

---

# lr580's 算法模板

---



Version 1.2.3, last built at 2025/3/5

## lr580's 算法模板

### 数学

#### 公式

常用符号

组合数学

排列组合

卡特兰数

斯特林数

生成函数

球盒问题

其他

分拆数

贝尔数

欧拉数

复杂度

数列

高等数学

线性代数

概率论

其他

### 数论

基础性质

素数

素性测试

埃氏筛

欧拉筛

pollard-rho算法

Meissel-Lehmer算法

拓展欧几里得定理

取模公式

费马小定理

卢卡斯定理

欧拉定理

线性快速幂

威尔逊定理

欧拉函数

中国剩余定理

BSGS

### 计算几何

向量

线段

点到直线距离

线段相交判定

直线交点

点在直线投影

多边形

极角排序

凸包

Graham

Andrew

直径

点与凸包相交

两凸包相交

扫描线

杂项

Pick定理  
平面最近点对  
随机增量法  
立体几何  
博弈论  
多项式  
    FFT  
    NTT  
    分治 FFT  
    FWT  
    拉格朗日插值法  
杂项  
    矩阵加速  
    高斯消元  
    康托展开  
    自适应辛普森法  
数据结构  
    ST表  
    线段树  
        常见应用  
            区间加法  
            区间加乘  
            区间查重  
            区间最值  
            历史最值  
        动态开点  
        zkw线段树  
            单点修改  
            区间修改  
        树上二分  
        猫树  
        主席树  
            可持久化数组  
            静态区间第k小  
    树状数组  
        静态区间最值  
        动态整体第k小  
        动态区间第k小  
        二维树状数组  
    平衡树  
        FHQ-Treap  
        AVL  
        Splay  
    K-D Tree  
    堆  
        可删堆  
        笛卡尔树  
        可并堆  
    珂朵莉树  
    并查集  
        种类并查集  
        加权并查集  
        可撤销并查集  
        可持久化并查集  
图论  
    树

重心  
直径  
最近公共祖先  
树上k级祖先  
LCA应用  
    前缀和差分  
    两点距离  
    路径相交  
    其他  
树链剖分  
启发式合并  
虚树  
点分治  
Prufer 序列  
二叉树遍历  
    先中求后  
    中后求先  
    先后求中  
杂项  
    随机游走  
    最小距离和

基本概念  
最短路  
    floyd  
    Bellman-Ford  
    SPFA  
    Dijkstra  
    Johnson  
    差分约束  
拓扑排序  
最小生成树  
    kruskal  
    prim  
    Borůvka  
    严格次小生成树  
连通性  
    连通分量  
        强连通分量  
        割点  
        桥  
        点双连通分量  
        边双连通分量  
    圆方树  
    2-SAT  
匹配问题  
    二分图  
    最大匹配  
    完美匹配  
    最大权匹配  
    最小边覆盖  
网络流  
    最大流  
        EK 算法  
        Dinic算法  
    最小费用最大流  
    最小割

杂项  
欧拉图  
哈密顿图  
竞赛图  
动态规划  
模板  
    背包问题  
    字符DP  
    树上DP  
    区间DP  
    状压DP  
    数位DP  
杂项  
    最长公共子序列  
    编辑距离  
    约瑟夫问题  
字符串  
    字符串哈希  
    KMP  
    manacher  
    字典树  
    AC自动机  
    后缀数组  
        基础  
        LCP  
        SA-IS  
    后缀自动机  
    FFT字符串匹配  
    子序列自动机  
    最小表示法  
    Lyndon分解  
杂项  
    排序  
        计数排序  
        基数排序  
        归并求逆序对  
    二分  
        最长单调序列  
        最长公共排列  
        二分答案  
            最小中位数子矩阵  
            最大子串平均值  
            其他  
            整体二分  
            wqs二分  
    前缀和/差分  
    搜索  
        IDA\*  
        双向搜索  
        折半搜索  
    随机化  
    莫队  
        普通  
        带修  
        树上  
        回滚

高精度  
C++  
FFT 乘法  
Java 高精度  
Python 高精度

悬线法

程序语法

常规运算

位运算

指针

I/O

其他

STL

函数

xx\_bound

字符串

正则表达式

随机数

杂项

数据结构

string

vector

set

map

bitset

其他

pb\_ds

卡常

快读/写

其他

# Ir580's 算法模板

## 数学

### 公式

#### 常用符号

1.  $x|y$  表示  $x$  整除  $y$ ，例如  $2|8$ ，
2.  $x \perp y$  表示  $x, y$  互质
3.  $\gcd(x, y) = (x, y)$  最大公约数
4.  $\text{lcm}(x, y) = [x, y]$  最小公倍数
5.  $\lfloor x \rfloor$  向下取整
6.  $\lceil x \rceil$  向上取整
7.  $C_n^m = C(n, m) = \binom{n}{m} = \frac{n!}{m!(n-m)!}$ ，若  $n < m$  或  $m < 0$  得 0,  $C(n, 0) = 1$
8.  $A_n^m = P_n^m = P(n, m) = \frac{n!}{(n-m)!} = \prod_{i=n-m+1}^n i$ ，若  $n < m$  得 0

## 组合数学

### 排列组合

元素依次排成一个圆圈的排列称作环排列， $S$  的  $r$  环排列数等于  $\frac{P(n, r)}{r}$

组合数性质：设正整数  $n, r$ ，有

- $C(n, r) = \frac{n}{r} C(n-1, r-1)$
- $C(n, r) = \frac{n-r+1}{r} C(n, r-1)$
- $C(n, r) = C(n, n-r)$
- $C(n, r) = C(n-1, r-1) + C(n-1, r)$  (Pascal公式/杨辉三角)

二项式定理  $n$  是正整数，对一切  $x, y$ ，有：

$$(x+y)^n = \sum_{k=0}^n \binom{n}{k} x^k y^{n-k}$$

多重组合数(即多重集的全排列)公式：

$$\binom{n}{n_1, n_2, \dots, n_r} = \frac{n!}{n_1! n_2! \dots n_r!}$$

二项式定理的多项式形式：

$$(x_1 + \cdots + x_t)^n = \sum_{\substack{\text{满足 } n_1 + \cdots + n_t = n \text{ 的非负整数解}}} \binom{n}{n_1, n_2, \dots, n_t} x_1^{n_1} x_2^{n_2} \cdots x_t^{n_t}$$

从多重集  $S = \{n_1 \cdot a_1, \dots, n_k \cdot a_k\}$  选  $r$  个元素的多重集组合数, 为  $x_1 + \cdots + x_k = r$  的非负整数解数目, 插板法知为  $\binom{r+k-1}{k-1}$  (条件:  $r < \min n_i$ ), 任意条件为:

$$\sum_{p=0}^k (-1)^p \sum_A \binom{k+r-1 - \sum_A n_{A_i} - p}{k-1}$$

$A$  是枚举子集, 满足  $|A| = p, A_i < A_{i+1}$

$n$  个自然数选  $k$  个, 任两个数都不相邻:  $\binom{n-k+1}{k}$

$i$  数错位排列数公式:  $d[0] = d[1] = 0, d[2] = 1$

$$d[n] = (n-1)(d[n-1] + d[n-2]) = n!(1 - \frac{1}{1!} + -\frac{1}{2!} + \cdots + (-1)^n \frac{1}{n!})$$

$d_n = (n-1)(d_{n-1} + d_{n-2}) = nd_{n-1} + (-1)^n$ 。且增长速度  $\lim_{n \rightarrow \infty} \frac{d_n}{n} = \frac{1}{e}$

$n$  个数有  $m$  个乱的情况数:  $C_n^m d[m]$

$n$  人围圈排列数  $Q_n^n = (n-1)!$ , 部分圆排列公式  $Q_n^r = \frac{A_n^r}{r}$

排序不等式: 逆序和  $\leq$  乱序和  $\leq$  顺序和。给定升序数组  $a, b$ , 设  $c$  是  $b$  的排列, 则求和  $\sum_{i=1}^n a_i b_{n-i+1} \leq \sum_{i=1}^n a_i c_i \leq \sum_{i=1}^n a_i b_i$

推论:

- $\sum_{k=0}^n \binom{n}{k} = 2^n, n \in N$
- $\sum_{k=0}^n (-1)^k \binom{n}{k} = 0, n \in N$
- $\sum_{l=0}^n \binom{l}{k} = \binom{n+1}{k+1}, n, k \in N$
- $\sum_{i=0}^m \binom{n}{i} \binom{m}{m-i} = \binom{m+n}{m}, n \geq m$
- $\sum_{i=0}^n \binom{n}{i}^2 = \binom{2n}{n}$
- $\binom{n}{r} \binom{r}{k} = \binom{n}{k} \binom{n-k}{r-k}, n \geq r \geq k, n, r, k \in N$
- $\sum_{k=0}^r \binom{m}{k} \binom{n}{r-k} = \binom{n+m}{r}, n, m, r \in N, r \leq \min(m, n)$
- $\sum_{k=0}^n \binom{m}{k} \binom{n}{k} = \binom{m+n}{m}, m, n \in N$

- $\sum_{i=0}^n i \binom{n}{i} = n2^{n-1}$
- $\sum_{i=0}^n \binom{n-i}{i} = fib_{n+1}$

某些时候可以两边求导再求

枚举所有组合可以使用 Gospers Hack 技巧：

```

1  unsigned int nextSubset(unsigned int subset) {
2      unsigned int u = subset & ~subset;
3      unsigned int v = subset + u;
4      return v | (((v ^ subset) / u) >> 2);
5  }
6  void gospersHack(int k, int n) {
7      unsigned int subset = (1 << k) - 1;
8      unsigned int limit = (1 << n);
9      while (subset < limit) {
10          for (int i = 0; i < n; ++i) {
11              if (subset & (1 << i)) {
12                  std::cout << i + 1 << " ";
13              } // Process the subset here
14          } // For example, printing it out
15          std::cout << std::endl;
16          subset = nextSubset(subset);
17      }
18  }

```

### 容斥原理：

$U$  中元素有  $n$  种不同属性  $p_i$ ，拥有属性  $p_i$  的元素构成集合  $S_i$

$$\begin{aligned}
 |\bigcup_{i=1}^n S_i| &= \sum_{m=1}^n (-1)^{m-1} \sum_{a_i < a_{i+1}} |\bigcap_{i=1}^m S_{a_i}| \\
 |\bigcap_{i=1}^n S_i| &= |U| - |\bigcup_{i=1}^n \overline{S_i}|
 \end{aligned}$$

### 抽屉原理：

把  $mn + 1$  个物体放到  $n$  个抽屉里，至少有一个抽屉至少有  $m + 1$  个物体

把  $mn - 1$  个物体放到  $n$  个抽屉里，至少有一个抽屉至少有  $m - 1$  个物体

将  $n$  物放到  $k$  抽屉，至少一个有不少于  $\lceil \frac{n}{k} \rceil$  个物体

应用：选  $n$  个正整数的若干个数， $n \geq m$  时选出的数必然存在可以被  $m$  整除

## 卡特兰数

$$H[0] = 1$$
$$H[n+1] = \sum_{i=0}^n H[i] \cdot H[n-i-1]$$

从 0 到 5 分别是 1, 1, 2, 5, 14, 42 ,  $H[35]$  为 19 位长度,  $H[36]$  爆 long long

$$H[n] = H[n-1] \times \frac{4n-2}{n+1}$$
$$H[n] = \frac{C(2n, n)}{n+1}$$
$$H[n] = C(2n, n) - C(2n, n-1)$$

常见应用:

- 矩阵连乘:  $P=a_1 \times a_2 \times a_3 \times \dots \times a_n$ , 依据乘法结合律, 不改变其顺序, 只用括号表示成对的乘积, 试问有几种括号化的方案。 $h(n)$  种
- 一个栈(无穷大)的进栈序列为 1, 2, 3, ..., n, 有多少个不同的出栈序列
- 在一个凸多边形中, 通过若干条互不相交的对角线, 把这个多边形划分成若干个三角形方案数如  $n=6, f(6)=14$
- 给定  $n$  个节点, 能构成多少不同的二叉搜索树
- 给定  $n$  对括号, 求括号正确配对的字符串数
- $2n$  个黑白, 使得前缀黑色小于白色的方案数
- 圆上  $2n$  点用  $n$  条不相交线段连接的方案数
- $n$  人有 50 元,  $n$  人有 100 元买 50 元物品, 排队方案数使得  $2n$  人都可找零
- $2n$  人交 5 元,  $n$  人只有 5 元一张, 另  $n$  人 10 元一张, 问多少种方案只要有 10 元人交钱就有 5 元找零
- $n$  个  $+1$  和  $n$  个  $-1$  构成  $2n$  项  $a_1, \dots, a_{2n}$  满足前缀和  $a_1 + \dots + a_k \geq 0 (k = 1, 2, \dots, 2n)$  的数列方案数
- 从  $(0, 0)$  到  $(m, n)$  的非降路径数  $\binom{n+m}{m}$
- 从  $(0, 0)$  到  $(n, n)$  除端点不接触  $y = x$  非降路径数  $2 \binom{2n-2}{n-1} - 2 \binom{2n-2}{n}$
- 从  $(0, 0)$  到  $(n, n)$  除端点不穿过  $y = x$  非降路径数  $\frac{2}{n+1} \binom{2n}{n}$

## 斯特林数

**第一类:** (无符号) 将  $n$  个不同元素构成  $m$  个非空圆排列的数目 ( $[A, B, C] \neq [C, B, A]$  为不同圆排列(轮换))。 (斯特林轮换数) 无符号为  $s_u(n, m)$ ,  $s(n, m)$  或  $\begin{bmatrix} n \\ m \end{bmatrix}$ , 有符号为  $s_s(n, m)$ , 分别表现升阶函数和降阶函数的各项系数。

$$\begin{aligned}
x^{n\downarrow} &= x(x-1)(x-2)\dots(x-n+1) = \sum_{k=0}^n s_s(n, k)x^k \\
x^{n\uparrow} &= x(x+1)(x+2)\dots(x+n-1) = \sum_{k=0}^n s_u(n, k)x^k \\
s_s(n, u) &= (-1)^{n+m} s_u(n, m) \\
s_u(n+1, m) &= s_u(n, m-1) + n \cdot s_u(n, m) \\
s_s(n+1, m) &= s_s(n, m-1) - n \cdot s_s(n, m)
\end{aligned}$$

例题：有  $n$  个仓库，每个仓库有两把钥匙，共  $2n$  把钥匙。同时又有  $n$  位官员。问如何放置钥匙使得所有官员都能够打开所有仓库？（只考虑钥匙怎么放到仓库中，而不考虑官员拿哪把钥匙。）那如果官员分成  $m$  个不同的部，部中的官员数量和管理的仓库数量一致。那么有多少方案使得，同部的所有官员可以打开所有本部管理的仓库，而无法打开其他部管理的仓库？（同样只考虑钥匙的放置。）

第一问就是打开将钥匙放入仓库构成一个环：1号仓库放2号钥匙，2号仓库放3号钥匙…… $n$ 号仓库放1号钥匙。这种情况相当于钥匙和仓库编号构成一个圆排列方案数是  $(n-1)!$  种。

而第二问就对应的将  $n$  个元素分成  $m$  个圆排列，方案数就是第一类无符号 Stirling 数  $s_u(n, m)$ 。如要要考虑官员的情况，只需再乘上  $n!$  即可。

p5409/p5408

**第二类：**(斯特林子集数) 把  $n$  个互异元素分成  $m$  个互不区分非空集合，记为  $S(n, m)$  或  $\left\{ \begin{matrix} n \\ m \end{matrix} \right\}$

$$\begin{aligned}
S(n, m) &= \frac{1}{m!} \sum_{k=0}^m (-1)^k \binom{m}{k} (m-k)^n = \sum_{i=0}^m \frac{(-1)^{m-i} i^n}{i! (m-i)!} \\
S(n+1, m) &= S(n, m-1) + m \cdot S(n, m)
\end{aligned}$$

递推上与第一类的区别在于  $m$  还是  $n$  作系数和正负问题。

求斯特林数，可以套用公式二，使用  $dp$ 。注意到初始状态是  $S(n, 1) = S(n, n) = 1$ 。 $S(n, 0) = 0^n$ 。求同一行 ( $n$  相同) 斯特林数可以用通项卷积，复杂度  $O(n \log n)$

(p5396/p5395)

## 生成函数

可以解决组合问题(排列则乘一个阶乘变成组合)

$$\begin{aligned}
e^x &= \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \dots, x \in \mathbf{R} \\
e^{-x} &= 1 - \frac{x}{1!} + \frac{x^2}{2!} - \dots \\
\frac{e^x + e^{-x}}{2} &= \left(1 + \frac{x^2}{2!} + \frac{x^4}{4!} \dots\right) \\
\frac{1}{1-x^a} &= 1 + x^a + x^{2a} + \dots
\end{aligned}$$

## 球盒问题

球区别	盒区别	空盒	模型	方案计数
有	有	有	选取	$m^n$
有	有	无	放球	$m! \binom{n}{m}$
有	无	有	放球	$\sum_{k=1}^m \binom{n}{k}$
有	无	无	放球	$\binom{n}{m}$
无	有	有	不定方程	$C_{n+m-1}^n$
无	有	无	不定方程	$C_{n-1}^{m-1}$
无	无	有	正整数拆分	$G(x) = \frac{1}{\prod_{i=1}^m (1-x^i)}, x^n$ 系数
无	无	无	正整数拆分	$G(x) = \frac{x^m}{\prod_{i=1}^m (1-x^i)}, x^n$ 系数

不定方程  $\sum_{i=1}^m x_i = n$  非负整数解个数是  $C_{n+m-1}^{m-1} = C_{n+m-1}^n$

正整数  $n$  拆分为  $m$  个正整数即  $a_1x_1 + a_2x_2 + \dots + a_mx_m = n$ ，若不允许重复，生成函数为  $\prod_{i=1}^m (1+x^{a_i})$ ，若允许重复，为  $\prod_{i=1}^m \frac{1}{1-x^i}$ 。均取系数  $x^n$ 。不允许就先每个至少取 1 所以乘  $x^m$

例洛谷P2386-  $m$  个相同苹果放  $n$  相同盘子,可空盘,求方案数

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define sc(x) scanf("%lld", &x)
4 typedef long long ll;
5 #define mn 30
6 ll n, m, t, dp[mn][mn]; //考虑前i个生成函数, x^j系数
7 signed main()
8 {
9     dp[0][0] = 1;
10    for (ll i = 1; i < mn; ++i) //暴力多项式乘法
11        for (ll j = 0; j < mn; j += i) // 1+x^i+x^{2i}+...
12            for (ll k = 0; k + j < mn; ++k) // 枚举dp[i-1][k]
13                dp[i][k + j] += dp[i - 1][k];
14    for (sc(t); t--;) {
15        sc(m), sc(n); // m无区别球放n无区别盒,可空盒
16        printf("%lld\n", dp[n][m]);
17    } //等价于前n个生成函数,系数为x^m
18    return 0;
19 }
```

一种非正整数划分的 DP 解法(时空  $O(nm)$ )

```

1 #include <bits/stdc++.h>
```

```

2  using namespace std;
3  #define sc(x) scanf("%lld", &x)
4  typedef long long ll;
5  #define mn 30
6  ll n, m, t, dp[mn][mn]; // m球n盒方案数
7  signed main()
8  {
9      for (ll i = 0; i < mn; ++i) //无球或单盒为1
10         dp[0][i] = dp[i][1] = 1;
11      for (ll i = 1; i < mn; ++i)
12          for (ll j = 1; j < mn; ++j)
13              if (j > i) //盒比球多, (?, 0)=(?)
14                  dp[i][j] = dp[i][i];
15              else //放满时每盒拿走一个;放不满时拿走空盒一个
16                  dp[i][j] = dp[i - j][j] + dp[i][j - 1];
17      for (sc(t); t--;}
18      sc(m), sc(n), printf("%lld\n", dp[m][n]);
19      return 0;
20 }

```

P5824- $n$  球  $m$  盒 ( $1 \leq n, m \leq 2 \times 10^5$ ), 求放球方案对 998244343 取模。

1. 球互不同, 盒互不同。  $m^n$
2. 球互不同, 盒互不同, 每盒最多装一个球。  $A_m^n$
3. 球互不同, 盒互不同, 每盒最少装一个球。 容斥枚举空盒数  $\sum_{i=0}^m (-1)^i C_m^i (m-i)^n$
4. 球互不同, 盒相同。 二项式反演求第二类斯特林数  $\sum_{i=1}^m S(n, i)$
5. 球互不同, 盒相同, 每盒最多装一个球。  $[n \leq m]$
6. 球互不同, 盒相同, 每盒最少装一个球。  $S(n, m)$
7. 球相同, 盒互不同。 隔板法  $C_{n+m-1}^{m-1}$
8. 球相同, 盒互不同, 每盒最多装一个球。  $C_m^n$
9. 球相同, 盒互不同, 每盒最少装一个球。 隔板法  $C_{n-1}^{m-1}$
10. 球相同, 盒相同。 分拆数  $p(n, m)$ , 使用生成函数多项式求  $\exp$  后求逆
11. 球相同, 盒相同, 每盒最多装一个球。  $[n \leq m]$
12. 球相同, 盒相同, 每盒最少装一个球。  $p(n - m, m)$

含:

- 排列组合, 快速幂, 欧拉筛求线性  $i^k$
- NTT, 多项式的求逆、求导、积分、指数、对数
- 求整行第二类斯特林数、求同  $k$  部的整列分拆数

的综合模板题:

```

1 // #pragma GCC optimize ("unroll-loops")
2 #include<bits/stdc++.h>
3 #define N 524293
4 #define p 998244353
5 using ll = long long;
6 #define add(x,y) (x+y>=p?x+y-p:x+y)
7 #define dec(x,y) (x<y?x-y+p:x-y)

```

```

8  using namespace std;
9
10 inline int power(int a,int t){
11     int res = 1;
12     while(t){
13         if(t&1) res = (ll)res*a%p;
14         a = (ll)a*a%p;
15         t >>= 1;
16     }
17     return res;
18 }
19
20 struct poly{
21     int a[N];
22     int t;
23     inline int operator [] (const int& x) const{ return a[x]; }
24     inline int& operator [] (const int& x){ return a[x]; }
25 };
26
27 int rev[N],rt[N],inv[N],fac[N],ifac[N],pw[N],pr[N>>1];
28 int siz;
29
30 void init(int n){
31     int w,lim = 1;
32     while(lim<=n) lim <= 1,++siz;
33     for(int i=1;i!=lim;++i) rev[i] = (rev[i>>1]>>1)|((i&1)<<(siz-1));
34     inv[1] = rt[lim>>1] = 1;
35     w = power(3,(p-1)>>siz);
36     for(int i=(lim>>1)+1;i!=lim;++i) rt[i] = (ll)rt[i-1]*w%p;
37     for(int i=(lim>>1)-1;i-->0) rt[i] = rt[i<<1];
38     for(int i=2;i<=lim;++i) inv[i] = (ll)(p-p/i)*inv[p%i]%p;
39 }
40
41 inline int getlen(int n){
42     return 1<<(32-__builtin_clz(n));
43 }
44
45 inline void NTT(poly &f,int type,int lim){
46     if(type==1) reverse(f.a+1,f.a+lim);
47     static unsigned long long a[N];
48     int x,shift = siz-__builtin_ctz(lim);
49     for(int i=0;i!=lim;++i) a[rev[i]>>shift] = f[i];
50     for(int mid=1;mid!=lim;mid<<=1)
51         for(int j=0;j!=lim;j+=(mid<<1))
52             for(int k=0;k!=mid;++k){
53                 x = a[j|k|mid]*rt[mid|k]%p;
54                 a[j|k|mid] = a[j|k]-x+p;
55                 a[j|k] += x;
56             }
57             for(int i=0;i!=lim;++i) f[i] = a[i]%p;
58             if(type==1) return;
59             x = inv[lim];
60             for(int i=0;i!=lim;++i) f[i] = (ll)f[i]*x%p;
61 }
62
63 poly inverse(poly f){

```

```

64     poly g,h,q;
65     memset(g.a,0,sizeof(g.a));
66     int lim = 1,n = f.t,top = 0;
67     int s[30];
68     while(n){
69         s[++top] = n;
70         n >= 1;
71     }
72     g[0] = power(f[0],p-2);
73     while(top--){
74         g.t = n = s[top+1];
75         while(lim<=(n<<1)) lim <<= 1;
76         memcpy(h.a,f.a,(n+1)<<2);
77         memcpy(q.a,g.a,(n+1)<<2);
78         memset(h.a+n+1,0,(lim-n)<<2);
79         NTT(g,1,lim),NTT(h,1,lim);
80         for(int i=0;i!=lim;++i) g[i] = (ll)g[i]*g[i]%p*h[i]%p;
81         NTT(g,-1,lim);
82         for(int i=0;i<=n;++i) g[i] = dec(add(q[i],q[i]),g[i]);
83         memset(g.a+n+1,0,(lim-n)<<2);
84     }
85     return g;
86 }
87
88 inline poly deriv(poly f){ //多项式求导
89     for(int i=0;i!=f.t;++i) f[i] = (ll)f[i+1]*(i+1)%p;
90     f[f.t] = 0;
91     return f;
92 }
93
94 inline poly integ(poly f){ //多项式积分
95     for(int i=f.t;i-->0) f[i] = (ll)f[i-1]*inv[i]%p;
96     f[0] = 0;
97     return f;
98 }
99
100 inline poly log(poly f){
101     poly g = inverse(f);
102     f = deriv(f);
103     int lim = getlen(f.t<<1);
104     NTT(f,1,lim),NTT(g,1,lim);
105     for(int i=0;i!=lim;++i) f[i] = (ll)f[i]*g[i]%p;
106     NTT(f,-1,lim);
107     memset(f.a+f.t+1,0,(lim-f.t)<<2);
108     return integ(f);
109 }
110
111 poly exp(poly f){
112     poly g,h;
113     memset(g.a,0,sizeof(g.a));
114     int n = f.t,lim = 1,top = 0;
115     int s[30];
116     while(n){
117         s[++top] = n;
118         n >= 1;
119     }

```

```

120     g[0] = 1;
121     while(top--){
122         g.t = n = s[top+1];
123         while(lim<=(n<<1)) lim <<= 1;
124         memcpy(h.a,g.a,(n+1)<<2);
125         memset(h.a+n+1,0,(lim-n)<<2);
126         g = log(g);
127         for(int i=0;i<=n;++i) g[i] = dec(f[i],g[i]);
128         g[0] = add(g[0],1);
129         NTT(g,1,lim),NTT(h,1,lim);
130         for(int i=0;i!=lim;++i) g[i] = (ll)g[i]*h[i]%p;
131         NTT(g,-1,lim);
132         memset(g.a+n+1,0,(lim-n)<<2);
133     }
134     return g;
135 }
136
137 inline int c(int n,int m){
138     if(n<m) return 0;
139     return (ll)fac[n]*ifac[m]%p*ifac[n-m]%p;
140 }
141
142 inline int A(int n,int m){
143     if(n<m) return 0;
144     return (ll)fac[n]*ifac[n-m]%p;
145 }
146
147 void sieve(int lim,int k){
148     int cnt = 0;
149     pw[1] = 1; //利用积性函数求 pw[i]=i^k
150     for(int i=2;i<=lim;++i){
151         if(!pw[i]){
152             pr[++cnt] = i; //欧拉筛
153             pw[i] = power(i,k);
154         }
155         for(int j=1;j<=cnt&&i*pr[j]<=lim;++j){
156             pw[i*pr[j]] = (ll)pw[i]*pw[pr[j]]%p;
157             if(i%pr[j]==0) break;
158         }
159     }
160 }
161
162 int n,m,lim,sum;
163 poly F,G;
164
165 int main(){
166     scanf("%d%d",&n,&m);
167     lim = max(n,m);
168     sieve(lim,n);
169     printf("%d\n",pw[m]); // I1
170     init(lim<<1);
171     fac[0] = fac[1] = ifac[0] = ifac[1] = 1;
172     for(int i=2;i<=n+m;++i) ifac[i] = fac[i] = (ll)fac[i-1]*i%p;
173     ifac[n+m] = power(fac[n+m],p-2);
174     for(int i=n+m-1;i;--i) ifac[i] = (ll)ifac[i+1]*(i+1)%p;
175     printf("%d\n",A(m,n)); // II2

```

```

176     for(int i=0;i<=n;++i){
177         F[i] = (ll)pw[i]*ifac[i]%p;
178         G[i] = (i&1)?p-ifac[i]:ifac[i];
179     }
180     lim = getlen(n<<1);
181     NTT(F,1,lim),NTT(G,1,lim);
182     for(int i=0;i!=lim;++i) F[i] = (ll)F[i]*G[i]%p;
183     NTT(F,-1,lim);
184     memset(F.a+n+1,0,(lim-n+2)<<2);
185     printf("%lld\n", (ll)fac[m]*F[m]%p); // III3
186     for(int i=1;i<=m;++i) sum = add(sum,F[i]);
187     printf("%d\n",sum); // IV4
188     printf("%d\n",n<=m); // V5
189     printf("%d\n",F[m]); // VI6
190     printf("%d\n",c(n+m-1,n)); // VII7
191     printf("%d\n",c(m,n)); // VIII8
192     printf("%d\n",c(n-1,m-1)); // IX9
193     F.t = n;
194     memset(F.a,0,sizeof(F.a));
195     for(int i=1;i<=m;++i){
196         for(int j=i;j<=n;j+=i)
197             F[j] = add(F[j],inv[j/i]);
198     }
199     F = exp(F);
200     printf("%d\n",F[n]); // X10
201     printf("%d\n",n<=m); // XI11
202     printf("%d\n",n<m?0:F[n-m]); // XII12
203     return 0;
204 }

```

## 其他

### 分拆数

若  $n, r \in \mathbb{Z}^+$ , 正整数  $n_1, \dots, n_r$  满足  $n = \sum_{i=1}^r n_i$  则为  $n$  的一个  $r$  拆分。

分拆数  $p_n$  是自然数  $n$  的分拆方法数, 有

$p_0 = p_1 = 1, p_2 = 2, p_3 = 3, p_4 = 5, p_5 = 7, p_6 = 11, p_7 = 15, p_8 = 22$ .

将  $n$  拆分为恰有  $k$  部分的分拆称为  $k$  部分分拆, 记作  $p(n, k)$ 。初始值为  $p(0, 0) = 1$ 。当且仅当  $n \geq k$  有解。

$$p(n, k) = p(n - 1, k - 1) + p(n - k, k)$$

$k$  部分拆数的生成函数为: (p4389/p5824)

$$\sum_{n,k=0}^{\infty} p(n, k) x^n y^k = \frac{1}{1 - xy} \frac{1}{1 - x^2 y} \frac{1}{1 - x^3 y} \dots$$

互异分拆数  $pd_n$ , 自然数  $n$  各部分不相同的分拆数。如

$pd_0 = pd_1 = pd_2 = 1, pd_3 = pd_4 = 2, pd_5 = 3, pd_6 = 4, pd_7 = 5, pd_8 = 6$

$$pd(n, k) = pd(n - k, k - 1) + pd(n - k, k)$$

计算可以压缩第二维, 使得时间复杂度  $O(nk)$  空间复杂度  $O(n)$ 。

奇分拆数  $po_n$ ，拆分出各部分都是奇数的拆分方法数。有:  $po_n = pd_n$ 。

最左边是互异拆分的生成函数，最右边是奇分拆数的生成函数。

$$\prod_{i=1}^{\infty} (1 + x^i) = \frac{\prod_{i=1}^{\infty} (1 - x^{2i})}{\prod_{i=1}^{\infty} (1 - x^i)} = \prod_{i=1}^{\infty} \frac{1}{1 - x^{2i-1}}$$

互异偶分拆数  $pde_n$  与互异奇分拆数  $pdo_n$  满足  $pd_n = pde_n + pdo_n$ 。

$$\text{广义五边形数为 } a_{2i} = \frac{i(3i-1)}{2}, \quad a_{2i+1} = \frac{i(3i+1)}{2}.$$

递推式:  $p_n = p_{n-1} + p_{n-2} - p_{n-5} - p_{n-7} + \dots$  即下标是五边形数，且两正两负交替。

那么可以简化为:

$$p_i = \sum_{j=1}^{a_j \leq i} (-1)^{\lfloor \frac{j+1}{2} \rfloor + 1} p_{i-a_j}$$

可以计算到  $n = 5 \times 10^4$  的  $O(n^2)$ 。

## 贝尔数

**贝尔数:**  $n$  元素集合划分方法数(开首是  $B_0 = B_1 = 1, B_2 = 2, B_3 = 5, B_4 = 15$  )

$$B_{n+1} = \sum_{k=0}^n \binom{n}{k} B_k, \quad B_n = \sum_{k=0}^n S(n, k)$$

可以用贝尔三角形  $a_{1,1} = 1, a_{n,1} = a_{n-1,n-1}, a_{n,m} = a_{n,m-1} + a_{n-1,m-1}$  求  $B_i = a_{i,i}$

## 欧拉数

**欧拉数:**  $n$  元素排列有  $m$  个元素大于上一个元素  $A(n, m)$ ，如  $A(3, 1) = 4$ 。

$$A(n, m) = 0 (m \geq n), A(0, m) = 0, A(n, 0) = 1$$

$$\text{否则 } A(n, m) = (n - m)A(n - 1, m - 1) + (m + 1)A(n - 1, m)$$

## 复杂度

若递归算法存在分治递推式:

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn^k & n > 1 \end{cases}$$

其中  $a, b, c, k$  是常数， $T(n)$  是非递减函数，则:

$$T(n) = \begin{cases} O(n^{\log_b a}) & a > b^k \\ O(n^k \log_b n) & a = b^k \\ O(n^k) & a < b^k \end{cases}$$

注:  $b = 1$  不适用，这是因为  $T(n) = aT(n) + cn^k$  显然是死循环

常用复杂度对  $10^8$  :

$n!(n = 11), 2^n(n = 26), n^3(n = 464), n^2 \log n(n = 3000), n \log n(n = 4.5 \times 10^6)$

## 数列

**等差数列** A.P. Arithmetic progression

$$a_n = a_1 + (n - 1)d = a_m + (n - m)d$$

$$d = \frac{a_m - a_n}{m - n}$$

$$\begin{aligned} S_n &= \frac{n(a_1 + a_n)}{2} \\ &= \frac{n(a_m + a_{n-m+1})}{2} \\ &= na_1 + \frac{n(n-1)}{2}d \\ &= \frac{d}{2}n^2 + (a_1 - \frac{d}{2})n \\ &= na_{\frac{n+1}{2}} \end{aligned}$$

部分性质:

$$1. \quad m + n = p + q \Rightarrow a_m + a_n = a_p + a_q$$

$$2. \quad \frac{a_n}{b_n} = \frac{S_{2n-1}}{T_{2n-1}}$$

$$3. \quad a_n = \frac{S_{2n-1}}{2n-1}$$

**等比数列** G.P. Geometric progression

$$a_n = a_1 q^{n-1} = a_m q^{n-m}$$

$$q = \sqrt[n-m]{\frac{a_n}{a_m}}$$

$$S_n = \frac{a_1(1 - q^n)}{1 - q}, \quad (q \neq 1)$$

部分性质:

$$1. \quad m + n = p + q \Rightarrow a_m \times a_n = a_p \times a_q$$

$$2. \quad S_{n+m} = S_m + q^m S_n$$

求和公式:

$$\begin{aligned}\sum_{i=1}^n i &= \frac{n(n+1)}{2} \\ \sum_{i=1}^n i^2 &= \frac{n(n+1)(2n+1)}{2} \\ \sum_{i=1}^n i^3 &= \left(\frac{n(n+1)}{2}\right)^2\end{aligned}$$

$$\sum_{i=1}^n \frac{1}{i} \approx 0.57722 + \ln n$$

$$\sum_{i=1}^{\infty} \left(\frac{1}{k}\right)^i = \frac{1}{k-1}$$

求  $a_n$  :

- $a_{n+1} = pa_n + q$  构造新数列, 使得  $b_{n+1} = pb_n$
- $a_{n+1} = a_n + f(n)$  累加法
- $a_{n+1} = a_n \cdot f(n)$  累乘法
- $a_{n+1} = \frac{pa_n}{ka_n + m}$  倒数法, 令  $b_n = a_n^{-1}$
- $\bar{k}, \bar{kk}, \bar{kkk} \cdots a_n = \frac{k}{9}(10^n - 1)$
- $a, b, a, b, \cdots a_n = \frac{a + b + (-1)^n(b - a)}{2}$

求  $S_n$  :

- 裂项相消  $b_n = \frac{1}{a_n \cdot a_{n+k}}, a_n \in A.P.$
- 错位相减  $c_n = a_n \cdot b_n, a_n \in A.P., b_n \in G.P.$

$$(1 - q)S_n = a_1b_1 - a_n b_{n+1} + \frac{db_2}{1 - q}(1 - q^{n-1})$$

$$c_n = (an + b)q^{n-1}, \text{ 则 } S_n = (An + B)q^n - B, \text{ 其中 } A = \frac{a}{q-1}, B = \frac{b - A}{q-1}$$

- 分组求和
- 倒序相加

## 高等数学

### 含部分初等数学

$$\begin{aligned}\cot x &= \frac{1}{\tan x}, \sec x = \frac{1}{\cos x}, \csc x = \frac{1}{\sin x} \\ \tan \frac{x}{2} &= \frac{1 - \cos x}{\sin x} = \frac{\sin x}{1 + \cos x}\end{aligned}$$

$$\begin{aligned}\sin a &= \frac{2 \tan \frac{a}{2}}{1 + \tan^2 \frac{a}{2}} \\ \cos a &= \frac{1 - \tan^2 \frac{a}{2}}{1 + \tan^2 \frac{a}{2}} \\ \tan a &= \frac{2 \tan \frac{a}{2}}{1 - \tan^2 \frac{a}{2}}\end{aligned}$$

$$\begin{aligned}\sin \alpha + \sin \beta &= 2 \sin \frac{\alpha + \beta}{2} \cos \frac{\alpha - \beta}{2} \\ \sin \alpha - \sin \beta &= 2 \cos \frac{\alpha + \beta}{2} \sin \frac{\alpha - \beta}{2} \\ \cos \alpha + \cos \beta &= 2 \cos \frac{\alpha + \beta}{2} \cos \frac{\alpha - \beta}{2} \\ \cos \alpha - \cos \beta &= -2 \sin \frac{\alpha + \beta}{2} \sin \frac{\alpha - \beta}{2} \\ \sin \alpha \cos \beta &= \frac{1}{2} (\sin(\alpha + \beta) + \sin(\alpha - \beta)) \\ \cos \alpha \sin \beta &= \frac{1}{2} (\sin(\alpha + \beta) - \sin(\alpha - \beta)) \\ \cos \alpha \cos \beta &= \frac{1}{2} (\cos(\alpha + \beta) + \cos(\alpha - \beta)) \\ \sin \alpha \sin \beta &= -\frac{1}{2} (\cos(\alpha + \beta) - \cos(\alpha - \beta))\end{aligned}$$

等价无穷小:  $a^x - 1 \sim x \ln a$ ,  $\arcsin(ax) \sim \sin(ax) \sim ax$ ,  $\arctan(ax) \sim \tan(ax) \sim ax$ ,  
 $\ln(1+x) \sim x$ ,  $\sqrt{1+x} - \sqrt{1-x} \sim x$ ,  $(1+ax)^b - 1 \sim abx$ ,  $\sqrt[b]{1+ax} - 1 \sim \frac{a}{b}x$   
 $, 1 - \cos x \sim \frac{x^2}{2}$ ,  $x - \ln(1+x) \sim \frac{x^2}{2}$ ,  $\tan x - \sin x \sim \frac{x^3}{2}$ ,  $x - \arctan x \sim \frac{x^3}{3}$   
 $, x - \sin x \sim \frac{x^3}{6}$ ,  $\arcsin x - x \sim \frac{x^3}{6}$

泰勒展开(幂级数展开式): (通常取  $x_0 = 0$ )

$$\begin{aligned}f(x) &= f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \cdots + \frac{f^n(x_0)}{n!}(x - x_0)^n + R_n \\ e^x &= 1 + \frac{1}{1!}x + \frac{1}{2!}x^2 + \cdots \\ \ln(1+x) &= x - \frac{1}{2}x^2 + \frac{1}{3}x^3 + \cdots \\ \sin x &= x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5 + \cdots \\ \arcsin x &= x + \frac{1}{2} \frac{x^3}{3} + \frac{1 \times 3}{2 \times 4} \frac{x^5}{5} + \cdots \\ \cos x &= 1 - \frac{1}{2!}x^2 + \frac{1}{4!}x^4 \\ \frac{1}{1-x} &= 1 + x + x^2 + x^3 + \cdots \\ (1+x)^a &= 1 + \frac{a}{1!}x + \frac{a(a-1)}{2!}x^2 + \cdots\end{aligned}$$

求导公式:

$$\begin{aligned}
(C)' &= 0 \\
(x^\mu)' &= \mu x^{\mu-1} \\
(\sin x)' &= \cos x \\
(\cos x)' &= -\sin x \\
(\tan x)' &= \sec^2 x \\
(\cot x)' &= -\csc^2 x = -\csc x \cot x \\
(\sec x)' &= \sec x \tan x \\
(\csc x)' &= -\csc x \cot x \\
(a^x)' &= a^x \ln a \quad (a > 0, a \neq 1) \\
(e^x)' &= e^x \\
(\log_a x)' &= \frac{1}{x \ln a} \quad (a > 0, a \neq 1) \\
(\ln x)' &= \frac{1}{x} \\
(\arcsin x)' &= \frac{1}{\sqrt{1-x^2}} \\
(\arccos x)' &= -\frac{1}{\sqrt{1-x^2}} \\
(\arctan x)' &= \frac{1}{1+x^2} \\
(\operatorname{arccot} x)' &= -\frac{1}{1+x^2} \\
\ln(x + \sqrt{a^2 + x^2})' &= \frac{1}{\sqrt{a^2 + x^2}} \\
(x^a)^{(n)} &= \begin{cases} A_a^n x^{a-n} & (n \leq a) \\ 0 & (n > a) \end{cases}, a \in Z_+ \\
(\sin x)^{(n)} &= \sin\left(x + \frac{n\pi}{2}\right), n \in Z_+ \\
(\cos x)^{(n)} &= \cos\left(x + \frac{n\pi}{2}\right), n \in Z_+ \\
(\ln(1+x))^{(n)} &= (-1)^{n-1} \frac{(n-1)!}{(1+x)^n}, n \in Z_+ \\
(a^x)^{(n)} &= a^x \ln^n a, n \in Z_+
\end{aligned}$$

积分公式：

$$\begin{aligned}
\int kdx &= kx + C \quad (k \text{ 是常数}) \\
\int x^\mu dx &= \frac{x^{\mu+1}}{\mu+1} + C \quad (\mu \neq -1) \\
\int \frac{dx}{x} &= \ln|x| + C \\
\int \frac{dx}{1+x^2} &= \arctan x + C \\
\int \frac{dx}{\sqrt{1-x^2}} &= \arcsin x + C \\
\int \cos x dx &= \sin x + C \\
\int \sin x dx &= -\cos x + C \\
\int \frac{dx}{\cos^2 x} &= \int \sec^2 x dx = \tan x + C \\
\int \frac{dx}{\sin^2 x} &= \int \csc^2 x dx = -\cot x + C \\
\int \sec x \tan x dx &= \sec x + C \\
\int \csc x \cot x dx &= -\csc x + C \\
\int e^x dx &= e^x + C \\
\int a^x dx &= \frac{a^x}{\ln a} + C \\
\int \tan x dx &= -\ln|\cos x| + C \\
\int \cot x dx &= \ln|\sin x| + C \\
\int \sec x dx &= \ln|\sec x + \tan x| + C = \ln|\tan(\frac{x}{2} + \frac{\pi}{4})| \\
\int \csc x dx &= \ln|\csc x - \cot x| + C = \ln|\tan \frac{x}{2}| \\
\int \frac{dx}{\sqrt{a^2 - x^2}} &= \arcsin \frac{x}{a} + C \\
\int \frac{dx}{x^2 + a^2} &= \frac{1}{a} \arctan \frac{x}{a} + C \\
\int \frac{dx}{x^2 - a^2} &= \frac{1}{2a} \ln \left| \frac{x-a}{x+a} \right| + C \\
\int \frac{xdx}{\sqrt{x^2 \pm a^2}} &= \sqrt{x^2 \pm a^2} + C \\
\int \frac{dx}{\sqrt{x^2 \pm a^2}} &= \ln|x + \sqrt{x^2 \pm a^2}| + C \\
\int \sqrt{a^2 - x^2} dx &= \frac{x}{2} \sqrt{a^2 - x^2} + \frac{a^2}{2} \arcsin \frac{x}{a} + C \\
\int \sqrt{x^2 \pm a^2} dx &= \frac{x}{2} \sqrt{x^2 \pm a^2} \pm \frac{a^2}{2} \ln|x + \sqrt{x^2 \pm a^2}| + C
\end{aligned}$$

$$\begin{aligned}
\int \frac{A}{x-a} dx &= A \ln|x-a| + C \\
\int \frac{A}{(x-a)^n} dx &= -\frac{A}{(n-1)(x-a)^{n-1}} + C \\
\int \frac{Bx+C}{x^2+px+q} dx \quad (p^2-4q < 0) \\
&= \frac{B}{2} \ln(x^2+px+q) + \frac{2C-Bp}{\sqrt{4q-p^2}} \arctan \frac{2x+p}{\sqrt{4q-p^2}} + C \\
\int \frac{Bx+C}{(x^2+px+q)^n} dx \quad (p^2-4q < 0) \\
&= \frac{B}{2(1-n)(x^2+px+q)^{n-1}} + \left(C - \frac{Bp}{2}\right) \int \frac{dx}{\left(\left(x+\frac{p}{2}\right)^2 + \left(\sqrt{q-\frac{p^2}{4}}\right)^2\right)^n}
\end{aligned}$$

分部积分技巧：

- 不动：反三角函数、对数函数、幂函数(幂不动性差些)
- 动：三角函数、常数函数、指数函数

偏导数定义：  $\lim_{\Delta x \rightarrow 0} \frac{f(x_0 + \Delta x, y_0) - f(x_0, y_0)}{\Delta x}$

全微分：  $\Delta z = f(x + \Delta x, y + \Delta y) - f(x, y)$

多元函数极值：  $u$  在开区域  $G \subset \mathbb{R}^2$  内有二阶连续偏导数，且  $P_0(x_0, y_0) \in G$  是  $f$  的驻点，令：

$$f''_{xx}(x_0, y_0) = A, f''_{xy}(x_0, y_0) = B, f''_{yy}(x_0, y_0) = C$$

则：

1. 若  $AC - B^2 > 0$ ，则在点  $P_0$  取得极值， $A < 0$  极大值， $A > 0$  极小值
2. 若  $AC - B^2 < 0$ ，则在点  $P_0$  不取极值
3. 若  $AC - B^2 = 0$ ，则无法判断

拉格朗日乘数法：  $n$  元函数在开区域内有一阶连续偏导数，且不全为 0，则在条件下的极值点一定是拉格朗日函数( $n+1$  元函数，新元是  $\lambda$ )  $f + \lambda\varphi$  的驻点

三元积分变换：

柱坐标：是仅变换  $x, y$  为极坐标(和  $x, y$  区域)的变换公式。

$$\iiint_{\Omega} f(x, y, z) dv = \iiint_{\Omega'} f(r \cos \theta, r \sin \theta, z) r dr d\theta dz$$

球坐标：  $\theta$  同柱坐标， $\varphi$  是与  $z$  轴正向的夹角

$$\iiint_{\Omega} f(x, y, z) dv = f(r \sin \varphi \cos \theta, r \sin \varphi \sin \theta, r \cos \varphi) r^2 \sin \varphi dr d\varphi d\theta$$

设曲线  $L$  由参数方程  $x = x(t), y = y(t)$  ( $\alpha \geq t \geq \beta$ ) 表示， $x = x(t), y = y(t)$  在  $[\alpha, \beta]$  上有一阶连续偏导数，且  $x'^2(t) + y'^2(t) \neq 0$  (即曲线  $L$  是光滑简单曲线)，函数  $f(x, y)$  在曲线上连续，则：

$$\int_L f(x, y) ds = \int_{\alpha}^{\beta} f(x(t), y(t)) \sqrt{x'^2(t) + y'^2(t)} dt$$

显然如果是显函数  $y = f(x)$ , 则直接化为  $f(x)dx$ ,  $x = f(y)$  同理。

$$\iint_{\Sigma} f(x, y, z) dS = \iint_D f(x, y, z(x, y)) \sqrt{1 + \left(\frac{\partial z}{\partial x}\right)^2 + \left(\frac{\partial z}{\partial y}\right)^2} dx dy$$

计算时需要满足投影面面积  $S \neq 0$ , 如果不为零, 需要更换投影面, 换元投影。

设有界闭区域  $D$  由分段光滑曲线  $L$  所围, 函数  $P(x, y), Q(x, y)$  在  $D$  上具有一阶连续偏导数,  $L$  是  $D$  取正向的边界曲线:

$$\iint_D \begin{vmatrix} \frac{\partial}{\partial x} & \frac{\partial}{\partial y} \\ P & Q \end{vmatrix} dx dy = \iint_D \left( \frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y} \right) dx dy = \oint_L P dx + Q dy$$

对于多连通区域, 格林公式右端应包括沿区域  $D$  的全部边界正向曲线积分

有向曲线弧  $x = \varphi(t), y = \psi(t)$ , 起点和终点对应参数  $\alpha, \beta (\alpha < \beta)$ , 在  $[\alpha, \beta]$  上具有一阶连续导数, 且  $\varphi'^2(t) + \psi'^2(t) \neq 0$ , 切向量是  $\vec{t} = (\varphi'(t), \psi'(t))$ , 指向  $t$  增大的方向, 那么显然方向余弦是:

$$\cos \alpha = \frac{\varphi'(t)}{\sqrt{\varphi'^2(t) + \psi'^2(t)}} \quad \cos \beta = \frac{\psi'(t)}{\sqrt{\varphi'^2(t) + \psi'^2(t)}}$$

则两类积分的转换公式为:

$$\begin{aligned} \int_L P dx + Q dy &= \int_L (P \cos \alpha + Q \cos \beta) ds \\ ds &= \sqrt{\varphi'^2(t) + \psi'^2(t)} dt, dx = \varphi'(t) dt, dy = \psi'(t) dt \end{aligned}$$

高斯公式: 设空间闭区域  $\Omega$  是由分片光滑的闭曲线  $\Sigma$  所围成, 函数  $P(x, y, z), Q(x, y, z), R(x, y, z)$  在  $\Omega$  以及  $\Sigma$  上具有关于  $x, y, z$  的连续偏导数,  $\Sigma$  是  $\Omega$  整个边界曲面的外侧, 则有:

$$\begin{aligned} \iint_{\Omega} \left( \frac{\partial P}{\partial x} + \frac{\partial Q}{\partial y} + \frac{\partial R}{\partial z} \right) dx dy dz &= \oint_{\Sigma} P dy dz + Q dz dx + R dx dy \\ &= \iint_{\partial \Omega} P \cos \alpha + Q \cos \beta + R \cos \gamma dS \end{aligned}$$

注:  $P = 0, Q = 0, R = 0$  也是函数, 所以也就是说缺少了一部分也是可用的

$z = z(x, y)$  表示有向曲面  $\Sigma$ , 在  $xOy$  的投影区域是  $D_{xy}$ , 函数  $z = z(x, y)$  在  $D_{xy}$  上具有一阶连续偏导数,  $R(x, y, z)$  在  $\Sigma$  连续且  $\Sigma$  取上侧, 曲面  $\Sigma$  上一点的法向量方向余弦是:

$$\begin{aligned} \cos \alpha &= \frac{-z_x}{\sqrt{1 + z_x^2 + z_y^2}}, \cos \beta = \frac{-z_y}{\sqrt{1 + z_x^2 + z_y^2}}, \cos \gamma = \frac{1}{\sqrt{1 + z_x^2 + z_y^2}} \\ \iint_{\Sigma} P dy dz + Q dz dx + R dx dy &= \iint_{\Sigma} P \cos \alpha + Q \cos \beta + R \cos \gamma dS \end{aligned}$$

## 线性代数

齐次线性方程组(等式右边全是 0)必有零解。

$$\mathbf{A} \times \mathbf{B} = \left( \sum_{k=1}^n a_{ik} \cdot b_{kj} \right)$$

矩阵乘法符合结合律、分配律，不符合交换律

运算可行的前提下，转置满足如下运算规律：

- $(A^T)^T = A$
- $(A + B)^T = A^T + B^T$
- $(\lambda A)^T = \lambda A^T$
- $(AB)^T = B^T A^T$

矩阵  $\mathbf{A} = \mathbf{0}$  的充要条件是  $\mathbf{A}\mathbf{A}^T = \mathbf{0}$

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11}(a_{22}a_{33} - a_{23}a_{32}) - a_{12}(a_{21}a_{33} - a_{23}a_{31}) + a_{13}(a_{21}a_{32} - a_{22}a_{31})$$

行列式  $\det$  的性质：

- 行列式和它的转置行列式相等
- 对换行列式的相邻两行或两列，行列式变号  
→ 如果行列式存在两行或两列完全相同，则行列式为 0
- 行列式某一行或某一列所有元素同时乘以同一个数  $k$ ，等于用  $k$  乘以整个行列式  
→ 行列式某一行或某一列所有公因子可以提取到行列式外面
- 行列式如果有两行或两列元素成比例，则这个行列式为 0
- 若行列式某一行元素都是两数之和，那么可以将该行列式拆成两个行列式，这两个行列式除了这一行值分别是两数外，其他行不变；列同理
- 行列式某一行或列各元素乘以同一个数加到另一行或列对应元素上，行列式不变
- $|\lambda A| = \lambda^n |A|$
- $|AB| = |A||B|$

在  $n$  阶行列式中，把  $a_{ij}$  所在行和列去掉后，留下的  $n - 1$  阶行列式叫做  $a_{ij}$  的余子式  $M_{ij}$ 。且有代数余子式  $A_{ij}$  定义如下：

$$A_{ij} = (-1)^{i+j} M_{ij}$$

行列式的按行（列）展开法则：行列式等于它的任一行（列）各元素与其对应的代数余子式乘积之和：

$$D = \sum_{i=1}^n a_{ij} A_{ij} = \sum_{j=1}^n a_{ij} A_{ij}$$

一个  $n$  阶矩阵  $\mathbf{A}$  各个元素的代数余子式  $A_{ij}$  的转置矩阵构成了矩阵  $\mathbf{A}$  的伴随矩阵  $\mathbf{A}^*$ ：

$$\mathbf{A}^* = (A_{ij})^T = (A_{ji}) = \begin{pmatrix} A_{11} & A_{21} & \cdots & A_{n1} \\ A_{12} & A_{22} & \cdots & A_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ A_{1n} & A_{2n} & \cdots & A_{nn} \end{pmatrix}$$

伴随阵满足如下运算：(可以求逆矩阵)

$$\mathbf{A}\mathbf{A}^* = \mathbf{A}^*\mathbf{A} = |\mathbf{A}|\mathbf{E}$$

若矩阵  $\mathbf{A}$  可逆，则  $|\mathbf{A}| \neq 0$ ，称为非奇异矩阵。

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

- $\lambda \neq 0, (\lambda \mathbf{A})^{-1} = \frac{1}{\lambda} \mathbf{A}^{-1}$
- $(\mathbf{AB})^{-1} = \mathbf{B}^{-1} \mathbf{A}^{-1}$
- $\Lambda^{-1} = \text{diag}(\lambda_1^{-1}, \dots, \lambda_m^{-1})$

初等变换得到的等价关系具有传递性。 $\mathbf{A} \sim \mathbf{B}$  的充要条件是存在  $m$  阶可逆矩阵  $\mathbf{P}$ 、 $n$  阶可逆矩阵  $\mathbf{Q}$  使得  $\mathbf{PAQ} = \mathbf{B}$ 。方阵  $\mathbf{A}$  可逆的充要条件是  $\mathbf{A} \stackrel{c}{\sim} \mathbf{B}$  或  $\mathbf{A} \stackrel{c}{\sim} \mathbf{B}$  或  $\mathbf{A} \sim \mathbf{B}$

若矩阵  $\mathbf{A}$  中有不为 0 的  $r$  阶子式  $D$ ，且所有  $r + 1$  阶子式(若存在)均为 0，则  $D$  称为  $\mathbf{A}$  的 **最高阶非零子式**，且  $r$  称为  $\mathbf{A}$  的 **秩**，记作  $R(\mathbf{A})$ 。可逆矩阵的秩等于矩阵阶数，不可逆矩阵的秩小于矩阵阶数

向量方程组  $\mathbf{Ax} = \mathbf{b}$ ，解为  $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$

- 无解的充要条件是  $R(\mathbf{A}) < R(\mathbf{A}, \mathbf{b})$
- 有唯一解的充要条件是  $R(\mathbf{A}) = R(\mathbf{A}, \mathbf{b}) = n$
- 有无穷解的充要条件是  $R(\mathbf{A}) = R(\mathbf{A}, \mathbf{b}) < n$

## 概率论

事件的运算规律：

- 若  $A \subset B$  则  $A \cup B = B, AB = A$
- $A - B = A\bar{B} = A - AB, A \cup B = A \cup (B - A)$
- 结合律  $(A \cup B) \cup C = A \cup (B \cup C), (A \cap B) \cap C = A \cap (B \cap C)$
- 分配律  $(A \cup B) \cap C = (A \cap C) \cup (B \cap C)$   
 $(A \cap B) \cup C = (A \cap C) \cup (B \cup C)$
- 对偶律  $\overline{A \cup B} = \overline{A} \cap \overline{B}, \overline{A \cap B} = \overline{A} \cup \overline{B}$

概率公式：

- $P(A - B) = P(A) - P(AB)$
- 若  $B \subset A$ ，有：  $P(A - B) = P(A) - P(B), P(A) \geq P(B)$
- $P(A \cup B) = P(A) + P(B) - P(AB)$ ，可由容斥原理推广到任意多事件

在事件  $A$  发生的条件下，事件  $B$  的条件概率  $P(B|A) = \frac{P(AB)}{P(A)}$

$$P(A_1 \cup \dots \cup A_n | A) = P(A_1 | A) + \dots + P(A_n | A)$$

$$P(A_1 \dots A_n) = P(A_1)P(A_2 | A_1) \dots P(A_n | A_1 \dots A_{n-1})$$

如两个事件  $A, B$  满足：  $P(AB) = P(A)P(B)$ ，那么称  $A, B$  独立

两点分布方差为  $p(1 - p)$ ，二项分布方差为  $np(1 - p)$ ，均匀分布期望  $\frac{a + b}{2}$ ，方差  $\frac{(b - a)^2}{12}$

数学期望的性质：

- 若  $C$  是常数，则  $E(C) = C, E(CX) = CE(X)$
- $E(X_1 + X_2) = E(X_1) + E(X_2)$

- 若  $X, Y$  相互独立, 则  $E(XY) = E(X)E(Y)$

方差计算公式  $D(X) = E(X^2) - [E(X)]^2$

- 设  $C$  是常数, 则  $D(C) = 0$ ,  $D(CX) = C^2D(X)$
- 若  $X, Y$  相互独立, 则  $D(X \pm Y) = D(X) + D(Y)$  (注意都是加)

## 其他

阶乘 极限值: 13!刚好爆int, 21!刚好爆long long

斐波那契数列:  $f(47)$ 刚好爆int,  $f(93)$ 刚好爆long long

通项  $f_n = \left( \left( \frac{1 + \sqrt{5}}{2} \right)^n - \left( \frac{1 - \sqrt{5}}{2} \right)^n \right) \div \sqrt{5}$ 。约等于去掉减式的最近整数

快速倍增  $[F_{n-1} \ F_n] = [F_{n-2} \ F_{n-1}] \cdot \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$ , 则

$F_{2k} = F_k(2F_{k+1} - F_k)$ ,  $F_{2k+1} = F_{k+1}^2 + F_k^2$ .

- 卡西尼性质  $f_{n-1}f_{n+1} - f_n^2 = (-1)^n$
- 附加性质  $f(n+m) = f(n+1)f(m) + f(n)f(m-1)$

令  $m = n$  得  $f_{2n} = f_n(f_{n+1} + f_{n-1})$ ,

可推导出  $f_n | f_{nk}$ , 且可逆即  $\forall f_a | f_b, a | b$

- $\gcd(f_m, f_n) = f_{\gcd(m, n)}$
- $\sum_{i=1}^n f(n) = f(n+2) - 1$
- $\sum_{i=1}^n f^2(i) = f(n) \times f(n+1)$
- $f(n+1) < 2f(n) < f(n+2)$
- $f(n+2) < 3f(n)$
- $4f(n) < 3f(n+1) \Rightarrow f(n) < f(n+1) < 3f(n-1)$

齐肯多夫定理: 任何正整数可以表示为若干个不连续的 Fibonacci 数之和

模数为  $p$ , 则  $f \bmod p$  的循环节长度  $\pi(p) \leq 6p$ 。 (皮萨诺周期)

- $\pi(2) = 3, \pi(5) = 20$
- 若模数  $a, b$  互质, 则  $\pi(ab) = \gcd(\pi(a), \pi(b))$
- 对奇素数  $p \equiv 1, 4 \pmod{5}$ ,  $p-1 | \pi(p)$
- 对奇素数  $p \equiv 2, 3 \pmod{5}$ ,  $2p+2 | \pi(p)$
- 若  $\pi(p^{k-1}) | m$ , 则  $\pi(p^k) | mp$ 。若  $\pi(p^{k-1}) = m$  则  $\pi(p^k) = mp$

牛顿法求函数零点:  $x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$

尾 0 数目可以将大数质因数分解, 求  $\min(num_2, num_5)$  即可。其他质因数都不能乘出 0

任意整数转 Excel 列名: (列名本质是从 1('A') 计数的 27 进制)

