 求 $\sum\limits_{i=l}^{r}f_{a_i}$ ，结果对 $1e9+7$ 取余，f_i 为斐波那契数列。

  考虑每个位置维护 T^{a_i} ，则区间加 k 变为区间乘 T^k ，区间求斐波那契数列和变为求 $\begin{bmatrix}a_1\\ a_0\end{bmatrix}\cdot\sum\limits_{i=l}^{r}T^{a_i}$ 。

- 表达特殊操作

  LOJ6208：树上每个点有权值 k_i,t_i ，有两个操作：选定一条路径 (u,v) 并给出 d ，使其上的点 $k_i\leftarrow k_i+d$ 或 $t_i\leftarrow t_i+k_i\cdot d$.

  直接维护难度较高，考虑构造矩阵 $\begin{bmatrix}k_i & t_i & 1\end{bmatrix}$ ，则操作一可写为 $X=\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ d & 0 & 1\end{bmatrix}$ ，操作二可写为 $Y=\begin{bmatrix}1 & d & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$ ，线段树维护区间乘法即可。

广义矩阵乘法

设 $\otimes, \oplus$ 是两种运算，规定一种矩阵运算 $A * B = C$ 满足 $C_{i,j} = \bigoplus_{k=1}^{n} A_{i,k} \otimes B_{k,j}$ ，若该运算满足结合律，则称 $A * B$ 为广义矩阵乘法。如 $\otimes = \times, \oplus = +$ 时，此矩阵乘法即为原本的矩阵乘法 。

若 $\otimes$ 满足交换、结合律，$\otimes$ 对 $\oplus$ 有分配律，则此矩阵运算满足结合律。

此时有 $((AB)C)_{i,j} = (A(BC))_{i,j} = \bigoplus_{k=1}^{p} \bigoplus_{t=1}^{q} (A_{i,t} \otimes B_{t,k} \otimes C_{k,j})$.

广义矩阵乘法同样可以快速幂，比如P5678 。

更多的时候，广义矩阵乘法用于维护 dp 或其他问题。

高斯消元

多数时候用的是高斯-约旦消元，即最终化成对角矩阵而不是梯形矩阵。（其实就是大伙平时解方程的办法。）

枚举主元 x_i --> 找到主元系数绝对值最大的一行 V_j --> 每行 V_k 减去 $\frac{V_{k,i}}{V_{j,i}} \cdot V_j$

复杂度：$O(n^3)$

板子题。

```
bool solve(){
    for(int i=1;i<=n;++i){
      int pos=i;
      for(int j=i+1;j<=n;++j)
        if(fabs(a[j][i])>fabs(a[pos][i]))
          pos=j;
      for(int k=1;k<=n+1;++k)
        swap(a[i][k],a[pos][k]);
      if(fabs(a[i][i])<esp)return 0;
      for(int j=1;j<=n;++j){
          if(j==i)continue;
          double rat=a[j][i]/a[i][i];
        for(int k=i;k<=n+1;++k)
              a[j][k]-=rat*a[i][k];
      }
    }
    return 1;
}
```

应用：

- 解异或方程

  题目：给定一个 $0,1$ 序列，其中某些位置相互关联，每次选定一个位置，将其以及与之关联的位置取反，求其变为指定序列的方案数。

  高斯消元解异或方程步骤不变（甚至因为没有浮点误差简单了），并且可以使用 bitset 优化从而使复杂度变为 $O(\frac{n^3}{\omega})$

  但是这题求的是方案数。

  考虑高斯消元后得到的对角矩阵，若第 k 行全为 0，则最终答案与 a_k 无关，从而 a_k 可以任取。

  所以最终有 x 行全零，则有 2^x 种方案。

- 求行列式（选）

  $$\det(A)=\sum_{\sigma\in S_n}(-1)^{\tau(\sigma)}\prod_{i=1}^{n}a_{i,\sigma_i}$$

  矩阵行列式相当于其所有列向量围成几何体的大小。

  有以下性质：

  - 行列对换（即矩阵转置），行列式不变。
  - 交换两行/列，行列式取反。
  - 行/列相减，行列式不变。
  - 矩阵乘以常数 k，行列式变为原来 k 倍。
  - 对角矩阵行列式为对角线上数的乘积。

  由最后一条，我们可以用高斯消元将矩阵化为对角矩阵求行列式，并根据中间交换的次数判断符号。

VI.组合数学

这部分比较考验数学功力，所以只讲一点基础计算。

- 组合数的计算
  - 单个计算$O(\log n)$：$C_n^m=\frac{n!}{m!(n-m)!}$，需要计算逆元。
  - 计算 $0\le i\le m,\ 0\le j\le n$ 的所有 C_j^i

$O(nm)$ 递推：$C_j^i = C_{j-1}^{i-1} + C_{j-1}^i$.

- 计算 $0 \le i \le m$ 的所有 C_n^i

  $O(m)$ 递推：$C_n^i = C_n^{i-1} \cdot \frac{n-i+1}{i}$ ，需要 $O(m)$ 预处理逆元。

- 抽屉（鸽巢）原理

  将 n 个物体划分为 k 组，至少存在一组有 $\lceil \frac{n}{k} \rceil$ 个物体。

  会怎么考呢?