```
1 #include <bits/stdc++.h>
2 int n; std::string x;
3 signed main()
4 {   for (scanf("%d", &n); n; n=(n-1)/26) x=(char)('A'+(n-1)%26)+x;
5   return std::cout<<x, 0; }
```

闰年公式: 满足其一 ① 为 4 的倍数不为 100 的倍数; ② 为 400 的倍数

## 数论

### 基础性质

设  $a, b, m \in \mathbb{Z}^+$ , 有:  $\lfloor \frac{a}{b} \rfloor = \lceil \frac{a+1}{b} \rceil - 1$ ,  $\lceil \frac{a}{b} \rceil = \lfloor \frac{a-1}{b} \rfloor + 1$ 。

$$\begin{aligned} ab \geq m &\Leftrightarrow a \geq \lceil \frac{m}{b} \rceil \Leftrightarrow b \geq \lceil \frac{m}{a} \rceil \\ ab > m &\Leftrightarrow a > \lceil \frac{m}{b} \rceil \Leftrightarrow b > \lceil \frac{m}{a} \rceil \\ ab \leq m &\Leftrightarrow a \leq \lfloor \frac{m}{b} \rfloor \Leftrightarrow b \leq \lfloor \frac{m}{a} \rfloor \\ ab < m &\Leftrightarrow a < \lfloor \frac{m}{b} \rfloor \Leftrightarrow b < \lfloor \frac{m}{a} \rfloor \end{aligned}$$

整除的性质: ( $x|y$  那么  $x$  是  $y$  倍数,  $y$  是  $x$  因数(约数),  $x, y \in \mathbb{Z}, x \neq 0$ )

- $a|b \Leftrightarrow -a|b \Leftrightarrow a|-b \Leftrightarrow |a| \mid |b|$
- $a|b$  且  $b|c \Rightarrow a|c$
- $a|b$  且  $a|c \Leftrightarrow \forall x, y \in \mathbb{Z}, a|\gcd(xb + yc)$  (字面理解:  $a$  是  $b, c$  公因数)
- $a|b$  且  $b|a \Rightarrow b = \pm a$
- $m \neq 0$  且  $a|b \Leftrightarrow ma|mb$
- $b \neq 0$  且  $a|b \Leftrightarrow |a| \leq |b|$
- $a \neq 0, b = qa + c, a|b \Leftrightarrow a|c$
- 0 是所有非零整数的倍数, 即  $0|x, x \neq 0$

真因数是对  $b \neq 0$  时除去  $\pm 1, \pm b$  外的全部因数。

素数的性质:

- 对合数  $a$ , 一定存在素数  $p \leq \sqrt{a}$  使得  $p|a$
- 所有大于 3 的素数都可以表示为  $6n \pm 1$  的形式
- 素数定理:  $\pi(x)$  为不超过  $x$  的素数个数, 则  $\pi(x) \approx \frac{x}{\ln x}$  ( $\pi(x)$  一般大于后者, 误差率约为 10%)

同余( $a \equiv b \pmod{m}$ ) 是  $m \neq 0, m|(a-b)$ , 称  $a$  同余于  $b$  模  $m$ ,  $b$  是  $a$  对模  $m$  的剩余)的性质:

- 传递性: 若  $a \equiv b \pmod{m}, b \equiv c \pmod{m}$  则  $a \equiv c \pmod{m}$

- 线性运算: 若  $a, b, c, d \in Z, m \in N^*, a \equiv b \pmod{m}, c \equiv d \pmod{m}$  则:
 
$$a \pm c \equiv b \pm d \pmod{m}, ac \equiv bd \pmod{m}$$
- 若  $a, b \in Z, k, m \in N^*, a \equiv b \pmod{m}$  则  $ak \equiv bk \pmod{mk}$
- 若  $a, b \in Z, d, m \in N^*, d|a, d|b, d|m, a \equiv b \pmod{m}$  则  $\frac{a}{d} \equiv \frac{b}{d} \pmod{\frac{m}{d}}$
- 若  $a, b \in Z, d, m \in N^*, d|m, a \equiv b \pmod{m}$  则  $a \equiv b \pmod{d}$
- 若  $a, b \in Z, d, m \in N^*, a \equiv b \pmod{m}$  则  $\gcd(a, m) = \gcd(b, m)$ , 若  $m|d$  且  $a|d, b|d$  成立其一, 那么  $a|d, b|d$  均成立

定义域是正整数的函数称为数论函数。

**积性函数**是  $f(1) = 1, \forall x, y \in N^*, \gcd(x, y) = 1 \Rightarrow f(xy) = f(x)f(y)$

**完全积性函数**是  $f(1) = 1, \forall x, y \in N^* \Rightarrow f(xy) = f(x)f(y)$

若  $f, g$  是积性函数, 则  $f(x^p), f^p(x), f(x)g(x), \sum_{d|x} f(d)g(\frac{x}{d})$  都是积性函数。

设  $x = \prod p_i^{k_i}$ , 积性函数满足  $f(x) = \prod f(p_i^{k_i})$ , 完全积性函数满足  $f(x) = \prod f(p_i)^{k_i}$

单位函数  $\epsilon(n) = [n = 1]$ , 除数函数

$\sigma_k(n) = \sum_{d|n} d^k, \sigma_0(n) = d(n) = \tau(n), \sigma_1(n) = \sigma(n)$

莫比乌斯函数  $\mu(n) = \begin{cases} 1 & n = 1 \\ 0 & \exists d > 1, d^2|n \\ (-1)^{\omega(n)} & otherwise \end{cases}$

函数

利用唯一分解定理质因数分解的复杂度为  $O(\sqrt{n})$ , 最坏复杂度在质数时取得(也可以枚举素数, 复杂度为  $O(\frac{n}{\ln n})$ , 但是需要素数筛预处理)

设  $x = \prod p_i^{a_i}$ , 则  $x$  因子数、因子和分别为  $\prod(a_i + 1), \prod(\sum_{j=0}^{a_i} p_i^j) = \prod \frac{p_i^{a_i+1} - 1}{p_i - 1}$

根据上式可知完全平方数(1, 4, 9, ...)有奇数个因数, 其他数有偶数个因数。因子和是积性函数, 设  $x$  分解出一个素数  $p$ , 则  $f(x) = f(\frac{x}{p})f(p)$ , 其中显然  $f(p) = p + 1$

互质/最大公因数/最小公倍数的性质:

- (辗转相除法)  $\gcd(a, b) = b? \gcd(b, a \% b) : a$  复杂度  $O(\log n)$ ;  $[a, b] = \frac{ab}{(a, b)}$
- 若  $a, b \in N^*, (a, b) = k$  则  $(\frac{a}{k}, \frac{b}{k}) = 1$ ; 若  $[a, b] = k$  则  $(\frac{k}{a}, \frac{k}{b}) = 1$
- 若  $p \perp q$ , 则  $(p^n, q^m) = 1$ , 也意味着  $p^n \pmod{q^m} \neq 0$
- 若  $x(a, b) = a, y(a, b) = b$  则  $(x, y) = 1$  即  $(\frac{a}{(a, b)}, \frac{b}{(a, b)}) = 1$
- 若  $(a \cdot b) \perp (c \cdot d)$  则  $a \perp c, a \perp d, b \perp c, b \perp d$

应用: 从长为  $n$  的环任意位置开始循环每次跳  $m$  格走, 能遍历全部位置充要条件是  $(n, m) = 1$

四平方和定理: 所有自然数至多只要用四个数的平方和就可以表示

## 素数

### 素性测试

判断单个数是否是素数模板 ( $O(n)$ ):

```
1 bool isprime(ll p)
2 {
3     if(p==2||p==3) return true;
4     if(p<2||(p%6!=1&&p%6!=5)) return false;
5     for(ll i=5;i*i<=p;i+=6) if(p%i==0||p%(i+2)==0) return false;
6     return true;
7 }
```

**Miller Rabin算法**, 复杂度  $O(k \log n)$ ,  $k$  是设定参数, 能够以较高准确率判定是否为素数, 建议设 10, 见 `pollard-rho` 代码。

100 内质数为 25 个

```
2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73,79,83,89,97
```

## 埃氏筛

埃氏筛(复杂度  $O(n \log \log n)$ )

```
1 // C++ Version
2 int Eratosthenes(int n) {
3     int p = 0;
4     for (int i = 0; i <= n; ++i) is_prime[i] = 1;
5     is_prime[0] = is_prime[1] = 0;
6     for (int i = 2; i <= n; ++i) {
7         if (is_prime[i]) {
8             prime[p++] = i; // prime[p]是i, 后置自增运算代表当前素数数量
9             if ((long long)i * i <= n)
10                 for (int j = i * i; j <= n; j += i)
11                     // 因为从 2 到 i - 1 的倍数我们之前筛过了, 这里直接从 i
12                     // 的倍数开始, 提高了运行速度
13                     is_prime[j] = 0; // 是i的倍数的均不是素数
14         }
15     }
16     return p;
17 }
```

## 欧拉筛

欧拉筛: (洛谷P3383)

```
1 #include <bits/stdc++.h>
2 #define MAXN 100000002
3 #define MAXP 6000002
4 using namespace std;
```

```

5  bool vis[MAXN]; //vis[0] = vis[1] = 1;
6  int Prime[MAXP], k, n, q, c;
7  void euler(int n)
8  {
9      for(int i=2;i<=n;i++)
10     {
11         if(!vis[i]) Prime[++k]=i;
12         for(int j=1;j<=k;j++)
13         {
14             if(Prime[j]*i>n) break;
15             vis[Prime[j]*i]=true;
16             if(i%Prime[j]==0) break;
17         }
18     }
19 }
20 int main()
21 {
22     scanf("%d%d", &n, &q);
23     euler(n);
24     while(q--)
25     {
26         scanf("%d", &c);
27         printf("%d\n", Prime[c]);
28     }
29     return 0;
30 }

```

求任意连续区间素数: (洛谷P1835-求素数个数  $1 \leq l \leq r < 2^{31}, r - l \leq 10^6$ )

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #define MAXN 1000002
5 #define SQRTR 50000
6 ll prime[MAXN], k, ans, lf, rf, p, bg;
7 bool vis[MAXN];
8 inline void euler()
9 {
10     for(ll i = 2; i <= SQRTR; ++i)
11     {
12         if (!vis[i]) prime[++k] = i;
13         for (ll j = 1; i * prime[j] <= SQRTR; ++j)
14         {
15             vis[i * prime[j]] = true;
16             if (i % prime[j] == 0) break;
17         }
18     }
19 }
20 signed main()
21 {
22     scanf("%lld%lld", &lf, &rf);
23     lf = (lf == 1) ? 2 : lf;
24     euler();
25     memset(vis, 0, sizeof vis);
26     for (ll i = 1; i <= k; ++i)

```

```

27     {
28         p = prime[i];
29         bg = (lf + p - 1) / p * p > 2 * p ? (lf + p - 1) / p * p : 2 * p;
30         for (ll j = bg; j <= rf; j += p) vis[j - lf + 1] = true;
31     }
32     for (ll i = 1; i <= rf - lf + 1; ++i) if (!vis[i]) ++ans;
33     printf("%lld", ans);
34     return 0;
35 }

```

SCNUOJ1745 求  $\sum_{i=1}^n \varphi(i^k) \cdot \sigma(i^k) \pmod{10^9 + 7}$ ,  $1 \leq n, k \leq 10^6$

对素数  $p$  和正整数  $k$ , 有  $\varphi(p^k) = p^k - p^{k-1}$ ,  $\sigma(p^k) = \frac{p^k - 1}{p - 1}$

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #define sc(x) scanf("%lld", &x)
5 #define mn 1000000
6 ll n, k, p[mn], pri[mn], e[mn], pe[mn], g[mn], cnt, ans = 1, mod = 1e9 + 7;
7 void euler(ll n)
8 { // e[i]是i质因数分解得到的最大的幂a_i,pe[i]是对应最大的(p^a[i])
9     for (ll i = 2; i <= n; ++i)
10     {
11         if (!p[i])
12         {
13             p[i] = i, pri[++cnt] = i, pe[i] = i, e[i] = 1;
14         }
15         for (ll j = 1; i * pri[j] <= n; ++j)
16         {
17             p[i * pri[j]] = pri[j];
18             if (pri[j] == p[i])
19             {
20                 e[i * pri[j]] = e[i] + 1;
21                 pe[i * pri[j]] = pe[i] * pri[j];
22                 break;
23             }
24             e[i * pri[j]] = 1;
25             pe[i * pri[j]] = pri[j];
26         }
27     }
28 }
29 ll qpow(ll a, ll b)
30 {
31     ll res = 1;
32     for (; b > 0; b >>= 1)
33     {
34         if (b & 1)
35         {
36             res = res * a % mod;
37         }
38         a = a * a % mod;
39     }
40     return res;

```

```

41 }
42 signed main()
43 {
44     sc(n), sc(k);
45     g[1] = 1;
46     euler(n);
47     for (ll i = 2; i <= n; ++i)
48     {
49         if (pe[i] == i)
50         {
51             g[i] = (qpow(p[i], e[i] * k + 1) - 1 + mod) % mod * qpow(p[i] - 1, mod - 2) % mod;
52             g[i] = g[i] * (qpow(p[i], e[i] * k) - qpow(p[i], e[i] * k - 1) + mod) % mod;
53         }
54         else
55         {
56             g[i] = g[i / pe[i]] * g[pe[i]] % mod;
57         }
58         ans = (ans + g[i]) % mod;
59     }
60     printf("%.11d", ans);
61     return 0;
62 }

```

$O(n)$  求积性函数  $f$ ，条件：任意质数  $p$  和正整数  $k$ ，可以  $O(1)$  求出  $f(p^k)$ ，那么设合数  $n = \prod_{i=1}^k p_i^{\alpha_i}$ ，线性筛记录  $g_n = p_1^{\alpha_1}$ （即  $n$  的第一个质数幂），若  $n$  被  $x \cdot p$  筛掉（ $p$  是质数），那么：

$$g_n = \begin{cases} g_x \cdot p & x \bmod p = 0 \\ p & \text{otherwise} \end{cases}$$

若  $n = g_n$ ，直接算  $f(n)$ ，否则  $f(n) = f\left(\frac{n}{g_n}\right)f(g_n)$

用欧拉筛可以在筛的时候求一些其他值，如例题：求  $\sum_{i=1}^n f(i^k) \bmod 10^9 + 7$ ,  $f$  是因子和

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ln;
4 typedef int ll;
5 #define mn 10000002
6 #define mod 1000000007
7 ll n, k, p[mn], pri[mn], e[mn], pe[mn], g[mn], cnt; ln ans = 1;
8 void euler(ll n)
9 {//e[i]是i质因数分解得到的最大的幂a_i,pe[i]是对应最大的(p^a[i])
10     for(ll i=2;i<=n;++i)
11     {
12         if (!p[i]) p[i] = i, pri[++cnt] = i, pe[i] = i, e[i] = 1;
13         for (ll j = 1; i * pri[j] <= n; ++j)
14         {
15             p[i * pri[j]] = pri[j];
16             if (pri[j] == p[i])
17             {
18                 e[i * pri[j]] = e[i] + 1;
19             }
20         }
21     }
22     for (ll i = 1; i <= n; ++i)
23     {
24         ans = (ans * e[i]) % mod;
25     }
26 }

```

```

19         pe[i * pri[j]] = pe[i] * pri[j];
20         break;
21     }
22     e[i * pri[j]] = 1;
23     pe[i * pri[j]] = pri[j];
24 }
25 }
26 }
27 ln qpow(ln a, ln b)
28 {
29     ln res = 1;
30     for (; b; b >= 1, (a *= a) %= mod) if (b & 1) (res *= a) %= mod;
31     return res;
32 }
33 signed main()
34 {
35     euler(100); g[1] = 1;
36     scanf("%d%d", &n, &k);
37     for(11 i=1;i<=n;++i)
38     {
39         if (pe[i] == i) g[i] = (qpow(p[i], 1LL * e[i] * k + 1) - 1 + mod) %
40         mod * qpow(p[i] - 1, mod - 2) % mod;
41         else g[i] = 1LL * g[i] / pe[i] * g[pe[i]] % mod;
42         (ans += g[i]) %= mod;
43     }
44     return printf("%lld", ans) & 0;
45 }

```

## pollard-rho算法

质因数分解的朴素复杂度是  $O(\sqrt{n})$ ，在预处理素数表的情况下质因数分解复杂度为  $O(\sqrt{\frac{n}{\ln n}})$ 。注意不能只预处理  $\sqrt{n}$ ，以防大质数乘小质数情况(此时对大质数可以除法求出，然后做素性测试)和本身是大质数。

若  $x|n$ ,  $1 < x < n$ (即 1 和自身外的因子)，则  $x$  是非平凡因子。能以  $O(n^{0.25})$  分解出随机一个非平凡因数。

P4718 给定  $t$  ( $\leq 350$ ) 个  $n$  ( $1 < n \leq 10^{18}$ )，判断  $n$  为质数或输出它的最大质因子

by NULL-SF

```

1 #include <iostream>
2 #include <algorithm>
3 #include <random>
4 #include <chrono>
5
6 #define int long long
7
8 using namespace std;
9
10 typedef long long ll;
11
12 auto seed = chrono::system_clock::now().time_since_epoch().count();

```

```

13 mt19937 gen(seed);
14
15 uniform_int_distribution<long long> dis(1, (long long)1e18);
16
17 ll qpow(ll a, ll b, ll mod)
18 {
19     ll r = 1;
20
21     for (; b; b >>= 1)
22     {
23         if (b & 1)
24             r = (__int128)r * a % mod;
25         a = (__int128)a * a % mod;
26     }
27
28     return r;
29 }
30
31 bool miller_rabin(ll p, ll k = 30) // 判断素数
32 {
33     if (p < 2) return false;
34     if (p == 2) return true;
35     if (p == 3) return true;
36
37     ll d = p - 1, r = 0;
38     while (!(d & 1)) ++r, d >>= 1; // 将d处理为奇数
39
40     while(k--) // k次随机运算提高准确率
41     {
42         ll a = dis(gen) % (p - 2) + 2;
43         ll x = qpow(a, d, p);
44
45         if (x == 1 || x == p - 1) continue;
46
47         for (ll i = 0; i < r - 1; ++i)
48         {
49             x = (__int128)x * x % p;
50
51             if (x == p - 1) break;
52         }
53
54         if (x != p - 1) return false;
55     }
56
57     return true;
58 }
59
60 ll pollard_rho(ll x)
61 {
62     ll c = dis(gen) % (x - 1) + 1;
63
64     for (ll goal = 1, s = 0, t; true; goal <= 1, s = t)
65     {
66         ll val = 1;
67
68         for (ll step = 1; step <= goal; ++step)

```

```

69     {
70         t = ((__int128)t * t + c) % x; //随机数生成器
71         val = ((__int128)val * abs(t - s)) % x;
72
73         if (step % 127 == 0)
74         {
75             if (d > 1) return d;
76         }
77     }
78 }
79
80     if (d > 1) return d;
81 }
82 }
83 }
84
85 ll get_maxi_prime_factor(ll x)
86 {
87     if (x < 2) return 0;
88
89     if (miller_rabin(x)) return x;
90
91     ll p = x;
92     while (p >= x) p = pollard_rho(x); //直到找到平凡
93     while (x % p == 0) x /= p;
94
95     return max(get_maxi_prime_factor(x), get_maxi_prime_factor(p)); //返回最
96     大质因子
97 }
98
99 signed main()
100 {
101     ios::sync_with_stdio(false), cin.tie(0), cout.tie(0);
102
103     ll t, n;
104
105     cin >> t;
106     while (t--)
107     {
108         cin >> n;
109
110         ll mx = get_maxi_prime_factor(n);
111
112         if (mx == n) cout << "Prime\n";
113         else cout << mx << '\n';
114     }
115
116     return 0;
117 }
118
119 /* 因式分解:
120 void get_prime_factors(ll base, ll index, map<ll, ll>& cnt_prime_fac)
121 {
122     if (base < 2) return;
123
124     if (miller_rabin(base))

```

```

124     {
125         cnt_prime_fac[base] += index;
126
127         return;
128     }
129
130     ll new_fac = base, multimes = 0;
131     while (new_fac >= base) new_fac = pollard_rho(base); //直到找到平凡
132     while (base % new_fac == 0) base /= new_fac, multimes++;
133
134     get_prime_factors(base, index, cnt_prime_fac);
135     get_prime_factors(new_fac, index * multimes, cnt_prime_fac);
136
137     return;
138 }
139
140 signed main()
141 {
142     ios::sync_with_stdio(false), cin.tie(0), cout.tie(0);
143
144     map<ll, ll> prime_factors;
145     get_prime_factors(27, 2, prime_factors);
146
147     for (auto x: prime_factors) cout << x.first << ':' << x.second << '\n';
148
149     return 0;
150 }
151
152 */

```

## Meissel-Lehmer算法

作用：准确地求出  $\pi(x)$  值和  $[1, x]$  内各素数，时间复杂度为  $O\left(\frac{x^{\frac{2}{3}}}{\log^2 x}\right)$ ，空间复杂度为  $O(x^{\frac{1}{3}} \log^3 x \log \log x)$

```

1 #include<bits/stdc++.h>
2 #define N 216000
3 #define ll long long
4 using namespace std;
5 int mn[N], pri[N/10], f1[N];
6 int tot, cnt, num, n;
7 int f[20005][55];
8 int inf=2e9;
9 int dp(int x, int y){
10     if (x<=20000&&y<=50) return f[x][y];
11     if (x==0||y==0) return x;
12     if (111*pri[y]*pri[y]>=x&&x<N) return max(0, mn[x]-y);
13     return dp(x, y-1)-dp(x/pri[y], y-1);
14 }
15 void pre(){
16     for (int i=2; i<N; i++){
17         if (!f1[i]) pri[++tot]=i;

```

```

18     for (int j=1;i*pri[j]<N&&j<=tot;j++){
19         fl[i*pri[j]]=1;
20         if (i*pri[j]==0) break;
21     }
22 }
23 for (int i=1;i<N;i++)
24     mn[i]=(cnt+=1-fl[i]);
25 for (int i=1;i<=20000;i++) f[i][0]=i;
26 for (int i=1;i<=20000;i++)
27     for (int j=1;j<=50;j++)
28         f[i][j]=f[i][j-1]-f[i/pri[j]][j-1];
29 }
30 int power(int x,int y){
31     int s=1;
32     while (y!=0){
33         if (y&1){
34             if (s>=inf/x) s=inf;
35             else s=s*x;
36         }
37         y/=2;
38         if (x>=inf/x) x=inf;
39         else x=x*x;
40     }
41     return s;
42 }
43 int yroot(ll x,int y){
44     int l=2,r=6666,ans=1;
45     while (l<=r){
46         int mid=(l+r)/2;
47         if (power(mid,y)<=x) ans=mid,l=mid+1;
48         else r=mid-1;
49     }
50     return ans;
51 }
52 int work(int m){
53     if (m<N) return mn[m]-1;
54     int y=yroot(m,3),n=mn[y];
55     int ans=dp(m,n)+n-1;
56     for (n++;pri[n]*pri[n]<=m;n++)
57         ans-=mn[m/pri[n]]-mn[pri[n]]+1;
58     return ans;
59 }
60 int main(){
61     pre();
62     scanf("%d",&n);
63     int an=work(n);
64     printf("%d\n",an);
65     for(int i=1;i<=an;++i) printf("%d ",pri[i]);
66 }

```

```

1 #include <bits/stdc++.h>
2 #define lowbit(x) ((x) & -(x))
3 typedef unsigned long long ull;
4 typedef unsigned int uint;
5 typedef long long ll;

```

```

6  using namespace std;
7
8  ull count_primes(ull x)
9  {
10     if (x <= 1)
11         return 0;
12
13     const int eps = 128;
14     const int eps_bit = 8;
15
16     uint s2 = sqrtl(x);
17     uint s3 = powl(x, 1.0 / 3);
18     uint s4 = sqrtl(s2);
19     uint B = std::max(ull, std::min(ull(s2), ull(s3 * std::max(1.0,
20     pow(log(x), 3) * 0.001)))) ;
21     uint u = x / B + eps;
22
23     uint a, b, t;
24     register uint p, q, r, u, v;
25     ll sum = 0;
26
27     vector<char> mu(B + 1);
28     mu[1] = 1;
29     for (a = 2; a <= B; ++a)
30         if (!mu[a])
31             for (b = a; b <= B; b += a)
32                 mu[b] = mu[b] < 0 ? 1 : -1;
33     for (a = 2; a * a <= B; ++a)
34         if (mu[a * a])
35             for (b = a * a; b <= B; b += a * a)
36                 mu[b] = 0;
37
38     vector<int> pi(u), primes(uint(u / log(u) * 1.25506)), md(u);
39     uint count = 0;
40     md[1] = 1 << 30, primes[0] = 1;
41     for (a = 6; a < u; a += 3)
42         md[a] = 3;
43     for (a = 4; a < u; a += 2)
44         md[a] = 2;
45     for (a = 4; a * a < u; ++a)
46         if (!md[a])
47             {
48                 for (b = a * a; b < u; b += 6 * a)
49                     if (!md[b])
50                         md[b] = a;
51                     for (b = a * (a + (3 - a % 3) * 2); b < u; b += 6 * a)
52                         if (!md[b])
53                             md[b] = a;
54             }
55     for (a = 2; a < u; ++a)
56         pi[a] = md[a] ? pi[a - 1] : (primes[++count] = md[a] = a, pi[a - 1]
57 + 1);
58
59     sum += pi[B] - 1;
60     for (p = pi[B] + 1; primes[p] <= s2; ++p)
61         sum -= pi[x / primes[p]] - p + 1;

```

```

60     for (a = 1; a <= B; ++a)
61         if (mu[a])
62             sum += mu[a] * (x / a);
63     for (p = sqrtl(x / primes[pi[B]]) + 1; p <= B; ++p)
64         if (pi[p] != pi[p - 1])
65             sum += pi[B] - pi[std::max(uint(x / p / p), p)];
66
67     for (p = S4 + 1; p <= S3; ++p)
68         if (pi[p] != pi[p - 1])
69         {
70             uint L = p + 1, R = std::min(ull)B, x / p / p);
71             sum += (11)(2 - pi[p]) * (pi[R] - pi[L - 1]);
72             ull m = x / p;
73             for (u = L; u <= R; u = v + 1)
74             {
75                 t = pi[m / primes[pi[u - 1] + 1]];
76                 v = m / primes[t];
77                 if (v > R)
78                     v = R;
79                 sum += (11)(pi[v] - pi[u - 1]) * t;
80             }
81         }
82
83     vector<int> bit(U);
84     vector<int> nxt(B + 2);
85     for (q = 1; q <= B; ++q)
86         nxt[q] = q + !mu[q];
87     nxt[B + 1] = B + 1;
88     for (r = 1; r <= U - eps; ++r)
89         bit[r] = r;
90     for (p = 1; p <= S4; ++p)
91         if (pi[p] != pi[p - 1])
92         {
93             ull m = x / p;
94             if (p <= eps_bit)
95             {
96                 for (q = std::max(p, B / p) + 1; q <= B; ++q)
97                 {
98                     while (q != nxt[q])
99                         q = nxt[q] = nxt[nxt[q]];
100                     if (q > B)
101                         break;
102                     if (md[q] <= p)
103                     {
104                         nxt[q] = q + 1;
105                         continue;
106                     }
107                     sum -= mu[q] * bit[m / q];
108                 }
109                 for (r = 1; r < U; ++r)
110                     bit[r] = md[r] > p;
111                 if (pi[p] < count && primes[pi[p] + 1] > eps_bit)
112                 {
113                     for (r = 1; r < U; ++r)
114                         if (r + lowbit(r) < U)
115                             bit[r + lowbit(r)] += bit[r];

```

```

116         }
117     else
118     {
119         for (r = 1; r < U; ++r)
120             bit[r] += bit[r - 1];
121     }
122 }
123 else
124 {
125     for (q = std::max(p, B / p) + 1; q <= B; ++q)
126     {
127         while (q != nxt[q])
128             q = nxt[q] = nxt[nxt[q]];
129         if (q > B)
130             break;
131         if (md[q] <= p)
132         {
133             nxt[q] = q + 1;
134             continue;
135         }
136         if (mu[q] == 1)
137         {
138             for (a = m / q; a; a -= lowbit(a))
139                 sum -= bit[a];
140         }
141         else
142         {
143             for (a = m / q; a; a -= lowbit(a))
144                 sum += bit[a];
145         }
146     }
147     for (a = p; a < U; a += lowbit(a))
148         --bit[a];
149     for (r = p * p; r < U; r += 6 * p)
150         if (md[r] == p)
151             for (a = r; a < U; a += lowbit(a))
152                 --bit[a];
153     for (r = p * (p + (3 - p % 3) * 2); r < U; r += 6 * p)
154         if (md[r] == p)
155             for (a = r; a < U; a += lowbit(a))
156                 --bit[a];
157     }
158 }
159
160     return sum;
161 }
162
163 int main()
164 {
165     ull n;
166     cin >> n;
167     cout << count_primes(n);
168 }

```

## 拓展欧几里得定理

求下列关于  $x, y$  的不定方程的一组可行解: ( $a, b, x, y$  都是整数)

$$ax + by = (a, b)$$

裴蜀定理又称贝祖定理 (Bézout's lemma) , 设  $a, b \in \mathbb{Z}, ab \neq 0$ , 则  $\exists x, y \in \mathbb{Z}$ :

$$ax + by = (a, b)$$

```
1 void exgcd(11 a, 11 b, 11 &x, 11 &y)
2 {
3     if (!b) x = 1, y = 0;
4     else exgcd(b, a % b, y, x), y -= a / b * x;
5 }
```

若要求  $ax + by = c$  的一组整数解, 需要先求出  $ax_0 + by_0 = (a, b)$  的一组解, 然后得到  $ax_0 \times \frac{c}{(a, b)} + by_0 \times \frac{c}{(a, b)} = (a, b) \times \frac{c}{(a, b)}$ , 若  $\frac{c}{(a, b)}$  非整数, 无解。

显然有恒等式  $a(x + db) + b(y - da) = c$ , 令  $d = \frac{1}{(a, b)}$ , 可以得到最小非负数解  $x' = (x \bmod \frac{b}{(a, b)} + \frac{b}{(a, b)}) \bmod \frac{b}{(a, b)}$ 。特别地若  $a < 0$  可以两边同时取负。

洛谷P1082-求  $ax \equiv 1 \pmod{b}$  的最小正整数解  $x$ , 保证有解

本质是  $ax = by + 1 \rightarrow ax - by = 1$ , 有解条件是  $(a, b) = 1$ , 输出  $(x \bmod b + x) \bmod b$  即可

洛谷P5656-输入  $1 \leq a, b, c \leq 10^9$ , 求  $ax + by = c$  的整数解。无解输出 `-1`, 有正整数解( $x, y \in \mathbb{Z}_+$ )输出正整数解数量, 所有正整数解中  $x, y$  的最小值, 最大值。若无正整数解, 输出  $x, y$  代表整数解里  $x, y$  的最小正整数值

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 ll exgcd(ll a, ll b, ll &x, ll &y)
5 {
6     ll d = a;
7     if (!b)
8         x = 1, y = 0;
9     else
10        d = exgcd(b, a % b, y, x), y -= a / b * x;
11    return d;
12 }
13 ll a, b, c, x, y, t, d, p, q, k;
14 signed main()
15 {
16     for (scanf("%lld", &t); t; --t)
17     {
18         scanf("%lld%lld%lld", &a, &b, &c);
19         d = exgcd(a, b, x, y);
```

```

20     if (c % d != 0)
21         printf("-1\n");
22     else
23     {
24         x *= c / d, y *= c / d, p = b / d, q = a / d;
25         if (x < 0)
26             k = ceil((1.0 - x) / p), x += p * k, y -= q * k;
27         else
28             k = (x - 1) / p, x -= p * k, y += q * k;
29         if (y > 0)
30         {
31             printf("%lld ", (y - 1) / q + 1);
32             printf("%lld %lld ", x, (y - 1) % q + 1);
33             printf("%lld %lld\n", x + (y - 1) / q * p, y);
34         }
35         else
36             printf("%lld %lld\n", x, y + q * (1)ceil((1.0 - y) / q));
37     }
38 }
39 return 0;
40 }
```

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 using ll = long long;
4 ll exgcd(ll a, ll b, ll &x, ll &y)
5 {
6     if (!b)
7     {
8         x = 1, y = 0;
9         return a;
10    }
11    ll d = exgcd(b, a % b, y, x);
12    y -= a / b * x;
13    return d;
14 }
15 ll t, a, b, c, x, y, xmi, xmx, ymi, ymx;
16 signed main()
17 {
18     cin.tie(0)->ios::sync_with_stdio(false);
19     cin >> t;
20     while (t--)
21     {
22         cin >> a >> b >> c;
23         ll g = exgcd(a, b, x, y);
24         if (c % g)
25         {
26             cout << "-1\n";
27             continue;
28         }
29         a /= g, b /= g, c /= g;
30         x *= c, y *= c;
31         x = (x % b + b) % b;
32         x = x == 0 ? b : x;
33         y = (c - a * x) / b;
```

```

34     if (y > 0)
35     {
36         xmi = x, ymx = y;
37         y %= a;
38         y = y == 0 ? a : y;
39         x = (c - b * y) / a;
40         xmx = x, ymi = y;
41         cout << (xmx - xmi) / b + 1 << ' ';
42         cout << xmi << ' ' << ymi << ' ' << xmx << ' ' << ymx << '\n';
43     }
44     else
45     {
46         cout << x << ' ' << y % a + a << '\n';
47     }
48 }
49 return 0;
50 }

```

## 取模公式

### 费马小定理

若  $p$  为素数,  $a \in Z_+, a \perp p$ , 则  $a^{p-1} \equiv 1 \pmod{p}$

线性逆元公式(从 1 开始):  $i^{-1} \equiv -\lfloor \frac{p}{i} \rfloor \times (p \bmod i)^{-1} \pmod{p}$  (实现时 + $p$  再取模)

任意  $n$  数线性求逆元公式: 记  $s_i$  是前缀积, 则  $a_i^{-1} \equiv s_{i-1} \cdot (s_i)^{-1} \pmod{p}$  (预处理  $s_{i-1}^{-1} \equiv s_i^{-1} \cdot a_i \pmod{p}$ )

阶乘线性逆元公式:  $((i-1)!)^{-1} = i \times (i!)^{-1}$  (推论:  $\frac{1}{i} \equiv (i!)^{-1} \times (i-1)! \pmod{p}$ )

### 卢卡斯定理

$n = sp + q, m = tp + r, 0 \leq q, r \leq p-1, p$  是质数, 则

$$\binom{n}{m} \pmod{p} = \binom{\lfloor \frac{n}{p} \rfloor}{\lfloor \frac{m}{p} \rfloor} \cdot \binom{n \pmod{p}}{m \pmod{p}} \pmod{p}$$

时间复杂度为  $O(f + g \log n)$ ,  $f$  是预处理组合数复杂度(如阶乘逆元),  $g$  是单次求组合数复杂度

洛谷P3807-求  $C_{n+m}^n \pmod{p}$ ,  $p$  是质数,  $1 \leq n, m, p \leq 10^5, 1 \leq T \leq 10$

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef double db;
4 typedef long long ll;
5 #define mn 100002
6 ll a[mn], p, t, n, m; // i!=a[i]
7 ll qpow(ll a, ll b)
8 {
9     ll res = 1;
10    for (; b; b >= 1, a = a * a % p) if (b & 1)
11        res = res * a % p;

```

```

12     return res;
13 }
14 ll c(ll n, ll m)
15 {
16     if (m > n) return 0;
17     return ((a[n] * qpow(a[m], p - 2)) % p * qpow(a[n - m], p - 2) % p);
18 }
19 ll lucas(ll n, ll m)
20 {
21     if (!m) return 1;
22     return c(n % p, m % p) * lucas(n / p, m / p) % p;
23 }
24 signed main()
25 {
26     a[0] = 1;
27     for (scanf("%lld", &t); t; --t)
28     {
29         scanf("%lld%lld%lld", &n, &m, &p);
30         for (ll i = 1; i <= p; ++i) a[i] = a[i - 1] * i % p;
31         printf("%lld\n", lucas(n + m, n)); //原题算c(m+n, n)
32     }
33     return 0;
34 }

```

## 扩展卢卡斯定理

求  $C_n^m \bmod p$ , 将  $p$  质因数分解为  $p_1^{\alpha_1} p_2^{\alpha_2} \dots$ , 用中国剩余定理求方程组  $C_n^m \bmod p_1^{\alpha_i}$  的最小解  $x$ 。如果  $p_i^{\alpha_1}$  都是素数且较小可以预处理阶乘, 直接对每个子算式上 lucas, 否则用下述代码:

不考虑 exCRT 复杂度时, 预处理  $n! \div n$  内  $p$  所有倍数乘积  $\bmod p$  复杂度为  $O(p + T \log p)$

洛谷P4720-求  $C_n^m \bmod p$ ,  $1 \leq m \leq n \leq 10^{18}$ ,  $2 \leq p \leq 10^6$ 。

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 using ll = long long;
4 void exgcd(ll a, ll b, ll &x, ll &y)
5 {
6     if (!b)
7         x = 1, y = 0;
8     else
9         exgcd(b, a % b, y, x), y -= a / b * x;
10 }
11 ll inv(ll a, ll p)
12 {
13     ll x, y;
14     exgcd(a, p, x, y);
15     return (x + p) % p;
16 }
17 ll qpow(ll a, ll b, ll p)
18 {
19     ll r = 1;
20     for (; b; b >= 1, a = a * a % p)
21         if (b & 1)
22             r = r * a % p;
23     return r;

```

```

24 }
25 ll CRT(int n, ll *a, ll *m)
26 {
27     ll M = 1, p = 0;
28     for (int i = 1; i <= n; i++)
29         M = M * m[i];
30     for (int i = 1; i <= n; i++)
31     {
32         ll w = M / m[i], x, y;
33         exgcd(w, m[i], x, y);
34         p = (p + a[i] * w * x % M) % M;
35     }
36     return (p % M + M) % M;
37 }
38 ll calc(ll n, ll x, ll P)
39 {
40     if (!n)
41         return 1;
42     ll s = 1;
43     for (ll i = 1; i <= P; i++)
44         if (i % x)
45             s = s * i % P;
46     s = qpow(s, n / P, P);
47     for (ll i = n / P * P + 1; i <= n; i++)
48         if (i % x)
49             s = i % P * s % P;
50     return s * calc(n / x, x, P) % P;
51 }
52 ll multilucas(ll n, ll m, ll x, ll P)
53 {
54     if (n < m)
55         return 0;
56     int cnt = 0;
57     for (ll i = n; i; i /= x)
58         cnt += i / x;
59     for (ll i = m; i; i /= x)
60         cnt -= i / x;
61     for (ll i = n - m; i; i /= x)
62         cnt -= i / x;
63     return qpow(x, cnt, P) % P * calc(n, x, P) % P * inv(calc(m, x, P), P) %
64         P * inv(calc(n - m, x, P), P) % P;
65 }
66 ll exlucas(ll n, ll m, ll P)
67 {
68     int cnt = 0;
69     ll p[20], a[20];
70     for (ll i = 2; i * i <= P; i++)
71     {
72         if (P % i == 0)
73         {
74             p[+cnt] = 1;
75             while (P % i == 0)
76                 p[cnt] = p[cnt] * i, P /= i;
77             a[cnt] = multilucas(n, m, i, p[cnt]);
78         }
79     }

```

```

80     if (P > 1)
81         p[+cnt] = P, a[cnt] = multilucas(n, m, P, P);
82     return CRT(cnt, a, p);
83 }
84 signed main()
85 {
86     cin.tie(0)->ios::sync_with_stdio(false);
87     ll n, m, p;
88     cin >> n >> m >> p;
89     cout << exlucas(n, m, p);
90     return 0;
91 }

```

## 欧拉定理

欧拉定理:  $\gcd(a, m) = 1$  则  $a^{\varphi(m)} \equiv 1 \pmod{m}$

拓展欧拉定理

$$a^b \equiv \begin{cases} a^{b \bmod \varphi(p)}, & \gcd(a, p) = 1 \\ a^b, & \gcd(a, p) \neq 1, b < \varphi(p) \pmod{p} \\ a^{b \bmod \varphi(p)+\varphi(p)}, & \gcd(a, p) \neq 1, b \geq \varphi(p) \end{cases}$$

合并为  $a^b \equiv a^{b \bmod \varphi(p)} \pmod{p}$ 。

SCNUOJ2117-求  $(n!)^{n!} \pmod{998244353}$ ,  $1 \leq n \leq 5000$ ,  $T \leq 5000$  次询问

```

1 p, phi, f, fp = 998244353, 998244352, [1, 1], [1, 1]
2 for i in range(2, 5001):
3     f.append((f[-1] * i) % p)
4     fp.append((fp[-1] * i) % phi)
5 for _ in range(int(input())):
6     n = int(input())
7     print(pow(f[n], fp[n], p))

```

洛谷P5091-求  $a^b \pmod{m}$ ,  $1 \leq a \leq 10^9$ ,  $1 \leq m \leq 10^8$ ,  $1 \leq b \leq 10^{2 \times 10^7}$

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef int ll;
4 ll a, b, m, t, phi, ans = 1;
5 char c; bool flag;
6 signed main()
7 {
8     scanf("%d%d", &a, &m);
9     t = phi = m;
10    for (ll i = 2; i * i <= m; ++i) if (t % i == 0)
11    {
12        phi = phi - phi / i;
13        while (t % i == 0) t /= i;
14    }
15    if (t > 1) phi = phi - phi / t;
16    while (!isdigit(c = getchar()));
17    for (; isdigit(c); c = getchar())

```

```

18     {
19         b = (b << 1) + (b << 3) + (c ^ '0');
20         if (b >= phi) flag = true, b %= phi;
21     }
22     if (flag) b += phi;
23     for (ans = 1; b; b >= 1, a = 1LL * a * a % m) if (b & 1)
24         ans = 1LL * ans * a % m;
25     printf("%d", ans);
26     return 0;
27 }

```

## 线性快速幂

同底数和模数，即对变化的  $x$  求  $a^x \bmod p$ ，可以时空  $O(\sqrt{p})$  预处理并  $O(1)$  计算快速幂。

对  $a^b \bmod p$ ，令  $s = \sqrt{p}$  或  $2^k$ （如  $k = 16$ ），可以将  $b$  拆成  $\lfloor \frac{b}{s} \rfloor s + (b \bmod s)$ ，线性预处理出  $a^i, a^{is}, 0 \leq i \leq s$ 。即计算  $(a^{\lfloor \frac{b}{s} \rfloor s} \times a^b \bmod s) \bmod s$ 。

LOJ 162 求  $n (\leq 5 \times 10^6)$  次  $x^a \bmod 998244352 (1 \leq a < p)$

```

1 #include <bits/stdc++.h> //LOJ 162
2 using namespace std;
3 using ll = long long;
4 const ll mn = 65536, mod = 998244352;
5 ll x, n, a, pw[mn + 10], pw2[mn + 10];
6 signed main()
7 {
8     cin.tie(0)->ios::sync_with_stdio(false);
9     cin >> x >> n;
10    pw[0] = pw2[0] = 1;
11    for (ll i = 1; i <= mn; ++i) pw[i] = pw[i - 1] * x % mod;
12    for (ll i = 1; i <= mn; ++i) pw2[i] = pw2[i - 1] * pw[mn] % mod;
13    for (ll i = 1; i <= n; ++i)
14    {
15        cin >> a;
16        cout << (pw2[a / mn] * pw[a % mn] % mod) << ' ';
17    }
18    return 0;
19 }

```

## 威尔逊定理

对素数  $p$  有  $(p - 1)! \equiv -1 \pmod{p}$

杂项：一个数对9取模等于数位和对9取模

## 欧拉函数

$\varphi(n)$  表示正整数  $[1, n]$  中与  $n$  互质的数的个数，如  $\varphi(0) = 0, \varphi(1) = 1, \varphi(8) = \text{len}\{1, 3, 5, 7\} = 4$

$$\varphi(x) = x \cdot \prod_{i=1}^n \left(1 - \frac{1}{p_i}\right), p_i \text{ 为 } x \text{ 的所有质因数}, x \in \mathbb{N}_+$$

欧拉函数性质：

1.  $p$  是素数时,  $\varphi(p) = p - 1$
2.  $p$  是素数时,  $k \in N^*, \varphi(p^k) = p^k - p^{k-1}$
3. 欧拉函数是积性函数,  $\forall a, b \in Z, (a, b) = 1$ , 有  $\varphi(ab) = \varphi(a)\varphi(b)$
4.  $\varphi(n) = \prod_{i=1}^m \varphi(p_i^{k_i})$
5. 若  $p$  是素数,  $\varphi(i \cdot p) = \begin{cases} \varphi(i) \cdot (p - 1) & , p \nmid i \\ \varphi(i) \cdot p & , p \mid i \end{cases}$
6.  $n = \sum_{d|n} \varphi(d)$

求一个数的欧拉函数:  $O(\sqrt{n})$

```
1 11 t = phi = m;
2 for (11 i = 2; i * i <= m; ++i) if (t % i == 0)
3 {
4     phi = phi - phi / i;
5     while (t % i == 0) t /= i;
6 }
7 if (t > 1) phi = phi - phi / t; //phi为所求
```

埃氏筛求  $[1, n]$  欧拉函数：

```
1 // C++ Version
2 void pre() {
3     memset(is_prime, 1, sizeof(is_prime));
4     int cnt = 0;
5     is_prime[1] = 0;
6     phi[1] = 1;
7     for (int i = 2; i <= 5000000; i++) {
8         if (is_prime[i]) {
9             prime[++cnt] = i;
10            phi[i] = i - 1;
11        }
12        for (int j = 1; j <= cnt && i * prime[j] <= 5000000; j++) {
13            is_prime[i * prime[j]] = 0;
14            if (i % prime[j])
15                phi[i * prime[j]] = phi[i] * phi[prime[j]];
16            else {
17                phi[i * prime[j]] = phi[i] * prime[j];
18                break;
19            }
20        }
21    }
22 }
```

(洛谷P3601, 求  $\sum_{i=l}^r (i - \varphi(i)) \bmod 666623333, 1 \leq l \leq r \leq 10^{12}, r - l \leq 10^6$ )

求区间欧拉函数, 复杂度为  $\sum_{p \in prime} \frac{r-l}{p} + 1 = \sqrt{r} + (r-l) \ln r$

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
```

```

4 #define MAXN 1000002
5 #define MOD 666623333
6 ll l, r, ans, cnt, prime[MAXN], a[MAXN], b[MAXN];
7 bool vis[MAXN];
8 signed main()
9 {
10     for(ll i=2;i<=MAXN;++i)
11     {
12         if (!vis[i]) prime[++cnt] = i;
13         for (ll j = i << 1; j <= MAXN; j += i) vis[j] = true;
14     }
15     scanf("%lld%lld", &l, &r);
16     for(ll i=1;i<=r;++i) a[i - 1] = b[i - 1] = i;
17     ll i = 1;
18     while (prime[i] * prime[i] <= r)
19     {
20         ll p = prime[i];
21         for (ll x = (p - 1 % p) % p; x + 1 <= r; x += p)
22         {
23             (a[x] /= p) *= p - 1;
24             while (b[x] % p == 0) b[x] /= p;
25         }
26         ++i;
27     }
28     for(ll i=0;i<=r-1;++i)
29     {
30         if (b[i] != 1) (a[i] /= b[i]) *= b[i] - 1;
31         (ans += 1 + i - a[i]) %= MOD;
32     }
33     printf("%lld", ans);
34     return 0;
35 }

```

## 中国剩余定理

CRT(Chinese Remainder Theorem) 求解如下方程最小正元  $x$  : (  $a_i$  两两互质)

$$\begin{cases} x \equiv b_1 \pmod{a_1} \\ x \equiv b_2 \pmod{a_2} \\ \vdots \\ x \equiv b_k \pmod{a_k} \end{cases}$$

设  $p = \prod a_i$ , 通解为  $x + kp$ 。方程组数为  $n$  则复杂度为  $O(n \log n)$ 。(exCRT 同)

洛谷P1495-输入不等式数  $n$  和  $a_i, b_i$  ( $a_i$  互质), 求最小  $x$

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #define mn 13
5 ll n, a[mn], b[mn], pd = 1, c, ci, z, ans;
6 void exgcd(ll a, ll b, ll &x, ll &y)
7 {
8     if (!b)

```

```

9         x = 1, y = 0;
10        else
11            exgcd(b, a % b, y, x), y -= a / b * x;
12        }
13    signed main()
14    {
15        scanf("%lld", &n);
16        for (ll i = 1; i <= n; ++i)
17        {
18            scanf("%lld%lld", &a[i], &b[i]);
19            pd *= a[i];
20        }
21        for (ll i = 1; i <= n; ++i)
22        {
23            c = pd / a[i], z = 0, ci = 0;
24            exgcd(c, a[i], ci, z);
25            ans += b[i] * c * ((ci + a[i]) % a[i]);
26        }
27        printf("%lld", ans % pd);
28        return 0;
29    }

```

**拓展中国定理:** ( $a_i$  不互质) 洛谷P4777-  $n \leq 10^5$ ,  $\gcd \leq 10^{18}$ ,  $1 \leq b_i, a_i \leq 10^{18}$

设前  $i - 1$  个不定方程的答案是  $ans$ , 前  $i - 1$  个模数的  $lcm$  为  $m$ , 有:

$ans + mx \equiv a_i \pmod{p_i}$ , 用  $exgcd$  求出新的  $x$ , 则新的  $ans + mx$  为新答案。

如果要求  $b_i x \equiv a_i \pmod{p_i}$ , 由于  $b_i$  逆元不一定存在不能拉过去。转化为求不定方程  $b_i(ans + mx) \equiv a_i \pmod{p_i}$ (那么初始值是  $an = 0, m = 1$  从  $i = 1$  开始推)

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #pragma GCC diagnostic ignored "-Wpedantic" using int128 = __int128; //忽略警告
5 #define mn 100002
6 ll n, ai[mn], bi[mn], x, y, k, m, ans, a, b, c, gcc, bg;
7 ll exgcd(ll a, ll b, ll &x, ll &y)
8 {
9     ll d = a;
10    if (!b)
11        x = 1, y = 0;
12    else
13        d = exgcd(b, a % b, y, x), y -= a / b * x;
14    return d;
15 }
16 signed main()
17 {
18     scanf("%lld", &n);
19     for (ll i = 1; i <= n; ++i)
20         scanf("%lld%lld", ai + i, bi + i);
21     m = ai[1], ans = bi[1];
22     for (ll i = 2; i <= n; ++i)
23     {
24         a = m, b = ai[i], c = (bi[i] - ans % b + b) % b;

```

```

25     gcc = exgcd(a, b, x, y), bg = b / gcc;
26     if (c % gcc != 0)
27         return printf("-1") & 0;
28     x = (__int128)x * (c / gcc) % bg;
29     ans += x * m, m *= bg, ans = (ans % m + m) % m;
30 }
31 printf("%lld", ans);
32 return 0;
33 }
```

## BSGS

求最小非负整数  $x$ ，满足  $a^x \equiv b \pmod{p}$ ,  $p$  是质数,  $O(\sqrt{p})$  (洛谷P4195)

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 using ll = long long;
4 ll bsgs(ll a, ll b, ll p)
5 { // min x of a^x%p=b
6     if (a % p == 0 && b)
7     {
8         return -1;
9     }
10    if (b == 1)
11    {
12        return 0;
13    }
14    map<ll, ll> h;
15    ll m = sqrt(p) + 1, s = 1;
16    for (ll i = 1; i <= m; ++i)
17    {
18        h[b] = i;
19        s = s * a % p, b = b * a % p;
20    }
21    a = s;
22    for (ll i = 1; i <= m; ++i)
23    {
24        if (h.count(a))
25        {
26            return i * m - h[a] + 1;
27        }
28        a = a * s % p;
29    }
30    return -1;
31 }
32 signed main()
33 {
34     ll a, b, p;
35     cin >> p >> a >> b;
36     ll ans = bsgs(a, b, p);
37     cout << (ans == -1 ? "no solution" : to_string(ans));
38     return 0;
39 }
```

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #define mn 1000700
5 #define mod 1000007
6 struct hasht
7 {
8     ll mp[mn], hsh[mn]; //避免关键字冲突map,hash
9     ll find(ll x)          //拉链法构造散列表
10    {
11        ll t = x % mod;
12        for (; mp[t] != x && mp[t] != -1; (t += 107) %= mod)
13            ;
14        return t;
15    }
16    void insert(ll x, ll i)
17    {
18        ll f = find(x);
19        mp[f] = x, hsh[f] = i;
20    }
21    bool in(ll x)
22    {
23        return mp[find(x)] == x;
24    }
25    ll get_hash(ll x)
26    {
27        return hsh[find(x)];
28    }
29    void init()
30    {
31        memset(hsh, -1, sizeof hsh), memset(mp, -1, sizeof mp);
32    }
33 } h;
34 ll a, b, p, m, s;
35 signed main()
36 {
37     scanf("%lld%lld%lld", &p, &a, &b); //a%p,b%p
38     h.init();
39     if (a % p == 0 && b)
40     {
41         return printf("no solution"), 0;
42     }
43     if (b == 1)
44     {
45         return !printf("0");
46     }
47     m = ceil(sqrt((double)p)) + 1, s = 1;
48     for (ll i = 1; i <= m; ++i)
49     {
50         h.insert(b, i);
51         s = (1LL * s * a) % p, b = (1LL * a * b) % p;
52     }
53     a = s;
54     for (ll i = 1; i <= m; ++i)
55     {

```

```

56     if (h.in(a))
57     {
58         return printf("%lld", i * m - h.get_hash(a) + 1) & 0;
59     }
60     a = (1LL * a * s) % p;
61 }
62 return printf("no solution"), 0;
63 }

```

若  $p$  不是质数, 使用 exBSGS 算法 (洛谷P4195) 复杂度  $O(\sqrt{\varphi(p)})$

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define LL long long
4 unordered_map<int, int> mp;
5 int gcd(int a, int b) { return b ? gcd(b, a % b) : a; }
6 int BSGS(int a, int n, int p, int ad)
7 {
8     mp.clear();
9     int m = ceil(sqrt(p));
10    int s = 1;
11    for (int i = 0; i < m; i++, s = 1LL * s * a % p)
12        mp[1LL * s * n % p] = i;
13    for (int i = 0, tmp = s, s = ad; i <= m; i++, s = 1LL * s * tmp % p)
14        if (mp.find(s) != mp.end())
15            if (1LL * i * m - mp[s] >= 0)
16                return 1LL * i * m - mp[s];
17    return -1;
18 }
19 int exBSGS(int a, int n, int p)
20 {
21     a %= p;
22     n %= p;
23     if (n == 1 || p == 1)
24         return 0;
25     int cnt = 0;
26     int d, ad = 1;
27     while ((d = gcd(a, p)) & 1)
28     {
29         if (n % d)
30             return -1;
31         cnt++;
32         n /= d;
33         p /= d;
34         ad = (1LL * ad * a / d) % p;
35         if (ad == n)
36             return cnt;
37     }
38     LL ans = BSGS(a, n, p, ad);
39     if (ans == -1)
40         return -1;
41     return ans + cnt;
42 }
43 signed main()
44 {

```

```

45     int a = read(), p = read(), n = read();
46     while (a || p || n)
47     {
48         int ans = exBSGS(a, n, p);
49         if (~ans)
50             printf("%d\n", ans);
51         else
52             puts("No solution");
53         a = read();
54         p = read();
55         n = read();
56     }
57     return 0;
58 }
```

BSGS 的结构是一条链(exBSGS cnt 部分)加一条环。可以用 BSGS 求环长的依据：有环必然存在  $a^p \equiv a \pmod{p}$  即  $a^{p-1} \equiv 1 \pmod{p}$ ，故环长为  $p - 1$ 。

## 计算几何

三角函数：使用弧度制，弧度制 `deg` 跟角度制 `arg` 的转换公式：( $\pi = 180^\circ$ )

$$1 = \frac{deg}{180^\circ} = \frac{arg}{\pi} \Rightarrow \begin{cases} arg = \frac{\pi \cdot deg}{180^\circ} \\ deg = \frac{180^\circ \cdot arg}{\pi} \end{cases}$$

三角函数是 `sin, cos, tan`，反三角函数是 `asin, acos, atan, atan2(y, x)`，其中 `atan2(y, x) = atan(y/x)`。反三角函数求出的角度值域范围是：`asin, atan` 为  $[-\frac{\pi}{2}, \frac{\pi}{2}]$ ，`acos` 为  $[0, \pi]$ ，`atan2` 为  $[-\pi, \pi]$ 。

注意这里 `atan2` 的含义是：向量  $(x, y)$  与  $x$  正半轴夹角的大小，即从  $X$  轴逆时针旋转的角度，如果为负数就表示顺时针旋转的角度。与 `atan` 是本质不同的函数

## 向量

向量点乘，有结论： $\vec{a} \cdot \vec{b}$  的符号大于 0 夹角在  $[0, 90^\circ]$ ，等于 0 为  $90^\circ$ ；小于 0 为  $(90^\circ, 180^\circ]$ 。点乘的几何意义是  $\vec{a}$  长度与  $\vec{b}$  投影到  $\vec{a}$  的长度的乘积或  $\vec{b}$  长度与  $\vec{a}$  投影到  $\vec{b}$  的长度的乘积。即  $\vec{a} \cdot \vec{b} = |\vec{a}| \cdot |\vec{b}| \cos \langle \vec{a}, \vec{b} \rangle$ 。如果是正的，说明投影后朝向一致，否则说明投影后朝向不一致

向量  $\vec{a}, \vec{b}$  叉乘得到一个向量  $\vec{c}$ ，其方向垂直这两向量形成的平面，如果  $\vec{b}$  满足  $\vec{a}$  经由  $180^\circ$  内的逆时针旋转可以与其平行，那么  $\vec{c}$  的竖坐标是正的；如果是  $180^\circ$  内的顺时针，那么是负的。可以用右手定则表示：当右手的四指从  $\vec{a}$  以不超过  $180^\circ$  的转角转向  $\vec{b}$  时，竖起的大拇指指向是  $\vec{a} \times \vec{b}$  的方向(向上正，向下负)，叉乘满足： $(\lambda \vec{a}) \times \vec{b} = \vec{a} \times (\lambda \vec{b}) = \lambda (\vec{a} \times \vec{b})$ ， $(\vec{a} + \vec{b}) \times \vec{c} = \vec{a} \times \vec{c} + \vec{b} \times \vec{c}$

判断向量共线(平行)：两非零向量  $\vec{a}, \vec{b}$  共线当且仅当存在唯一实数  $\lambda$ ，使得  $\vec{b} = \lambda \vec{a}$ ，或  $|\vec{a} \cdot \vec{b}| = |\vec{a}| |\vec{b}|$  (也可以用  $\vec{a} \times \vec{b} = 0$ )

判断向量垂直： $\vec{a} \cdot \vec{b} = 0$

向量与原点的夹角： $\theta = \arctan \frac{y}{x}$ ，由角和长度  $l$  逆求坐标  $(l \cos \theta, l \sin \theta)$

两向量的夹角:  $\cos \langle \vec{a}, \vec{b} \rangle = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| |\vec{b}|}$  , 再来个反三角函数  $\arccos$  即可。

若  $A, B, C$  三点共线: 那么对任意点  $O$  有  $\vec{OB} = \lambda \vec{OA} + (1 - \lambda) \vec{OC}$

向量旋转公式: 设向量  $\vec{a} = (x, y)$  , 倾角为  $\alpha$  , 长度为  $l = \sqrt{x^2 + y^2}$  , 即  $\vec{a} = (l \cos \alpha, l \sin \alpha)$  , 逆时针旋转  $\beta$  , 得到  $\vec{a}' = (l \cos(\alpha + \beta), l \sin(\alpha + \beta))$  。展开化简得:

$$\begin{aligned}\vec{a}' &= (l(\cos \alpha \cos \beta - \sin \alpha \sin \beta), l(\sin \alpha \cos \beta + \cos \beta \sin \alpha)) \\ &= (x \cos \beta - y \sin \beta, x \sin \beta + y \cos \beta)\end{aligned}$$

依次取  $\beta = 90^\circ, 180^\circ, 270^\circ$  , 对  $(x, y)$  得到的分别为:  $(-y, x), (-x, -y), (y, -x)$

```

1 #define EPS (1e-10)
2 #define equals(a, b) (fabs((a) - (b)) < EPS)
3 class Point
4 {
5 public:
6     double x, y;
7     Point(double x = 0, double y = 0) : x(x), y(y) {}
8     Point operator+(const Point &p) const { return Point(x + p.x, y + p.y); }
9     Point operator-(const Point &p) const { return Point(x - p.x, y - p.y); }
10    Point operator*(const double &a) const { return Point(x * a, y * a); }
11    Point operator/(const double &a) const { return Point(x / a, y / a); }
12    double norm() const { return x * x + y * y; }
13    double abs() const { return sqrt(norm()); }
14    bool operator<(const Point &p) const
15    {
16        return x != p.x ? x < p.x : y < p.y;
17    }
18    bool operator==(const Point &p) const
19    {
20        return fabs(x - p.x) < EPS && fabs(y - p.y) < EPS;
21    }
22};
23 typedef Point Vector;
24 double dot(const Point &a, const Point &b)
25 {
26     return a.x * b.x + a.y * b.y;
27 }
28 double cross(const Point &a, const Point &b)
29 {
30     return a.x * b.y - a.y * b.x;
31 }
32 bool isParallel(const Vector &a, const Vector &b)
33 {
34     return equals(cross(a, b), 0.0);
35 }
36 bool isOrthogonal(const Vector &a, const Vector &b)
37 {
38     return equals(dot(a, b), 0.0);
39 }
40 double getDistance(const Point &a, const Point &b)

```

```

41 {
42     return (a - b).abs();
43 }
44 //极坐标互换
45 double arg(const vector &p) { return atan2(p.y, p.x); }
46 Vector polar(const double &a, const double &r)
47 {
48     return Point(cos(r) * a, sin(r) * a);
49 }

```

## 线段

### 点到直线距离

$$\text{点 } P \text{ 到直线 } AB \text{ 的距离 } |PD| = \frac{|\vec{AB} \times \vec{AP}|}{|AB|}$$

```

1 double dis_lp(const Point &a, const Point &b, const Point &p)
2 {
3     return abs(cross(b - a, p - a)) / (b - a).abs();
4 }

```

点到线段的距离：

```

1 double dis_sp(const Point &a, const Point &b, const Point &p)
2 {
3     if (dot(b - a, p - a) < 0.0)
4     {
5         return (p - a).abs();
6     }
7     if (dot(a - b, p - b) < 0.0)
8     {
9         return (p - b).abs();
10    }
11    return dis_lp(a, b, p);
12 }

```

### 线段相交判定

判断线段  $AB, CD$  是否相交：

```

1 bool isIntersect(const Point &a, const Point &b, const Point &c, const Point
&d)
2 {
3     return cross(b - a, c - a) * cross(b - a, d - a) <= 0 && cross(d - c, a -
c) * cross(d - c, b - c) <= 0;
4 } // 条件：A,B,C,D 不共线

```

```

1 11 f(const Point &a, const Point &b)
2 {

```

```

3     if (cross(a, b) > eps)
4     {
5         return 1; //逆时针
6     }
7     if (cross(a, b) < -eps)
8     {
9         return -1; //顺时针
10    }
11    if (dot(a, b) < -eps)
12    {
13        return 2; // P在AB左方
14    }
15    if (a.abs() < b.abs())
16    {
17        return -2; // P在AB右方
18    }
19    return 0; // P在AB内部
20 }
21 bool isIntersect(const Point &a, const Point &b, const Point &c, const Point
&d)
22 {
23     return f(b - a, c - a) * f(b - a, d - a) <= 0 && f(d - c, a - c) * f(d -
c, b - c) <= 0;
24 }

```

直线  $AB$  与线段  $CD$ ，那么只需要判断  $C, D$  是否在  $AB$  两端即可，不必反过来再判断一次，即只需要判断：

$$f(\vec{AB}, \vec{AC}) \cdot f(\vec{AB}, \vec{AD}) \leq 0$$

线段  $AB, CD$  的距离，如果两线段相交，距离为 0，否则分别计算  $A, B$  到线段  $CD$  的  $C, D$  到线段  $AB$  的距离，取四者最小值即可

## 直线交点

直线  $AB, CD$  交于点  $P$ ，则  $\vec{OP} = \vec{OC} + \frac{\vec{CA} \times \vec{AB}}{\vec{CD} \times \vec{AB}} \cdot \vec{OD}$

```

1 Point intersect(const Point &a, const Point &b, const Point &c, const Point
&d)
2 {
3     return c + (d - c) * (cross(a - c, b - a) / cross(d - c, b - a));
4 }

```

线段  $AB, CD$  交点，那么可以先判断  $AB, CD$  是否相交，如果是的话同直线交点的处理方法，如果不相交，输出不存在

## 点在直线投影

点  $P$  在直线  $AB$  上投影点  $C$  满足  $\vec{OC} = \vec{OA} + \frac{\vec{AB} \cdot \vec{AP}}{|\vec{AB}|^2} \cdot \vec{AB}$

```
1 Point project(const Point &a, const Point &b, const Point &p)
2 {
3     return a + (b - a) * (dot(b - a, p - a) / (b - a).norm());
4 }
```

$P$  关于  $AB$  对称的点  $P'$ ，可以先求投影点  $P_0$ ，那么根据  $\vec{PP'} = 2\vec{PP}_0$ ，可知  $\vec{OP'} = \vec{OP} + \vec{PP'} = \vec{OP} + 2\vec{PP}_0$

求圆  $P$  与直线  $AB$  的交点  $C, D$ ，求出  $P$  在  $AB$  的投影点及距离  $|PE|$ ，则：

$$\vec{OC} = \vec{OE} + \vec{EC} = \vec{OE} - \frac{\sqrt{r^2 - |PE|^2}}{|\vec{AB}|} \vec{AB}$$
$$\vec{OD} = \vec{OE} - \vec{EC} = \vec{OE} + \frac{\sqrt{r^2 - |PE|^2}}{|\vec{AB}|} \vec{AB}$$

求两圆  $P, Q$  的交点：由余弦定理，求出夹角

$\cos \theta = \cos \angle APQ = \cos \angle BPQ = \frac{|PQ|^2 + r_1^2 - r_2^2}{2r_1|PQ|}$ ，可得角度  $\theta$ 。根据反三角函数可知  $\vec{OP}$

的夹角  $\alpha$ ，则  $\vec{PA} = (r_1 \cos(\alpha + \theta), r_1 \sin(\alpha + \theta))$  且  $\vec{PB} = (r_1 \cos(\alpha - \theta), r_1 \sin(\alpha - \theta))$ ，那么  $\vec{OA} = \vec{OP} + \vec{PA}, \vec{OB} = \vec{OP} + \vec{PB}$

两圆是否相交(等)：圆心距离  $d$  小于等于  $r_1 + r_2$  即  $d^2 \leq (r_1 + r_2)^2$

## 多边形

由在同一平面且不在同一直线上的三条或三条以上的线段首尾顺次连结且不相交所组成的封闭图形叫做多边形。又称简单多边形。不满足“不相交”的就是复杂多边形。

凸多边形是没有任何一个内角为优角(范围在区间  $(180^\circ, 360^\circ)$  的角)的多边形。凹多边形则至少有一个优角。凸多边形以任意一条边延长为直线，其他所有边都必然在直线的同一侧。凹多边形则存在不在同一侧的情况。凸多边形上的点或内部点间连线位于多边形都在多边形内部或边上。凹多边形存在两个点的连线不全在多边形内部

多边形的性质：

- 无论凹凸， $n$  边形的内角和是  $180^\circ \cdot (n - 2)$
- 凸多边形外角和是  $360^\circ$
- $n$  边形的对角线数量是  $\frac{n(n - 3)}{2}$

三角形的性质：

- 设  $p = \frac{1}{2}(a + b + c)$ ,  $R$  是外接圆半径， $r$  是内接圆半径，则面积公式：
  - $S = \sqrt{p(p - a)(p - b)(p - c)} = \frac{abc}{4R} = rp$
- 三角形重心坐标是三点坐标的平均值。

- 三角形外角和是  $360^\circ$ ，等于不相邻内角之和。
- 三角形中线将三角形划分为面积相等的两部分。
- 直角三角形斜边中线等于斜边一半。
- 三线合一：等腰三角形顶角平分线、底边中线、底边高是同一条直线

**多边形面积公式**：(点按逆时针排序)

$$S = \frac{1}{2} \sum_{i=1}^n \vec{OA}_i \times \vec{OA}_{i+1}$$

**PIP 问题**：点是否在多边形内部(或边上)问题。

- **光线投射算法**，从该点出发作射线(注意不是直线)，如果与奇数个点相交在多边形内，偶数个点相交在多边形外。实现上，可以作平行  $x$  轴的射线，枚举每条线段，如果线段两端点坐标值都小于该点，那么忽略该线段，否则套用上文直线与线段相交的模板进行判断即可。特别注意，如果相交在端点，那么应当算相交了两次(相交时特判横坐标是否与其中一个端点相同即可)。
- **回转数算法**求这个问题。计算该点与每相邻两个顶点的夹角弧度制(取范围  $[-\pi, \pi]$ )，然后将其求和，判断是否为  $0$ ，是就在多边形外。实现上， $O(n)$  枚举相邻边，利用

$$\angle \vec{a}, \vec{b} = \arccos \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| |\vec{b}|} \text{ 求角，累加，最后与 } 0 \text{ 判断即可(注意浮点误差)}$$

是否位于多边形边上，可以通过检查  $p$  与相邻两端点的向量( $p$ 为起点)是否在同一直线且方向相反(相同是可以内包的)

```

1 int contains(Polygon g, Point p) //伪代码
2 {
3     int n = g.size();
4     bool x = false;
5     for(int i=0;i<n;i++)
6     {
7         Point a = g[i] - p, b = g[(i+1)%n] - p;
8         if(abs(cross(a,b))<EPS&&dot(a,b)<EPS)
9             return 1;
10        if(a.y<b.y) swap(a,b);
11        if(a.y<EPS && EPS<b.y && cross(a,b)>EPS)
12            x=!x;
13    }
14    return (x?2:0);
15 }
```

p3454-任意多边形的对称轴数：边长平方-两边差积(内角)交替组成序列，化环为二倍链，使用字符串哈希求每个长为  $2n$  的序列区间是否是回文串。(/KMP/mahacher)

## 极角排序

如果极角大小值相同，可以规定按照离原点的远近排序(越近越小)

atan2 参考代码：(范围是  $[-\pi, \pi]$ )

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
```

```

4  typedef double db;
5  #define sc(x) scanf("%lld", &x)
6  #define cp const point &
7  struct point
8  {
9      db x, y;
10     point(db a = 0, db b = 0) : x(a), y(b) {}
11     point operator-(cp r) const { return point(x - r.x, y - r.y); }
12     db ang() const { return atan2(y, x); }
13     db norm() const { return x * x + y * y; }
14 } p[1010], p0;
15 db cross(cp a, cp b) { return a.x * b.y - a.y * b.x; }
16 bool operator<(cp a, cp b)
17 {
18     point p1 = (a - p[1]), p2 = (b - p[1]);
19     return p1.ang() != p2.ang() ? p1.ang() < p2.ang() : p1.norm() <
p2.norm();
20 }
21 int n, b;
22 signed main()
23 {
24     sc(n);
25     for (int i = 1; i <= n; ++i)
26     {
27         scanf("%lf%lf", &p[i].x, &p[i].y);
28     }
29     sort(p + 2, p + 1 + n); //假设以第一个点为坐标原点
30     for (int i = 1; i <= n; ++i)
31     {
32         printf("%lf, %lf) %lf\n", p[i].x, p[i].y, (p[i] - p[1]).ang());
33     }
34     return 0;
35 }

```

如果范围要求  $[0, 2\pi]$  的排序, 那么更改 `ang` 函数的定义:

```

1 db ang() const
2 {
3     db v = atan2(y, x);
4     return v < 0 ? v + 2 * acos(-1) : v;
5 }

```

叉乘(极角取值为  $[0, 2\pi]$  , 最左下为原点):

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef Long Long ll;
4 typedef double db;
5 #define sc(x) scanf("%lld", &x)
6 #define mn 100010
7 #define cp const point &
8 int n, ss;
9 struct point

```

```

10 {
11     db x, y;
12     point(db a = 0, db b = 0) : x(a), y(b) {}
13     point operator-(cp r) const { return point(x - r.x, y - r.y); }
14     db norm() const { return x * x + y * y; }
15 } p[mn];
16 db cross(cp a, cp b) { return a.x * b.y - a.y * b.x; }
17 bool operator<(cp a, cp b)
18 {
19     db v = cross(a - p[1], b - p[1]);
20     return v > 0 || (v == 0 && (a - p[1]).norm() < (b - p[1]).norm());
21 }
22 signed main()
23 {
24     sc(n);
25     for (ll i = 1; i <= n; ++i)
26     {
27         scanf("%lf%lf", &p[i].x, &p[i].y);
28         if (p[i].y < p[1].y || (p[i].y == p[1].y && p[i].x < p[1].x))
29         {
30             swap(p[i], p[1]);
31         }
32     }
33     sort(p + 2, p + 1 + n);
34     for (ll i = 1; i <= n; ++i)
35     {
36         printf("(%.1f, %.1f)\n", p[i].x, p[i].y);
37     }
38     return 0;
39 }

```

任意点为原点：

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 typedef double db;
5 #define sc(x) scanf("%lld", &x)
6 #define mn 100010
7 #define cp const point &
8 ll n, ss;
9 struct point
10 {
11     db x, y;
12     point(db a = 0, db b = 0) : x(a), y(b) {}
13     point operator-(cp r) const { return point(x - r.x, y - r.y); }
14     db norm() const { return x * x + y * y; }
15     ll quadrant() const
16     {
17         return 1 * (x > 0 && y >= 0) + 2 * (x <= 0 && y > 0) + 3 * (x < 0 &&
18         y <= 0) + 4 * (x >= 0 && y < 0);
19     }
20 } p[mn], s[mn], p0;
21 db cross(cp a, cp b) { return a.x * b.y - a.y * b.x; }
22 bool operator<(cp a, cp b)

```

```

22 {
23     if ((a - p0).quadrant() != (b - p0).quadrant())
24     {
25         return (a - p0).quadrant() < (b - p0).quadrant();
26     }
27     db v = cross(a - p0, b - p0);
28     return v > 0 || (v == 0 && (a - p0).norm() < (b - p0).norm());
29 }
30 signed main()
31 {
32     sc(n);
33     for (ll i = 1; i <= n; ++i)
34     {
35         scanf("%lf%lf", &p[i].x, &p[i].y);
36     }
37     sort(p + 1, p + 1 + n);
38     for (ll i = 1; i <= n; ++i)
39     {
40         printf("(%.1f, %.1f)\n", p[i].x, p[i].y);
41     }
42     return 0;
43 }

```

## 凸包

平面上能包含所有给定点的周长最小的凸多边形是凸包。(注意不是面积最小)。上凸壳是圆的第一第二象限逆时针方向朝向(含  $x$  负半轴), 其余是下凸壳。

### Graham

Graham算法: (洛谷P2742)

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 typedef double db;
5 #define sc(x) scanf("%lld", &x)
6 #define mn 100010
7 #define cp const point &
8 ll n, ss;
9 struct point
10 {
11     db x, y;
12     point(db a = 0, db b = 0) : x(a), y(b) {}
13     point operator+(cp r) const { return point(x + r.x, y + r.y); }
14     point operator-(cp r) const { return point(x - r.x, y - r.y); }
15     db norm() const { return x * x + y * y; }
16 } p[mn], s[mn];
17 db cross(cp a, cp b) { return a.x * b.y - a.y * b.x; }
18 bool operator<(cp a, cp b)
19 {
20     db v = cross(a - p[1], b - p[1]);
21     return v > 0 || (v == 0 && a.norm() < b.norm());
22 }

```