  NOI2021D2T1

  题目描述：字典 S 由 n 个 256 位的 01 串构成。有 m 次询问，每次给出一个 256 位的 01 串 T ，问 S 中是否存在一个 01 串 S_i 与 T 有至多 15 位不同。**保证字典随机生成**

  注意到不同的位数至多 15 位，那么把 256 位的串 16 等分，则 S_i 与 T 至少有一份完全相同。

  用类似 hash 的方法，设立 2^{16} 个桶，将字典中的每个串 16 等分后，每份转换成对应的值，并将串存入相应下标的桶中。

  每个桶中期望元素个数是 $O(\frac{n}{2^{12}})$ ，每次询问即在 T 相应的 16 个桶中查找，复杂度 $O(\frac{nm}{2^8})$ 。

- 容斥原理

$$|\bigcup_{i=1}^{n} S_i| = \sum_{m=1}^{n} (-1)^{m-1} \sum_{a_i < a_{i+1}} |\bigcap_{i=1}^{m} S_{a_i}|$$

- 错位排列

  n 个数的排列满足 $a_i \neq i$ 的情况数。

  $D_n = (n-1) \cdot (D_{n-2} + D_{n-1})$

- pop_count 预处理

  NOI 出题人不讲武德在标算中使用了 $O(1)$ 的 `__builtin_popcountll()` ，但双下划线开头函数咱们不能用。所以很多时候我们需要预处理一定范围内所有数的 pop_count 。有一个显然的公式：

  $popcnt[i] = popcnt[i >> 1] + (i \& 1)$

VII.博弈论

公平组合游戏

公平组合游戏满足一下要素：

- 有两个玩家参与游戏，且均知道场上的所有信息。
- 对同一局面两玩家可以进行的操作相同。
- 不能多次出现同一局面，游戏在有限步中以非平局结束。

  指出下面游戏是否为公平组合游戏：五子棋、围棋、象棋、取数游戏（两人依次从序列两端取走数字，比较总和）。

  博弈图与状态：

- 必胜状态：先手必胜的局面。
- 必败状态：后手必胜的局面。
- 后继状态：玩家一次操作后可使当前局面变化成的状态。

定理：

- 没有后继状态的状态为必败状态。
- 一个状态是必胜状态当且仅当存在至少一个必败状态为它的后继状态。
- 一个状态是必败状态当且仅当它的所有后继状态均为必胜状态。

应用：

给出一个数 n，两人依次选择其一个因数 $x,(x\neq 1或n)$，使 n 变为 $n-x$，不能操作者败，求先手必胜或必败。

```
const int WIN=1,LOSE=2;
bool dfs(int n){
    if(st[n])return st[n]==WIN;
    for(int i=2;i<n;++i)
      if(n%i==0)
        if(!dfs(n-i))
            return st[n]=WIN;
    st[n]=LOSE;
    return 0;
}
```

```
 1: 0 , 2: 0 , 3: 0 , 4: 1 , 5: 0 , 6: 1 , 7: 0 , 8: 0 , 9: 0 ,10: 1 ,
11: 0 ,12: 1 ,13: 0 ,14: 1 ,15: 0 ,16: 1 ,17: 0 ,18: 1 ,19: 0 ,20: 1 ,
21: 0 ,22: 1 ,23: 0 ,24: 1 ,25: 0 ,26: 1 ,27: 0 ,28: 1 ,29: 0 ,30: 1 ,
31: 0 ,32: 0 .
```

Nim博弈

有 n 堆石子，每堆有 k_i 个，两人依次从任一堆中取走任意个，先取完者胜。求先手必胜还是必败。

定义 Nim 和为 $S = \bigoplus_{1 \le i \le n} k_i$ ，其中 $\oplus$ 表示异或。

则 $S = 0$ 时先手必败，否则必胜。

证明：只需证明其满足三条定理即可。

即证：

- 无后继状态的状态为必败状态，应有 $S = 0$ 。
- 一个状态是必胜状态当且仅当存在至少一个必败状态为它的后继状态：$S \neq 0$ 时存在一种操作使下个局面 $S = 0$ 。
- 一个状态是必败状态当且仅当它的所有后继状态均为必胜状态：$S = 0$ 的下个局面一定有 $S \neq 0$ 。

VIII.知识点交叉

其他知识解决数学问题，或数学知识解决其他问题。

矩阵树定理

已经不是提高组考点了，只提一下基本概念。

定义无向无权图的基尔霍夫矩阵 $K = D - A$ ，D 为度数矩阵（$D_{i,i} = degree(i)$，其余为 0），A 为邻接矩阵。

将K去除最后一行和最后一列得该矩阵的 $n-1$ 阶主子式 K' ，则该图生成树个数为 $\det(K')$.

矩阵加速图上问题

求从一个点出发，**恰**经过 k 条边，两点间的最短路/路径数目。

- 求最短路：P2886

考虑在floyd 的三重循环中，如果将枚举 k 的循环放在最内层，得到的结果是什么？

记 $A_{i,j}=\min\limits_{k}\{dis_{i,k}+dis_{k,j}\}$，显然 $A_{i,j}$ 表示 i,j 之间经过一条边的最短路。

定义广义矩阵乘法 $C=A*B$ 使 $C_{i,j}=\min\limits_{k}\{A_{i,k}+B_{k,j}\}$

$+$ 满足结合律，$+$ 对 $\min$ 有分配律 （即 $a+\min(b,c)=min(a+b,a+c)$ ），即该乘法满足结合律。

因为这部分没啥难度，所以也会和小规模 DEG 图 dp 结合起来考（尤其是AC自动机dp），不太常考（因为它套的dp一般挺难），也了解一下。

复杂度 $O(n^3\log k)$

- 求路径数：P5789 可乐

这时我们不关心每条边的长度（因为都是1），只求两点间的路径数。所以邻接矩阵 $A_{i,j}$ 的含义就变为 i,j 之间的路径条数。设$S(k,i,j)$ 表示经过 k 条边后 i,j 两点间路径数，则 $S(k+1,i,j)=\sum\limits_{x}S(k,i,x)*A_{x,j}$.

不难发现这是一个经典的矩阵乘法，即 $S(k)=A^k$ 。 复杂度同上。

这部分有很多寄巧，比较超纲，看我的博客吧。