```

23 signed main()
24 {
25     sc(n);
26     for (ll i = 1; i <= n; ++i)
27     {
28         scanf("%lf%lf", &p[i].x, &p[i].y);
29         if (p[i].y < p[1].y || (p[i].y == p[1].y && p[i].x < p[1].x))
30         {
31             swap(p[i], p[1]);
32         }
33     }
34     sort(p + 2, p + 1 + n);
35     s[ss] = p[1];
36     for (ll i = 2; i <= n; ++i)
37     {
38         while (ss > 1 && cross(s[ss] - s[ss - 1], p[i] - s[ss]) < 0)
39         { //根据题目要求选择 < 或 <=
40             --ss;
41         }
42         s[ss] = p[i];
43     }
44     for (ll i = 1; i <= ss; ++i) //输出各点
45     {
46         printf("%lf %lf\n", s[i].x, s[i].y);
47     }
48     return 0;
49 }

```

## Andrew

Andrew算法: (洛谷P2742)

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 typedef double db;
5 #define sc(x) scanf("%lld", &x)
6 #define mn 100010
7 #define cp const point &
8 ll n, ss, top;
9 struct point
10 {
11     db x, y;
12     point(db a = 0, db b = 0) : x(a), y(b) {}
13     point operator+(cp r) const { return point(x + r.x, y + r.y); }
14     point operator-(cp r) const { return point(x - r.x, y - r.y); }
15     db norm() const { return x * x + y * y; }
16 } p[mn], s[mn * 2];
17 db cross(cp a, cp b) { return a.x * b.y - a.y * b.x; }
18 bool operator<(cp a, cp b)
19 {
20     return a.x != b.x ? a.x < b.x : a.y < b.y;
21 }
22 void f(ll i)
23 {

```

```

24     while (ss > top && cross(s[ss] - s[ss - 1], p[i] - s[ss]) < 0)
25     { //根据题目要求选择 < 或 <=
26         --ss;
27     }
28     s[ss] = p[i];
29 }
30 signed main()
31 {
32     sc(n);
33     for (ll i = 1; i <= n; ++i)
34     {
35         scanf("%lf%lf", &p[i].x, &p[i].y);
36     }
37     sort(p + 1, p + 1 + n);
38     top = 1, s[ss = 1] = p[1];
39     for (ll i = 2; i <= n; ++i)
40     {
41         f(i);
42     }
43     top = ss;
44     for (ll i = n - 1; i >= 1; --i)
45     {
46         f(i);
47     }
48     for (ll i = 1; i < ss; ++i) //输出各点
49     {
50         printf("%lf %lf\n", s[i].x, s[i].y);
51     }
52     return 0;
53 }

```

## 直径

旋转卡壳模板-求凸包直径: (洛谷P1452)

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #define sc(x) scanf("%lld", &x)
5 #define mn 50010
6 #define cp const point &
7 struct point
8 {
9     ll x, y;
10    point(ll a = 0, ll b = 0) : x(a), y(b) {}
11    point operator-(cp r) const { return point(x - r.x, y - r.y); }
12    ll norm() const { return x * x + y * y; }
13 } p[mn], s[mn];
14 ll cross(cp a, cp b) { return a.x * b.y - a.y * b.x; }
15 bool operator<(cp a, cp b)
16 {
17     ll v = cross(a - p[1], b - p[1]);
18     return v > 0 || (v == 0 && (a - p[1]).norm() < (b - p[1]).norm());
19 }
20 ll area(cp a, cp b, cp c) { return cross(b - a, c - a); }

```

```

21  ll n, ss, ans;
22  signed main()
23  {
24      sc(n);
25      for (ll i = 1; i <= n; ++i)
26      {
27          sc(p[i].x), sc(p[i].y);
28          if (p[i].x < p[1].x || (p[i].x == p[1].x && p[i].y < p[1].y))
29          {
30              swap(p[i], p[1]);
31          }
32      }
33      sort(p + 2, p + 1 + n);
34      for (ll i = 1; i <= n; ++i)
35      {
36          while (ss > 1 && cross(s[ss] - s[ss - 1], p[i] - s[ss]) <= 0)
37          {
38              --ss;
39          }
40          s[ss + 1] = s[1];
41          if (ss == 2) // not n==2
42          {
43              printf("%lld", (s[2] - s[1]).norm());
44              return 0;
45          }
46          for (ll i = 1, j = 3; i <= ss; ++i)
47          {
48              while (area(s[i], s[i + 1], s[j]) <= area(s[i], s[i + 1], s[j % ss + 1]))
49              {
50                  j = j % ss + 1;
51              }
52              ans = max(ans, max((s[j] - s[i]).norm(), (s[j] - s[i + 1]).norm()));
53          }
54          printf("%lld", ans);
55      }
56      return 0;
57  }

```

如果每次以某个端点为原点进行极角排序，再枚举线段第二端点，配合单调栈，可以用  $O(n^2 \log n)$  实现枚举(但是处理概率连乘积乘除特别是 1 (对应概率为 0) 比较麻烦)，暂时没有什么好的实现方法

## 点与凸包相交

取其中的一个点，他和其他点可以组成  $n-2$  个三角形，利用二分判断差积，判断他是在当前三角形内，还是在三角形左边，还是在三角形右边，或者是三角形外 (UVaLive 7281)

或者用另一个实现(下文)。

```

1  bool check(Point A, Point*p, int n) //判断点在凸包内模板 O(logn)
2  {
3      int l=1, r=n-2, mid;
4      while(l<=r)
5      {
6          mid=(l+r)>>1;

```

```

7     double a1=cross(p[mid]-p[0],A-p[0]);
8     double a2=cross(p[mid+1]-p[0],A-p[0]);
9     if(a1>=0&&a2<=0)
10    {
11        if(cross(p[mid+1]-p[mid],A-p[mid])>=0) return true;
12        return false;
13    }
14    else if(a1<0)
15    {
16        r=mid-1;
17    }
18    else
19    {
20        l=mid+1;
21    }
22 }
23 return false;
24 }
```

## 两凸包相交

点集  $A, B$  的闵可夫斯基和(Minkowski sum)是  $A + B = \{a + b | a \in A, b \in B\}$ , 即对  $a$  每个点位置放一个  $B$  的并。

两凸包的闵可夫斯基和是凸包, 可以  $O(n \log n)$  求出。使用滑窗拼出然后复杂度在凸包重排。

应用: 判断凸多边形是否相交。(如果不凸, 枚举一多边形每点 PIP  $O(nm)$ )

判定:  $\exists a \in A, b \in B, a = b$  即  $a - b = 0$ , 故设  $B = -B$ , 求  $A + B$  的闵可夫斯基和, 然后判断原点( $B$  左下角)是否在凸包内。

p4557-给定凸包(不全共线, 点不同)  $A, B(n, m \leq 10^5)$ , 有  $q \leq 10^5$  个询问, 每次将  $B$  每个点平移  $d(dx, dy)$ , 问移动后  $A, B$  交不交。

$A = B + d$  可以转化为  $A - B = d$ , 所以对每个  $d$  判断在不在闵可夫斯基和凸包即可。

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 using ll = long long;
4 const ll N=1e5+10;
5 struct Node
6 {
7     ll x,y;
8     Node operator - (Node A) {return (Node){x-A.x,y-A.y};}
9     Node operator + (Node A) {return (Node){x+A.x,y+A.y};}
10    ll operator * (Node A) const {return x*A.y-y*A.x;}
11    ll len() const {return x*x+y*y;}
12 }A[N],C1[N],C2[N],s1[N],s2[N],bs;
13 ll cmp1(const Node&A,const Node&B) {return A.y<B.y||(A.y==B.y&&A.x<B.x);}
14 ll cmp2(const Node&A,const Node&B) {return A*x>0||(A*x==0&&A.len()<B.len());}
15 ll n,m,sta[N],top,q,tot;
16 void Convex(Node *A,ll &n)
17 {
18     sort(A+1,A+n+1,cmp1);
19     bs=A[1];sta[top=1]=1;
20     for(ll i=1;i<=n;i++) A[i]=A[i]-bs;
```

```

21     sort(A+2,A+n+1,cmp2);
22     for(11 i=2;i<=n;sta[++top]=i,i++)
23         while(top>=2&&(A[i]-A[sta[top-1]])*(A[sta[top]]-A[sta[top-1]])>=0)
24             top--;
25         for(11 i=1;i<=top;i++) A[i]=A[sta[i]]+bs;
26     n=top;A[n+1]=A[1];
27 }
28 void Minkowski()
29 {
30     for(11 i=1;i<n;i++) s1[i]=c1[i+1]-c1[i];s1[n]=c1[1]-c1[n];
31     for(11 i=1;i<m;i++) s2[i]=c2[i+1]-c2[i];s2[m]=c2[1]-c2[m];
32     A[tot=1]=c1[1]+c2[1];
33     11 p1=1,p2=1;
34     while(p1<=n&&p2<=m) ++tot,A[tot]=A[tot-1]+(s1[p1]*s2[p2]>=0?
35         s1[p1++]:s2[p2++]);
36     while(p1<=n) ++tot,A[tot]=A[tot-1]+s1[p1++];
37     while(p2<=m) ++tot,A[tot]=A[tot-1]+s2[p2++];
38 }
39 11 in(Node a)
40 {
41     if(a*A[1]>0||A[tot]*a>0) return 0;
42     11 ps=lower_bound(A+1,A+tot+1,a,cmp2)-A-1;
43     return (a-A[ps])*(A[ps%tot+1]-A[ps])<=0;
44 }
45 int main()
46 {
47     cin>>n>>m>>q;
48     for(11 i=1;i<=n;i++)
49         scanf("%lld%lld",&c1[i].x,&c1[i].y);
50     Convex(c1,n);
51     for(11 i=1;i<=m;i++)
52     {
53         scanf("%lld%lld",&c2[i].x,&c2[i].y);
54         c2[i].x=-c2[i].x;c2[i].y=-c2[i].y;
55     }
56     Convex(c2,m);
57     Minkowski();
58     Convex(A,tot);
59     bs=A[1];for(11 i=tot;i>=1;i--) A[i]=A[i]-A[1];
60     while(q--)
61     {
62         scanf("%lld%lld",&A[0].x,&A[0].y);
63         printf("%lld\n",in(A[0]-bs));
64     }
65     return 0;
66 }

```

## 扫描线

矩形面积并

洛谷P5490-输入  $n$  和矩形左下角、右上角，输出  $n$  矩形并集总面积

复杂度  $O(n \log n)$ ，先离散化，再线段树维护

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #define mn 300010
5 unordered_map<ll, ll> h;
6 struct scline
7 {
8     ll x, ay, by, c;
9     bool operator<(const scline &p) const { return x < p.x; }
10 } sc[mn];
11 ll ax, ay, bx, by, n, xs[mn], xss, c[mn << 2], lp[mn << 2], rp[mn << 2],
12 v[mn << 2], lf, rf, k, ans;
13 #define lfs h << 1
14 #define rfs h << 1 | 1
15 #define mkcf ll cf = (lf + rf) >> 1
16 void init(ll lf, ll rf, ll h)
17 {
18     lp[h] = lf, rp[h] = rf;
19     if (lf != rf)
20     {
21         mkcf;
22         init(lf, cf, lfs);
23         init(cf + 1, rf, rfs);
24     }
25 }
26 void add(ll h)
27 {
28     if (rf < lp[h] || lf > rp[h])
29         return;
30     if (lf <= lp[h] && rp[h] <= rf)
31         c[h] += k;
32     else
33         add(lfs), add(rfs);
34     if (c[h])
35         v[h] = xs[rp[h] + 1] - xs[lp[h]];
36     else
37         v[h] = v[lfs] + v[rfs];
38 }
39 signed main()
40 {
41     scanf("%lld", &n);
42     for (ll i = 1; i <= n; ++i)
43     {
44         scanf("%lld%lld%lld%lld", &ax, &ay, &bx, &by);
45         sc[i] = {ax, ay, by, 1};
46         sc[i + n] = {bx, ay, by, -1};
47         xs[+xss] = by;
48         xs[+xss] = ay;
49     }
50     n <= 1;
51     sort(sc + 1, sc + 1 + n);
52     sort(xs + 1, xs + 1 + n);
53     xss = unique(xs + 1, xs + 1 + n) - xs - 1;
54     for (ll i = 1; i <= xss; ++i)
55         h[xs[i]] = i;

```

```

55     init(1, n, 1);
56     for (ll i = 1; i < n; ++i)
57     {
58         lf = h[sc[i].ay], rf = h[sc[i].by] - 1, k = sc[i].c;
59         add(1);
60         ans += v[1] * (sc[i + 1].x - sc[i].x);
61     }
62     printf("%lld", ans);
63     return 0;
64 }

```

### 矩形周长并 (洛谷P1856)

```

1 #include <bits/stdc++.h>
2 #define mn 10002
3 using namespace std;
4 typedef long long ll;
5 ll n, ax, ay, bx, by, xs[mn], m, bf, ans;
6 struct scline
7 {
8     ll y, ax, bx, c;
9     bool inline operator < (const scline& b) const
10    {
11        if(y==b.y) return c>b.c;
12        return y<b.y;
13    }
14} sc[mn];
15 struct segtr
16 {
17     ll c, s, l, r, len;
18     bool lc, rc;
19} t[mn<<2];
20 #define lfi (i<<1)
21 #define rfi (i<<1|1)
22 void init(ll i, ll lf, ll rf)
23 {
24     t[i].l=lf, t[i].r=rf;
25     if(lf!=rf)
26     {
27         ll cf=lf+rf>>1;
28         init(lfi, lf, cf);
29         init(rfi, cf+1, rf);
30     }
31 }
32 void pushup(ll i)
33 {
34     ll l=t[i].l, r=t[i].r;
35     if(t[i].s)
36     {
37         t[i].len=xs[r+1]-xs[l];
38         t[i].lc=t[i].rc=true;
39         t[i].c=1;
40         return;
41     }
42 }

```

```

41     }
42     t[i].len=t[lfi].len+t[rfi].len;
43     t[i].lc=t[lfi].lc,t[i].rc=t[rfi].rc;
44     t[i].c=t[lfi].c+t[rfi].c;
45     if(t[lfi].rc&&t[rfi].lc) t[i].c--;
46 }
47 void add(ll i, ll lf, ll rf, ll ci)
48 {
49     ll lt=xs[t[i].l],rt=xs[t[i].r+1];
50     if(lt>=rf||rt<=lf) return;
51     if(lt>=lf&&rt<=rf)
52     {
53         t[i].s+=ci;
54         pushup(i);
55         return;
56     }
57     add(lfi,lf,rf,ci);
58     add(rfi,lf,rf,ci);
59     pushup(i);
60 }
61 signed main()
62 {
63     scanf("%lld",&n);
64     for(ll i=1;i<=n;++i)
65     {
66         scanf("%lld%lld%lld%lld",&ax,&ay,&bx,&by);
67         sc[i]={ay,ax,bx,1};
68         sc[i+n]={by,ax,bx,-1};
69         xs[i]=ax;
70         xs[i+n]=bx;
71     }
72     n<=1;
73     sort(sc+1,sc+1+n);
74     sort(xs+1,xs+1+n);
75     m=unique(xs+1,xs+1+n)-xs-1;
76     init(1,1,m-1);
77     for(ll i=1;i<n;++i)
78     {
79         add(1,sc[i].ax,sc[i].bx,sc[i].c);
80         ans+=abs(bf-t[1].len); bf=t[1].len;
81         ans+=2*t[1].c*(sc[i+1].y-sc[i].y);
82     }
83     printf("%lld",ans+sc[n].bx-sc[n].ax);
84     return 0;
85 }

```

## 杂项

### Pick定理

顶点为整点的简单多边形，面积  $A$ ，内部格点数目  $i$ ，边上格点数目  $b$  满足： $A = i + \frac{b}{2} - 1$ ，其中  $1$  是多边形的欧拉特征数。

边上格点数目，对于每一条边，其贡献量为： $\gcd(\Delta x, \Delta y)$  (闭环之后刚好每个顶点都被算了一次，所以不需要额外  $+1$ ，单边时就需要)，于是可以根据公式求出  $i$

如果是平行四边形格点，同上；如果是三角形格点， $A = 2i + b - 2$ ；高维推广 Ehrhart 多项式

欧拉公式： $V - E + F = 2$ ， $V, E, F$  是简单几何体顶点数、边数、面数

POJ1265-有多组测试，每组测试  $n (3 \leq n \leq 100)$  个点，接下来输入若干坐标偏量，代表当前点离上一个点的坐标位移。求这个多边形的内部包含格点整点数、边上包含的格点整点数和多边形面积

```
1 #include <cstdio>
2 using namespace std;
3 typedef long long ll;
4 #define sc(x) scanf("%lld", &x)
5 #define mn 200
6 ll t, n, x[mn], y[mn], s, edge, kase;
7 ll gcd(ll a, ll b)
8 {
9     return b ? gcd(b, a % b) : a;
10 }
11 ll abs(ll x)
12 {
13     return x < 0 ? -x : x;
14 }
15 signed main()
16 {
17     for (sc(t); t--;) {
18         sc(n);
19         s = edge = 0;
20         for (ll i = 1, dx, dy; i <= n; ++i)
21         {
22             sc(dx), sc(dy);
23             x[i] = x[i - 1] + dx, y[i] = y[i - 1] + dy;
24             edge += gcd(abs(dx), abs(dy));
25         }
26         x[n + 1] = x[1], y[n + 1] = y[1];
27         for (ll i = 1, dx, dy; i <= n; ++i)
28         {
29             s += x[i] * y[i + 1] - x[i + 1] * y[i];
30         }
31         s = abs(s);
32         printf("Scenario #%lld: %lld %lld %.1f\n\n", ++kase, (s - edge + 2)
33 / 2, edge, s * 0.5f);
34     }
35     return 0;
36 }
```

## 平面最近点对

(洛谷P7883) 归并排序复杂度  $O(n \log n)$  , 若快排复杂度  $O(n \log^2 n)$

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #define sc(x) scanf("%lld", &x)
5 #define mn 400010
6 ll n, ans = 1e18;
7 struct node
8 {
9     ll x, y;
10    bool operator<(const node &r) const { return x < r.x; }
11 } p[mn];
12 void solve(ll lc, ll rc)
13 {
14     if (lc == rc)
15     {
16         return;
17     }
18     ll cc = (lc + rc) >> 1;
19     ll cx = p[cc].x; //修改前的
20     solve(lc, cc), solve(cc + 1, rc);
21     inplace_merge(p + lc, p + cc + 1, p + rc + 1, [] (const node &x, const
node &y)
22                 { return x.y < y.y; });
23     vector<node> v;
24     for (ll i = lc; i <= rc; ++i)
25     {
26         if ((cx - p[i].x) * (cx - p[i].x) <= ans)
27         {
28             v.emplace_back(p[i]);
29         }
30     }
31     for (auto lf = v.begin(), rf = lf; lf != v.end(); lf++)
32     {
33         double sans = sqrt(ans);
34         while (rf != v.end() && rf->y <= lf->y + sans)
35         {
36             ++rf;
37         }
38         for (auto i = lf + 1; i != rf; ++i)
39         {
40             ans = min(ans, (lf->x - i->x) * (lf->x - i->x) + (lf->y - i->y)
* (lf->y - i->y));
41         }
42     }
43 }
44 signed main()
45 {
46     sc(n);
47     for (ll i = 1; i <= n; ++i)
48     {
```

```

49         sc(p[i].x), sc(p[i].y);
50     }
51     sort(p + 1, p + 1 + n);
52     solve(1, n);
53     printf("%lld", ans);
54     return 0;
55 }

```

拓展-洛谷P4423-求平面周长最小三角形(含共线三点)

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 typedef double db;
5 #define sc(x) scanf("%lld", &x)
6 #define mn 200010
7 ll n;
8 db ans = 1e21;
9 struct node
10 {
11     ll x, y;
12     bool operator<(const node &r) const { return x < r.x; }
13 } p[mn];
14 db dis(const node &a, const node &b)
15 {
16     return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));
17 }
18 db tri(const node &a, const node &b, const node &c)
19 {
20     return dis(a, b) + dis(b, c) + dis(c, a);
21 }
22 void solve(ll lc, ll rc)
23 {
24     if (lc == rc)
25         return;
26     ll cc = (lc + rc) >> 1;
27     ll cx = p[cc].x; //修改前的
28     solve(lc, cc), solve(cc + 1, rc);
29     inplace_merge(p + lc, p + cc + 1, p + rc + 1, [] (const node &x, const
node &y)
30                 { return x.y < y.y; });
31     vector<node> v;
32     for (ll i = lc; i <= rc; ++i)
33         if (abs(cx - p[i].x) < ans / 2)
34             v.emplace_back(p[i]);
35     for (auto lf = v.begin(), rf = lf; lf != v.end(); lf++)
36     {
37         while (rf != v.end() && abs(rf->y - lf->y) < ans / 2)
38             ++rf;
39         for (auto i = lf + 1; i != rf; ++i)
40             for (auto j = lf + 1; j != i; ++j)
41                 ans = min(ans, tri(*lf, *i, *j));
42     }
43 }
44 signed main()

```

```

45 {
46     sc(n);
47     for (ll i = 1; i <= n; ++i)
48         sc(p[i].x), sc(p[i].y);
49     sort(p + 1, p + 1 + n);
50     solve(1, n);
51     printf("%lf", ans);
52     return 0;
53 }

```

## 随机增量法

洛谷P1742-最小圆覆盖(求圆的半径和坐标, 使得圆覆盖  $n$  个给定点)  $O(n)$

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 typedef double db;
5 #define mn 100010
6 #define cp const point &
7 ll n;
8 struct point
9 {
10     db x = 0, y = 0;
11     point(db a = 0, db b = 0) : x(a), y(b) {}
12     point operator+(cp r) const { return point(x + r.x, y + r.y); }
13     point operator-(cp r) const { return point(x - r.x, y - r.y); }
14     point operator*(const db r) const { return point(x * r, y * r); }
15     point operator/(const db r) const { return point(x / r, y / r); }
16     db norm() const { return x * x + y * y; }
17     point rotate() const { return point(y, -x); } //顺90
18 } p[mn], o;
19 db r2; // r*r
20 db cross(cp a, cp b) { return a.x * b.y - a.y * b.x; }
21 point intersect(cp p0, cp v0, cp p1, cp v1)
22 {
23     return p0 + v0 * (cross(p1 - p0, v1) / cross(v0, v1));
24 }
25 void solve(cp a, cp b, cp c)
26 {
27     o = intersect((a + b) / 2, (a - b).rotate(), (a + c) / 2, (a -
28 c).rotate());
29     r2 = (a - o).norm();
30 }
31 signed main()
32 {
33     scanf("%lld", &n);
34     for (ll i = 1; i <= n; ++i)
35     {
36         scanf("%lf%lf", &p[i].x, &p[i].y);
37         random_shuffle(p + 1, p + 1 + n);
38         for (ll i = 1; i <= n; ++i)
39         {

```

```

40     if ((p[i] - o).norm() > r2)
41     {
42         o = p[i], r2 = 0;
43         for (ll j = 1; j < i; ++j)
44         {
45             if ((p[j] - o).norm() > r2)
46             {
47                 o = (p[i] + p[j]) / 2, r2 = ((p[i] - p[j]) / 2).norm();
48                 for (ll k = 1; k < j; ++k)
49                 {
50                     if ((p[k] - o).norm() > r2)
51                     {
52                         solve(p[i], p[j], p[k]);
53                     }
54                 }
55             }
56         }
57     }
58 }
59 printf("%.12lf\n%.12lf %.12lf", sqrt(r2), o.x, o.y);
60 return 0;
61 }

```

## 立体几何

直线参数方程:  $A + t\vec{AB}$

点  $P(x, y, z)$  到直线  $A(x_0, y_0, z_0) + t(a, b, c)$  的距离  $\frac{|\vec{PA} \times (a, b, c)|}{\sqrt{a^2 + b^2 + c^2}}$  (四边形面积(叉乘)=底( $AB$  长度)乘以高(距离))

$$\text{直线夹角 } \cos \varphi = \frac{|m_1m_2 + n_1n_2 + p_1p_2|}{\sqrt{m_1^2 + n_1^2 + p_1^2} \sqrt{m_2^2 + n_2^2 + p_2^2}}$$

直线垂直:  $m_1m_2 + n_1n_2 + p_1p_2 = 0$

空间向量的叉乘:

$$\vec{a} \times \vec{b} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ a_x & a_y & a_z \\ b_x & b_y & b_z \end{vmatrix} = (a_z b_x - a_x b_z) \vec{i} + (a_x b_y - a_y b_x) \vec{j} + (a_y b_z - a_z b_y) \vec{k}$$

叉乘绝对值:  $\vec{i}, \vec{j}, \vec{k}$  的系数平方和开根。

直线平行:  $a_z b_x - a_x b_z = a_x b_y - a_y b_x = a_y b_z - a_z b_y = 0$  等价于  $\frac{x_1}{x_2} = \frac{y_1}{y_2} = \frac{z_1}{z_2}$

求过不共线三点  $A, B, C$  的平面  $ax + by + cz + d = 0$

- 求  $\vec{AB}, \vec{AC}$ , 求出法向量为  $\vec{n} = \vec{AB} \times \vec{AC} = (a, b, c)$
  - 法向量垂直平面任一点  $P(x, y, z)$ , 设存在投影点  $P_0(x_0, y_0, z_0)$  在平面内, 则与其点积为  $\vec{n} \cdot \vec{PP_0} = 0$ , 即  $a(x - x_0) + b(y - y_0) + c(z - z_0)$
- 用  $A, B, C$  任一取代  $P_0$ , 求出  $d = -(ax_0 + by_0 + cz_0)$

$$\text{直线 } (n, m, p) \text{ 与平面的夹角正弦值 } \sin \varphi = \frac{|am + bn + cp|}{\sqrt{a^2 + b^2 + c^2} \sqrt{m^2 + n^2 + p^2}}$$

$$\text{直线与平面平行 } am + bn + cp = 0, \text{ 垂直 } \frac{a}{m} = \frac{b}{n} = \frac{c}{p}$$

$$\text{点到平面距离公式: } d = \frac{|Ax + By + Cz + D|}{\sqrt{A^2 + B^2 + C^2}}$$

球面一般方程:  $x^2 + y^2 + z^2 + Ax + By + Cz + D = 0$  若  $A^2 + B^2 + C^2 - 4D > 0$ , 是球面

圆锥面方程:  $z^2 = k^2(x^2 + y^2)$ ,  $z = \pm k\sqrt{x^2 + y^2}$

椭球面方程:  $\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1$

单叶双曲面(沿着  $z$  延展的曲线)方程:  $\frac{x^2}{a^2} + \frac{y^2}{b^2} - \frac{z^2}{c^2} = 1$ , 双叶:  $\frac{x^2}{a^2} + \frac{y^2}{b^2} - \frac{z^2}{c^2} = -1$

抛物面(沿着  $z$  正向)方程:  $\frac{x^2}{p} + \frac{y^2}{q} = 2z$

双曲面方程:  $\frac{x^2}{p} - \frac{y^2}{q} = 2z$

## 博弈论

P-position: (p:previous) 先手必败, N-position: (n:next) 先手必胜

局面的性质: (总结: 只要能移动到  $P$  就是  $N$ , 否则都是  $P$ )

1. 合法操作集合为空的局面是P-position
2. 可以移动到P-position的局面是N-position
3. 所有移动都只能到N-position的局面是P-position

mex 函数: 最小不属于某集合的非负整数  $\text{mex}(A) = \min\{k | k \in \mathbb{C}_N A\}$

SG 函数: 所有后继中都未出现的最小值  $\text{sg}(x) = \text{mex}\{\text{sg}(y) | y \text{ 为 } x \text{ 的直接儿子}\}$

输出任意先手必胜方案: 找最大 sg 将它变成剩余的 sg 和, 可证一定能找到

顶点  $x$  所代表的局面是 P-position 当且仅当  $\text{sg}(x) = 0$

**SG定理:** 由  $n$  个有向图组成的ICG, 设它们的起点分别为  $s_1, s_2, \dots, s_n$ , 当且仅当  $SG(s_1) \oplus SG(s_2) \oplus \dots \oplus SG(s_n) \neq 0$  时, 这个游戏先手必胜

经典博弈:

- **巴什博弈** 有  $n$  个物品, 两人轮流取物, 每次至少取一个, 最多取  $m$  个, 最后取完者胜  
先手必败条件:  $n \bmod (m + 1) = 0$
- **Nim游戏** (所有ICG游戏都可以转化为Nim游戏求解) 有  $n$  堆数, 每堆有  $s_i$  个, 每次可以从任意堆里取 1 到任意多个数

先手必败条件:  $\bigoplus_{i=1}^n s_i = 0$

先手必胜方案输出: 当前异或和  $x \neq 0$  时,  $O(\log a)$  找  $b_i \oplus x < b_i$ , 将  $b_i$  更改为  $b_i \oplus x$ ; 否则任取完一堆; 回合数可以不超  $2n$  (SCNUOJ1701)

- **威佐夫博弈** 有两堆物品, 轮流从一堆拿任意个或同时从两堆拿同样的任意个, 取完者胜

先手必败条件为  $(b, a), a > b, b = \lfloor \frac{1 + \sqrt{5}}{2} \times (a - b) \rfloor$

- **斐波那契博弈** 一堆物品，先手最少取一个且不能取完，之后每次取不能少于1或超过上次的二倍，取完者胜

先手必败条件：物品是斐波那契数列

在普通SG游戏的基础上，新增一条规则：每次行动，每个可以移动的SG游戏都要移动，是Every-SG游戏

例如棋盘上有n个棋子在不同的位置，双方每次行动都要将每一个可以移动的棋子按规则移动，这就是典型的Every-SG游戏。

$$step(v) = \begin{cases} 0, & v \text{ 为终止状态} \\ \max(step(u)) + 1, & sg(v) > 0, sg(u) = 0 \\ \min(step(u)) + 1, & sg(v) = 0 \end{cases}$$

定理：Every-SG游戏先手必胜当且仅当单一游戏中最大的 step 为奇数

## 多项式

一个多项式全家桶模板见上文公式-组合数学-球盒问题。

### FFT

系数表示法： $F(x) = y = \sum_{i=0}^n a_i x^i$ ，其中 $\{a_i\}$ 是每一项的系数。

点值表示法： $F(x)$ 可以被n个点 $(x_i, f(x_i))$ 唯一确定

两多项式相乘即求卷积： $f(x) \times g(x) = (x_i, f(x_i) \times g(x_i))$ ，点值表示法下  $O(n)$

将系数表示法转点值表示法为快速傅里叶变换(FFT)，点值转系数为IFFT，均  $O(n \log n)$

洛谷P3803-输入  $n, m$  和  $n$  次多项式  $F(x)$ ， $m$  次多项式  $G(x)$ ，求  $F(x) \cdot G(x)$ ，系数都是个位数

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 typedef double db;
5 #define mn ((1 << 21) + 1)
6 ll n1, n2, rev[mn], ans[mn], k, s = 1, len, n;
7 db pi = acos(-1), v;
8 typedef complex<db> cp;
9 cp a[mn], b[mn];
10 void fft(cp *a, ll n, ll flag)
11 {
12     for (ll i = 0; i < n; ++i)
13     {
14         if (i < rev[i])
15         {
16             swap(a[i], a[rev[i]]);
17         }
18     }
19     for (ll h = 1; h < n; h <= 1)
20     {
21         cp wn = exp(cp(0, flag * pi / h));
22         for (ll j = 0; j < n; j += h << 1)
23         {
24             cp w(1, 0);

```

```

25         for (ll k = j; k < j + h; ++k)
26     {
27         cp x = a[k], y = w * a[k + h];
28         a[k] = x + y;
29         a[k + h] = x - y;
30         w *= wn;
31     }
32 }
33 }
34 if (flag == -1)
35 {
36     for (ll i = 0; i < n; ++i)
37     {
38         a[i] /= n;
39     }
40 }
41 }
42 #pragma GCC diagnostic ignored "-Wformat" //忽略db转complex警告
43 signed main()
44 {
45     scanf("%lld%lld", &n1, &n2), n = n1 + n2;
46     for (ll i = 0; i <= n1; ++i)
47     {
48         scanf("%lf", a + i);
49     }
50     for (ll i = 0; i <= n2; ++i)
51     {
52         scanf("%lf", b + i);
53     }
54     while (s <= n)
55     {
56         s <= 1, ++k;
57     }
58     for (ll i = 0; i < s; ++i)
59     {
60         rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (k - 1));
61     }
62     fft(a, s, 1), fft(b, s, 1);
63     for (ll i = 0; i <= s; ++i)
64     {
65         a[i] *= b[i];
66     }
67     fft(a, s, -1);
68     for (ll i = 0; i <= n; ++i)
69     {
70         printf("%lld ", (ll)(a[i].real() + 0.5));
71     }
72     return 0;
73 }

```

## NTT

快速数论变换, FNTT, 简称为NTT number theory transform

FFT 在模意义下的一种实现, 功能上与 FFT 类似。FFT 的计算量大且存在浮点误差。

欧拉定理:  $a \in \mathbb{Z}, m \in \mathbb{N}^*$ , 若  $(a, m) = 1$ , 则  $a^{\varphi(m)} \equiv 1 \pmod{m}$

阶: 满足  $a^n \equiv 1 \pmod{m}$  的最小正整数  $n$ , 记作  $\delta_m(a)$

原根:  $m \in \mathbb{N}_+, a \in \mathbb{Z}$ , 若  $(a, m) = 1, \delta_m(a) = \varphi(m)$ , 则  $a$  是模  $m$  的原根

998244353 的原根是 3,  $10^9 + 7$  不能做 NTT(MTT 考虑 P4245)

洛谷P3803

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 #define sc(x) scanf("%lld", &x)
4 typedef long long ll;
5 const ll mn = 3e6 + 10, p = 998244353, g = 3, gi = 332748118;
6 ll n, m, lim = 1, l, r[mn], a[mn], b[mn]; // 2^l=lim
7 ll qpow(ll a, ll b)
8 {
9     ll r = 1;
10    for (; b; b >= 1)
11    {
12        if (b & 1)
13        {
14            r = r * a % p;
15        }
16        a = a * a % p;
17    }
18    return r;
19 }
20 void ntt(ll *a, ll type)
21 {
22     for (ll i = 0; i < lim; ++i)
23     {
24         if (i < r[i])
25         {
26             swap(a[i], a[r[i]]);
27         }
28     }
29     for (ll mid = 1; mid < lim; mid <= 1)
30     {
31         ll wn = qpow(type == 1 ? g : gi, (p - 1) / (mid << 1));
32         for (ll j = 0; j < lim; j += (mid << 1))
33         {
34             ll w = 1;
35             for (ll k = 0; k < mid; ++k, w = w * wn % p)
36             {
37                 ll x = a[j + k], y = w * a[j + k + mid] % p;
38                 a[j + k] = (x + y) % p;
39                 a[j + k + mid] = (x - y + p) % p;
40             }
41         }
42     }
43 }
```

```

42     }
43 }
44 signed main()
45 {
46     sc(n), sc(m);
47     for (ll i = 0; i <= n; ++i)
48     {
49         sc(a[i]);
50     }
51     for (ll i = 0; i <= m; ++i)
52     {
53         sc(b[i]);
54     }
55     while (lim <= n + m)
56     {
57         ++l, lim <= 1;
58     }
59     for (ll i = 0; i < lim; ++i)
60     {
61         r[i] = (r[i >> 1] >> 1 | ((i & 1) << (l - 1)));
62     }
63     ntt(a, 1), ntt(b, 1);
64     for (ll i = 0; i < lim; ++i)
65     {
66         a[i] = a[i] * b[i] % p;
67     }
68     ntt(a, -1);
69     ll inv = qpow(lim, p - 2);
70     for (ll i = 0; i <= n + m; ++i)
71     {
72         printf("%lld ", a[i] * inv % p);
73     }
74     return 0;
75 }

```

## 分治 FFT

SCNUOJ1806-求  $\prod_{i=1}^n (1 + a_i x)$  的  $x^k$  系数对 998244353 取模,  
 $1 \leq k \leq n \leq 10^5, 1 \leq a_i \leq 10^9$

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 using ll = long long;
4 using p11 = pair<ll, ll>;
5 const ll mod = 998244353;
6
7 ll qui(ll a, ll x)
8 {
9     ll ret = 1;
10    while (x)
11    {
12        if (x & 1)
13            ret = ret * a % mod;

```

```

14         a = a * a % mod;
15         x >>= 1;
16     }
17     return ret;
18 }
19
20 using Poly = vector<ll>;
21 const int BIT = 20;
22 int p[1 << BIT];
23 const ll maxn = 1e5 + 10;
24 ll fac[maxn], inv[maxn];
25
26 Poly operator*(const Poly &a, const Poly &b)
27 {
28     int n = a.size() - 1, m = b.size() - 1;
29     int L, l = 0;
30     for (L = 1; L <= n + m; l++, L = L << 1)
31     ;
32
33     vector<int> p(L);
34
35     for (int i = 1; i < L; i++)
36         p[i] = ((p[i] >> 1) >> 1) | ((i & 1) << (l - 1));
37     auto u = a, v = b;
38     u.resize(L, 0), v.resize(L, 0);
39     auto ntt = [&L, &l, &p](Poly &g, int type)
40     {
41         for (int i = 0; i < L; i++)
42             if (i < p[i])
43                 swap(g[i], g[p[i]]);
44         for (int i = 1; i < L; (i <<= 1))
45         {
46             ll wn = qui(3, (mod - 1) / (i << 1));
47             for (int j = 0; j < L; j += (i << 1))
48             {
49                 ll w = 1;
50                 for (int k = j; k < j + i; w = w * wn % mod, k++)
51                 {
52                     assert(k + i < L);
53                     assert(k < L);
54                     ll t = g[k + i] * w % mod;
55                     g[k + i] = (g[k] - t + mod) % mod;
56                     g[k] = (g[k] + t) % mod;
57                 }
58             }
59         }
60         if (type == 1)
61             return;
62         reverse(g.begin() + 1, g.begin() + L);
63         ll ni = qui(L, mod - 2);
64         for (int i = 0; i < L; i++)
65             g[i] = g[i] * ni % mod;
66     };
67     ntt(u, 1), ntt(v, 1);
68
69     Poly g(L, 0);

```

```

70     for (int i = 0; i < L; i++)
71         g[i] = u[i] * v[i] % mod;
72     ntt(g, -1);
73
74     return g;
75 }
76
77 signed main()
78 {
79     cin.tie(0)->sync_with_stdio(false);
80     int n, k;
81     cin >> n >> k;
82
83     vector<int> b(n + 1);
84     for (int i = 1; i <= n; i++)
85         cin >> b[i];
86
87     function<Poly(int, int)> calc = [&](int l, int r)
88     {
89         if (l >= r)
90         {
91             assert(l == r);
92             return Poly{1, b[l]};
93         }
94         int mid = (l + r) / 2;
95         return calc(l, mid) * calc(mid + 1, r);
96     };
97
98     Poly ep = calc(0, n);
99     cout << ep[k];
100    return 0;
101 }

```

## FWT

快速沃尔什变换。求  $C_i = \sum_{j \oplus k} A_j B_k$ , 其中  $\oplus$  是与/或/异或/同或。复杂度与 FFT 看齐。

模板题 P4717-给定长为  $2^n (n \leq 17)$  的序列, 分别求对或/与/异或后的序列, 对  $P = 998244353$  取模

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 using ll = long long;
4 const ll P = 998244353;
5 ll qpow(ll a, ll b = P - 2)
6 {
7     ll r = 1;
8     for (; b; b >= 1, a = a * a % P)
9         if (b & 1)
10             r = r * a % P;
11     return r;
12 }
13 struct modint
14 {
15     ll v;

```

```

16     modint(l1 v_ = 0) : v(v_) {}
17     modint operator+(const modint &o) const { return modint((v + o.v) % P); }
18     modint operator-(const modint &o) const { return modint((v - o.v + P) % P); }
19     modint operator*(const modint &o) const { return modint(v * o.v % P); }
20     modint operator/(const modint &o) const { return modint(v * qpow(o.v) % P); }
21     modint operator+=(const modint &o) { return *this = *this + o; }
22     modint operator-=(const modint &o) { return *this = *this - o; }
23     modint operator*=(const modint &o) { return *this = *this * o; }
24     modint operator/=(const modint &o) { return *this = *this / o; }
25 };
26 #define rd(x) scanf("%d", &x)
27 const int N = 1 << 17 | 1;
28 int n, m;
29 modint A[N], B[N], a[N], b[N];
30
31 inline void in()
32 {
33     for (int i = 0; i < n; i++)
34         a[i] = A[i], b[i] = B[i];
35 }
36
37 inline void get()
38 {
39     for (int i = 0; i < n; i++)
40         a[i] *= b[i];
41 }
42
43 inline void out()
44 {
45     for (int i = 0; i < n; i++)
46         printf("%d%c", a[i], " \n"[i == n - 1]);
47 }
48
49 inline void OR(modint *f, modint x = 1)
50 {
51     for (int o = 2, k = 1; o <= n; o <= 1, k <= 1)
52         for (int i = 0; i < n; i += o)
53             for (int j = 0; j < k; j++)
54                 f[i + j + k] += f[i + j] * x;
55 }
56
57 inline void AND(modint *f, modint x = 1)
58 {
59     for (int o = 2, k = 1; o <= n; o <= 1, k <= 1)
60         for (int i = 0; i < n; i += o)
61             for (int j = 0; j < k; j++)
62                 f[i + j] += f[i + j + k] * x;
63 }
64
65 inline void XOR(modint *f, modint x = 1)
66 {
67     for (int o = 2, k = 1; o <= n; o <= 1, k <= 1)
68         for (int i = 0; i < n; i += o)

```

```

69         for (int j = 0; j < k; j++)
70     {
71         f[i + j] += f[i + j + k];
72         f[i + j + k] = f[i + j] - f[i + j + k] - f[i + j + k];
73         f[i + j] *= x, f[i + j + k] *= x;
74     }
75 }
76
77 int main()
{
78     rd(m), n = 1 << m;
79     for (int i = 0; i < n; i++)
80         rd(A[i]);
81     for (int i = 0; i < n; i++)
82         rd(B[i]);
83     in(), OR(a), OR(b), get(), OR(a, P - 1), out();
84     in(), AND(a), AND(b), get(), AND(a, P - 1), out();
85     in(), XOR(a), XOR(b), get(), XOR(a, (modint)1 / 2), out();
86     return 0;
87 }
88 }
```

lc982-给定整数数组  $a$ ,  $|a| \leq 10^3$ ,  $0 \leq a_i < 2^{16}$ , 求  $a_i \& a_j \& a_k = 0$  的三元对的数目( $i, j, k$  可以取等)

设桶数组  $a$ , 即求  $\sum_{i \& j \& k=0} a_i a_j a_k = \sum_{i \& p=0} a_i \sum_{j \& k=p} a_j a_k$ , 即先对  $a$  做 FWT, 然后算  $fwf(b) = fwt(a)^3$ , 再逆运算得到  $b_0$  作为答案。

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 int w;
4 int B[1 << 16];
5 void FWTand(int *a, int opt)
6 {
7     int N = 1 << w;
8     for (int mid = 1; mid < N; mid <= 1)
9         for (int R = mid << 1, j = 0; j < N; j += R)
10             for (int k = 0; k < mid; k++)
11                 if (opt == 1)
12                     a[j + k] += a[j + k + mid];
13                 else
14                     a[j + k] -= a[j + k + mid];
15 }
16 int countTriplets(vector<int> &A)
17 {
18     for (auto a : A)
19     {
20         B[a]++;
21         while ((1 << w) <= a)
22             w++;
23     }
24     FWTand(B, 1);
25     for (int i = 0; i < (1 << w); i++)
26         B[i] *= B[i] * B[i];
```

```

27     FWTand(B, -1);
28     return B[0];
29 }
```

## 拉格朗日插值法

洛谷P4781-给定由  $n$  个点确定的多项式  $f(x)$ , 求  $f(k) \bmod 998244353$ ,  $x_i$  各异

复杂度为  $O(n^2)$ , 比高斯消元快 (但高斯消元对给出点坐标无要求) (若  $x_i = i$  可以朴素  $O(n^2)$  差分)

$$f(k) = \sum_{i=0}^n y_i \prod_{i \neq j} \frac{k - x_j}{x_i - x_j}$$

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #define MAXN 2002
5 #define MOD 998244353
6 ll n, k, x[MAXN], y[MAXN], ans, s1, s2;
7 ll qpow(ll a, ll x)
8 {
9     ll res = 1, t = a;
10    for (; x; t = t * t % MOD, x >= 1) if (x & 1)
11        res = res * t % MOD;
12    return res;
13 }
14 inline ll inv(ll x) { return qpow(x, MOD - 2); }
15 signed main()
16 {
17     scanf("%lld%lld", &n, &k);
18     for (ll i = 1; i <= n; ++i) scanf("%lld%lld", x + i, y + i);
19     for (ll i = 1; i <= n; ++i)
20     {
21         s1 = y[i] % MOD, s2 = 1;
22         for (ll j = 1; j <= n; ++j) if (i != j)
23             s1 = s1 * (k - x[j]) % MOD,
24             s2 = s2 * ((x[i] - x[j]) % MOD) % MOD;
25         ans += s1 * inv(s2) % MOD;
26         ans = (ans + MOD) % MOD;
27     }
28     printf("%lld", ans);
29     return 0;
30 }
```

# 杂项

## 矩阵加速

常见建模：

$$f_i = af_{i-1} + bf_{i-2} \Rightarrow \begin{pmatrix} a & b \\ 1 & 0 \end{pmatrix} \begin{pmatrix} f_{i-1} \\ f_{i-2} \end{pmatrix} = \begin{pmatrix} f_i \\ f_{i-1} \end{pmatrix}$$

$$f_i = af_{i-1} + b^i \Rightarrow \begin{pmatrix} a & 1 \\ 0 & b \end{pmatrix} \begin{pmatrix} f_{i-1} \\ b^{i-1} \end{pmatrix} = \begin{pmatrix} f_i \\ b^i \end{pmatrix}$$

$$f_i = af_{i-1} + i^3 \Rightarrow \begin{pmatrix} a & 1 & 0 & 0 & 0 \\ 0 & 1 & 3 & 3 & 1 \\ 0 & 0 & 1 & 2 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} f_{i-1} \\ i^3 \\ i^2 \\ i \\ 1 \end{pmatrix} = \begin{pmatrix} f_i \\ (i+1)^3 \\ (i+1)^2 \\ i+1 \\ 1 \end{pmatrix}$$

矩阵前  $k$  项和：(SCNUOI1068 / GDCPC2022K) (直接用 A.P. 的问题是不一定可逆)

已知  $n$  阶方阵  $A$ , 求  $A + A^2 + A^3 + \dots + A^k$ 。

令  $B_i = A + A^2 + \dots + A^i$ ,  $n$  阶单位阵为  $E$ 。若  $k$  为偶数, 上式可以转化为:

$$B_k = (A + A^2 + \dots + A^{\frac{k}{2}}) + A^{\frac{k}{2}}(A + A^2 + \dots + A^{\frac{k}{2}}) = B_{\frac{k}{2}} + A^{\frac{k}{2}}B_{\frac{k}{2}} = (E + A^{\frac{k}{2}})B_{\frac{k}{2}}$$

若  $k$  为奇数, 可以直接用上一个偶数, 即  $B_k = B_{k-1} + A^k$ 。

其中,  $A^i$  可由矩阵快速幂计算, 复杂度为  $O(n^3 \log i)$ 。而  $B_k$  的递推需要  $\log k$  次, 故总复杂度为  $O(n^3 \log^2 k)$ 。

```

1 matrix psum(11 k)
2 {
3     if (k == 0)
4         return matrix(n);
5     if (k == 1)
6         return *this;
7     if (k % 2 == 1)
8         return psum(k - 1) + pow(k);
9     return (unit(n) + this->pow(k / 2)) * psum(k / 2);
10}
```

## 高斯消元

优化的Gauss-Jordan消元法, 复杂度  $O(n^3)$

洛谷P3389-求实数线性方程组的唯一解或 `No solution`, 值域  $|a| \leq 10^4$

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 typedef double db;
5 #define repe(i, a, b) for(ll i=a; i<=b; ++i)
```

```

6 #define limn 102
7 db a[limn][limn];
8 ll n;
9 signed main()
10 {
11     scanf("%d", &n);
12     repe(i, 1, n) repe(j, 1, n + 1) scanf("%lf", &a[i][j]);
13     repe(i, 1, n)
14     {
15         ll mx = i;
16         repe(j, i + 1, n) if (fabs(a[j][i]) > fabs(a[mx][i])) mx = j;
17         repe(j, 1, n + 1) swap(a[i][j], a[mx][j]);
18         if (!a[i][i]) return !printf("No Solution");
19         repe(j, 1, n) if(j!=i)
20         {
21             db tmp = a[j][i] / a[i][i];
22             repe(k, i + 1, n + 1) a[j][k] -= a[i][k] * tmp;
23         }
24     }
25     repe(i, 1, n) printf("%.2lf\n", a[i][n + 1] / a[i][i]);
26     return 0;
27 }

```

洛谷P4783-求逆矩阵对  $10^9 + 7$  取模或输出 No solution

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #define repe(i,a,b) for(ll i=a;i<=b;++i)
5 #define MAXN 402
6 #define MOD 1000000007
7 ll n, m, f[MAXN][MAXN << 1], r, ret;
8 ll inv(ll u, ll v)
9 {
10     for (ret = 1; v; u = u * u % MOD, v >>= 1) if (v & 1)
11         ret = ret * u % MOD;
12     return ret;
13 }
14 signed main()
15 {
16     scanf("%lld", &n), m = n << 1;
17     repe(i, 1, n)
18     {
19         repe(j, 1, n) scanf("%lld", &f[i][j]);
20         f[i][n + i] = 1;
21     }
22     repe(i, 1, n)
23     {
24         repe(j, i, n) if (f[j][i])
25         {
26             repe(k, 1, m) swap(f[i][k], f[j][k]);
27             break;
28         }
29         if (!f[i][i]) return !printf("No Solution");
30         r = inv(f[i][i], MOD - 2);

```

```

31     repe(j, i, m) f[i][j] = f[i][j] * r % MOD;
32     repe(j, 1, n) if (j != i)
33     {
34         r = f[j][i];
35         repe(k, i, m) f[j][k] = (f[j][k] - r * f[i][k] % MOD + MOD) %
36 MOD;
37     }
38     repe(i, 1, n)
39     {
40         repe(j, n + 1, m) printf("%lld ", f[i][j]);
41         printf("\n");
42     }
43     return 0;
44 }

```

## 康托展开

树状数组优化后  $O(n \log n)$  求  $[1, n]$  的任意排列的排名(字典序排序序号), 公式:

$$1 + \sum_{i=1}^n (n-i)! \times \left( \sum_{j=i}^n [a[j] < a[i]] \right)$$

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #define MAXN 10000002
5 #define MOD 998244353
6 ll a, c[MAXN], f[MAXN], ans, v, n, lb[MAXN], k;
7 ll sum(ll p)
8 {
9     ll res = 0;
10    while (p) res += c[p], p -= lb[p];
11    return res;
12 }
13 void add(ll x, ll&& k)
14 {
15     while (x <= n) c[x] += k, x += lb[x];
16 }
17 signed main()
18 {
19     scanf("%lld", &n);
20     f[0] = 1;
21     for (ll i = 1; i <= n; ++i) f[i] = f[i - 1] * i % MOD;
22     for (ll i = 1; i <= n; ++i) lb[i] = i & -i;
23     for (ll i = 1; i <= n; ++i) add(i, 1);
24     for (ll i = 1; i <= n; ++i)
25     {
26         scanf("%lld", &a);
27         (ans += (sum(a) - 1) * f[n - i] % MOD) %= MOD;

```

```

28     add(a, -1);
29 }
30 printf("%lld", (ans)%MOD);
31 return 0;
32 }

```

逆康托展开(UVA11525)-给定每个  $\sum_{j=i}^n [a[j] < a[i]]$  , 求排列:

```

1 #include<iostream>
2 #include<csstdio>
3 #define N (500000+21)
4 //define一定一定要记得加括号
5 using namespace std;
6 struct SegmentTree{int l,r,cnt;}t[N*4];//线段树数组要开四倍
7 int T,n,x;
8 //下面是线段树模板, 基本和区间加差不多, 就是注意查询操作类似于平衡树找第k名
9 void build(int p,int l,int r){
10     t[p].l=l,t[p].r=r;
11     if(l==r){
12         t[p].cnt=1;
13         return;
14     }
15     int mid=(l+r)>>1;
16     build(p*2,l,mid);
17     build(p*2+1,mid+1,r);
18     t[p].cnt=t[p*2].cnt+t[p*2+1].cnt;
19 }
20 int modify(int p,int v){
21     if(t[p].l==t[p].r){
22         t[p].cnt=0;//这里随着查询也改变了数值
23         return t[p].l;
24     }
25     int res=0;
26     if(v<=t[p*2].cnt) res=modify(p*2,v);
27     else res=modify(p*2+1,v-t[p*2].cnt);
28     t[p].cnt=t[p*2].cnt+t[p*2+1].cnt;
29     return res;
30 }
31 int main(){
32     scanf("%d",&T);
33     while(T--){
34         scanf("%d",&n);
35         build(1,1,n);
36         for(int i=1;i<=n;i++){
37             scanf("%d",&x);
38             printf("%d",modify(1,x+1));//因为从0开始所以要把x加1
39             if(i!=n) printf(" ");
40         }
41         printf("\n");
42     }
43     return 0;
44 }

```

## 自适应辛普森法

洛谷P4525-求  $\int_L^R \frac{cx+d}{ax+b} dx$  , 输入  $a, b, c, d, L, R$

$$\int_a^b f(x)dx \approx \frac{(b-a)(f(a) + f(b) + 4f(\frac{a+b}{2}))}{6}$$

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef double db;
4 #define sc(x) scanf("%lf", &x)
5 db a, b, c, d, l, r, eps = 1e-8; // eps可以比题目要求的细一两位
6 db f(db x) //这里根据题目不同编写不同的f
7 {
8     return (c * x + d) / (a * x + b);
9 }
10 db simpson(db lf, db rf)
11 {
12     return (rf - lf) * (f(lf) + f(rf) + 4 * f((lf + rf) / 2)) / 6;
13 }
14 db solve(db lf, db rf, db now) //当前区间估计值为now
15 {
16     db cf = (lf + rf) / 2;
17     db lfans = simpson(lf, cf), rfans = simpson(cf, rf);
18     if (abs(lfans + rfans - now) <= eps) //二分后也是now
19     {
//那就不需要细分了
20         return lfans + rfans;
21     }
22     return solve(lf, cf, lfans) + solve(cf, rf, rfans);
23 }
24 signed main()
25 {
26     sc(a), sc(b), sc(c), sc(d), sc(l), sc(r);
27     printf("%lf", solve(l, r, 0)); //估计值随便填一个值
28     return 0;
29 }
```

## 数据结构

### ST表

以区间 gcd 为例(时间复杂度没有两个  $\log$  乘积, 具体为  $O(n(\log w + \log n))$ )

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef int ll;
4 typedef double db;
5 #define sc(x) scanf("%d", &x)
6 #define mn 2000010
7 #define lg 23
```

```

8 11 n, m, st[mn][lg], l, r, lg2[mn];
9 signed main()
10 {
11     sc(n), sc(m);
12     for (ll i = 1; i <= n; ++i)
13     {
14         sc(st[i][0]);
15     }
16     for (ll i = 2; i <= n; ++i)
17     {
18         lg2[i] = lg2[i / 2] + 1;
19     }
20     for (ll j = 1; j < lg; ++j)
21     {
22         for (ll i = 1; i + (1 << j) - 1 <= n; ++i)
23         {
24             st[i][j] = __gcd(st[i][j - 1], st[i + (1 << (j - 1))][j - 1]);
25         }
26     }
27     while (m--)
28     {
29         sc(l), sc(r);
30         ll p = lg2[r - 1 + 1];
31         ll ans = __gcd(st[l][p], st[r - (1 << p) + 1][p]);
32         printf("%d\n", ans);
33     }
34     return 0;
35 }

```

笛卡尔树 LCA ±1 RMQ 问题，预处理、查询复杂度是  $O(n), O(1)$ 。空间复杂度  $O(n)$ 。(以区间max为例洛谷P3865,  $n \leq 10^5, m \leq 2 \times 10^6$ )

```

1 #include<iostream>
2 #include<cstdio>
3 #include<cstring>
4 using namespace std;
5 const int inf=1e9;
6 int n,m,w[2000000],lg2[1500000];
7 int pos[1500000],lst[1500000],dep[1500000],id,mm,s,ch[1500000]
[2],stack[1500000];
8 struct LIM_RMQ
9 {
10     int
w[1500000],b1[1500000],blo,L[150000],R[150000],pos[10000],val[150000],minn[150000],minpos[150000],t[1000],n_st;
11     struct ST_node{int f,id;bool operator <(const ST_node &tmp) const{return f<tmp.f;}};
12     struct STable
13     {
14         ST_node a[4][12];//n=1e6的话logn/2只有9...所以放心开
15         void make(int w[],int n)
16         {
17             for(int i=1;i<=n;i++)a[0][i]=(ST_node){w[i],i};
18             for(int i=1;(1<<i)<=n;i++)//st表，不过要同时处理出最小值所在的位置
19                 for(int j=1;j+(1<<i)-1<=n;j++)

```

```

20             a[i][j]=min(a[i-1][j],a[i-1][j+(1<<(i-1))]);
21         }
22         ST_node query(int l,int r)
23     {
24         int len=lg2[r-l+1];
25         return min(a[len][l],a[len][r-(1<<len)+1]);
26     }
27 }st[10000];//这里只有sqrt(n)个，也不用开这么大
28 struct STable_block
29 {
30     ST_node a[20][150000];//这个占空间最大了吧...不过也只是O(n)的
31     void make(int w[],int n)
32     {
33         for(int i=1;i<=n;i++)a[0][i]=(ST_node){w[i],i};
34         for(int i=1;(1<<i)<=n;i++)
35             for(int j=1;j+(1<<i)-1<=n;j++)
36                 a[i][j]=min(a[i-1][j],a[i-1][j+(1<<(i-1))]);
37     }
38     ST_node query(int l,int r)
39     {
40         int len=lg2[r-l+1];//细节，math库的log2函数不能看做O(1)的，要提前处理
41         return min(a[len][l],a[len][r-(1<<len)+1]);
42     }
43 }st_block;
44 void make(int a[],int n)
45 {
46     for(int i=1;i<=n;i++)w[i]=a[i];
47     lg2[1]=0;for(int i=2;i<=n;i++)lg2[i]=lg2[i>>1]+1;//处理log2
48     b1=max(lg2[n]>>1,1);//分块
49     for(int i=1;i<=n;i++)b1[i]=(i-1)/b1+1;
50     for(int i=1;i<=b1[n];i++)L[i]=(i-
51     1)*b1+1,R[i]=min(i*b1,n),minn[i]=inf;
52     w[0]=w[1]-1;
53     for(int i=1;i<=b1[n];i++)
54     {
55         int tmp=0,nn=0;
56         for(int j=L[i];j<=R[i];j++)
57         {
58             t[++nn]=w[j],tmp=tmp<<1|(w[j]-w[j-1]==1?1:0);
59             if(w[j]<minn[i])minn[i]=w[j],minpos[i]=j;
60             //可以用一个状压来表示本质
61         }
62         if(!pos[tmp])st[pos[tmp]]=++n_st;
63         val[i]=pos[tmp];//记下每个块属于哪一个本质
64     }
65     st_block.make(minn,b1[n]);//块间rmq
66 }
67 ST_node query_block(int id,int l,int r){ST_node t=st[val[id]].query(l-
68 L[id]+1,r-L[id]+1);return (ST_node){t.f,t.id+L[id]-1};}
69 //实际位置=块左端点+块内查询位置-1，如果你把块内查询从0开始写就可以省略-1
70 int query(int l,int r)
71 {
72     int b1l=b1[l],b1r=b1[r];
73     if(b1l==b1r) return query_block(b1l,l,r).id;//一个块
74     int
75     m1=query_block(b1l,l,R[b1l]).id, m2=query_block(b1r,L[b1r],r).id, m3;

```

```

73         if(w[m1]<w[mr])mm=m1;else mm=mr;//两端零散块
74         if(b11+1<=b1r-1)//整块
75         {
76             int mmid=minpos[st_block.query(b11+1,b1r-1).id];
77             if(w[mmid]<w[mm])mm=mmid;
78         }
79         return mm;
80     }
81 }
82 void dfs(int u,int depth)
83 {
84     lst[u]=++id, pos[id]=u, dep[id]=depth;//处理欧拉序列
85     for(int i=0;i<=1;i++)
86     {
87         if(ch[u][i])
88         {
89             dfs(ch[u][i], depth+1);
90             pos[+id]=u, dep[id]=depth;
91         }
92     }
93     int getin()
94     {
95         int x=0;char ch=getchar();
96         while(ch<'0'||ch>'9')ch=getchar();
97         while(ch>='0'&&ch<='9')x=x*10+ch-48, ch=getchar();
98         return x;
99     }
100    int wt[30];
101    void putout(int x)
102    {
103        if(!x){putchar('0');return;}
104        int l=0;
105        while(x)wt[++l]=x%10, x/=10;
106        while(l)putchar(wt[l--]+48);
107        puts("");
108    }
109    void build_tree()//笛卡尔树
110    {
111        int top=1;stack[1]=s=1;//开始根为1
112        for(int i=2;i<=n;i++)
113        {
114            int lst=0;
115            while(top&&w[i]>w[stack[top]])lst=stack[top--];
116            if(lst)ch[i][0]=lst;
117            if(top)ch[stack[top]][1]=i;else s=i;
118            stack[+top]=i;
119        }
120        dfs(s,1);
121    }
122    int main()
123    {
124        n=getin(), m=getin();
125        for(int i=1;i<=n;i++)w[i]=getin();
126        build_tree();
127        a.make(dep,id);
128        for(int i=1;i<=m;i++)
129        {

```

```

129         int l=lst[getin()],r=lst[getin()];
130         if(l>r)swap(l,r); //可能第一次出现的位置是反过来的
131         putout(w[pos[a.query(l,r)]]);
132     }
133 }

```

## 线段树

凡是符合结合律的二目操作都可以使用线段树

### 常见应用

#### 区间加法

洛谷P3372: 给定长为  $n(1 \leq n \leq 10^5)$  的序列和  $m(1 \leq m \leq 10^5)$  次操作:

1. `1 x y k` 区间  $[x, y]$  每个数加上  $k$
2. `2 x y` 输出区间  $[x, y]$  的和

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #define sc(x) scanf("%lld", &x)
5 #define mn 100010
6 ll t[mn << 2], laz[mn << 2], n, m, c, lc, rc, v, a[mn], ans;
7 #define lfs p << 1
8 #define rfs p << 1 | 1
9 #define mkcf ll cf = (lf + rf) >> 1
10 void build(ll p, ll lf, ll rf)
11 {
12     if (lf == rf)
13     {
14         t[p] = a[lf];
15         return;
16     }
17     mkcf;
18     build(lfs, lf, cf);
19     build(rfs, cf + 1, rf);
20     t[p] = t[lfs] + t[rfs];
21 }
22 void pushdown(ll p, ll lf, ll rf)
23 {
24     mkcf;
25     t[lfs] += (cf - lf + 1) * laz[p];
26     t[rfs] += (rf - cf) * laz[p];
27     laz[lfs] += laz[p];
28     laz[rfs] += laz[p];
29     laz[p] = 0;
30 }
31 void update(ll p, ll lf, ll rf)
32 {
33     if (lf >= lc && rf <= rc)
34     {

```

```

35         t[p] += v * (rf - lf + 1);
36         laz[p] += v;
37         return;
38     }
39     pushdown(p, lf, rf);
40     mkcf;
41     if (cf >= lc)
42     {
43         update(lfs, lf, cf);
44     }
45     if (cf < rc)
46     {
47         update(rfs, cf + 1, rf);
48     }
49     t[p] = t[lfs] + t[rfs];
50 }
51 void query(l1 p, l1 lf, l1 rf)
52 {
53     if (lf >= lc && rf <= rc)
54     {
55         ans += t[p];
56         return;
57     }
58     pushdown(p, lf, rf);
59     mkcf;
60     if (cf >= lc)
61     {
62         query(lfs, lf, cf);
63     }
64     if (cf < rc)
65     {
66         query(rfs, cf + 1, rf);
67     }
68     //t[p] = t[lfs] + t[rfs];
69 }
70 signed main()
71 {
72     sc(n), sc(m);
73     for (l1 i = 1; i <= n; ++i)
74     {
75         sc(a[i]);
76     }
77     build(1, 1, n);
78     while (m--)
79     {
80         sc(c), sc(lc), sc(rc);
81         if (c == 1)
82         {
83             sc(v);
84             update(1, 1, n);
85         }
86         else
87         {
88             ans = 0;
89             query(1, 1, n);
90             printf("%lld\n", ans);

```

```

91     }
92 }
93 return 0;
94 }

```

## 区间加乘

洛谷P3373, 给定长为  $n(1 \leq n \leq 10^5)$  的序列和  $m(1 \leq m \leq 10^5)$  次操作和  $p$  :

1. `1 x y k` 区间每个数乘  $k$
2. `2 x y k` 区间每个数加  $k$
3. `3 x y` 输出区间每个数的和对  $p$  取模

## 先乘后加

```

1 #include <iostream>
2 #include <cstdio>
3 using namespace std;
4 int p;
5 long long a[100007];
6 //线段树结构体, v表示此时的答案, mul表示乘法意义上的lazytag, add是加法意义上的
7 struct node{
8     long long v, mul, add;
9 }st[400007];
10 void bt(int root, int l, int r){
11     //初始化lazytag
12     st[root].mul=1;
13     st[root].add=0;
14     if(l==r){
15         st[root].v=a[l];
16     }
17     else{
18         int m=(l+r)/2;
19         bt(root*2, l, m);
20         bt(root*2+1, m+1, r);
21         st[root].v=st[root*2].v+st[root*2+1].v;
22     }
23     st[root].v%=p;
24     return ;
25 }
26 //核心代码, 维护lazytag
27 void pushdown(int root, int l, int r){
28     int m=(l+r)/2;
29     //根据我们规定的优先度, 儿子的值=此刻儿子的值*父亲的乘法lazytag+儿子的区间长度*父亲的加
30     //法lazytag
31     st[root*2].v=(st[root*2].v*st[root].mul+st[root].add*(m-l+1))%p;
32     st[root*2+1].v=(st[root*2+1].v*st[root].mul+st[root].add*(r-m))%p;
33     //很好维护的lazytag
34     st[root*2].mul=(st[root*2].mul*st[root].mul)%p;
35     st[root*2+1].mul=(st[root*2+1].mul*st[root].mul)%p;
36     st[root*2].add=(st[root*2].add*st[root].mul+st[root].add)%p;
37     st[root*2+1].add=(st[root*2+1].add*st[root].mul+st[root].add)%p;
38     //把父节点的值初始化

```

```

38     st[root].mul=1;
39     st[root].add=0;
40     return ;
41 }
42 //update1, 乘法, stdl此刻区间的左边, stdr此刻区间的右边, l给出的左边, r给出的右边
43 void ud1(int root, int stdl, int stdr, int l, int r, long long k){
44 //假如本区间和给出的区间没有交集
45     if(r<stdl || stdr<l){
46         return ;
47     }
48 //假如给出的区间包含本区间
49     if(l<=stdl && stdr<=r){
50         st[root].v=(st[root].v*k)%p;
51         st[root].mul=(st[root].mul*k)%p;
52         st[root].add=(st[root].add*k)%p;
53         return ;
54     }
55 //假如给出的区间和本区间有交集, 但是也有不交叉的部分
56 //先传递lazytag
57     pushdown(root, stdl, stdr);
58     int m=(stdl+stdr)/2;
59     ud1(root*2, stdl, m, l, r, k);
60     ud1(root*2+1, m+1, stdr, l, r, k);
61     st[root].v=(st[root*2].v+st[root*2+1].v)%p;
62     return ;
63 }
64 //update2, 加法, 和乘法同理
65 void ud2(int root, int stdl, int stdr, int l, int r, long long k){
66     if(r<stdl || stdr<l){
67         return ;
68     }
69     if(l<=stdl && stdr<=r){
70         st[root].add=(st[root].add+k)%p;
71         st[root].v=(st[root].v+k*(stdr-stdl+1))%p;
72         return ;
73     }
74     pushdown(root, stdl, stdr);
75     int m=(stdl+stdr)/2;
76     ud2(root*2, stdl, m, l, r, k);
77     ud2(root*2+1, m+1, stdr, l, r, k);
78     st[root].v=(st[root*2].v+st[root*2+1].v)%p;
79     return ;
80 }
81 //访问, 和update一样
82 long long query(int root, int stdl, int stdr, int l, int r){
83     if(r<stdl || stdr<l){
84         return 0;
85     }
86     if(l<=stdl && stdr<=r){
87         return st[root].v;
88     }
89     pushdown(root, stdl, stdr);
90     int m=(stdl+stdr)/2;
91     return (query(root*2, stdl, m, l, r)+query(root*2+1, m+1, stdr, l,
92     r))%p;
92 }

```

```

93 int main(){
94     int n, m;
95     scanf("%d%d%d", &n, &m, &p);
96     for(int i=1; i<=n; i++){
97         scanf("%lld", &a[i]);
98     }
99     bt(1, 1, n);
100    while(m--){
101        int chk;
102        scanf("%d", &chk);
103        int x, y;
104        long long k;
105        if(chk==1){
106            scanf("%d%d%lld", &x, &y, &k);
107            ud1(1, 1, n, x, y, k);
108        }
109        else if(chk==2){
110            scanf("%d%d%lld", &x, &y, &k);
111            ud2(1, 1, n, x, y, k);
112        }
113        else{
114            scanf("%d%d", &x, &y);
115            printf("%lld\n", query(1, 1, n, x, y));
116        }
117    }
118    return 0;
119 }

```

## 区间查重

SCNUOJ1454  $n$ 个柜子(编号从1开始), 每个柜子一开始放了一个编号为 $a_i$ 的物品。维护下面的操作:

- 1  $x$  取走第 $x$ 个柜子的物品(保证此时存在物品)
- 2  $l \ r$  判断 $[l, r]$ 内的未被取走的物品是否有两个编号相同

$1 \leq n, q \leq 5 \times 10^5, 1 \leq a_i \leq 10^6, 1 \leq x, l \leq r \leq n$

思路见代码注释:

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef int ll;
4 #define sc(x) scanf("%d", &x)
5 #define mn 500010
6 ll n, q, a[mn], v, i;
7 // 维护编号为v的双向链表
8 ll index_before[mn]; // 第i个柜子的编号上一次出现的下标 即 链表prev
9 ll last_pos[mn]; // v最后一次出现的下标是last_pos[v] 即链表v的尾部指针
10 ll index_after[mn]; // 第i个柜子的编号下一次出现的下标 即 链表next
11
12 // 区间[l, r]内不重复的充要条件是index_before[i]或index_after[i]均不在这个区间内
13 // 任取其一判断即可, 这里取index_before[i]
14 // 合并物品, 构造线段树, 每个节点维护的区间代表这个区间内的编号最近会在上一次重复的下标

```

```

15 //只需要对区间内每个index_before取最值; 显然max操作可以维护线段树
16 ll t[mn << 2], lc, rc, res, cmd, pos;
17 #define lfs p << 1
18 #define rfs p << 1 | 1
19 #define mkcf ll cf = (lf + rf) >> 1
20 void build(ll lf, ll rf, ll p)
21 {
22     if (lf == rf)
23     {
24         t[p] = index_before[lf];
25         return;
26     }
27     mkcf;
28     build(lf, cf, lfs);
29     build(cf + 1, rf, rfs);
30     t[p] = max(t[lfs], t[rfs]);
31 }
32 void query(ll lf, ll rf, ll p)
33 {
34     if (lc <= lf && rf <= rc)
35     {
36         res = max(res, t[p]);
37         return;
38     }
39     mkcf;
40     if (cf >= lc)
41         query(lf, cf, lfs);
42     if (cf < rc)
43         query(cf + 1, rf, rfs);
44 }
45 void fix(ll lf, ll rf, ll p)
46 {
47     if (lf == rf)
48     {
49         t[p] = v;
50         return;
51     }
52     mkcf;
53     if (cf >= pos)
54         fix(lf, cf, lfs);
55     else
56         fix(cf + 1, rf, rfs);
57     t[p] = max(t[lfs], t[rfs]);
58 }
59
60 signed main()
61 {
62     sc(n), sc(q);
63     for (ll i = 1; i <= n; ++i)
64     {
65         sc(v);
66         //下面三行代码实际上在维护编号为v的双向链表
67         index_before[i] = last_pos[v];
68         index_after[last_pos[v]] = i;
69         last_pos[v] = i;
70         a[i] = v;

```

```

71     }
72     build(1, n, 1); //不build见祖宗
73     while (q--)
74     {
75         sc(cmd);
76         if (cmd == 2)
77         {
78             sc(lc), sc(rc), res = 0;
79             query(1, n, 1);
80             printf("%d\n", res >= lc);
81         }
82     else
83     {
84         sc(i);
85         //双向链表删除节点
86         lc = index_before[i], rc = index_after[i];
87         if (lc)
88             index_after[lc] = rc;
89         if (rc)
90             index_before[rc] = lc;
91
92         //线段树修改
93         //编号为0的双向链表不计重复，均头结点指向0
94         pos = i, v = 0, fix(1, n, 1);
95         //双向链表的后继节点指向更新 对线段树生效
96         if (rc)
97             pos = rc, v = lc, fix(1, n, 1);
98     }
99 }
100 return 0;
101 }

```

## 区间最值

维护操作：① 区间每个数与  $t$  取  $\min$  ② 输出区间和 ③ 输出区间最大值

对每个节点区间：维护该区间最大值  $mx$ 、最大值出现次数  $cnt$ 、严格次大值  $se$

对操作 1：

- 若  $mx \leq t$ ，整个区间都没  $t$  大，不操作 (相等也等于没操作)
- 若  $se < t < mx$ ，那么所有最大值都变成  $t$ ，次大值和更小的值都不变，所以  $cnt$  个  $mx$  被删掉，再加上  $cnt$  个  $t$ ，即区间更新  $cnt(t - mx)$ ，并记录懒标记
- 若  $t \leq se$ ，递归往下处理子区间

对操作 2,3，有手就行

处理 `pushup`：

- 若左右子区间最值一样， $cnt$  取左右子之和， $se$  取左右  $se$  较大者， $mx$  任取左右(都一样)
- 否则， $mx, cnt$  取大者， $se$  让小区间  $mx$  和大区间  $se$  取最值

处理 `pushdown`：

- 懒标记是  $t$  的懒标记，根据题意可以取一些诸如无穷小代表无懒标记

处理建树：

- 初始次大值建无穷小

采用二分法知，对  $m$  次询问，复杂度为  $O(m \log n)$

以 SCNUOJ1703 为例

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #define sc(x) scanf("%lld", &x)
5 #define mn 400010
6 #define lfs p << 1
7 #define rfs p << 1 | 1
8 #define mkcf ll cf = (lf + rf) >> 1
9 #define neg -1
10 ll n, m, c, lc, rc, v, a[mn], s[mn], mx[mn], se[mn], cnt[mn], laz[mn], ans;
11 void pushup(ll p)
12 {
13     s[p] = s[lfs] + s[rfs];
14     if (mx[lfs] == mx[rfs])
15     {
16         mx[p] = mx[lfs];
17         cnt[p] = cnt[lfs] + cnt[rfs];
18         se[p] = max(se[lfs], se[rfs]);
19     }
20     else if (mx[lfs] > mx[rfs])
21     {
22         mx[p] = mx[lfs];
23         cnt[p] = cnt[lfs];
24         se[p] = max(se[lfs], mx[rfs]);
25     }
26     else if (mx[lfs] < mx[rfs])
27     {
28         mx[p] = mx[rfs];
29         cnt[p] = cnt[rfs];
30         se[p] = max(se[rfs], mx[lfs]);
31     }
32 }
33 void build(ll p, ll lf, ll rf)
34 {
35     laz[p] = -1;
36     if (lf == rf)
37     {
38         s[p] = mx[p] = a[lf];
39         cnt[p] = 1;
40         se[p] = -1;
41         return;
42     }
43     mkcf;
44     build(lfs, lf, cf);
45     build(rfs, cf + 1, rf);
46     pushup(p);
47 }
48 void pushd(ll p, ll nw)
```

```

49 {
50     if (mx[p] <= nw)
51     {
52         return;
53     }
54     s[p] += cnt[p] * (nw - mx[p]);
55     mx[p] = laz[p] = nw;
56 }
57 void pushdown(l1 p)
58 {
59     if (laz[p] == -1)
60     {
61         return;
62     }
63     pushd(lfs, laz[p]);
64     pushd(rfs, laz[p]);
65     laz[p] = -1;
66 }
67 void update(l1 p, l1 lf, l1 rf)
68 {
69     if (mx[p] <= v)
70     {
71         return;
72     }
73     if (lf >= lc && rf <= rc && se[p] < v)
74     {
75         return pushd(p, v);
76     }
77     pushdown(p);
78     mkcf;
79     if (cf >= lc)
80     {
81         update(lfs, lf, cf);
82     }
83     if (cf < rc)
84     {
85         update(rfs, cf + 1, rf);
86     }
87     pushup(p);
88 }
89 void query_mx(l1 p, l1 lf, l1 rf)
90 {
91     if (lf >= lc && rf <= rc)
92     {
93         ans = max(ans, mx[p]);
94         return;
95     }
96     pushdown(p);
97     mkcf;
98     if (cf >= lc)
99     {
100         query_mx(lfs, lf, cf);
101     }
102     if (cf < rc)
103     {
104         query_mx(rfs, cf + 1, rf);

```

```

105     }
106 }
107 void query_sum(ll p, ll lf, ll rf)
108 {
109     if (lf >= lc && rf <= rc)
110     {
111         ans += s[p];
112         return;
113     }
114     pushdown(p);
115     mkcf;
116     if (cf >= lc)
117     {
118         query_sum(lfs, lf, cf);
119     }
120     if (cf < rc)
121     {
122         query_sum(rfs, cf + 1, rf);
123     }
124 }
125 signed main()
126 {
127     sc(n), sc(m);
128     for (ll i = 1; i <= n; ++i)
129     {
130         sc(a[i]);
131     }
132     build(1, 1, n);
133     while (m--)
134     {
135         sc(c), sc(lc), sc(rc);
136         if (c == 0)
137         {
138             sc(v);
139             update(1, 1, n);
140         }
141         else if (c == 1)
142         {
143             ans = -1;
144             query_mx(1, 1, n);
145             printf("%lld\n", ans);
146         }
147         else
148         {
149             ans = 0;
150             query_sum(1, 1, n);
151             printf("%lld\n", ans);
152         }
153     }
154     return 0;
155 }

```

## 历史最值

洛谷P6242:长为  $n$  的数列  $A$ ,  $m$  次操作, 每次操作后  $B_i = \max(B_i, A_i)$  :

1. `1 l r k` 对  $A$  区间加法(  $k$  可以是负数 )
2. `2 l r v` 区间求  $A_i = \min(A_i, v)$
3. `3 l r` 对  $A$  区间求和
4. `4 l r` 对  $A$  求区间最大值
5. `5 l r` 对  $B$  求区间最大值

$1 \leq n, m \leq 5 \times 10^5, -5 \times 10^8 \leq A_i, v \leq 5 \times 10^8, -2000 \leq k \leq 2000$

代码解释:

四种不同的懒标记:

- `add_a` : 该区间最大值的加法的懒标记。
- `add_a1` : 该区间非最大值的加法的懒标记。
- `add_b` : 该区间最大的历史最大值的加法的懒标记。
- `add_b1` : 该区间非最大的历史最大值的加法的懒标记。

可能后面两个懒标记比较难理解, 但是一定要理解。

其他就简单了:

- `l`: 该区间的左端点。
- `r`: 该区间的右端点。
- `sum`: 该区间的的和。
- `maxa` : 该区间的最大值。
- `se` : 该区间的严格次大值。
- `cnt` : 该区间最大值的个数。
- `maxb` : 该区间的历史最大值。
- `k1`: 最大值要加的数。
- `k2`: 最大的历史最大值要加的数。
- `k3`: 非最大值要加的数。
- `k4`: 非最大的历史最大值要加的数。

```
1 #include<bits/stdc++.h>
2 请勿抄袭
3 #define ll long long
4 using namespace std;
5 int n,m,op,l,r,k;
6 struct tree{
7     ll sum;
8     int add_a,add_a1,add_b,add_b1;
9     int l,r,maxa,se,maxb,cnt;
10 }s[2000005];
11 char buf[1<<21],*p1=buf,*p2=buf,obuf[1<<21],*o=obuf;
12 #define g() (p1==p2&&(p2=(p1=buf)+fread(buf,1,1<<21,stdin),p1==p2)?EOF: *p1++)
```

```

12 inline int read()
13 {
14     int s=0,f=1;char c=g();
15     for(;c<'0'||c>'9';c=g())
16         if(c=='-')f=-1;
17     for(;c>='0'&&c<='9';c=g())
18         s=s*10+c-'0';
19     return s*f;
20 }
21 inline void write(ll x)
22 {
23     static char buf[16];
24     static int len=-1;
25     if(x<0)putchar('-'),x=-x;
26     do buf[++len]=x%10,x/=10;while(x);
27     while(len>=0)putchar(buf[len--]+'0');
28     putchar('\n');
29 }
30 inline void push_up(int p)
31 {
32     s[p].maxa=max(s[p*2].maxa,s[p*2+1].maxa);
33     s[p].maxb=max(s[p*2].maxb,s[p*2+1].maxb);
34     s[p].sum=s[p*2].sum+s[p*2+1].sum;
35     if(s[p*2].maxa==s[p*2+1].maxa)
36     {
37         s[p].se=max(s[p*2].se,s[p*2+1].se);
38         s[p].cnt=s[p*2].cnt+s[p*2+1].cnt;
39     }
40     if(s[p*2].maxa>s[p*2+1].maxa)
41     {
42         s[p].se=max(s[p*2].se,s[p*2+1].maxa);
43         s[p].cnt=s[p*2].cnt;
44     }
45     if(s[p*2].maxa<s[p*2+1].maxa)
46     {
47         s[p].se=max(s[p*2].maxa,s[p*2+1].se);
48         s[p].cnt=s[p*2+1].cnt;
49     }
50 }
51 void build(int l,int r,int p)
52 {
53     s[p].l=l,s[p].r=r;
54     if(l==r)
55     {
56         s[p].sum=s[p].maxa=s[p].maxb=read();
57         s[p].se=-1e9;
58         s[p].cnt=1;
59         return;
60     }
61     int mid=(l+r)/2;
62     build(l,mid,p*2);
63     build(mid+1,r,p*2+1);
64     push_up(p);
65 }
66 inline void update(int k1,int k2,int k3,int k4,int p)
67 {

```

```

68     s[p].sum+=111*k1*s[p].cnt+111*k3*(s[p].r-s[p].l+1-s[p].cnt);
69     s[p].maxb=max(s[p].maxb,s[p].maxa+k2);
70     s[p].add_b=max(s[p].add_b,s[p].add_a+k2);
71     s[p].add_b1=max(s[p].add_b1,s[p].add_a1+k4);
72     s[p].maxa+=k1,s[p].add_a+=k1;
73     s[p].add_a1+=k3;
74     if(s[p].se!=-1e18)s[p].se+=k3;
75 }
76 inline void push_down(int p)
77 {
78     int maxn=max(s[p*2].maxa,s[p*2+1].maxa);
79     if(s[p*2].maxa==maxn)
80         update(s[p].add_a,s[p].add_b,s[p].add_a1,s[p].add_b1,p*2);
81     else update(s[p].add_a1,s[p].add_b1,s[p].add_a1,s[p].add_b1,p*2);
82     if(s[p*2+1].maxa==maxn)
83         update(s[p].add_a,s[p].add_b,s[p].add_a1,s[p].add_b1,p*2+1);
84     else update(s[p].add_a1,s[p].add_b1,s[p].add_a1,s[p].add_b1,p*2+1);
85     s[p].add_a=s[p].add_b=s[p].add_a1=s[p].add_b1=0;
86 }
87 void update_add(int p)
88 {
89     if(s[p].l>r||s[p].r<l) return;
90     if(l<=s[p].l&&s[p].r<=r)
91         return update(k,k,k,k,p);
92     push_down(p);
93     update_add(p*2),update_add(p*2+1);
94     push_up(p);
95 }
96 void update_min(int p)
97 {
98     if(s[p].l>r||s[p].r<l||k>=s[p].maxa) return;
99     if(l<=s[p].l&&s[p].r<=r&&k>s[p].se)
100        return update(k-s[p].maxa,k-s[p].maxa,0,0,p);
101     push_down(p);
102     update_min(p*2),update_min(p*2+1);
103     push_up(p);
104 }
105 int query_add(int p)
106 {
107     if(s[p].l>r||s[p].r<l) return 0;
108     if(l<=s[p].l&&s[p].r<=r) return s[p].sum;
109     push_down(p);
110     return query_add(p*2)+query_add(p*2+1);
111 }
112 int query_maxa(int p)
113 {
114     if(s[p].l>r||s[p].r<l) return -1e9;
115     if(l<=s[p].l&&s[p].r<=r) return s[p].maxa;
116     push_down(p);
117     return max(query_maxa(p*2),query_maxa(p*2+1));
118 }
119 int query_maxb(int p)
120 {
121     if(s[p].l>r||s[p].r<l) return -1e9;
122     if(l<=s[p].l&&s[p].r<=r) return s[p].maxb;
123     push_down(p);

```

```

124     return max(query_maxb(p*2), query_maxb(p*2+1));
125 }
126 int main()
127 {
128     n=read(),m=read();
129     build(1,n,1);
130     while(m--)
131     {
132         op=read(),l=read(),r=read();
133         if(op==1)k=read(),update_add(1);
134         if(op==2)k=read(),update_min(1);
135         if(op==3)write(query_add(1));
136         if(op==4)write(query_maxa(1));
137         if(op==5)write(query_maxb(1));
138     }
139     return 0;
140 }

```

## 动态开点

动态开点线段树模板题，强制在线维护：(lc715)

- 将区间  $[l, r]$  的所有点染色。
- 将区间  $[l, r]$  的所有点取消染色。
- 查询区间  $[l, r]$  是否每个点都被染色。

其中操作数  $q = 10^4$ ，值域  $n = 10^9$ 。

本质：懒开点的线段树，即仍然构造值域为  $[1, n]$  的普通线段树。但是对所有节点，只有需要访问时，才使用内存分配给该点。对大部分不需要访问的点不开内存。

点数估计：可以考虑走大约  $2q \log n$ ；如果  $q > n$ ，最坏点数约  $2n - 1$ 。

```

1 using ll = long long;
2 class RangeModule
3 {
4     constexpr static ll inf = 1e9;
5     constexpr static ll maxn = 60 * 1e4; // log(inf)*q
6     struct node
7     {
8         ll ls, rs, sum, laz;
9         } t[maxn] = {};
10    int cnt = 1;
11 #define c11 const ll &
12 #define mkcf ll cf = (lf + rf) >> 1
13    void pushdown(int r, c11 lf, c11 rf)
14    {
15        if (!t[r].ls)
16            t[r].ls = ++cnt;
17        if (!t[r].rs)
18            t[r].rs = ++cnt;
19        if (!t[r].laz)
20            return;
21        if (t[r].laz == 1)
22        {

```

```

23         mkcf;
24         t[t[r].ls].sum = (cf - lf + 1); // len-len/2
25         t[t[r].rs].sum = rf - cf; // len/2
26     }
27     else
28     {
29         t[t[r].ls].sum = t[t[r].rs].sum = 0;
30     }
31     t[t[r].ls].laz = t[t[r].rs].laz = t[r].laz;
32     t[r].laz = 0;
33 }
34 void pushup(int r)
35 {
36     t[r].sum = t[t[r].ls].sum + t[t[r].rs].sum;
37 }
38 void update(int r, ll lf, ll rf, cl1 lc, cl1 rc, cl1 v)
39 {
40     if (lc <= lf && rf <= rc)
41     {
42         t[r].sum += (v == 1) * (rf - lf + 1);
43         t[r].laz = v;
44         return;
45     }
46     pushdown(r, lf, rf);
47     mkcf;
48     if (lc <= cf)
49         update(t[r].ls, lf, cf, lc, rc, v);
50     if (rc >= cf + 1)
51         update(t[r].rs, cf + 1, rf, lc, rc, v);
52     pushup(r);
53 }
54 ll query(int r, ll lf, ll rf, cl1 lc, cl1 rc)
55 {
56     if (lc <= lf && rf <= rc)
57         return t[r].sum;
58     pushdown(r, lf, rf);
59     ll res = 0;
60     mkcf;
61     if (lc <= cf)
62         res += query(t[r].ls, lf, cf, lc, rc);
63     if (rc >= cf + 1)
64         res += query(t[r].rs, cf + 1, rf, lc, rc);
65     return res;
66 }
67 public:
68     RangeModule() {}
69     void addRange(int left, int right)
70     {
71         update(1, 1, inf, left, right - 1, 1);
72     }
73     bool queryRange(int left, int right)
74     {
75         return query(1, 1, inf, left, right - 1) == right - left;
76     }
77     void removeRange(int left, int right)
78     {

```

```
79     update(1, 1, inf, left, right - 1, -1);
80 }
81 };
```

## zkw线段树

### 单点修改

单点增加修改，区间求和模板(洛谷P3374)

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #define sc(x) scanf("%lld", &x)
5 #define mn 500010
6 ll t[mn << 2], n, N, m, c, x, y;
7 signed main()
8 {
9     sc(n), sc(m);
10    for (N = 1; N <= n + 1; N <= 1)
11        ;
12    for (ll i = N + 1; i <= N + n; ++i)
13    {
14        sc(t[i]);
15    }
16    for (ll i = N - 1; i >= 1; --i)
17    {
18        t[i] = t[i << 1] + t[i << 1 | 1];
19    }
20    while (m--)
21    {
22        sc(c), sc(x), sc(y);
23        if (c == 1)
24        {
25            for (ll i = x + N; i; i >= 1)
26            {
27                t[i] += y;
28            }
29        }
30        else
31        {
32            ll ans = 0;
33            for (ll s = N + x - 1, r = N + y + 1; s ^ r ^ 1; s >= 1, r >=
34 1)
35            {
36                if (~s & 1)
37                {
38                    ans += t[s ^ 1];
39                }
40                if (r & 1)
41                {
42                    ans += t[r ^ 1];
43                }
44            }
45        }
46    }
47 }
```

```

44         printf("%lld\n", ans);
45     }
46 }
47 return 0;
48 }

```

## 区间修改

区间增加修改，区间求和模板 (洛谷P3372)

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #define mn 400010
5 #define sc(x) scanf("%lld", &x)
6 ll n, m, a[mn], dt, laz[mn], c, x, y, k;
7 signed main()
8 {
9     sc(n), sc(m);
10    for (dt = 1; dt <= n + 1; dt <= 1)
11    {
12        for (ll i = dt + 1; i <= dt + n; ++i)
13        {
14            sc(a[i]);
15        }
16        for (ll i = dt - 1; i >= 1; --i)
17        {
18            a[i] = a[i << 1] + a[i << 1 | 1];
19        }
20        while (m--)
21        {
22            sc(c), sc(x), sc(y);
23            if (c == 1)
24            {
25                sc(k);
26                ll lf = 0, rf = 0, layer = 1, s = dt + x - 1, t = dt + y + 1;
27                for (; s & t & 1; s >>= 1, t >>= 1, layer <= 1)
28                {
29                    a[s] += k * lf, a[t] += k * rf;
30                    if (~s & 1)
31                    {
32                        laz[s & 1] += k, a[s & 1] += k * layer, lf += layer;
33                    }
34                    if (t & 1)
35                    {
36                        laz[t & 1] += k, a[t & 1] += k * layer, rf += layer;
37                    }
38                }
39                for (; s; s >>= 1, t >>= 1)
40                {
41                    a[s] += k * lf, a[t] += k * rf;
42                }
43            }
44            else
45            {

```

```

46         ll lf = 0, rf = 0, layer = 1, s = dt + x - 1, t = dt + y + 1;
47         ll ans = 0;
48         for (; s ^ t ^ 1; s >= 1, t >= 1, layer <= 1)
49         {
50             if (laz[s])
51             {
52                 ans += laz[s] * lf;
53             }
54             if (laz[t])
55             {
56                 ans += laz[t] * rf;
57             }
58             if (~s & 1)
59             {
60                 ans += a[s ^ 1], lf += layer;
61             }
62             if (t & 1)
63             {
64                 ans += a[t ^ 1], rf += layer;
65             }
66         }
67         for (; s; s >= 1, t >= 1)
68         {
69             ans += laz[s] * lf + laz[t] * rf;
70         }
71         printf("%lld\n", ans);
72     }
73 }
74 return 0;
75 }
```

## 树上二分

对单调问题,  $O(\log n)$  单次查询出固定左/右端点下, 最长的右端点/左端点, 区间内满足性质。

2023GDCPC-F 给定长为  $n$  ( $\leq 10^5$ ) 的序列, 有颜色  $c_i$  ( $1 \leq c_i \leq n$ ) 和值  $v_i$  ( $1 \leq v_i \leq 10^9$ ),  $q$  ( $\leq 10^5$ ) 次询问, 时限 5s, 空间 1GB, 要求维护:

1. `1 p x` 修改  $c_p$  为  $x$
2. `2 p x` 修改  $v_p$  为  $x$
3. `3 x k a_1 a_2 ... a_k`  $\sum k \leq 10^6$

找到包含  $x$  的最长区间  $[l, r]$  满足  $k$  个颜色  $a$  在该区间全部覆盖

求某个区间出现的颜色是否都在  $A$ , 即求  $A$  所有颜色在区间出现次数之和是否等于区间长度。可以维护线段树/树状数组, 每个节点保存哈希表, 表示该区间出现的颜色及其次数。每个维护值重复  $\log n$  次, 故空间复杂度为  $O(n \log n)$ 。可以考虑用 `pb_ds` `unmap` 卡常。

二分找出从  $x$  往左往右能到达的最大左右端点, 求区间和。如果朴素二分, 复杂度为  $O(\sum k \log^2 n)$ , 考虑使用线段树/树状数组上二分/倍增优化, 为  $O(\sum k \log n)$ 。

线段树上二分:

```

1 #include <bits/stdc++.h>
2
3 using namespace std;
```

```

4  using ll = long long;
5  const ll mn = 3e5 + 10, mt = mn * 4;
6
7  // 比 unordered_map 更快的哈希表
8  #include <ext/pb_ds/assoc_container.hpp>
9  using namespace __gnu_pbds;
10 const int RANDOM =
11     chrono::high_resolution_clock::now().time_since_epoch().count();
12 struct chash
13 {
14     int operator()(int x) const { return x ^ RANDOM; }
15 };
16 typedef gp_hash_table<int, int, chash> hash_t;
17
18 ll s[mt];
19 hash_t h[mt];
20 #define lfs p << 1
21 #define rfs p << 1 | 1
22 #define mkcf ll cf = (lf + rf) >> 1
23 #define ci const int &
24 void modifyS(int p, int lf, int rf, ci pos, ci v)
25 {
26     if (lf == rf)
27     {
28         s[p] += v;
29         return;
30     }
31     mkcf;
32     if (pos <= cf)
33     {
34         modifyS(lfs, lf, cf, pos, v);
35     }
36     else
37     {
38         modifyS(rfs, cf + 1, rf, pos, v);
39     }
40     s[p] = s[lfs] + s[rfs];
41 }
42 ll queryS(int p, int lf, int rf, ci lc, ci rc)
43 {
44     if (lc <= lf && rf <= rc)
45     {
46         return s[p];
47     }
48     mkcf;
49     ll ans = 0;
50     if (lc <= cf)
51     {
52         ans += queryS(lfs, lf, cf, lc, rc);
53     }
54     if (cf + 1 <= rc)
55     {
56         ans += queryS(rfs, cf + 1, rf, lc, rc);
57     }
58     return ans;
59 }

```

```

59 void modifyC(int p, int lf, int rf, ci pos, ci c, ci v)
60 {
61     h[p][c] += v;
62     if (lf == rf)
63     {
64         return;
65     }
66     mkcf;
67     if (pos <= cf)
68     {
69         modifyC(lfs, lf, cf, pos, c, v);
70     }
71     else
72     {
73         modifyC(rfs, cf + 1, rf, pos, c, v);
74     }
75 }
76 #define colors const vector<ll> &
77 ll check(ll p, ll lf, ll rf, colors cs)
78 {
79     ll sum = 0;
80     for (auto &c : cs)
81     {
82         sum += h[p][c];
83     }
84     return sum == rf - lf + 1;
85 }
86 const ll fail = -1;
87 ll queryCR(int p, int lf, int rf, int lc, colors cs)
88 {
89     if (rf < lc)
90     {
91         return fail;
92     }
93     if (lc <= lf)
94     {
95         if (check(p, lf, rf, cs))
96         { // still ok, move right
97             return -1;
98         }
99         if (lf == rf)
100         { // first to fail is lf, then ok is lf-1
101             return lf - 1;
102         }
103     }
104     mkcf;
105     ll pos = queryCR(lfs, lf, cf, lc, cs);
106     if (pos != fail)
107     {
108         return pos;
109     }
110     return queryCR(rfs, cf + 1, rf, lc, cs);
111 }
112 ll queryCL(int p, int lf, int rf, int rc, colors cs)
113 {
114     if (lf > rc)

```

```

115     {
116         return fail;
117     }
118     if (rf <= rc)
119     {
120         if (check(p, lf, rf, cs))
121         {
122             return -1;
123         }
124         if (lf == rf)
125         {
126             return lf + 1;
127         }
128     }
129     mkcf;
130     ll pos = queryCL(rfs, cf + 1, rf, rc, cs);
131     if (pos != fail)
132     {
133         return pos;
134     }
135     return queryCL(lfs, lf, cf, rc, cs);
136 }
137
138 ll n, q, c[mn], v[mn];
139 void solve()
140 {
141     cin >> n >> q;
142     for (ll i = 1; i <= n; ++i)
143     {
144         cin >> c[i];
145         modifyC(1, 1, n, i, c[i], 1);
146     }
147     for (ll i = 1; i <= n; ++i)
148     {
149         cin >> v[i];
150         modifyS(1, 1, n, i, v[i]);
151     }
152     for (ll op, x, y; q--;")
153     {
154         cin >> op >> x >> y;
155         if (op == 1)
156         {
157             modifyC(1, 1, n, x, c[x], -1);
158             c[x] = y;
159             modifyC(1, 1, n, x, c[x], 1);
160         }
161         else if (op == 2)
162         {
163             modifyS(1, 1, n, x, y - v[x]);
164             v[x] = y;
165         }
166         else
167         {
168             bool ok = false;
169             vector<ll> vec;
170             for (ll i = 1, z; i <= y; ++i)

```

```

171     {
172         cin >> z;
173         if (c[x] == z)
174         {
175             ok = true;
176         }
177         vec.emplace_back(z);
178     }
179     if (!ok)
180     {
181         cout << "0\n";
182         continue;
183     }
184
185     ll l = queryCL(1, 1, n, x, vec);
186     ll r = queryCR(1, 1, n, x, vec);
187     l = l == fail ? 1 : l;
188     r = r == fail ? n : r;
189     cout << queryS(1, 1, n, l, r) << '\n';
190 }
191 }
192
193 for (ll i = 1; i <= 4 * n; ++i)
194 {
195     h[i].clear();
196     s[i] = 0;
197 }
198
199
200 signed main()
201 {
202     ios::sync_with_stdio(false), cin.tie(0);
203     ll t;
204     cin >> t;
205     while (t--)
206     {
207         solve();
208     }
209     return 0;
210 }

```

树状数组上倍增：

```

1 #include <bits/stdc++.h>
2 #define MAXN ((int) 3e5)
3 using namespace std;
4
5 // 比 unordered_map 更快的哈希表
6 #include <ext/pb_ds/assoc_container.hpp>
7 using namespace __gnu_pbds;
8 const int RANDOM =
9     chrono::high_resolution_clock::now().time_since_epoch().count();
10 struct hash {
11     int operator()(int x) const { return x ^ RANDOM; }
12 };

```

```

12 typedef gp_hash_table<int, int, chash> hash_t;
13
14 int n, q, C[MAXN + 10], V[MAXN + 10];
15
16 hash_t colTree[MAXN + 10];
17 long long smTree[MAXN + 10];
18
19 int lb(int x) { return x & (-x); }
20
21 // val == -1: 把位置 pos 的颜色 c 删掉
22 // val == 1: 把位置 pos 的颜色设为 c
23 void addCol(int pos, int c, int val) {
24     for (; pos <= n; pos += lb(pos)) colTree[pos][c] += val;
25 }
26
27 // 查询 vec 里的所有颜色在前 pos 个位置中一共出现了几次
28 int queryCol(int pos, vector<int> &vec) {
29     int ret = 0;
30     for (; pos; pos -= lb(pos)) for (int c : vec) {
31         auto it = colTree[pos].find(c);
32         if (it != colTree[pos].end()) ret += it->second;
33     }
34     return ret;
35 }
36
37 // 树状数组上倍增,
38 // 返回值 l 满足 vec 里所有颜色在区间 [l, lim] 中出现的总次数等于区间长度, 且 l 最小
39 int gao1(int lim, vector<int> &vec) {
40     int base = queryCol(lim, vec);
41     if (base == lim) return 1;
42
43     int b;
44     for (b = 1; b <= n; b <= 1);
45
46     int now = 0, cnt = 0;
47     for (b >= 1; b; b >= 1) {
48         int nxt = now | b, tmp = 0;
49         for (int c : vec) {
50             auto it = colTree[nxt].find(c);
51             if (it != colTree[nxt].end()) tmp += it->second;
52         }
53         if (nxt > lim || base - (cnt + tmp) == lim - nxt) {
54             // do nothing
55         } else {
56             now = nxt; cnt += tmp;
57         }
58     }
59     return now + 2;
60 }
61
62 // 树状数组上倍增,
63 // 返回值 r 满足 vec 里所有颜色在区间 [lim, r] 中出现的总次数等于区间长度, 且 r 最大
64 int gao2(int lim, vector<int> &vec) {
65     int base = queryCol(lim, vec);
66
67     int b;

```

```

68     for (b = 1; b <= n; b <= 1) {
69
70     int now = 0, cnt = 0;
71     for (b >= 1; b; b >= 1) {
72         int nxt = now | b, tmp = 0;
73         for (int c : vec) {
74             auto it = colTree[nxt].find(c);
75             if (it != colTree[nxt].end()) tmp += it->second;
76         }
77         if (nxt < lim || (cnt + tmp) - base == nxt - lim) {
78             now = nxt; cnt += tmp;
79         } else {
80             // do nothing
81         }
82     }
83     return now;
84 }
85
86 // 位置 pos 的权值增加 val
87 void addSm(int pos, long long val) {
88     for (; pos <= n; pos += 1b(pos)) smTree[pos] += val;
89 }
90
91 // 求前 pos 个位置的权值之和
92 long long querySm(int pos) {
93     long long ret = 0;
94     for (; pos; pos -= 1b(pos)) ret += smTree[pos];
95     return ret;
96 }
97
98 void solve() {
99     scanf("%d%d", &n, &q);
100    for (int i = 1; i <= n; i++) {
101        scanf("%d", &c[i]);
102        addCol(i, c[i], 1);
103    }
104    for (int i = 1; i <= n; i++) {
105        scanf("%d", &v[i]);
106        addSm(i, v[i]);
107    }
108
109    while (q--) {
110        int op, x, y; scanf("%d%d%d", &op, &x, &y);
111        if (op == 1) {
112            addCol(x, c[x], -1);
113            addCol(x, y, 1);
114            c[x] = y;
115        } else if (op == 2) {
116            addSm(x, y - v[x]);
117            v[x] = y;
118        } else {
119            bool ok = false;
120            vector<int> vec;
121            for (int i = 1; i <= y; i++) {
122                int z; scanf("%d", &z);
123                if (c[x] == z) ok = true;

```

```

124         vec.push_back(z);
125     }
126     if (!ok) { printf("0\n"); continue; }
127
128     int L = gao1(x, vec), R = gao2(x, vec);
129     printf("%lld\n", querySm(R) - querySm(L - 1));
130 }
131 }
132
133 for (int i = 1; i <= n; i++) colTree[i].clear();
134 for (int i = 1; i <= n; i++) smTree[i] = 0;
135 }
136
137 int main() {
138     int tcase; scanf("%d", &tcase);
139     while (tcase--) solve();
140     return 0;
141 }

```

## 猫树

洛谷P3865-求区间  $\max$ ,  $1 \leq n \leq 10^5$ ,  $1 \leq m \leq 2 \times 10^6$ ,  $a_i \in [0, 10^9]$

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define sc(x) scanf("%lld", &x)
4 typedef long long ll;
5 #define mn 131082 // 1<<log(1e5,2)+10
6 #define ml 20
7 ll n, m, a[mn], n2, lg[mn * 2], l, r;
8 ll pos[mn], gcd[mn][ml];
9 #define lfs p << 1
10 #define rfs p << 1 | 1
11 #define mkcf ll cf = (lf + rf) >> 1
12 void build(ll p, ll lf, ll rf, ll dep)
13 {
14     if (lf == rf)
15     {
16         pos[lf] = p;
17         return;
18     }
19     mkcf;
20     build(lfs, lf, cf, dep + 1);
21     build(rfs, cf + 1, rf, dep + 1);
22     gcd[cf][dep] = a[cf];
23     for (ll i = cf - 1; i >= lf; --i)
24     {
25         gcd[i][dep] = max(gcd[i + 1][dep], a[i]);
26     }
27     for (ll i = cf + 1; i <= rf; ++i)
28     {
29         gcd[i][dep] = max(gcd[i - 1][dep], a[i]);
30     }

```

```

31 }
32 signed main()
33 {
34     sc(n), sc(m);
35     for (ll i = 1; i <= n; ++i)
36     {
37         sc(a[i]);
38     }
39     for (n2 = 1; n2 < n; n2 <= 1) //满二叉树下节点有n2个
40     ;
41     for (ll i = 2; i <= n2 * 2; ++i)
42     {
43         lg[i] = lg[i / 2] + 1;
44     }
45     build(1, 1, n2, 1);
46     while (m--)
47     {
48         sc(l), sc(r);
49         if (l == r) //猫树不存叶子节点
50         {
51             printf("%lld\n", a[l]);
52             continue;
53         }
54         ll k = lg[pos[r]] - lg[pos[l] ^ pos[r]];
55         printf("%lld\n", max(gcd[l][k], gcd[r][k]));
56     }
57     return 0;
58 }

```

## 主席树

### 可持久化数组

洛谷P3919: 给定长为  $n(1 \leq n \leq 10^6)$  值域在  $[-10^9, 10^9]$  初始版本号为 0 的数组, 有  $m(1 \leq m \leq 10^6)$  次操作:

1. 修改版本号为  $ver$  的数组下标为  $loc$  位置值为  $x$
2. 查询版本为  $ver$  的数组下标为  $loc$  位置值

每次操作结束后, 将当前版本号数组复制为  $ver$  数组。

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef int ll;
4 #define sc(x) scanf("%d", &x)
5 #define mn 1000010
6 #define lg 24 // log(mn)+4mn
7 struct node
8 {
9     ll l, r, v;
10 } t[mn * lg];
11 ll n, m, rot[mn], a[mn], cnt, cmd, loc, ver, x;
12 #define mkcf ll cf = (lf + rf) >> 1
13 ll build(ll lf, ll rf)

```

```

14 {
15     ll p = ++cnt;
16     if (lf == rf)
17     {
18         t[p].v = a[lf];
19     }
20     else
21     {
22         mkcf;
23         t[p].l = build(lf, cf);
24         t[p].r = build(cf + 1, rf);
25     }
26     return p;
27 }
28 ll update(ll r, ll lf, ll rf)
29 {
30     ll p = ++cnt;
31     if (lf == rf)
32     {
33         t[p].v = x;
34     }
35     else
36     {
37         mkcf;
38         if (loc <= cf)
39         {
40             t[p].l = update(t[r].l, lf, cf);
41             t[p].r = t[r].r;
42         }
43         else
44         {
45             t[p].l = t[r].l;
46             t[p].r = update(t[r].r, cf + 1, rf);
47         }
48     }
49     return p;
50 }
51 void query(ll r, ll lf, ll rf)
52 {
53     if (lf == rf)
54     {
55         x = t[r].v;
56         return;
57     }
58     mkcf;
59     if (loc <= cf)
60     {
61         query(t[r].l, lf, cf);
62     }
63     else
64     {
65         query(t[r].r, cf + 1, rf);
66     }
67 }
68 signed main()
69 {

```

```

70     sc(n), sc(m);
71     for (ll i = 1; i <= n; ++i)
72     {
73         sc(a[i]);
74     }
75     rot[0] = build(1, n);
76     for (ll i = 1; i <= m; ++i)
77     {
78         sc(ver), sc(cmd), sc(loc);
79         if (cmd == 1)
80         {
81             sc(x);
82             rot[i] = update(rot[ver], 1, n);
83         }
84         else
85         {
86             query(rot[ver], 1, n);
87             rot[i] = rot[ver];
88             printf("%d\n", x);
89         }
90     }
91     return 0;
92 }
```

## 静态区间第k小

洛谷P3824: 长为  $n$  ( $1 \leq n \leq 2 \times 10^5$ ) 值域的序列, 有  $m$  ( $1 \leq m \leq 2 \times 10^5$ ) 次询问, 问区间  $[l, r]$  内严格第  $k$  小(不去重)的值是什么

设去重后长度为  $n'$ , 设主席树, 版本为  $i$  ( $0 \leq i \leq n$ ) 的主席树代表前  $i$  个数, 当前节点维护排名(从小到大)区间  $[l, r]$ , 其值代表在该区间内有多少个数。对叶子节点, 表示特定排名数字数目。

可以通过前缀和思想来建树, 每次继承上一个根节点主席树。通过二分查找找到第  $i$  个数的排名。

对于查询, 对前缀和求差分, 对查询的  $[l, r]$  区间, 取版本为  $r$  的主席树节点与版本为  $l - 1$  的同一区间主席树节点作差, 即得  $[1, r]$  出现个数减去  $[1, l - 1]$  出现个数即  $[l, r]$  出现个数。如果左子区间出现个数  $s$  大于等于  $k$ , 也就是说左子区间里有当前子区间的第 1 到第  $s$  小, 包含了第  $k$  小, 所以需要在左子区间里继续找。否则, 证明第  $k$  小在右子区间, 因为左子区间有前  $s$  小, 所以分配到右子时, 要减去  $s$ 。

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #define sc(x) scanf("%lld", &x)
5 #define lg 22
6 #define mn 200010
7 ll m, n, q, a[mn], l[mn * lg], r[mn * lg], v[mn * lg], rot[mn], lc, rc, k,
8 cnt, a0[mn];
9 #define mkcf ll cf = (lf + rf) >> 1
10 ll build(ll lf, ll rf)
11 {
12     ll p = ++cnt;
13     if (lf != rf)
14     {
```

```

14     mkcf;
15     l[p] = build(lf, cf);
16     r[p] = build(cf + 1, rf);
17 }
18 return p;
19 }
20 l update(l l, l lf, l rf)
21 {
22     l p = ++cnt;
23     l[p] = l[q], r[p] = r[q], v[p] = v[q] + 1;
24     if (lf != rf)
25     {
26         mkcf;
27         if (k <= cf)
28         {
29             l[p] = update(l[q], lf, cf);
30         }
31         else
32         {
33             r[p] = update(r[q], cf + 1, rf);
34         }
35     }
36     return p;
37 }
38 l query(l p, l q, l lf, l rf, l i)
39 {
40     if (lf == rf)
41     {
42         return lf;
43     }
44     mkcf;
45     l j = v[l[q]] - v[l[p]];
46     if (j >= i)
47     {
48         return query(l[p], l[q], lf, cf, i);
49     }
50     else
51     {
52         return query(r[p], r[q], cf + 1, rf, i - j);
53     }
54 }
55 signed main()
56 {
57     sc(m), sc(q);
58     for (l i = 1; i <= m; ++i)
59     {
60         sc(a[i]);
61         a0[i] = a[i];
62     }
63     sort(a + 1, a + 1 + m);
64     n = unique(a + 1, a + 1 + m) - (a + 1);
65     rot[0] = build(1, n);
66     for (l i = 1; i <= m; ++i)
67     {
68         k = lower_bound(a + 1, a + 1 + n, a0[i]) - a;
69         rot[i] = update(rot[i - 1], 1, n);

```

```

70     }
71     while (q--)
72     {
73         sc(lc), sc(rc), sc(k);
74         ll i = query(rot[lc - 1], rot[rc], 1, n, k);
75         printf("%lld\n", a[i]);
76     }
77     return 0;
78 }

```

## 树状数组

一种  $O(n)$  建树方法：

```

1 // C++ Version
2 // O(n)建树
3 void init() {
4     for (int i = 1; i <= n; ++i) {
5         t[i] += a[i];
6         int j = i + lowbit(i);
7         if (j <= n) t[j] += t[i];
8     }
9 }

```

## 静态区间最值

以静态区间最小值为例。不支持可以反向变化的最值维护

```

1 #include <bits/stdc++.h>
2 // #pragma warning(disable:6031)
3 #define MAXN 100002
4 const int BIG = 0xffffffff;
5 using namespace std;
6 int tr[MAXN], n, m, lf, rf, a[MAXN];
7 inline int lowbit(int& k)
8 {
9     return k & -k;
10 }
11 inline void add(int x, int& k)
12 {
13     while (x <= n)
14     {
15         if (k < tr[x]) tr[x] = k; //这两行换成min(tr[x], k)也可
16         else return; //但是那样的话会慢些
17         x += lowbit(x);
18     }
19 }
20 inline int query(int lf, int rf)
21 {
22     int dtr = rf, res = BIG;
23     while (dtr >= lf)

```

```

24     {
25         if (drt - lowbit(drt) > 1f)
26         {
27             res = min(res, tr[drt]);
28             drt -= lowbit(drt);
29         }
30     else
31     {
32         res = min(res, a[drt]);
33         --drt;
34     }
35 }
36     return res;
37 }
38 signed main()
39 {
40     scanf("%d%d", &n, &m);
41     for (int i = 1; i <= n; ++i)//不能在下面那个for
42     {
43         tr[i] = BIG;
44     }
45     for (int i = 1; i <= n; ++i)
46     {
47         scanf("%d", &a[i]);
48         add(i, a[i]);
49     }
50     while (m--)
51     {
52         scanf("%d%d", &lf, &rf);
53         printf("%d ", query(lf, rf));
54     }
55     return 0;
56 }

```

## 动态整体第k小

洛谷P1168以求数组前  $1, 3, 5, \dots$  项中位数为例  $n \leq 10^5, a \leq 10^9$

```

1 #include<iostream>
2 #include<cstdio>
3 #include<algorithm>
4 using namespace std;
5
6 int n,tot;
7 const int maxn=1e5+10;
8 int bit[maxn];
9 int a[maxn],b[maxn];
10
11 inline int lowbit(int x)
12 {
13     return x&-x;
14 }
15 inline void add(int pos,int x)

```

```

16 {
17     for(int i=pos;i<=tot;i+=lowbit(i))bit[i]+=x;
18 }
19 inline int find_kth(int k)
20 {
21     int ans=0,now=0; //这里主要解释一下这个的原理 ans就是答案, now是
比当前找到的数的大小的数字的个数。
22     for(int i=20;i>=0;i--) //2^20可以说很大了, 满足我们的需求了, 我们按照
20倍增就可以
23     {
24         ans+=(1<<i); //先让答案加上去, 试试
25         if(ans>tot||now+bit[ans]>=k)ans-=(1<<i); //如果超了总体的最大值(防止数组越
界), 或者是 超过了k个, 就退回去, 这里注意是大于等于, 因为要考虑有重复元素, 所以我们找的其实
是一个满足小于他的个数小于k的最大数
26         else now+=bit[ans]; //能加就加上, 这里不用怕加到了原来的数, 因为树状数组的结构
使这个新增出来的数就是多出来的那一条枝
27     }
28     return ans+1; //然后加上1就是答案啦
29 }
30
31 int main()
32 {
33     scanf("%d",&n);
34     for(int i=1;i<=n;i++)
35     {
36         scanf("%d",&a[++tot]); //读个入
37         b[tot]=a[tot];
38     }
39     sort(a+1,a+1+n); //排个序
40     tot=unique(a+1,a+1+tot)-a-1; //去个重
41     for(int i=1;i<=n;i++)b[i]=lower_bound(a+1,a+1+tot,b[i])-a; //离散化一下
42     for(int i=1;i<=n;i++)
43     {
44         add(b[i],1); //动态加点
45         if(i&1)printf("%d\n",a[find_kth((i+1)>>1)]); //查kth
46     }
47     return 0;
48 }

```

## 动态区间第k小

洛谷P2617:给定长为  $n$  的数列  $a$  , 支持  $m$  次两种操作:  $1 \leq n, m \leq 10^5, 0 \leq a_i, y \leq 10^9$

- `Q l r k` 查询区间第  $k$  小
- `C x y` 将  $a_x$  改为  $y$

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define sc(x) scanf("%lld", &x)
4 typedef long long ll;
5 #define mn 100010
6 #define lg 310
7 ll v[mn * lg], l[mn * lg], r[mn * lg], cnt, rot[mn];

```

```

8 11 n, m, a[mn], h[mn * 2], hs;
9 #define mkcf 11 cf = (lf + rf) >> 1
10 #define lowbit(x) (x & -x)
11 void update(11 &p, 11 lf, 11 rf, const 11 &pos, const 11 &x)
12 {
13     if (!p)
14     {
15         p = ++cnt;
16     }
17     v[p] += x;
18     if (lf != rf)
19     {
20         mkcf;
21         if (pos <= cf)
22         {
23             update(l[p], lf, cf, pos, x);
24         }
25         else
26         {
27             update(r[p], cf + 1, rf, pos, x);
28         }
29     }
30 }
31 void updates(11 pos, 11 x)
32 {
33     11 k = lower_bound(h + 1, h + 1 + hs, a[pos]) - h;
34     for (11 i = pos; i <= n; i += lowbit(i))
35     {
36         update(rot[i], 1, hs, k, x);
37     }
38 }
39 11 addcnt, addid[lg], subcnt, subid[lg];
40 11 query(11 lf, 11 rf, 11 k)
41 {
42     if (lf == rf)
43     {
44         return lf;
45     }
46     mkcf;
47     11 res = 0;
48     for (11 i = 1; i <= addcnt; ++i)
49     {
50         res += v[l[addid[i]]];
51     }
52     for (11 i = 1; i <= subcnt; ++i)
53     {
54         res -= v[l[subid[i]]];
55     }
56     if (k <= res)
57     {
58         for (11 i = 1; i <= addcnt; ++i)
59         {
60             addid[i] = l[addid[i]];
61         }
62         for (11 i = 1; i <= subcnt; ++i)
63         {

```

```

64         subid[i] = l[subid[i]];
65     }
66     return query(lf, cf, k);
67 }
68 else
69 {
70     for (ll i = 1; i <= addcnt; ++i)
71     {
72         addid[i] = r[addid[i]];
73     }
74     for (ll i = 1; i <= subcnt; ++i)
75     {
76         subid[i] = r[subid[i]];
77     }
78     return query(cf + 1, rf, k - res);
79 }
80 }
81 ll querys(ll lf, ll rf, ll k)
82 {
83     addcnt = subcnt = 0;
84     for (ll i = rf; i; i -= lowbit(i))
85     {
86         addid[++addcnt] = rot[i];
87     }
88     for (ll i = lf - 1; i; i -= lowbit(i))
89     {
90         subid[++subcnt] = rot[i];
91     }
92     return query(1, hs, k);
93 }
94 struct queries
95 {
96     ll cmd, l, r, k;
97 } q[mn * 2];
98 char c[10];
99 signed main()
100 {
101     sc(n), sc(m), hs = n;
102     for (ll i = 1; i <= n; ++i)
103     {
104         sc(a[i]), h[i] = a[i];
105     }
106     for (ll i = 1; i <= m; ++i)
107     {
108         scanf("%s", c), sc(q[i].l), sc(q[i].r);
109         if (c[0] == 'Q')
110         {
111             sc(q[i].k);
112         }
113         else
114         {
115             q[i].cmd = 1, h[++hs] = q[i].r;
116         }
117     }
118     sort(h + 1, h + 1 + hs);
119     hs = unique(h + 1, h + 1 + hs) - (h + 1);

```

```

120     for (ll i = 1; i <= n; ++i)
121     {
122         updates(i, 1);
123     }
124     for (ll i = 1; i <= m; ++i)
125     {
126         if (q[i].cmd == 0)
127         {
128             printf("%lld\n", h[querys(q[i].l, q[i].r, q[i].k)]);
129         }
130         else
131         {
132             updates(q[i].l, -1);
133             a[q[i].l] = q[i].r;
134             updates(q[i].l, 1);
135         }
136     }
137     return 0;
138 }
```

## 二维树状数组

维护矩阵，支持单点修改和区间查询。可以维护一维能维护的内容，区间即二维上的子矩阵。注意空间复杂度是  $O(n^2)$

以区间求和为例，更新和查询函数如下：

```

1 for (int x = i; x < A.length; x += lowbit(x))
2     for (int y = j; y < A[i].length; y += lowbit(y))
3         c[x][y] += delta;
```

```

1 int result = 0;
2 for (int x = i; x > 0; x -= lowbit(x))
3     for (int y = j; y > 0; y -= lowbit(y))
4         result += c[x][y];
5 return result;
```

输入  $x \ n \ m$  给定  $n \times m$  矩阵，接下来输入若干行，每行可能输入  $l \ a \ b \ c \ d \ delta$  代表将顶点为  $(a, b), (c, d)$  的矩形区域每个数字加上  $delta$ ，或输入  $k \ a \ b \ c \ d$  代表求矩形区域和。

$1 \leq n, m \leq 2048, -500 \leq delta \leq 500, line \leq 2 \times 10^5$ ，过程在  $int$  内

考虑用二维差分数组  $d[n][m]$  维护区间修改的二维前缀和查询。设原数组为  $a[n][m]$ ，二维前缀和数组为  $s[n][m]$ ，则：

$$\begin{aligned}
 s[x][y] &= \sum_{h=1}^x \sum_{k=1}^y a[h][k] \\
 a[i][j] &= \sum_{i=1}^h \sum_{j=1}^k d[i][j] \\
 \therefore s[x][y] &= \sum_{h=1}^x \sum_{k=1}^y \sum_{i=1}^h \sum_{j=1}^k d[i][j]
 \end{aligned}$$

对求和化简，容易可得：

$$s[x][y] = \sum_{i=1}^x \sum_{j=1}^y d[i][j] \times (x - i + 1) \times (y - j + 1)$$

分解得：

$$\sum_{i=1}^x \sum_{j=1}^y d[i][j] \cdot (xy + x + y + 1) - d[i][j] \cdot i(y + 1) - d[i][j] \cdot j(x + 1) + d[i][j] \cdot ij$$

可以维护四个差分数组： $d[i][j]$ ,  $d[i][j] \cdot i$ ,  $d[i][j] \cdot j$ ,  $d[i][j] \cdot ij$ 。

接下来就可以跑二维差分模板和二维树状数组模板了：

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef int ll;
4 #define sc(x) scanf("%d", &x)
5 #define mn 2050
6 ll n, m, a, b, c, d, delta;
7 char op[3];
8 ll lowbit(ll &x) { return x & -x; }
9 struct BIT
10 {
11     ll t[mn][mn];
12     void add(ll lf, ll rf, ll v)
13     {
14         for (ll i = lf; i <= n; i += lowbit(i))
15             for (ll j = rf; j <= m; j += lowbit(j))
16                 t[i][j] += v;
17     }
18     ll query(ll lf, ll rf)
19     {
20         ll res = 0;
21         for (ll i = lf; i; i -= lowbit(i))
22             for (ll j = rf; j; j -= lowbit(j))
23                 res += t[i][j];
24     }
25     return res;
26 }
27 } x, xi, xj, xij;
28 void add(ll lf, ll rf, ll v)
29 {
30     x.add(lf, rf, v), xi.add(lf, rf, v * lf), xj.add(lf, rf, v * rf),
31     xij.add(lf, rf, v * lf * rf);
32 }
33 ll query(ll lf, ll rf)
34 {
35     return x.query(lf, rf) * (lf * rf + lf + rf + 1) - xi.query(lf, rf) *
36     (rf + 1) - xj.query(lf, rf) * (lf + 1) + xij.query(lf, rf);
37 }
38 signed main()
39 {
40     sc("X"), sc(n), sc(m);
41     while (~scn("%s", op))
42     {
43         sc(a), sc(b), sc(c), sc(d);
```

```

41     if (op[0] == 'L')
42     {
43         sc(delta);
44         add(a, b, delta), add(a, d + 1, -delta), add(c + 1, b, -delta),
45         add(c + 1, d + 1, delta);
46     }
47     else
48         printf("%d\n", query(c, d) - query(a - 1, d) - query(c, b - 1) +
49             query(a - 1, b - 1));
50     }
51     return 0;
52 }

```

## 平衡树

使用示例: (洛谷P6136) (目前该题平衡树均为别人的代码,后续版本可能改)

题意: 输入  $n, m (1 \leq n \leq 10^5, 1 \leq m \leq 10^6)$ , 输入  $a_i (0 \leq a_i < 2^{30})$ , 接下来有 6 种在线操作 ( $x = x \oplus last$ ):

1. 插入整数  $x (0 \leq x < 2^{30})$
2. 删除整数  $x$  (若有多个相同, 只删一个)
3. 查询整数  $x$  的排名(比它小的数个数 +1)
4. 查询排名为  $x$  的数(不存在时查小于  $x$  的最大数, 保证  $x$  不越当前界)
5. 求  $x$  前驱(小于  $x$  的最大的数)
6. 求  $x$  后继(大于  $x$  的最小的数)

此题用 `pb_ds` 解法见本模板后面

## FHQ-Treap

```

1 #include<cstdio>
2 #define maxn 1100010
3 struct pair{
4     int a,b;
5     pair(int a_=0,int b_=0) { a=a_; b=b_; }
6 };
7 int read(){
8     int ans=0; char ch=getchar();
9     while(ch>'9' || ch<'0') ch=getchar();
10    while(ch<='9' && ch>='0'){
11        ans=ans*10+ch-'0';
12        ch=getchar();
13    }
14    return ans;
15 }
16 int key[maxn],wei[maxn],size[maxn],son[maxn][2];
17 int n,m,cnt,ans,seed=1,root,last;
18 int rand1() { return seed*=19260817; }
19 inline void pushup(int u)
20     { size[u]=size[son[u][0]]+size[son[u][1]]+1; }
21 pair split(int u,int k){

```

```

22     if(!u) return pair(0,0);
23     if(key[u]<k){
24         pair t=split(son[u][1],k);
25         son[u][1]=t.a;
26         pushup(u);
27         return pair(u,t.b);
28     }else{
29         pair t=split(son[u][0],k);
30         son[u][0]=t.b;
31         pushup(u);
32         return pair(t.a,u);
33     }
34 }
35 int merge(int u,int v){
36     if(!u||!v) return u+v;
37     if(wei[u]<wei[v]){
38         son[u][1]=merge(son[u][1],v);
39         pushup(u);
40         return u;
41     }else{
42         son[v][0]=merge(u,son[v][0]);
43         pushup(v);
44         return v;
45     }
46 }
47 void insert(int k){
48     key[++cnt]=k; wei[cnt]=rand1(); size[cnt]=1;
49     pair t=split(root,k);
50     root=merge(merge(t.a,cnt),t.b);
51 }
52 void eraser(int k){
53     pair x,y;
54     x=split(root,k);
55     y=split(x.b,k+1);
56     y.a=merge(son[y.a][0],son[y.a][1]);
57     root=merge(x.a,merge(y.a,y.b));
58 }
59 int find1(int k){
60     int re;
61     pair t=split(root,k);
62     re=size[t.a]+1;
63     root=merge(t.a,t.b);
64     return re;
65 }
66 int find2(int k){
67     int pos=root;
68     while(pos){
69         if(k==size[son[pos][0]]+1) return key[pos];
70         if(k<=size[son[pos][0]]) pos=son[pos][0];
71         else { k-=size[son[pos][0]]+1; pos=son[pos][1]; }
72     }
73 }
74 int lst(int k) { return find2(find1(k)-1); }
75 int nxt(int k) { return find2(find1(k+1)); }
76 int main(){
77     n=read(); m=read();

```

```

78     for(int i=1;i<=n;i++){
79         int a=read();
80         insert(a);
81     }
82     for(int i=1;i<=m;i++){
83         int o=read(),x; x=read();
84         if(o==1) insert(x^last);
85         if(o==2) eraser(x^last);
86         if(o==3) { last=find1(x^last); ans^=last; }
87         if(o==4) { last=find2(x^last); ans^=last; }
88         if(o==5) { last=lst(x^last); ans^=last; }
89         if(o==6) { last=nxt(x^last); ans^=last; }
90     }
91     printf("%d\n",ans);
92     return 0;
93 }

```

## AVL

```

1 #include <cstdio>
2 const int N=1e5+1e6+5;
3 int n,m,k,x,cnt,root,last,ans;
4 struct jd {
5     int l,r,val,size;
6     int ht;//树高
7 }t[N];
8 inline int read(){
9     int x=0,flag=0;char ch=getchar();
10    while(ch<'0' || ch>'9') {flag|=(ch=='-');ch=getchar();}
11    while(ch>='0' && ch<='9') {x=(x<<3)+(x<<1)+ch-'0';ch=getchar();}
12    return flag?-x:x;
13 }
14 inline int newt(int val) {
15     t[++cnt]=(jd){0,0,val,1,0};
16     return cnt;
17 }
18 inline int mx(int x,int y) {return x>y?x:y;}
19
20 inline int bf(int now) {return t[t[now].l].ht-t[t[now].r].ht;}
21 inline void update(int now) {
22     t[now].size=t[t[now].l].size+t[t[now].r].size+1;
23     t[now].ht=mx(t[t[now].l].ht,t[t[now].r].ht)+1;
24 }
25 inline void lt(int &now) {//左旋
26     int y=t[now].r;
27     t[now].r=t[y].l;
28     t[y].l=now;
29     now=y;
30     update(t[now].l);
31     update(now);
32 }
33 inline void rt(int &now) {//右旋
34     int y=t[now].l;

```

```

35     t[now].l=t[y].r;
36     t[y].r=now;
37     now=y;
38     update(t[now].r);
39     update(now);
40 }
41
42 inline void ch(int &now) { //对now子树进行维护
43     int s=bf(now);
44     if(s>1) {
45         int ss=bf(t[now].l);
46         if(ss>0) rt(now); //LL
47         else lt(t[now].l),rt(now); //LR
48     }
49     else if(s<-1){
50         int ss=bf(t[now].r);
51         if(ss<0) lt(now); //RR
52         else rt(t[now].r),lt(now); //RL
53     }
54     else if(now) update(now);
55 }
56
57 void ins(int &now,int val) {
58     if(!now) now=newt(val);
59     else if(t[now].val<=val) ins(t[now].r,val);
60     else ins(t[now].l,val);
61     ch(now);
62 }
63
64 void del(int &now,int val) { //我是使用将其旋转至叶子节点后删除的
65     if(t[now].val==val) {
66         if(!t[now].l||!t[now].r) now=t[now].l^t[now].r;
67         else if(bf(now)<0) rt(now),del(now,val); //右子树高, 旋转至左儿子位置可略
68        减小维护的常数
69         else lt(now),del(now,val); //同上
70     }
71     else if(val>=t[now].val) del(t[now].r,val);
72     else del(t[now].l,val);
73     ch(now);
74 }
75
76 inline int rank(int val) {
77     int now=root,s=0;
78     while(now) {
79         if(t[now].val>=val) now=t[now].l;
80         else s+=t[t[now].l].size+1,now=t[now].r;
81     }
82     return s+1;
83 }
84
85 inline int num(int val) {
86     int now=root;
87     while(now) {
88         if(t[t[now].l].size+1==val)
89             return t[now].val;
90         if(t[t[now].l].size>=val) now=t[now].l;
91     }
92 }

```

```

90         else val=val-t[t[now].l].size-1,now=t[now].r;
91     }
92 }
93 int main() {
94     n=read();m=read();
95     for(int i=1;i<=n;++i)
96         ins(root,read());
97     while(m--) {
98         k=read();x=read();
99         x^=last;
100        switch(k) {
101            case 1:
102                ins(root,x);
103                break;
104            case 2:
105                del(root,x);
106                break;
107            case 3:
108                ans^=last=rank(x);
109                break;
110            case 4:
111                ans^=last=num(x);
112                break;
113            case 5:
114                ans^=last=num(rank(x)-1);
115                break;
116            case 6:
117                ans^=last=num(rank(x+1));
118                break;
119        }
120    }
121    printf("%d\n",ans);
122    return 0;
123 }

```

## Splay

洛谷P3391：对长为  $n(1 \leq n \leq 10^5)$  的区间，进行  $m(1 \leq m \leq 10^5)$  次  $[l, r]$  翻转，输出最后结果

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef int ll;
4 #define mn 200002
5 struct node
6 {
7     ll ch[2], ff, v, size, mark;
8     void init(ll x, ll fa)
9     {
10         ff = ch[0] = ch[1] = 0;
11         size = 1;
12         v = x;
13         ff = fa;

```

```

14     }
15 } t[mn];
16 ll n, rot, m, tot, lf, rf;
17 void pushuf(ll x)
18 {
19     t[x].size = t[t[x].ch[0]].size + t[t[x].ch[1]].size + 1;
20 }
21 void pushdf(ll x)
22 {
23     if (t[x].mark)
24     {
25         t[t[x].ch[0]].mark ^= 1;
26         t[t[x].ch[1]].mark ^= 1;
27         t[x].mark = 0;
28         swap(t[x].ch[0], t[x].ch[1]);
29     }
30 }
31 void rotate(ll x)
32 {
33     ll y = t[x].ff;
34     ll z = t[y].ff;
35     ll k = (t[y].ch[1] == x);
36     t[z].ch[t[z].ch[1] == y] = x;
37     t[x].ff = z;
38     t[y].ch[k] = t[x].ch[k ^ 1];
39     t[t[x].ch[k ^ 1]].ff = y;
40     t[x].ch[k ^ 1] = y;
41     t[y].ff = x;
42     pushuf(y), pushuf(x);
43 }
44 void splay(ll x, ll tg)
45 {
46     while (t[x].ff != tg)
47     {
48         ll y = t[x].ff;
49         ll z = t[y].ff;
50         if (z != tg)
51             ((t[z].ch[1] == y) ^ (t[y].ch[1] == x)) ? rotate(x) :
52             rotate(y);
53         rotate(x);
54     }
55     if (!tg)
56         rot = x;
57 }
58 void insert(ll x)
59 {
60     ll u = rot, ff = 0;
61     while (u)
62     {
63         ff = u, u = t[u].ch[x > t[u].v];
64         u = ++tot;
65         if (ff)
66             t[ff].ch[x > t[ff].v] = u;
67         t[u].init(x, ff);
68         splay(u, 0);
69     }
70 ll kth(ll k)
71 {
72     if (k == 0)
73         return rot;
74     ll u = rot, ff = 0;
75     while (u)
76     {
77         ff = u, u = t[u].ch[k > t[u].v];
78         u = ++tot;
79         if (ff)
80             t[ff].ch[k > t[ff].v] = u;
81         k -= t[u].size;
82     }
83     return u;
84 }
85

```

```

69  {
70      ll u = rot;
71      while (1)
72      {
73          pushdf(u);
74          if (t[t[u].ch[0]].size >= k)
75              u = t[u].ch[0];
76          else if (t[t[u].ch[0]].size + 1 == k)
77              return u;
78          else
79              k -= t[t[u].ch[0]].size + 1, u = t[u].ch[1];
80      }
81  }
82  void write(ll u)
83  {
84      pushdf(u);
85      if (t[u].ch[0])
86          write(t[u].ch[0]);
87      if (t[u].v > 1 && t[u].v < n + 2)
88          printf("%d ", t[u].v - 1);
89      if (t[u].ch[1])
90          write(t[u].ch[1]);
91  }
92  void wk(ll lf, ll rf)
93  {
94      lf = kth(lf), rf = kth(rf + 2);
95      splay(lf, 0), splay(rf, lf);
96      t[t[t[rot].ch[1]].ch[0]].mark ^= 1;
97  }
98  signed main()
99  {
100     scanf("%d%d", &n, &m);
101     for (ll i = 1; i <= n + 2; ++i)
102         insert(i);
103     while (m--)
104         scanf("%d%d", &lf, &rf), wk(lf, rf);
105     write(rot);
106     return 0;
107 }

```

## K-D Tree

洛谷P4475-有  $n$  人和  $m$  巧克力，每个人有参数  $a, b, c$ ，巧克力有参数  $x, y, h$ ，每人只要对他自己  $ax + by \leq c$  的巧克力，求每人这样的巧克力的  $h$  值之和  $1 \leq n, m \leq 5 \times 10^4, -10^9 \leq$  参数  $\leq 10^9$

对二元函数  $f(x, y) = ax + by$ ，满足  $x, y$  上单调性，所以节点维护的超长方体若边界都在肯定全部点都在，所以可以进行二分下去。

```

1 #include<cstdio>
2 #include<algorithm>
3 #define ll long long
4 using namespace std;

```

```

5  const int N=5e5+50;
6  int n,now,m,rt;
7  ll a,b,c;
8  struct data
9  {
10    int d[2],mx[2],mn[2],lc,rc,val;
11    ll sum;
12    friend bool operator < (data a,data b)
13        {return a.d[now]<b.d[now];}
14 }dat[N],t[N];
15 void getmax(int&a,int b){if(a<b)a=b;}
16 void getmin(int&a,int b){if(a>b)a=b;}
17 void pushup(int x)
18 {
19    int lc=t[x].lc,rc=t[x].rc;
20    for(int i=0;i<2;i++)
21    {
22        t[x].mn[i]=t[x].mx[i]=t[x].d[i];
23        if(lc) getmin(t[x].mn[i],t[lc].mn[i]),
24            getmax(t[x].mx[i],t[lc].mx[i]);
25        if(rc) getmin(t[x].mn[i],t[rc].mn[i]),
26            getmax(t[x].mx[i],t[rc].mx[i]);
27    }
28    t[x].sum=t[lc].sum+t[rc].sum+t[x].val;
29 }
30
31 int build(int l,int r,int pl)
32 {
33    now=pl; int mid=(l+r)>>1;
34    nth_element(dat+l,dat+mid,dat+r+1);
35    t[mid]=dat[mid];
36    if(l<mid) t[mid].lc=build(l,mid-1,!pl);
37    if(r>mid) t[mid].rc=build(mid+1,r,!pl);
38    pushup(mid); return mid;
39 }
40 bool check(ll x,ll y) {return x*a+y*b<c;}
41 ll query(int x)
42 {
43    int tot=0;
44    tot+=check(t[x].mx[0],t[x].mx[1]);
45    tot+=check(t[x].mn[0],t[x].mx[1]);
46    tot+=check(t[x].mx[0],t[x].mn[1]);
47    tot+=check(t[x].mn[0],t[x].mn[1]);
48    if(tot==4) return t[x].sum; // 都满足
49    if(tot==0) return 0; // 都不满足
50    ll res=0;
51    if(check(t[x].d[0],t[x].d[1])) res+=t[x].val;
52    if(t[x].lc) res+=query(t[x].lc);
53    if(t[x].rc) res+=query(t[x].rc);
54    return res;
55 }
56 int main()
57 {
58    scanf("%d%d",&n,&m);
59    for(int i=1;i<=n;i++)
60        scanf("%d%d%d",&dat[i].d[0],&dat[i].d[1],&dat[i].val);

```

```

61     rt=build(1,n,0); while(m--)
62     {
63         scanf("%lld%lld%lld", &a, &b, &c);
64         printf("%lld\n", query(rt));
65     }
66     return 0;
67 }

```

洛谷P4148-对  $n \times n$  ( $n \leq 5 \times 10^5$ ) 矩阵进行强制在线维护(每个值异或  $lastans$ ):

1. `1 x y A` 单点加  $A$  (下标值在  $[1, n]$ )
2. `2 x1 y1 x2 y2` 区间求和
3. `3 EOF` (操作数不大于  $2 \times 10^5$ )

```

1 #include <algorithm>
2 #include <cstdio>
3 #include <cstring>
4 using namespace std;
5 const int maxn = 200010;
6 int n, op, xl, xr, yl, yr, lastans;
7
8 struct node {
9     int x, y, v;
10 } s[maxn];
11
12 bool cmp1(int a, int b) { return s[a].x < s[b].x; }
13
14 bool cmp2(int a, int b) { return s[a].y < s[b].y; }
15
16 double a = 0.725;
17 int rt, cur, d[maxn], lc[maxn], rc[maxn], L[maxn], R[maxn], D[maxn],
18     U[maxn],
19     siz[maxn], sum[maxn];
20 int g[maxn], t;
21
22 void print(int x) {
23     if (!x) return;
24     print(lc[x]);
25     g[+t] = x;
26     print(rc[x]);
27 }
28
29 void maintain(int x) {
30     siz[x] = siz[lc[x]] + siz[rc[x]] + 1;
31     sum[x] = sum[lc[x]] + sum[rc[x]] + s[x].v;
32     L[x] = R[x] = s[x].x;
33     D[x] = U[x] = s[x].y;
34     if (lc[x])
35         L[x] = min(L[x], L[lc[x]]), R[x] = max(R[x], R[lc[x]]),
36         D[x] = min(D[x], D[lc[x]]), U[x] = max(U[x], U[lc[x]]);
37     if (rc[x])
38         L[x] = min(L[x], L[rc[x]]), R[x] = max(R[x], R[rc[x]]),
39         D[x] = min(D[x], D[rc[x]]), U[x] = max(U[x], U[rc[x]]);

```

```

39 }
40
41 int build(int l, int r) {
42     if (l > r) return 0;
43     int mid = (l + r) >> 1;
44     double av1 = 0, av2 = 0, va1 = 0, va2 = 0;
45     for (int i = l; i <= r; i++) av1 += s[g[i]].x, av2 += s[g[i]].y;
46     av1 /= (r - l + 1);
47     av2 /= (r - l + 1);
48     for (int i = l; i <= r; i++)
49         va1 += (av1 - s[g[i]].x) * (av1 - s[g[i]].x),
50         va2 += (av2 - s[g[i]].y) * (av2 - s[g[i]].y);
51     if (va1 > va2)
52         nth_element(g + l, g + mid, g + r + 1, cmp1), d[g[mid]] = 1;
53     else
54         nth_element(g + l, g + mid, g + r + 1, cmp2), d[g[mid]] = 2;
55     lc[g[mid]] = build(l, mid - 1);
56     rc[g[mid]] = build(mid + 1, r);
57     maintain(g[mid]);
58     return g[mid];
59 }
60
61 void rebuild(int& x) {
62     t = 0;
63     print(x);
64     x = build(1, t);
65 }
66
67 bool bad(int x) { return a * siz[x] <= (double)max(siz[lc[x]], siz[rc[x]]); }
68
69 void insert(int& x, int v) {
70     if (!x) {
71         x = v;
72         maintain(x);
73         return;
74     }
75     if (d[x] == 1) {
76         if (s[v].x <= s[x].x)
77             insert(lc[x], v);
78         else
79             insert(rc[x], v);
80     } else {
81         if (s[v].y <= s[x].y)
82             insert(lc[x], v);
83         else
84             insert(rc[x], v);
85     }
86     maintain(x);
87     if (bad(x)) rebuild(x);
88 }
89
90 int query(int x) {
91     if (!x || xr < L[x] || xl > R[x] || yr < D[x] || yl > U[x]) return 0;
92     if (xl <= L[x] && R[x] <= xr && yl <= D[x] && U[x] <= yr) return sum[x];
93     int ret = 0;

```

```

94     if (x1 <= s[x].x && s[x].x <= xr && y1 <= s[x].y && s[x].y <= yr)
95         ret += s[x].v;
96     return query(lc[x]) + query(rc[x]) + ret;
97 }
98
99 int main() {
100    scanf("%d", &n);
101    while (~scanf("%d", &op)) {
102        if (op == 1) {
103            cur++, scanf("%d%d%d", &s[cur].x, &s[cur].y, &s[cur].v);
104            s[cur].x ^= lstans;
105            s[cur].y ^= lstans;
106            s[cur].v ^= lstans;
107            insert(rt, cur);
108        }
109        if (op == 2) {
110            scanf("%d%d%d%d", &x1, &y1, &xr, &yr);
111            x1 ^= lstans;
112            y1 ^= lstans;
113            xr ^= lstans;
114            yr ^= lstans;
115            printf("%d\n", lstans = query(rt));
116        }
117        if (op == 3) return 0;
118    }
119 }

```

## 堆

### 可删堆

例题: P2056洛谷的一部分)

以  $O(\log n)$  均摊实现: ①插入任意值 ②删除指定值 ③求最大值

若要求次大值, 可以弹两次堆顶, 再把第一次弹的压进去, 最后返回第二次弹的结果

```

1 struct heap //可删堆(大根堆)
2 {
3     priority_queue<ll> a, b;
4     void insert(ll x) { a.push(x); }
5     void erase(ll x) { b.push(x); }
6     ll top()
7     {
8         while (!b.empty() && a.top() == b.top())
9             a.pop(), b.pop();
10        return a.top();
11    }
12    ll pop()
13    {
14        ll t = top(); a.pop(); return t;
15    }
16    ll size() { return a.size() - b.size(); }

```

## 笛卡尔树

节点由  $(k, w)$  组成,  $k$  满足二叉搜索树,  $w$  满足堆, 若  $(k, w)$  分别互不相同, 那么结构唯一可以  $O(n)$  构建。例题P5854: 给定排列, 构建(满足小根堆的)笛卡尔树, 输出给节点  $i$  的左右儿子(不存在为 0)的  $\oplus_{i=1}^n i(l_i + 1)$ ,  $\oplus_{i=1}^n i(r_i + 1)$

笛卡尔树不平衡, 单调序列可以卡单次查询到  $O(n)$

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef int ll;
4 #define MAXN 10000002
5 inline ll read()
6 {
7     char p = 0;
8     ll r = 0, o = 0;
9     for (; p < '0' || p > '9'; o |= p == '-' , p = getchar())
10    ;
11     for (; p >= '0' && p <= '9'; r = (r << 1) + (r << 3) + (p ^ 48), p = getchar())
12    ;
13     return o ? (~r) + 1 : r;
14 }
15 ll n, a[MAXN], stk[MAXN], ls[MAXN], rs[MAXN], pos, top;
16 long long lf, rf;
17 signed main()
18 {
19     n = read();
20     for (ll i = 1; i <= n; ++i)
21     {
22         a[i] = read();
23         while (pos && a[stk[pos]] > a[i])
24             --pos;
25         if (pos)
26             rs[stk[pos]] = i;
27         if (pos < top)
28             ls[i] = stk[pos + 1];
29         stk[top = ++pos] = i;
30     }
31     for (long long i = 1; i <= n; ++i)
32         lf ^= i * (ls[i] + 1), rf ^= i * (rs[i] + 1);
33     printf("%lld %lld", lf, rf);
34     return 0;
35 }
```

## 可并堆

左偏树。支持在  $O(\log n)$  的时间复杂度内进行合并的堆式数据结构

**外结点**：左儿子或右儿子是空结点的结点。

**距离**：一个结点  $x$  的距离  $dist_x$  定义为其子树中与结点  $x$  最近的外结点到  $x$  的距离。特别地，定义空结点的距离为 -1。

满足小根堆/大根堆性质，且满足左偏：每个节点  $dist_{lc} \geq dist_{rc}$

基本结论：

- 节点  $x$  的距离  $dist_x = dist_{rc} + 1$
- 距离为  $n$  的左偏树至少有  $2^{n+1} - 1$  节点且是满二叉树

但是向左的链也是左偏树

例题洛谷P3377：一开始有  $n(\leq 10^5)$  个小根堆，实现  $m(\leq 10^5)$  次操作：

- `1 x y` 将第  $x$  个数和第  $y$  个数所在堆合并(若删掉或已在无视)
- `2 x` 输出第  $x$  个数所在堆最小数，并删掉这个数(多个最小删先输入的)，已删输出 `-1` 并无视

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef int ll;
4 #define MAXN 100002
5 ll n, m, op, x, y, lc[MAXN], rc[MAXN], d[MAXN], rt[MAXN];
6 bool tf[MAXN];
7 struct node
8 {
9     ll i, v;
10    bool operator<(const node &x) const
11    {
12        return v == x.v ? i < x.i : v < x.v;
13    }
14 } v[MAXN];
15 inline ll finds(ll x)
16 {
17     while (x != rt[x])
18         x = rt[x] = rt[rt[x]];
19     return x;
20 }
21 ll merge(ll x, ll y)
22 {
23     if (!x || !y)
24         return x + y;
25     if (v[y] < v[x])
26         swap(x, y);
27     rc[x] = merge(rc[x], y);
28     if (d[lc[x]] < d[rc[x]])
29         swap(lc[x], rc[x]);
30     d[x] = d[rc[x]] + 1;
31     return x;
32 }
33 signed main()
```

```

34 {
35     d[0] = -1;
36     scanf("%d%d", &n, &m);
37     for (ll i = 1; i <= n; ++i)
38         scanf("%d", &v[i].v), v[i].i = i, rt[i] = i;
39     while (m--)
40     {
41         scanf("%d%d", &op, &x);
42         if (op == 1)
43         {
44             scanf("%d", &y);
45             if (tf[x] || tf[y])
46                 continue;
47             x = finds(x), y = finds(y);
48             if (x != y)
49                 rt[x] = rt[y] = merge(x, y);
50         }
51     else
52     {
53         if (tf[x])
54         {
55             printf("-1\n");
56             continue;
57         }
58         x = finds(x);
59         printf("%d\n", v[x].v);
60         tf[x] = true;
61         rt[lc[x]] = rt[rc[x]] = rt[x] = merge(lc[x], rc[x]);
62         lc[x] = rc[x] = d[x] = 0;
63     }
64 }
65     return 0;
66 }

```

## 珂朵莉树

骗分数据结构。核心操作：把值相同的区间合并成一个结点保存在 set 里面。在数据随机的情况下，复杂度为  $O(n \log \log n)$ ，链表为  $O(n \log n)$

进行求取区间左右端点操作时，必须先 split 右端点，再 split 左端点。若先 split 左端点，返回的迭代器可能在 split 右端点的时候失效，可能会导致 RE

(SCNUOJ1731)你有一个长度为  $n$  的序列，下标从 1 开始，一开始每个数都是 1236895，你需要维护  $m$  次下列操作：

1. `1 l r x` 将区间  $[l, r]$  的每个值都设为  $x$
2. `2 l r x` 将区间  $[l, r]$  的每个值都加上  $x$
3. `3 l r` 查询区间  $[l, r]$  的值之和

$n, m (1 \leq n, m \leq 10^5)$ ，数据随机， $1 \leq x \leq 10^7$

```

1 #include <bits/stdc++.h>
2 using namespace std;

```

```

3 #define sc(x) scanf("%lld", &x)
4
5
6 struct node_t
7 {
8     ll l, r;
9     mutable ll v; // mutable使得const可变(set元素是const)
10    node_t(const ll &l, const ll &r, const ll &v) : l(l), r(r), v(v)
11 }
12
13 inline bool operator<(const node_t &o) const { return l < o.l; }
14
15 set<node_t> odt; //建立一棵珂朵莉树
16 auto split(ll x) //珂朵莉基操
17 {
18     if (x > n)
19         return odt.end();
20     auto it = --odt.upper_bound(node_t{x, 0, 0});
21     if (it->l == x)
22         return it;
23     ll l = it->l, r = it->r, v = it->v;
24     odt.erase(it);
25     odt.insert(node_t(l, x - 1, v));
26     return odt.insert(node_t(x, r, v)).first;
27 }
28 void assign(ll l, ll r, ll v) //区间赋值为v, 均摊O(loglogn)
29 {
30     auto itr = split(r + 1), itl = split(l);
31     odt.erase(itl, itr);
32     odt.insert(node_t(l, r, v));
33 }
34 void add(ll l, ll r, ll v) //区间增加v, 均摊O(loglogn)
35 {
36     auto itr = split(r + 1), itl = split(l);
37     for (; itl != itr; ++itl) //枚举每个子区间
38     {
39         itl->v = itl->v + v;
40     }
41 }
42 ll query(ll l, ll r) //区间查询, 均摊O(loglogn)
43 {
44     ll res = 0;
45     auto itr = split(r + 1), itl = split(l);
46     for (; itl != itr; ++itl)
47     { // itl->l, itl->r, itl->v 是当前子区间的左右端点和值
48         res += (itl->r - itl->l + 1) * (itl->v);
49     }
50     return res;
51 }
52 signed main()
53 {
54     sc(n);
55     odt.insert({1, n, 1236895}); //初始化区间
56     ll m, cmd, l, r, x;
57     for (sc(m); m--;) {
58         sc(cmd), sc(l), sc(r);

```

```

58     if (cmd == 1)
59     {
60         sc(x), assign(l, r, x);
61     }
62     else if (cmd == 2)
63     {
64         sc(x), add(l, r, x);
65     }
66     else
67     {
68         printf("%lld\n", query(l, r));
69     }
70 }
71 return 0;
72 }
```

## 并查集

### 种类并查集

种类并查集就是有多少个种类开多少倍长即可。

洛谷P1892: 有n个人m条关系, E a b代表a和b是敌人, F a b代表a和b是朋友; 敌人的敌人是朋友, 朋友的朋友是朋友; 朋友间一定会结成同一帮派, 求帮派数

使用反集。解法二是照搬二分图。

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 int s,n,m,a,b,f[2500];
4 char ch;
5 int find(int x){
6     if(f[x]!=x)f[x]=find(f[x]);
7     return f[x];
8 }
9 int main(){
10    cin>>n>>m;
11    for(int i=1;i<=2*n;i++){
12        f[i]=i;
13    }
14    for(int i=1;i<=m;i++){
15        cin>>ch>>a>>b;
16        if(ch=='F'){
17            f[find(a)]=find(b); //合并
18        }else{
19            f[find(a+n)]=find(b);
20            f[find(b+n)]=find(a); //反集合并
21        }
22    }
23    for(int i=1;i<=n;i++){
24        if(f[i]==i)s++;
25    }
26    cout<<s; //祖先数就是团伙数
27 }
```

例题洛谷P2024:存在食物环A->B->C->A, 有n种动物, m条判断。判断类型为1 i j表示i,j同属A/B/C大类,2 i j表示i->j。先出现的判断在不产生矛盾的情况下变成真理。求m条判断中谬论数。(输入n,m,依次输入判断)  $1 \leq n \leq 5 \times 10^5$ ,  $1 \leq k \leq 10^5$

构建三倍并查集, 其中i是自身, i+n是匿名猎物, i+2n是匿名天敌。另一种解法是加权并查集, 三种关系分别是0, 1, 2, 用向量加减模3得到新关系。加权并查集写法见下文

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define MAXN 150002
4 #define ANS \
5     { \
6         \
7         ++ans; \
8         continue; \
9     }
10 int fa[MAXN], n, k, ans, cmd, x, y, n3;
11 inline int finds(int k)
12 {
13     while (k != fa[k])
14         k = fa[k] = fa[fa[k]];
15     return k;
16 }
17 signed main()
18 {
19     scanf("%d%d", &n, &k);
20     for (int i = 1; i <= 3 * n; ++i)
21         fa[i] = i;
22     while (k--)
23     {
24         scanf("%d%d%d", &cmd, &x, &y);
25         if (x > n || y > n)
26             ANS;
27         if (cmd == 1)
28         {
29             if (finds(x + n) == finds(y) || finds(x) == finds(y + n))
30                 ANS; //或后者改为x+2n与y
31             fa[finds(x)] = finds(y);
32             fa[finds(x + n)] = finds(y + n);
33             fa[finds(x + n + n)] = finds(y + n + n);
34         }
35         else
36         {
37             if (finds(x) == finds(y) || finds(x) == finds(y + n))
38                 ANS; //如果同类相食或反向捕食(y+n是天敌), 是谬论
39             fa[finds(x + n)] = finds(y);
40             fa[finds(x + n + n)] = finds(y + n);
41             fa[finds(x)] = finds(y + n + n);
42         }
43     }
44     printf("%d", ans);
45     return 0;
46 }
```

## 加权并查集

洛谷P1196 有30000 战舰，每次 `M i j` 将  $i$  所在编队连在  $j$  所在编队尾部成链。每次 `C i j` 查询  $i, j$  链之间有多少个战舰。若  $i, j$  不同编队输出 `-1`。 $t \leq 5 \times 10^5$

设前缀和 `cnt` 为离队首的距离，则所求为  $|cnt_i - cnt_j| + 1 - 2$

每次合并并查集时，`fa[find(i)]=find(j)`，在这之前，更新  $cnt_x$ ，距离增加  $num_y$ ，即舰队数。且  $num_y$  增加  $num_x$ ，并且清零  $num_x$  (否则反复合并会出锅)。`find` 用递归式，每次 `find` 时更新  $cnt_x = cnt_{fa_x}$ ，注意在更新 `fa` 之前做，不然等于我赋值我自己。

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 #define sc(x) scanf("%lld", &x)
4 typedef long long ll;
5 #define mn 500010
6 ll t, fa[mn], cnt[mn], sum[mn];
7 char cmd[5];
8 ll findf(ll x)
9 {
10     if (x == fa[x])
11     {
12         return x;
13     }
14     ll res = findf(fa[x]);
15     cnt[x] += cnt[fa[x]];
16     return fa[x] = res;
17 }
18 signed main()
19 {
20     sc(t);
21     for (ll i = 1; i < mn; ++i)
22     {
23         fa[i] = i, sum[i] = 1;
24     }
25     while (t--)
26     {
27         ll u, v, fu, fv;
28         scanf("%s%lld%lld", cmd, &u, &v);
29         fu = findf(u), fv = findf(v);
30         if (cmd[0] == 'M')
31         {
32             cnt[fu] += sum[fv];
33             fa[fu] = fv;
34             sum[fv] += sum[fu];
35             sum[fu] = 0;
36         }
37         else
38         {
39             if (fu != fv)
40             {
41                 printf("-1\n");
42             }
43             else
```

```

44         {
45             printf("%lld\n", abs(cnt[u] - cnt[v]) - 1);
46         }
47     }
48 }
49 return 0;
50 }

```

洛谷P2024 题面见上文

构造关系  $u \rightarrow v$ ，设为  $ty[u]$  表示  $u$  与根  $v$  的关系，设 0 表示同类，1 表示捕食，2 表示被捕食，显然构造出来的关系模 3 结果符合传递性，即满足性质

$$\because u \rightarrow v \rightarrow w \therefore ty[u \rightarrow w] = (ty[u \rightarrow v] + ty[v \rightarrow w]) \bmod 3$$

那么可以构造加权并查集，使得压缩路径后，若  $u, v$  同根，则根据向量加法， $u, v$  见关系可以表示为  $(ty[u] - ty[v]) \bmod 3$ 。并且如果要新建  $u, v$  之间的关系，那么新建的  $fa_u, fa_v$  之间，若设  $fa[fa_u] = fa_v$ ，则  $ty[fa_u] = (-ty[u] + new + ty[v]) \bmod 3$

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define sc(x) scanf("%lld", &x)
4 typedef long long ll;
5 #define mn 50010
6 ll n, k, fa[mn], ty[mn], fake;
7 ll findf(ll x)
8 {
9     if (x == fa[x])
10    {
11        return x;
12    }
13    ll res = findf(fa[x]);
14    ty[x] = (ty[x] + ty[fa[x]]) % 3;
15    return fa[x] = res;
16 }
17 signed main()
18 {
19     sc(n), sc(k);
20     for (ll i = 1; i <= n; ++i)
21     {
22         fa[i] = i;
23     }
24     while (k--)
25     {
26         ll cmd, u, v, fu, fv;
27         sc(cmd), sc(u), sc(v), fu = findf(u), fv = findf(v);
28         if (u > n || v > n || (u == v && cmd == 2))
29         {
30             ++fake;
31             continue;
32         }
33         if (cmd == 1)
34         {
35             if (fu == fv && ty[u] != ty[v])
36             {

```

```

37         ++fake;
38     }
39     else
40     {
41         ty[fu] = (3 - ty[u] + ty[v]) % 3;
42         fa[fu] = fv;
43     }
44 }
45 else
46 {
47     if (fu == fv && (ty[u] - ty[v] + 3) % 3 != 1)
48     {
49         ++fake;
50     }
51     else
52     {
53         ty[fu] = (3 - ty[u] + ty[v] + 1) % 3;
54         fa[fu] = fv;
55     }
56 }
57 }
58 printf("%lld", fake);
59 return 0;
60 }

```

P2294-有  $t(< 100)$  组询问，每次给定  $n(< 100)$  个月和  $m(< 10^3)$  条账本，账本表示  $[u, v]$  月份收入和为  $w$ 。问账本是否矛盾。

解法一：差分约束。

设前  $i$  个月收入之和为  $s_i$ ，则每条账本为： $s_v - s_{u-1} = w$ ，即：

$$\begin{cases} s_{u-1} \leq s_v - w \\ s_v \leq s_{u-1} + w \end{cases}$$

直接建图，建出来的图不一定是连通的，则可以对每个非连通子图都跑一次 SPFA，如果存在一个子图有负环就 false，否则 true。不管跑最短路还是最长路都能过(分别设不同的无穷初始值)，记得初始化全(不要漏了 `cnt` 等数组)。

解法二：并查集。

若已知  $s_a - s_b, s_b - s_c$ ，且发现新的  $s_a - s_c$ ，需要判断新的是否符合原有的。维护带权并查集，设  $c_u = s_{fa} - s_u$ 。当我们发现输入的  $a, b$  在同一个并查集时，有  $c_a - c_b = s_{fa} - s_a - s_{fa} + s_b$ ，故判断原有的  $c_u - c_v$  是否与  $w$  相等即可。

初始设  $c = 0$ ，合并两并查集  $u, v$  时，设  $fa_v$  并到  $fa_u$  去，因为已知  $s_v - s_u = w$ ，且原有  $c_u = s_{fa_u} - s_u, c_v = s_{fa_v} - s_v$ 。要求出新的  $s_{fa_u} - s_{fa_v} = (c_u + s_u) - (c_v + s_v) = c_u - c_v - (s_v - s_u) = c_u - c_v - w$ 。(类比向量加减法)

对于路径压缩，设已知  $c_u = s_r - s_u, c_v = s_u - s_v$ ，需要更新  $s_r - s_v$ ，等于  $c_u + c_v$ ，即直接求边权和即可。

```

1 #include<bits/stdc++.h>
2 int fa[105], cha[105];
3 int find(int x)

```

```

4  {
5      if(x!=fa[x])
6      {
7          int t=find(fa[x]);
8          cha[x]+=cha[fa[x]];
9          fa[x]=t;
10     }
11     return fa[x];
12 }
13 int main()
14 {
15     int T,n,m,i,x,y,z,flag;
16     scanf("%d",&T);
17     while (T--)
18     {
19         flag=0;
20         scanf("%d%d",&n,&m);
21         for(i=0;i<=n;i++)
22         {
23             fa[i]=i;
24             cha[i]=0;
25         }
26         for(i=1;i<=m;i++)
27         {
28             scanf("%d%d%d",&x,&y,&z);
29             x--;
30             if(find(x)!=find(y))
31             {
32                 cha[fa[y]]=cha[x]-cha[y]-z;
33                 fa[fa[y]]=fa[x];
34             }
35             else
36                 if(cha[x]-cha[y]!=z) flag=1;
37         }
38         if(flag==0) printf("true\n"); else printf("false\n");
39     }
40     return 0;
41 }

```

### 解法三：贪心

先按左端点为第一排序关键字，再排右端点，作为比较依据动态维护一个堆。之后就开始两两比较，如果左端点相等，就比较右端点，如果相等，就比较值，如果值不同，就直接输出false，否则输出true，如果右端点不等，就把相同的部分抵消掉，把新的区间再压入优先队列。直到不能操作，就输出true。

### 其他例题：

- (洛谷P2661)出度均 1 无自环  $n$  点有向图，求最小环长度

路径压缩时维护每个点到根的距离。枚举每条边，若端点在同一并查集那么环长

$= d[u] + d[v] + 1$ ，否则合并并查集 ( $d[u] = d[v] + 1, f_{f_u} = f_v$ )；也可以 DFS (带深度, 重复访问时得环, 深度相减)

## 可撤销并查集

洛谷5787  $n(1 \leq n \leq 10^5)$  点  $m(1 \leq m \leq 2 \times 10^5)$  边无向图, 每条边生存周期是  $[l, r)(1 \leq l < r \leq 10^5)$ , 问每个时刻图是不是二分图。

二分图判断: 可撤销种类并查集 + 染色法。

对节点  $i$ , 拆分为两个节点分属  $S, T$ 。若存在一个点, 使得拆分的两点并在一起了, 那么就不是二分图。

按时间轴建立线段树, 第  $i$  个线段树节点维护时间区间内存在的所有边(从 1 编号), 那么每次加一条边, 会向最多  $\log k$  个节点加边。

然后按线段树顺序遍历节点, 先序时把线段树节点的所有边在并查集上合并, 即对边  $(u, v)$  入图的话, 将  $(u, v + n)$  与  $(u + n, v)$  在并查集合并, 然后每次合并时, 按秩合并, 树高小的合到大的, 并记录进行合并的操作顺序, 然后记录这个节点一开始是在哪个操作记号, 在再次回到该节点时(后序), 逐个撤销掉并查集操作, 进行复原。

设最大生存时间是  $k$ , 空间复杂度是  $O(m \log k)$ , 时间复杂度是  $O(k \log k + 4k + m \log k + 2n \log n) = O(m \log k + n \log n)$

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 #define sc(x) scanf("%d", &x)
4 using ll = int;
5 using trip = tuple<ll, ll, ll>; //感觉pair就行
6 struct stk
7 {
8     vector<trip> a;
9     ll top;
10    void push(trip v) { a[++top] = v; }
11    trip pop() { return a[top--]; }
12 } s;
13 signed main()
14 {
15     ll n, m, k;
16     sc(n), sc(m), sc(k);
17     using edge = pair<ll, ll>;
18     vector<vector<edge>> e(4 * k + 10, vector<edge>());
19 #define lfs p << 1
20 #define rfs p << 1 | 1
21 #define mkcf ll cf = (lf + rf) >> 1
22     auto insert = [&](auto self, ll p, ll lf, ll rf, ll lc, ll rc, edge v)
23     -> void
24     {
25         if (lf >= lc && rf <= rc)
26         {
27             e[p].push_back(v);
28             return;
29         }
30         mkcf;
31         if (lc <= cf)
32         {
33             self(self, lfs, lf, cf, lc, rc, v);
34         }
35         if (rc >= cf + 1)
```

```

35         {
36             self(self, rfs, cf + 1, rf, 1c, rc, v);
37         }
38     };
39     for (ll i = 1, x, y, l, r; i <= m; ++i)
40     {
41         sc(x), sc(y), sc(l), sc(r), ++l;
42         insert(insert, 1, 1, k, l, r, {x, y});
43     }
44     vector<ll> fa(2 * n + 10), h(2 * n + 10, 1);
45     for (ll i = 1; i <= 2 * n; ++i)
46     {
47         fa[i] = i;
48     }
49     auto findf = [&](ll x)
50     {
51         while (x != fa[x])
52         {
53             x = fa[x] = fa[fa[x]];
54         }
55         return x;
56     };
57     ll cnt = 0;
58     for (ll i = 1; i <= 4 * k; ++i)
59     {
60         cnt += e[i].size();
61     }
62     s.a.resize(cnt * 2 + 10); //我也不知道怎么定的，cnt都行
63     auto merge = [&](ll x, ll y)
64     {
65         ll fx = findf(x), fy = findf(y);
66         if (h[fx] > h[fy])
67         {
68             swap(fx, fy);
69         }
70         s.push({fx, fy, h[fx] == h[fy]});
71         fa[fx] = fy, h[fy] += h[fx] == h[fy];
72     };
73     auto solve = [&](auto self, ll p, ll lf, ll rf) -> void
74     {
75         bool ok = true;
76         ll lastTop = s.top;
77         for (edge x : e[p])
78         {
79             ll u = x.first, v = x.second;
80             if (findf(u) == findf(v))
81             {
82                 ok = false;
83                 for (ll i = lf; i <= rf; ++i)
84                 {
85                     printf("No\n");
86                 }
87                 break;
88             }
89             merge(u, v + n), merge(u + n, v);
90         }

```

```

91     if (ok)
92     {
93         if (lf == rf)
94         {
95             printf("Yes\n");
96             return;
97         }
98         mkcf;
99         self(self, lfs, lf, cf), self(self, rfs, cf + 1, rf);
100    }
101   while (s.top > lastTop)
102   {
103       trip nd = s.pop();
104       l1 x, y, dt;
105       tie(x, y, dt) = nd;
106       h[fa[x]] -= dt;
107       fa[x] = x;
108   }
109 };
110 solve(solve, 1, 1, k);
111 return 0;
112 }
```

实现版本 2: CF813F(每次更改一条边  $(x, y)$  ( $x < y$ ), 若本来有就删否则加),  $n, m \leq 10^5, m$  是操作次数

```

1 #include <bits/stdc++.h>
2 const int N = 200000 + 10;
3 int n, m, ans[N];
4 struct E
5 {
6     int u, v;
7     E(int a, int b) : u(a), v(b) {}
8 };
9 std::map<int, int> g[N];
10
11 int L[N << 2], R[N << 2];
12 std::vector<E> tag[N << 2];
13 #define lc (o << 1)
14 #define rc ((o << 1) | 1)
15 void build(int o, int l, int r)
16 {
17     L[o] = l;
18     R[o] = r;
19     if (l == r)
20         return;
21     int mid = (l + r) >> 1;
22     build(lc, l, mid);
23     build(rc, mid + 1, r);
24 }
25 // 把一条边拆到 logm 个区间上去.
26 // 可以说线段树只是用来保证复杂度的...
27 void query(int o, int l, int r, const E &e)
28 {
29     if (L[o] > r || R[o] < l)
```

```

30         return;
31     if (l <= L[o] && R[o] <= r)
32     {
33         tag[o].push_back(e);
34         return;
35     }
36     query(lc, l, r, e);
37     query(rc, l, r, e);
38 }
39
40 // merged tree(root=x)->tree(root=y)
41 struct P
42 {
43     int x, y;
44     P(int a, int b) : x(a), y(b) {}
45 };
46 int fa[N], size[N], dis[N];
47 inline int getpar(int x)
48 {
49     while (fa[x] != x)
50         x = fa[x];
51     return x;
52 }
53
54 // 到并查集根所经过的边数mod2.
55 // mod2意义下的加法即为xor.
56 // 用来查询x和根是不是相同颜色.
57 inline int getdis(int x)
58 {
59     int d = 0;
60     while (fa[x] != x)
61     {
62         d ^= dis[x];
63         x = fa[x];
64     }
65     return d;
66 }
67 // 按照size合并.
68 inline int solve(int x, int y, std::stack<P> &stk)
69 {
70     int dx = getdis(x), dy = getdis(y);
71     x = getpar(x);
72     y = getpar(y);
73     if (x == y)
74         return (dx ^ dy);
75     if (size[x] > size[y])
76     {
77         std::swap(x, y);
78         std::swap(dx, dy);
79     }
80     size[y] += size[x];
81     fa[x] = y;
82     dis[x] = (dx ^ dy ^ 1);
83     stk.push(P(x, y));
84     return 1;
85 }

```

```

86 // 撤销(x->y)的合并,拆出来, size改回去. dis改成0
87 inline void back(P p)
88 {
89     int x = p.x, y = p.y;
90     size[y] -= size[x];
91     dis[fa[x] = x] = 0;
92 }
93 // 递归线段树,进入区间[l,r]时
94 // 保证时间轴上[l,r]范围内一直可用的边都被加入了
95 void dfs(int o, int flag)
96 {
97     std::stack<P> stk;
98     for (auto e : tag[o])
99         flag &= solve(e.u, e.v, stk);
100    if (L[o] == R[o])
101        ans[L[o]] = flag;
102    else
103    {
104        dfs(lc, flag);
105        dfs(rc, flag);
106    }
107    // 撤销操作...不然从爸爸进入兄弟节点的时候开始讲到的那个性质会被破坏.
108    // 注意撤销顺序...
109    while (!stk.empty())
110    {
111        back(stk.top());
112        stk.pop();
113    }
114 }
115
116 int read()
117 {
118     int x = 0;
119     char c;
120     do
121     {
122         c = getchar();
123     } while (!isdigit(c));
124     do
125     {
126         x = x * 10 + c - '0';
127         c = getchar();
128     } while (isdigit(c));
129     return x;
130 }
131 int main()
132 {
133     n = read();
134     m = read();
135     build(1, 1, m);
136     int x, y;
137     for (int i = 1; i <= m; i++)
138     {
139         x = read();
140         y = read();
141         if (x > y)

```

```

142         std::swap(x, y);
143         if (g[x][y] == 0)
144             g[x][y] = i;
145         else
146         {
147             query(1, g[x][y], i - 1, E(x, y));
148             g[x][y] = 0;
149         }
150     }
151     for (int i = 1; i <= n; i++)
152     {
153         size[fa[i] = i] = 1;
154         for (auto p : g[i])
155             if (p.second > 0)
156                 query(1, p.second, m, E(i, p.first));
157     }
158     dfs(1, 1);
159     for (int i = 1; i <= m; i++)
160         puts(ans[i] ? "YES" : "NO");
161     return 0;
162 }

```

## 可持久化并查集

洛谷U208135-给定  $n(1 \leq n \leq 10^5)$  个集合, 第  $i$  个集合初始只有数  $i$ , 维护  $m(1 \leq m \leq 2 \times 10^5)$  次操作:

- 1 a b 合并  $a, b$  所在集合
- 2 k 回到第  $k$  次操作之后的状态
- 3 a b 问  $a, b$  是否在同一集合(是输出 1 否则 0 )

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define sc(x) scanf("%d", &x)
4 typedef pair<int, int> pii;
5 const int maxn = 1e5 + 5;
6 const int maxm = 2e5 + 5;
7 const int maxt = 4e6;//4n+mlogn
8 #define fi first
9 #define se second
10 struct Node
11 {
12     int ls, rs, val, rnk;
13 };
14 #define mid ((l + r) / 2)
15 int n, m;
16 int rt[maxm];
17 struct SegTree
18 {
19     Node s[maxt];
20     int tot;
21     int build(int l, int r)

```

```

22     {
23         int k = ++tot;
24         if (l == r)
25             s[k] = {0, 0, 1, 1};
26         else
27             s[k] = {build(l, mid), build(mid + 1, r), 0, 0};
28         return k;
29     }
30     pii new_node(int l, int r, int u, int p)
31     {
32         int k = ++tot;
33         s[k] = s[u];
34         if (l == r)
35             return {k, k};
36         int res;
37         if (p <= mid)
38         {
39             pii pos = new_node(l, mid, s[u].ls, p);
40             s[k].ls = pos.fi, res = pos.se;
41         }
42         else
43         {
44             pii pos = new_node(mid + 1, r, s[u].rs, p);
45             s[k].rs = pos.fi, res = pos.se;
46         }
47         return {k, res};
48     }
49     int qry(int l, int r, int k, int p)
50     { // node
51         if (l == r)
52             return k;
53         if (p <= mid)
54             return qry(l, mid, s[k].ls, p);
55         else
56             return qry(mid + 1, r, s[k].rs, p);
57     }
58     int merge(int k, int u, int v)
59     { // node
60         pii p1 = new_node(1, n, k, s[u].val);
61         s[p1.se].val = s[v].val;
62         pii p2 = new_node(1, n, p1.fi, s[v].val);
63         s[p2.se].rnk = max(s[p2.se].rnk, s[p1.se].rnk + 1);
64         return p2.fi;
65     }
66 } sgt;
67
68 struct DSU
69 {
70     Node *s;
71     void init()
72     {
73         rt[0] = sgt.build(1, n);
74         s = sgt.s;
75     }
76     int fd(int k, int u)
77     { // node

```

```

78     int fa = sgt.qry(1, n, k, u);
79     if (s[fa].val == u)
80         return fa;
81     int res = fd(k, s[fa].val);
82     s[fa].rnk = s[res].rnk + 1;
83     return res;
84 }
85 int mg(int k, int u, int v)
86 {
87     u = fd(k, u), v = fd(k, v);
88     if (s[u].val == s[v].val)
89         return k;
90     if (s[u].rnk < s[v].rnk)
91         return sgt.merge(k, u, v);
92     else
93         return sgt.merge(k, v, u);
94 }
95 } dsu;
96
97 int main()
98 {
99     sc(n), sc(m);
100    dsu.init();
101    for (int i = 1, o, x, y; i <= m; ++i)
102    {
103        sc(o);
104        if (o == 1)
105            sc(x), sc(y), rt[i] = dsu.mg(rt[i - 1], x, y);
106        if (o == 2)
107            sc(x), rt[i] = rt[x];
108        if (o == 3)
109        {
110            sc(x), sc(y);
111            rt[i] = rt[i - 1];
112            int u = dsu.fd(rt[i], x), v = dsu.fd(rt[i], y);
113            printf("%d\n", u == v);
114        }
115    }
116    return 0;
117 }

```

其他写法:

```

1 #include<stdio.h>
2 const int N=1e5+5;
3 int L[N<<6],R[N<<6],rt[N<<6],fa[N<<6],sz[N<<6],cnt;
4 #define mid (l+r>>1)
5 #define ls L[k],l,mid
6 #define rs R[k],mid+1,r
7 int n,m;
8 void swap(int &x,int &y){
9     int z=x;x=y;y=z;
10 }
11 void build(int &k,int l,int r){
12     k=++cnt;

```

```

13     if(l==r) return fa[k]=1,sz[k]=1,void();
14     build(ls),build(rs);
15 }
16 void modify(int &k,int l,int r,int x,int y,int z){
17     ++cnt;L[cnt]=L[k],R[cnt]=R[k],fa[cnt]=fa[k],sz[cnt]=sz[k];k=cnt;
18     if(l==r) return fa[k]=y,sz[k]=z,void();
19     if(x<=mid)modify(ls,x,y,z);
20     else modify(rs,x,y,z);
21 }
22 int query(int k,int l,int r,int x){
23     if(l==r) return k;
24     if(x<=mid) return query(ls,x);
25     else return query(rs,x);
26 }
27 int find(int k,int x){
28     int F=query(k,1,n,x);
29     if(fa[F]==x) return F;
30     return find(k,fa[F]);
31 }
32 int main(){
33     scanf("%d%d",&n,&m);
34     build(rt[0],1,n);
35     for(int i=1,opt,x,y;i<=m;i++){
36         scanf("%d%d",&opt,&x);rt[i]=rt[i-1];
37         if(opt!=2)scanf("%d",&y);
38         if(opt==1){
39             int fx=find(rt[i],x),fy=find(rt[i],y);
40             if(fa[fx]!=fa[fy]){
41                 if(sz[fx]>sz[fy])swap(fx,fy);
42                 modify(rt[i],1,n,fa[fx],fa[fy],sz[fx]);
43                 modify(rt[i],1,n,fa[fy],fa[fx],sz[fx]+sz[fy]);
44             }
45         }
46         else if(opt==2)rt[i]=rt[x];
47         else{
48             int fx=find(rt[i],x),fy=find(rt[i],y);
49             printf("%d\n",fa[fx]==fa[fy]);
50         }
51     }
52     return 0;
53 }

```

# 图论

## 树

### 重心

以树的重心为根时，所有子树的大小都不超过整棵树大小的一半。

树中所有点到某个点的距离和中，到重心的距离和是最小的；如果有两个重心，那么到它们的距离和一样。

把两棵树通过一条边相连得到一棵新的树，那么新的树的重心在连接原来两棵树的重心的路径上。

在一棵树上添加或删除一个叶子，那么它的重心最多只移动一条边的距离。

例题POJ1655：给定  $t(\leq 20)$  询问，每次节点为  $n(n \leq 2 \times 10^4)$  的树，求最小编号重心及其最大子树节点数

```
1 #include <cstdio>
2 #include <algorithm>
3 #include <cstring>
4 using namespace std;
5 typedef long long ll;
6 #define mn 20010
7 struct edge
8 {
9     ll to, nx;
10 } e[mn << 1];
11 ll t, n, hd[mn], cnt, cf, cfv, siz[mn], w[mn];
12 void adde(ll u, ll v)
13 {
14     e[++cnt] = {v, hd[u]};
15     hd[u] = cnt;
16 }
17 #define sc(x) scanf("%lld", &x)
18 void dfs(ll u, ll fa)
19 {
20     siz[u] = 1, w[u] = 0;
21     for (ll i = hd[u], v; i; i = e[i].nx)
22     {
23         v = e[i].to;
24         if (v == fa)
25         {
26             continue;
27         }
28         dfs(v, u);
29         siz[u] += siz[v];
30         w[u] = max(w[u], siz[v]);
31     }
32     w[u] = max(w[u], n - siz[u]);
33     if (w[u] <= n / 2 && u < cf)
34     {
35         cf = u, cfv = w[u];
36     }
37 }
38 signed main()
39 {
40     for (sc(t); t; --t)
41     {
42         sc(n), cnt = 0, memset(hd, 0, sizeof hd), cf = n + 1;
43         for (ll i = 1, u, v; i < n; ++i)
44         {
45             sc(u), sc(v), adde(u, v), adde(v, u);
46         }
47         dfs(1, 0);
48         printf("%lld %lld\n", cf, cfv);
49     }
50     return 0;
51 }
```

## 直径

直径是树的全源最长路径

两次 DFS 法求直径

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #define sc(x) scanf("%lld", &x)
5 #define mn 200002
6 ll n, cnt, hd[mn], rot, mxd;
7 bool vis[mn];
8 struct edge
9 {
10     ll to, nx;
11 } e[mn << 1];
12 void adde(ll &u, ll &v)
13 {
14     e[++cnt] = {v, hd[u]};
15     hd[u] = cnt;
16 }
17 void dfs(ll x, ll fa, ll d)
18 {
19     vis[x] = true;
20     if (d > mxd)
21     {
22         mxd = d, rot = x;
23     }
24     for (ll i = hd[x]; i; i = e[i].nx)
25     {
26         ll toi = e[i].to;
27         if (!vis[toi])
28             dfs(toi, x, d + 1);
29     }
30 }
31 signed main()
32 {
33     sc(n);
34     for (ll i = 1, u, v; i < n; ++i)
35         sc(u), sc(v), adde(u, v), adde(v, u);
36     dfs(1, 0, 0);
37     mxd = 0;
38     memset(vis, 0, sizeof vis);
39     dfs(rot, 0, 0);
40     printf("%lld", mxd);
41     return 0;
42 }
```

## 最近公共祖先

解法零：暴力

```
1 if d[u] < d[v]:
2     u, v = v, u
3     while d[u] > d[v]:
4         u = fa[u]
5     while u != v:
6         u, v = fa[u], fa[v]
7 return u
```

解法一：倍增法求 LCA 可以  $O(n \log n)$  预处理和  $O(\log n)$  查询

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 #define sc(x) scanf("%lld", &x)
4 typedef long long ll;
5 #define mn 500010
6 #define mm 20
7 struct edge
8 {
9     ll to, nx;
10 } e[mn << 1];
11 fa[mn][mm], n, m, hd[mn], cnt, d[mn], u, v, lg[mn], s;
12 void adde(ll &u, ll &v)
13 {
14     e[++cnt] = {v, hd[u]};
15     hd[u] = cnt;
16 }
17 void dfs(ll h, ll faa)
18 {
19     fa[h][0] = faa;
20     d[h] = d[faa] + 1;
21     for (ll i = 1; i <= lg[d[h]]; ++i)
22     {
23         fa[h][i] = fa[fa[h][i - 1]][i - 1];
24     }
25     for (ll i = hd[h]; i; i = e[i].nx)
26     {
27         ll toi = e[i].to;
28         if (toi != faa)
29             dfs(toi, h);
30     }
31 }
32 ll lca(ll u, ll v)
33 {
34     if (d[u] < d[v])
35         swap(u, v);
36     while (d[u] > d[v])
37         u = fa[u][lg[d[u]] - d[v] - 1];
38     assert(d[u] == d[v]);
39     if (u == v)
40         return u;
41     for (ll k = lg[d[u]] - 1; k >= 0; --k)
```

```

42     {
43         if (fa[u][k] != fa[v][k])
44         {
45             u = fa[u][k];
46             v = fa[v][k];
47         }
48     }
49     return fa[u][0];
50 }
51 signed main()
52 {
53     sc(n), sc(m), sc(s); // s是根节点
54     for (ll i = 1; i < n; ++i)
55         sc(u), sc(v), adde(u, v), adde(v, u);
56     for (ll i = 1; i <= n; ++i) // 1+floor(log(,2))
57         lg[i] = lg[i / 2] + 1; // 如1,2,2,3,3
58     dfs(s, 0);
59     while (m--)
60     {
61         sc(u), sc(v);
62         printf("%lld\n", lca(u, v));
63     }
64     return 0;
65 }

```

解法二：欧拉序上 RMQ，可以  $O(n \log n)$  预处理和  $O(1)$  查询

欧拉序就是每个点进入时记录一次，从每一个子树出来时记录一次

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define sc(x) scanf("%d", &x)
4 typedef int ll;
5 const ll N = 1e6 + 10; //开二倍长
6 ll n, m, s, tot, cnt;
7 ll head[N], to[N], Next[N], Log[N];
8 inline void addedge(ll x, ll y)
9 {
10     to[++tot] = y;
11     Next[tot] = head[x];
12     head[x] = tot;
13 }
14 ll a[N], dep[N], mn[21][N], p[N];
15 inline void dfs(ll x, ll fa)
16 {
17     a[++cnt] = x;
18     p[x] = cnt;
19     dep[x] = dep[fa] + 1;
20     for (ll i = head[x]; i; i = Next[i])
21     {
22         ll u = to[i];
23         if (u == fa)
24             continue;
25         dfs(u, x);
26         a[++cnt] = x;

```

```

27     }
28 }
29 signed main()
30 {
31     sc(n), sc(m), sc(s);
32     Log[0] = -1;
33     for (ll i = 1, x, y; i < n; ++i)
34         sc(x), sc(y), addedge(x, y), addedge(y, x);
35     dfs(s, 0);
36     for (ll i = 1; i <= cnt; ++i)
37         mn[0][i] = a[i];
38     for (ll i = 1; i <= 20; ++i)
39         for (ll j = 1; j + (1 << i) <= cnt; ++j)
40             if (dep[mn[i - 1][j]] < dep[mn[i - 1][j + (1 << (i - 1))]])
41                 mn[i][j] = mn[i - 1][j];
42             else
43                 mn[i][j] = mn[i - 1][j + (1 << (i - 1))];
44     for (ll i = 1; i <= cnt; ++i)
45         Log[i] = Log[i >> 1] + 1;
46     for (ll i = 1, x, y; i <= m; ++i)
47     {
48         sc(x), sc(y);
49         if (p[x] > p[y])
50             swap(x, y);
51         ll k = Log[p[y] - p[x] + 1], ans;
52         if (dep[mn[k][p[x]]] < dep[mn[k][p[y] - (1 << k) + 1]])
53             ans = mn[k][p[x]];
54         else
55             ans = mn[k][p[y] - (1 << k) + 1];
56         printf("%d\n", ans);
57     }
58     return 0;
59 }

```

解法三：tarjan 离线，复杂度取并查集的

```

1 #include <algorithm>
2 #include <cstring>
3 #include <iostream>
4 using namespace std;
5
6 class Edge {
7 public:
8     int toVertex, fromVertex;
9     int next;
10    int LCA;
11    Edge() : toVertex(-1), fromVertex(-1), next(-1), LCA(-1){};
12    Edge(int u, int v, int n) : fromVertex(u), tovertex(v), next(n), LCA(-1)
13    {};
14 };
15 const int MAX = 100;
16 int head[MAX], queryHead[MAX];
17 Edge edge[MAX], queryEdge[MAX];
18 int parent[MAX], visited[MAX];

```

```

19 int vertexCount, queryCount;
20
21 void init() {
22     for (int i = 0; i <= vertexCount; i++) {
23         parent[i] = i;
24     }
25 }
26
27 int find(int x) {
28     if (parent[x] == x) {
29         return x;
30     } else {
31         return find(parent[x]);
32     }
33 }
34
35 void tarjan(int u) {
36     parent[u] = u;
37     visited[u] = 1;
38
39     for (int i = head[u]; i != -1; i = edge[i].next) {
40         Edge& e = edge[i];
41         if (!visited[e.toVertex]) {
42             tarjan(e.toVertex);
43             parent[e.toVertex] = u;
44         }
45     }
46
47     for (int i = queryHead[u]; i != -1; i = queryEdge[i].next) {
48         Edge& e = queryEdge[i];
49         if (visited[e.toVertex]) {
50             queryEdge[i ^ 1].LCA = e.LCA = find(e.toVertex);
51         }
52     }
53 }
54
55 int main() {
56     memset(head, 0xff, sizeof(head));
57     memset(queryHead, 0xff, sizeof(queryHead));
58
59     cin >> vertexCount >> queryCount;
60     int count = 0;
61     for (int i = 0; i < vertexCount - 1; i++) {
62         int start = 0, end = 0;
63         cin >> start >> end;
64
65         edge[count] = Edge(start, end, head[start]);
66         head[start] = count;
67         count++;
68
69         edge[count] = Edge(end, start, head[end]);
70         head[end] = count;
71         count++;
72     }
73
74     count = 0;

```

```

75  for (int i = 0; i < queryCount; i++) {
76      int start = 0, end = 0;
77      cin >> start >> end;
78
79      queryEdge[count] = Edge(start, end, queryHead[start]);
80      queryHead[start] = count;
81      count++;
82
83      queryEdge[count] = Edge(end, start, queryHead[end]);
84      queryHead[end] = count;
85      count++;
86  }
87
88  init();
89  tarjan(1);
90
91  for (int i = 0; i < queryCount; i++) {
92      Edge& e = queryEdge[i * 2];
93      cout << "(" << e.fromVertex << "," << e.toVertex << ")" " << e.LCA <<
94      endl;
95  }
96
97  return 0;
}

```

解法四：LCA 为两个游标跳转到同一条重链上时深度较小的那个游标所指向的点。

树链剖分的预处理时间复杂度为线性，单次查询的时间复杂度为对数，并且常数较小。

## 树上k级祖先

倍增：(lc1483, 不存在输出 -1)

```

1  class TreeAncestor {
2      vector<vector<int>> pa;
3  public:
4      TreeAncestor(int n, vector<int> &parent) {
5          int m = 32 - __builtin_clz(n); // n 的二进制长度
6          pa.resize(n, vector<int>(m, -1));
7          for (int i = 0; i < n; i++)
8              pa[i][0] = parent[i];
9          for (int i = 0; i < m - 1; i++)
10             for (int x = 0; x < n; x++)
11                 if (int p = pa[x][i]; p != -1)
12                     pa[x][i + 1] = pa[p][i];
13     }
14
15     int getKthAncestor(int node, int k) {
16         int m = 32 - __builtin_clz(k); // k 的二进制长度
17         for (int i = 0; i < m; i++) {
18             if ((k >> i) & 1) { // k 的二进制从低到高第 i 位是 1
19                 node = pa[node][i];
20                 if (node < 0) break;
21             }
22         }
23         return node;
}

```

```

24 }
25
26 // 另一种写法, 不断去掉 k 的最低位的 1
27 int getKthAncestor2(int node, int k) {
28     for (; k && node != -1; k &= k - 1) // 也可以写成 ~node
29         node = pa[node][__builtin_ctz(k)];
30     return node;
31 }
32 };

```

例题P5903: 给定  $n (2 \leq n \leq 5 \times 10^5)$  点有根树,  $q (1 \leq q \leq 5 \times 10^6)$  询问, 随机种子  $s$ 。接下来输入每个点的父亲 (0 为根), 然后有根据随机种子生成的询问, 每次求节点  $x$  的  $k$  级祖先。输出  $\bigoplus_{i=1}^q i \times ans_i \circ s$

长链剖分, 先一次性跳到长链, 而后在链上  $O(1)$  解决。结论: 跳  $\lceil \log k \rceil$  次一定在长链上

长链: 重子节点表示其子节点中子树深度最大的子结点。长链剖分从一个节点到根的路径的轻边切换条数是  $\sqrt{n}$  级别的

设重链长  $d$ , 且存在  $i$  满足  $2^i < k < 2^{i+1}$ , 那么  $k - 2^i < 2^i \leq d$

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef int ll;
4 #define MAXN 500002
5 #define ui unsigned int
6 #define sc(x) scanf("%d", &x)
7 ll n, g[MAXN], d[MAXN], f[MAXN][20], son[MAXN], dep[MAXN], top[MAXN];
8 ll q, rt, ans, o, x, k;
9 vector<ll> e[MAXN], u[MAXN], v[MAXN];
10 long long res;
11 ui s;
12 inline ui get(ui x)
13 {
14     x ^= x << 13, x ^= x >> 17, x ^= x << 5;
15     return s = x;
16 }
17 void dfs1(ll x)
18 {
19     dep[x] = d[x] = d[f[x][0]] + 1;
20     for (auto y : e[x])
21     {
22         f[y][0] = x;
23         for (ll i = 0; f[y][i]; ++i)
24             f[y][i + 1] = f[f[y][i]][i];
25         dfs1(y);
26         if (dep[y] > dep[x])
27             dep[x] = dep[y], son[x] = y;
28     }
29 }
30 void dfs2(ll x, ll p)
31 {
32     top[x] = p;
33     if (x == p)

```

```

34    {
35        o = x;
36        for (ll i = 0; i <= dep[x] - d[x]; ++i)
37            u[x].push_back(o), o = f[o][0];
38        o = x;
39        for (ll i = 0; i <= dep[x] - d[x]; ++i)
40            v[x].push_back(o), o = son[o];
41    }
42    if (son[x])
43        dfs2(son[x], p);
44    for (auto y : e[x])
45        if (y != son[x])
46            dfs2(y, y);
47 }
48 inline ll ask(ll x, ll k)
49 {
50     if (!k)
51         return x;
52     x = f[x][g[k]], k -= 1 << g[k];
53     k -= d[x] - d[top[x]], x = top[x];
54     return k >= 0 ? u[x][k] : v[x][-k];
55 }
56 signed main()
57 {
58     scanf("%d%d%d", &n, &q, &s);
59     g[0] = -1;
60     for (ll i = 1; i <= n; ++i)
61     {
62         scanf("%d", &f[i][0]);
63         e[f[i][0]].push_back(i), g[i] = g[i >> 1] + 1;
64     }
65     rt = e[0][0];
66     dfs1(rt);
67     dfs2(rt, rt);
68     for (ll i = 1; i <= q; ++i)
69     {
70         x = (get(s) ^ ans) % n + 1;
71         k = (get(s) ^ ans) % d[x];
72         res ^= 1LL * i * (ans = ask(x, k));
73     }
74     printf("%lld", res);
75     return 0;
76 }

```

## LCA应用

### 前缀和差分

路径点权和:  $s[u] + s[v] - s[lca] - s[fa_{lca}]$

路径边权和:  $s[u] + s[v] - 2s[lca]$

路径点差分:  $s[u], s[v]$  加一,  $s[lca], s[fa_{lca}]$  减一

路径边差分:  $s[u], s[v]$  加一,  $s[lca]$  减二

## 两点距离

在 LCA 的基础上，设深度为  $d[x]$ ，则距离为  $d[u] + d[v] - 2d[lca]$ 。

也可以表示为： $|d[u] - d[LCA]| + |d[v] - d[LCA]|$

注：暴力思路是从  $u$  出发 DFS 到  $v$  即可

## 路径相交

对  $AB, CD$ ，分别求出 LCA 为  $X, Y$ ，若两点距离满足  $|AY| + |BY| = |AB|$  或  $|CX| + |DX| = |CD|$ ，证明相交

## 其他

从树上三个点出发，求一个目的点，使得路径和最小，输出最小路径和

两两求 LCA，可以发现必然有一个重复点，并且可以发现不重复点必然是最优解，设深度为  $d$ ，最小路径和为：

$$d_a + d_b + d_c - (d_{lca(a)} + d_{lca(b)} + d_{lca(c)})$$

此外，目的是三个 LCA 深度最大的一个，可以得知，最深的是不重复的

## 树链剖分

性质：

1. DFS 序新编号的重建树，保证子树  $x$  的节点范围是  $[x, x+siz[x]-1]$ 。因此，对于重建树，可以使用线段树等结构维护节点值。
2. 任意一条路径可以表示为不超过  $O(\log n)$  条连续重链(或部分重链)。

因此初始化的时间复杂度是  $O(n)$ ，使用线段树维护时，单路径查询的时间复杂度是  $O(\log^2 n)$ ，单子树查询的时间复杂度是  $O(\log n)$ 。

P3384 例题：给定  $n, m, r, p$ ， $r$  是根节点编号。对  $m$  次操作：

1.  $[1 \ x \ y \ z]$  将  $x, y$  路径上全部点加上  $z$
2.  $[2 \ x \ y]$  查询路径和对  $p$  取模
3.  $[3 \ x \ z]$  将  $x$  为根节点的子树(含根下同)全部点加上  $z$
4.  $[4 \ x]$  查询子树节点和对  $p$  取模

$$1 \leq n, m \leq 10^5, 1 \leq p \leq 2^{31} - 1$$

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #define sc(x) scanf("%lld", &x)
5 #define mn 100020
```

```

6 11 n, m, r, mod, ui, vi, k, x, y, z;
7 #define mkcf 11 cf = (lf + rf) >> 1
8 #define lfs p << 1
9 #define rfs p << 1 | 1
10 11 hd[mn], cnt, w[mn], wt[mn];
11 struct edge
12 {
13     11 to, nx;
14 } e[mn];
15 void adde(11 u, 11 v)
16 {
17     e[++cnt] = {v, hd[u]};
18     hd[u] = cnt;
19 }
20 11 a[mn << 2], laz[mn << 2], lc, rc, res, updv;
21 11 hvson[mn], id[mn], fa[mn], dfn, dep[mn], siz[mn], top[mn];
22
23 void pushdown(11 lf, 11 rf, 11 p)
24 {
25     mkcf;
26     laz[lfs] += laz[p], laz[rfs] += laz[p];
27     (a[lfs] += laz[p] * (cf - lf + 1)) %= mod;
28     (a[rfs] += laz[p] * (rf - cf)) %= mod;
29     laz[p] = 0;
30 }
31 void build(11 lf = 1, 11 rf = n, 11 p = 1)
32 {
33     if (lf == rf)
34     {
35         a[p] = wt[lf] % mod;
36         return;
37     }
38     mkcf;
39     build(lf, cf, lfs), build(cf + 1, rf, rfs);
40     a[p] = (a[lfs] + a[rfs]) % mod;
41 }
42 void query(11 lf = 1, 11 rf = n, 11 p = 1)
43 {
44     if (lc <= lf && rf <= rc)
45     {
46         (res += a[p]) %= mod;
47         return;
48     }
49     pushdown(lf, rf, p);
50     mkcf;
51     if (lc <= cf)
52     {
53         query(lf, cf, lfs);
54     }
55     if (rc > cf)
56     {
57         query(cf + 1, rf, rfs);
58     }
59 }
60 void update(11 lf = 1, 11 rf = n, 11 p = 1)
61 {

```

```

62     if (lc <= lf && rf <= rc)
63     {
64         laz[p] += updv;
65         a[p] += updv * (rf - lf + 1) % mod;
66         return;
67     }
68     pushdown(lf, rf, p);
69     mkcf;
70     if (lc <= cf)
71     {
72         update(lf, cf, lfs);
73     }
74     if (rc > cf)
75     {
76         update(cf + 1, rf, rfs);
77     }
78     a[p] = (a[lfs] + a[rfs]) % mod;
79 }
80
81 ll qrange(ll x, ll y)
82 {
83     ll ans = 0;
84     while (top[x] != top[y])
85     {
86         if (dep[top[x]] < dep[top[y]])
87         {
88             swap(x, y);
89         }
90         res = 0, lc = id[top[x]], rc = id[x], query();
91         (ans += res) %= mod;
92         x = fa[top[x]];
93     }
94     if (dep[x] > dep[y])
95     {
96         swap(x, y);
97     }
98     res = 0, lc = id[x], rc = id[y], query();
99     return (ans + res) % mod;
100 }
101
102 void frange(ll x, ll y, ll k)
103 {
104     updv = k % mod;
105     while (top[x] != top[y])
106     {
107         if (dep[top[x]] < dep[top[y]])
108         {
109             swap(x, y);
110         }
111         lc = id[top[x]], rc = id[x], update();
112         x = fa[top[x]];
113     }
114     if (dep[x] > dep[y])
115     {
116         swap(x, y);
117     }

```

```

118     lc = id[x], rc = id[y], update();
119 }
120
121 ll qson(ll x)
122 {
123     res = 0, lc = id[x], rc = id[x] + siz[x] - 1, query();
124     return res;
125 }
126
127 void fson(ll x, ll k)
128 {
129     updv = k % mod, lc = id[x], rc = id[x] + siz[x] - 1, update();
130 }
131
132 void dfs(ll x, ll f, ll deep)
133 {
134     dep[x] = deep, fa[x] = f, siz[x] = 1;
135     ll hvsonv = -1, y;
136     for (ll i = hd[x]; i; i = e[i].nx)
137     {
138         y = e[i].to;
139         if (y == f)
140         {
141             continue;
142         }
143         dfs(y, x, deep + 1);
144         siz[x] += siz[y];
145         if (siz[y] > hvsonv)
146         {
147             hvson[x] = y, hvsonv = siz[y];
148         }
149     }
150 }
151
152 void dfs(ll x, ll topi)
153 {
154     id[x] = ++dfn, wt[dfn] = w[x], top[x] = topi;
155     if (!hvson[x])
156     {
157         return;
158     }
159     dfs(hvson[x], topi);
160     for (ll i = hd[x]; i; i = e[i].nx)
161     {
162         ll y = e[i].to;
163         if (y == fa[x] || y == hvson[x])
164         {
165             continue;
166         }
167         dfs(y, y);
168     }
169 }
170
171 signed main()
172 {
173     sc(n), sc(m), sc(r), sc(mod);

```

```

174     for (ll i = 1; i <= n; ++i)
175     {
176         sc(w[i]);
177     }
178     for (ll i = 1; i < n; ++i)
179     {
180         sc(ui), sc(vi), adde(ui, vi), adde(vi, ui);
181     }
182     dfs(r, 0, 1), dfs(r, r), build();
183     while (m--)
184     {
185         sc(k), sc(x);
186         if (k == 1)
187         {
188             sc(y), sc(z), frange(x, y, z);
189         }
190         else if (k == 2)
191         {
192             sc(y), printf("%lld\n", qrange(x, y));
193         }
194         else if (k == 3)
195         {
196             sc(y), fson(x, y);
197         }
198         else
199         {
200             printf("%lld\n", qson(x));
201         }
202     }
203     return 0;
204 }

```

## 启发式合并

dsu on tree

预处理过程：DFS 一次，得到轻重儿子关系和 DFN 序(后者可选)

核心 DFS 合并过程：

1. DFS 轻儿子，合并所有轻子树上答案值
2. DFS 重儿子，合并重子树上的答案并保存当前总合并答案
3. 遍历当前节点的轻子树和当前节点，累加子树各点的答案值，保存答案
4. 遍历当前节点的轻子树，删掉轻子树合并值

简单理解就是，先 DFS 轻的部分，然后直接计算并保存其答案；然后因为其兄弟节点子树也有轻的，而当前子树不会给兄弟节点子树贡献，所以要把当前的答案全部丢弃，然后再计算兄弟节点子树。当轻子树都计算完后，计算重子树，但是不进行丢弃，此时回到当前的根，再次遍历全部轻子树永久加上轻子树的贡献。

在树上启发式合并里，整个板子是基本不用动的，唯一需要更改的地方是合并函数的增删处理。

用途：求出静态有根树的每个子树(显然共  $n$  个子树)的某个要求的值，这个过程是  $O(n \log n)$  的

根据轻重链剖分的性质可知，经过一条轻边时，子树的大小至少会除以二，所以树上任意一条路径最多包含  $\log n$  条轻边。如果某点到根节点经过了  $x$  条轻边，那么它的大小  $y$  满足：  $y < \frac{n}{2^x}$

一个节点被遍历的次数等于它到根节点上轻边的数目+1(重节点只被遍历一次)，根据上述性质，即一个节点最多被遍历  $\log n + 1$  次，所以时间复杂度是  $O(n \log n)$

边权问题可以把边权在 DFS1 时叠到点权(`v==fa continue` 后)，把 `add(u)` 和 `ans[u]=tot` 顺序对换

例题洛谷U41492：  $n(n \leq 10^5)$  个节点的树，每个节点有不同的颜色( $10^5$  种)，问每个节点为根的子树中，所有节点有多少种不同的颜色；  $m(m \leq 10^5)$  次询问，每次问一棵子树

写法一：DFN序

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 typedef double db;
5 #define sc(x) scanf("%lld", &x)
6 #define mn 100010
7 ll n, m, cnt, dfn[mn], big[mn], siz[mn], tot, c[mn], ecnt, hd[mn];
8 ll lf[mn], rf[mn], s[mn], ans[mn], x;
9 struct edge
10 {
11     ll to, nx;
12 } e[mn * 2];
13 void add(ll u, ll v)
14 {
15     e[++ecnt] = {v, hd[u]};
16     hd[u] = ecnt;
17 }
18 void dfs1(ll u, ll fa)
19 {
20     lf[u] = ++cnt, dfn[cnt] = u, siz[u] = 1;
21     for (ll i = hd[u], v; i; i = e[i].nx)
22     {
23         v = e[i].to;
24         if (v == fa)
25         {
26             continue;
27         }
28         dfs1(v, u);
29         siz[u] += siz[v];
30         if (siz[big[u]] < siz[v])
31         {
32             big[u] = v;
33         }
34     }
35     rf[u] = cnt;
36 }
37 void add(ll u)
38 {
39     tot += (s[c[u]]++ == 0);
40 }
41 void remove(ll u)
```

```

42     {
43         tot -= (c[u] == 0);
44     }
45     void dfs2(ll u, ll fa, bool save)
46     {
47         for (ll i = hd[u], v; i; i = e[i].nx)
48         {
49             v = e[i].to;
50             if (v != fa && v != big[u])
51             {
52                 dfs2(v, u, false);
53             }
54         }
55         if (big[u])
56         {
57             dfs2(big[u], u, true);
58         }
59         for (ll i = hd[u], v; i; i = e[i].nx)
60         {
61             v = e[i].to;
62             if (v != fa && v != big[u])
63             {
64                 for (ll j = lf[v]; j <= rf[v]; ++j)
65                 {
66                     add(dfn[j]);
67                 }
68             }
69         }
70         add(u);
71         ans[u] = tot;
72         if (!save)
73         {
74             for (ll j = lf[u]; j <= rf[u]; ++j)
75             {
76                 remove(dfn[j]);
77             }
78         }
79     }
80     signed main()
81     {
82         sc(n);
83         for (ll i = 1, u, v; i < n; ++i)
84         {
85             sc(u), sc(v), adde(u, v), adde(v, u);
86         }
87         for (ll i = 1; i <= n; ++i)
88         {
89             sc(c[i]);
90         }
91         dfs1(1, 0), dfs2(1, 0, false);
92         for (sc(m); m; --m)
93         {
94             sc(x);
95             printf("%lld\n", ans[x]);
96         }
97         return 0;

```

## 写法二：不带DFN序

```

1 #include <cstdio>
2 #include <cstring>
3 #define maxn 100010
4
5 int n,m,col[maxn];
6 struct edge{int y,next;};
7 edge e[maxn*2];
8 int first[maxn];
9 void buildroad(int x,int y)
10 {
11     static int len=0;
12     e[++len]=(edge){y,first[x]};
13     first[x]=len;
14 }
15 int size[maxn],mson[maxn];
16 void dfs1(int x,int fa)//求重儿子
17 {
18     size[x]=1;
19     for(int i=first[x];i;i=e[i].next)
20     {
21         int y=e[i].y;
22         if(y==fa)continue;
23         dfs1(y,x);
24         if(size[y]>size[mson[x]])mson[x]=y;
25         size[x]+=size[y];
26     }
27 }
28 int tong[maxn],ans[maxn],now_ans=0;
29 void go(int x,int fa,int type)
30 {
31     tong[col[x]]+=type;
32     if(type==1&&tong[col[x]]==1)now_ans++;
33     if(type==-1&&tong[col[x]]==0)now_ans--;
34     for(int i=first[x];i;i=e[i].next)
35         if(e[i].y!=fa)go(e[i].y,x,type);
36 }
37 void dfs2(int x,int fa,bool del)
38 //求解, del表示求完x的子树的答案后需不需要清空x的子树的信息
39 {
40     for(int i=first[x];i;i=e[i].next)//先统计轻儿子的答案
41         if(e[i].y!=fa&&e[i].y!=mson[x])dfs2(e[i].y,x,true);
42         if(mson[x]!=0)dfs2(mson[x],x,false);//最后统计重儿子的答案
43
44     tong[col[x]]++;if(tong[col[x]]==1)now_ans++;//统计自己以及轻子树的信息
45     for(int i=first[x];i;i=e[i].next)
46         if(e[i].y!=fa&&e[i].y!=mson[x])go(e[i].y,x,1);
47         ans[x]=now_ans;//得到自己的答案
48
49     if(del)go(x,fa,-1);//假如要删掉自己的信息, 就暴力地删掉
50 }
51

```

```

52 int main()
53 {
54     scanf("%d", &n);
55     for(int i=1, x, y; i<n; i++)
56         scanf("%d %d", &x, &y), buildroad(x, y), buildroad(y, x);
57     for(int i=1; i<=n; i++)
58         scanf("%d", &col[i]);
59     dfs1(1, 0);
60     dfs2(1, 0, false);
61     scanf("%d", &m);
62     for(int i=1, x; i<=m; i++)
63         scanf("%d", &x), printf("%d\n", ans[x]);
64 }
65

```

## 虚树

virtual tree

对于原树上的  $k$  个关键点，它们及其两两LCA所组成的树是虚树

实现方法是把关键点按 DFS 序排序，对两两相邻的关键点求 LCA，哈希表判重。根据原树的祖先后代关系建树。

在虚树里，只要保证祖先后代的关系没有改变，就可以随意添加原树的非关键点。

使用单调栈建立虚树。先把根节点 1 加入栈。按 DFS 序遍历节点，若当前节点与栈顶节点的 LCA 是栈顶，直接入栈；否则，把 LCA 连栈顶建虚树，栈顶弹栈。根据栈顶节点 DFS 序和次大节点(栈顶下方节点) DFS 序与 LCA 作比较，判断 LCA 是否入栈过，如果 LCA 未入栈，则 LCA 入栈。然后才新节点入栈。遍历结束后，栈内形成链，两两相连即可。

细节优化是可以把清空邻接表改成有一个从未入栈的元素入栈的时候清空该元素对应的邻接表。

例题洛谷P2495：有  $n(2 \leq n \leq 2.5 \times 10^5)$  节点的树，边带权，有  $m(1 \leq m \leq 5 \times 10^5)$  次询问，每次询问有  $k(1 \leq k \leq n, \sum k \leq 5 \times 10^5)$  个点，对每次询问，删掉树上若干边，使得删的边权和最小，满足删后根节点 1 不能到达任一个给定询问点

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #define mn 500010
5 #define mm 10000
6 #define big 0x3fffffffffffffa
7 #define mlg 20
8 struct edge
9 {
10     ll to, nx, w;
11 } e[mn << 1], e2[mn << 1];
12 ll hd[mn], hd2[mn], cnt, cnt2, n, q, k;
13 void adde(const ll &u, const ll &v, const ll &w, ll *hd, ll &cnt, edge *e)
14 {
15     e[++cnt] = {v, hd[u], w};
16     hd[u] = cnt;

```

```

17 }
18 ll dfn[mn], dep[mn], fa[mn][m1g], mi[mn], m[mn], lst[mn];
19 bool vis[mn];
20 ll num, top, dfscnt, stk[mn];
21
22 void dfs1(ll u)
23 {
24     ll k = 0;
25     for (; fa[u][k]; ++k)
26     {
27         fa[u][k + 1] = fa[fa[u][k]][k];
28     }
29     m[u] = k;
30     dfn[u] = ++dfscnt;
31     for (ll i = hd[u], v; i; i = e[i].nx)
32     {
33         v = e[i].to;
34         if (!dfn[v])
35         {
36             dep[v] = dep[u] + 1;
37             mi[v] = min(mi[u], e[i].w);
38             fa[v][0] = u;
39             dfs1(v);
40         }
41     }
42 }
43
44 ll lca(ll x, ll y)
45 {
46     if (dep[x] < dep[y])
47     {
48         swap(x, y);
49     }
50     for (ll i = m[x]; i >= 0; --i)
51     {
52         if (dep[fa[x][i]] >= dep[y])
53         {
54             x = fa[x][i];
55         }
56     }
57     if (x == y)
58     {
59         return x;
60     }
61     for (ll i = m[x]; i >= 0; --i)
62     {
63         if (fa[x][i] != fa[y][i])
64         {
65             x = fa[x][i], y = fa[y][i];
66         }
67     }
68     return fa[x][0];
69 }
70
71 ll dfs2(ll u) //dp
72 {

```

```

73     ll sum = 0;
74     for (ll i = hd2[u], v; i; i = e2[i].nx)
75     {
76         v = e2[i].to;
77         sum += dfs2(v);
78     }
79     ll res = vis[u] ? mi[u] : min(mi[u], sum); //vis[u] hd.d af
80     hd2[u] = 0, vis[u] = false; //clear
81     return res;
82 }
83
84 #define sc(x) scanf("%lld", &x)
85 signed main()
86 {
87     sc(n);
88     for (ll i = 1, u, v, w; i < n; ++i)
89     {
90         sc(u), sc(v), sc(w);
91         adde(u, v, w, hd, cnt, e), adde(v, u, w, hd, cnt, e);
92     }
93     mi[1] = big;
94     dfs1(1);
95     for (sc(q); q; --q)
96     {
97         sc(k);
98         for (ll i = 1; i <= k; ++i)
99         {
100             sc(lst[i]), vis[lst[i]] = true;
101         }
102         sort(lst + 1, lst + 1 + k, [](const ll &x, const ll &y)
103             { return dfn[x] < dfn[y]; });
104         stk[top = 1] = lst[1];
105         for (ll i = 2; i <= k; ++i)
106         {
107             ll now = lst[i], lc = lca(now, stk[top]);
108             while (true)
109             {
110                 if (dep[lc] >= dep[stk[top - 1]])
111                 {
112                     if (lc != stk[top])
113                     {
114                         adde(lc, stk[top], 0, hd2, cnt2, e2);
115                         if (lc != stk[top - 1])
116                         {
117                             stk[top] = lc;
118                         }
119                     else
120                     {
121                         --top;
122                     }
123                 }
124                 break;
125             }
126             else
127             {
128                 adde(stk[top - 1], stk[top], 0, hd2, cnt2, e2);

```

```

129         --top;
130     }
131     }
132     stk[++top] = now;
133 }
134 while (--top)
135 {
136     adde(stk[top], stk[top + 1], 0, hd2, cnt2, e2);
137 }
138 printf("%lld\n", dfs2(stk[1])), cnt2 = 0;
139 }
140 return 0;
141 }

```

## 点分治

求解问题：枚举所有路径，计算相关的信息

树上路径可以分为两种：一种是经过根的，一种是不经过的；对前者，一种根作为端点，一种不作为端点，其中后者等效于两条前者。

预处理：求重心，然后求以重心为根各子树的大小。然后从重心开始点分治。

在每次点分治过程：当前重心为根，DFS计算所有以该点为端点的路径所要求的值，然后删掉该点，对该点的每个子树，再次求重心，然后求新重心为根各子树大小，然后以新重心为根继续递归执行上述过程。

经过点分治，可以求出满足可加的每一条路径的值。点分治的复杂度是  $O(n \log n)$

例题洛谷P3806： $n(1 \leq n \leq 10^4)$  点带边权树， $m(1 \leq m \leq 100)$  次询问，每次询问求树上是否有距离为  $k$  的点对，有就输出 AYE

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #define sc(x) scanf("%lld", &x)
5 #define mn 10010
6 #define mm 105
7 #define mb 10000010
8 struct edge
9 {
10     ll to, nx, w;
11 } e[mn << 1];
12 ll hd[mn], ecnt;
13 void adde(ll &u, ll &v, ll &w)
14 {
15     e[++ecnt] = {v, hd[u], w};
16     hd[u] = ecnt;
17 }
18 bool died[mn], bin[mb], suc[mm];
19 ll n, m, q[mn];
20 ll p, sz[mn], n2, smx[mn];
21 void dfs(ll u, ll fa)
22 {

```

```

23     sz[u] = 1, smx[u] = 0;
24     for (ll i = hd[u], v; i; i = e[i].nx)
25     {
26         v = e[i].to;
27         if (v == fa || died[v])
28         {
29             continue;
30         }
31         dfs(v, u);
32         sz[u] += sz[v];
33         smx[u] = max(smx[u], sz[v]);
34     }
35     smx[u] = max(smx[u], n2 - smx[u]);
36     if (smx[u] < smx[p])
37     {
38         p = u;
39     }
40 }
41 ll ds[mn], dcnt, d[mn];
42 void dfs2(ll u, ll fa)
43 {
44     ds[dcnt] = d[u];
45     for (ll i = hd[u], v; i; i = e[i].nx)
46     {
47         v = e[i].to;
48         if (v == fa || died[v])
49         {
50             continue;
51         }
52         d[v] = d[u] + e[i].w;
53         dfs2(v, u);
54     }
55 }
56 stack<ll> qd;
57 void dfz(ll u, ll fa)
58 {
59     bin[0] = true;
60     qd.push(0);
61     died[u] = true;
62     for (ll i = hd[u], v; i; i = e[i].nx)
63     {
64         v = e[i].to;
65         if (v == fa || died[v])
66         {
67             continue;
68         }
69         d[v] = e[i].w;
70         dfs2(v, u);
71         for (ll j = 1; j <= dcnt; ++j)
72         {
73             for (ll k = 1; k <= m; ++k)
74             {
75                 if (qd[k] >= ds[j])
76                 {
77                     suc[k] |= bin[qd[k] - ds[j]];
78                 }

```

```

79         }
80     }
81     for (ll j = 1; j <= dcnt; ++j)
82     {
83         if (ds[j] < mb)
84         {
85             bin[ds[j]] = true;
86             qd.push(ds[j]);
87         }
88     }
89     dcnt = 0;
90 }
91 while (!qd.empty())
92 {
93     bin[qd.top()] = false;
94     qd.pop();
95 }
96 for (ll i = hd[u], v; i; i = e[i].nx)
97 {
98     v = e[i].to;
99     if (v == fa || died[v])
100    {
101        continue;
102    }
103    n2 = sz[v], p = 0;
104    dfs(v, u), dfs(p, 0);
105    dfz(p, u);
106 }
107 }
108 signed main()
109 {
110     sc(n), sc(m), n2 = n, smx[0] = 0x7fffffff;
111     for (ll i = 1, u, v, w; i < n; ++i)
112     {
113         sc(u), sc(v), sc(w), adde(u, v, w), adde(v, u, w);
114     }
115     for (ll i = 1; i <= m; ++i)
116     {
117         sc(q[i]);
118     }
119     dfs(1, 0), dfs(p, 0);
120     dfz(p, 0);
121     for (ll i = 1; i <= m; ++i)
122     {
123         printf(suc[i] ? "AYE\n" : "NAY\n");
124     }
125     return 0;
126 }

```

## Prufer 序列

常用于解决与度数相关的树上计数问题

将无根代表号树与整数序列一一对应的方法

树转 Prufer 序列：每次选择一个编号最小的叶结点并删掉它，然后在序列中记录下它连接到的那个结点。重复  $n - 2$  次后就只剩下两个结点

Prufer 序列转树：每次选择度为 1 的节点与当前点连接

Prufer 序列性质：剩下的 2 个节点其中一个是编号最大节点；每个节点在序列出现的次数是度数减一

Cayley 公式：完全图  $K_n$  有  $n^{n-2}$  棵生成树

有  $n$  点树，给定度数，则树可能的形态数为  $\frac{(n-2)!}{\prod_{i=1}^n (d_i - 1)!}$

含未知度数时，设已知点数为  $k$ ，已知度数和为  $\sum(d_i - 1) = s$ ，则答案为

$$\frac{C_{n-2}^s s! (n-k)^{n-2-s}}{\prod_{i=1}^k (d_i - 1)!}$$

$n$  点  $m$  边带标号无向图有  $k$  个连通块，添加  $k - 1$  条边使得图连通，方案数为  $n^{k-2} \prod_{i=1}^k s_i$ ， $s_i$  是每个连通块数量，有  $\sum_{i=1}^n s_i = n$

例题洛谷P6086：输入  $n, m (2 \leq n \leq 5 \times 10^6, 1 \leq m \leq 2)$ ，设根为  $n$ ， $m = 1$  时输入

$[1, n - 1]$  的父亲序列， $m = 2$  输入 Prufer 序列，将其转为另一方。设一个序列的值为

$\oplus_{i=1}^{\text{len}} i \times a_i$ ，求转换后序列权值

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef int ll;
4 #define MAXN 5000002
5 ll n, m, f[MAXN], p[MAXN], d[MAXN];
6 long long ans;
7 signed main()
8 {
9     scanf("%d%d", &n, &m);
10    if (m == 1) //树转Prufer
11    {
12        for (ll i = 1; i < n; ++i)
13            scanf("%d", f + i), ++d[f[i]];
14        for (ll i = 1, j = 1; i <= n - 1; ++i, ++j)
15        {
16            while (d[j])
17                ++j;
18            p[i] = f[j];
19            while (i <= n - 2 && !--d[p[i]] && p[i] < j)
20                p[i + 1] = f[p[i]], ++i;
21        }
22        for (ll i = 1; i <= n - 2; ++i)
23            ans *= 1LL * i * p[i];
24    }
25    else // Prufer转树
26    {
27        for (ll i = 1; i <= n - 2; ++i)
28            scanf("%d", p + i), ++d[p[i]];
29    }
30 }
```

```

29         p[n - 1] = n;
30         for (ll i = 1, j = 1; i < n; ++i, ++j)
31         {
32             while (d[j])
33                 ++j;
34             f[j] = p[i];
35             while (i < n && !--d[p[i]] && p[i] < j)
36                 f[p[i]] = p[i + 1], ++i;
37         }
38         for (ll i = 1; i < n; ++i)
39             ans *= 1LL * i * f[i];
40     }
41     printf("%lld", ans);
42     return 0;
43 }

```

## 二叉树遍历

### 先中求后

离散化优化，复杂度  $O(n)$ ，以洛谷P1827为例(顶点值不重复)

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 string pre, mid, post;
4 typedef long long ll;
5 ll n, idx, h[256];      // index of pre
6 void dfs(ll lf, ll rf) // search in [lf,rf)
7 {
8     if (lf >= rf || idx == n)
9         return;
10    char v = pre[idx++];
11    ll cf = h[v];
12    dfs(lf, cf);
13    dfs(cf + 1, rf);
14    post.push_back(v);
15 }
16 signed main()
17 {
18     cin >> mid >> pre;
19     n = mid.size();
20     for (ll i = 0, ie = mid.size(); i < ie; ++i)
21         h[mid[i]] = i;
22     dfs(0, n);
23     cout << post;
24     return 0;
25 }

```

## 中后求先

离散化优化，复杂度  $O(n)$ ，洛谷P1030为例

```
1 //输入中序和后续，返回先序
2 #include <bits/stdc++.h>
3 using namespace std;
4 typedef long long ll;
5 ll h[256], n;
6 char mid[256], post[256];
7 void dfs(ll mlf, ll mrf, ll plf, ll prf) // mid,post
8 {
9     if (mlf > mrf || plf > prf)
10         return;
11     char c = post[prf];
12     int cf = h[(ll)c];
13     cout << c;
14     dfs(mlf, cf - 1, plf, plf + cf - mlf - 1);
15     dfs(cf + 1, n - 1, plf + cf - mlf, prf - 1);
16 }
17 signed main()
18 {
19     cin >> mid >> post;
20     n = strlen(mid);
21     for (ll i = 0, ie = n; i < ie; ++i)
22         h[(ll)mid[i]] = i;
23     dfs(0, n - 1, 0, n - 1); // [0,n-1], [0,n-1]
24     return 0;
25 }
```

## 先后求中

离散化，答案数为先序中  $AB$  与后序中  $BA$  相等数  $x$  的  $2^x$ ，复杂度  $O(n)$  (SCNUOJ 1423为例)

```
1 #include <bits/stdc++.h>
2 #define mod 1000000007
3 #define mn 100002
4 typedef long long ll;
5 ll n, a[mn], b[mn], c[mn], r = 1;
6 signed main()
7 {
8     scanf("%lld", &n);
9     for (ll i = 1; i <= n; ++i) //先序
10         scanf("%lld", a + i);
11     for (ll i = 1; i <= n; ++i) //后序
12         scanf("%lld", b + i), c[b[i]] = i;
13     for (ll i = 1; i <= n; ++i)
14     {
15         ll k = c[a[i]];
16         if (i != n && a[i + 1] == b[k - 1])
17             (r *= 2) %= mod;
18     }
19     printf("%lld", r);
20 }
```

```
20     return 0;
21 }
```

## 杂项

### 随机游走

给定一棵树，树的某个结点上有一个硬币，在某一时刻硬币会等概率地移动到邻接结点上，问硬币移动到邻接结点上的期望距离。

设  $f(u)$  为  $u$  节点走到父节点  $p_u$  的期望距离，有：

$$f(u) = \sum_{(u,t) \in E} w(u,t) + \sum_{v \in son_u} f(v)$$

若  $w = 1$  则即  $u$  子树所有节点的度数和，即  $u$  子树大小的二倍减一。

设  $g(u)$  代表  $p_u$  走向子节点  $u$  的期望距离，有：

$$g(u) = g(p_u) + f(p_u) - f(u)$$

显然初始值是  $f_{leaf} = 1, g_{root} = 0$ 。

```
1 vector<int> G[maxn];
2 void dfs1(int u, int p) {
3     f[u] = G[u].size();
4     for (auto v : G[u]) {
5         if (v == p) continue;
6         dfs1(v, u);
7         f[u] += f[v];
8     }
9 }
10 void dfs2(int u, int p) {
11     if (u != 1) g[u] = g[p] + f[p] - f[u];
12     for (auto v : G[u]) {
13         if (v == p) continue;
14         dfs2(v, u);
15     }
16 }
```

## 最小距离和

例题P1364：给定  $n (1 \leq n \leq 100)$  节点二叉树(输入为  $w, u, v$  为第  $i$  个节点点权和左右儿子(0是无子))，求最小距离和(以某点为原点各点点权乘以路径长) ( $1 \leq w \leq 10^5$ )

设  $dp$  为距离和，对  $u$  的邻点  $v$ ，有  $dp[v] = dp[u] + size[1] - 2size[v]$

在  $dp[u]$  的基础上当根从  $u$  变为  $v$  的时候， $v$  的子树的所有结点的距离都减少 1，那么总距离就减少  $size[v]$ ，同时，以  $v$  为根的子树以外的所有节点路程都增加了 1，总路程就增加了  $size[1] - size[v]$

```
1 #include <bits/stdc++.h>
2 using namespace std;
```

```

3  typedef long long ll;
4  #define mn 105
5  #define sc(x) scanf("%lld", &x)
6  struct edge
7  {
8      ll to, nx;
9  } e[mn << 1];
10 ll hd[mn], cnt, siz[mn], w[mn], dp[mn], n, ans = 0x7fffffff;
11 void adde(ll u, ll v)
12 {
13     e[++cnt] = {v, hd[u]};
14     hd[u] = cnt;
15 }
16 void dfs1(ll u, ll fa, ll d)
17 {
18     siz[u] = w[u];
19     for (ll i = hd[u], v; i; i = e[i].nx)
20     {
21         v = e[i].to;
22         if (v == fa)
23         {
24             continue;
25         }
26         dfs1(v, u, d + 1);
27         siz[u] += siz[v];
28     }
29     dp[1] += d * w[u];
30 }
31 void dfs2(ll u, ll fa)
32 {
33     for (ll i = hd[u], v; i; i = e[i].nx)
34     {
35         v = e[i].to;
36         if (v == fa)
37         {
38             continue;
39         }
40         dp[v] = dp[u] + (siz[1] - siz[v]) - siz[v];
41         dfs2(v, u);
42     }
43     ans = min(ans, dp[u]);
44 }
45 signed main()
46 {
47     scanf("%lld", &n);
48     for (ll i = 1, x, y; i <= n; ++i)
49     {
50         sc(w[i]), sc(x), sc(y);
51         if (x)
52         {
53             adde(i, x), adde(x, i);
54         }
55         if (y)
56         {
57             adde(i, y), adde(y, i);
58         }
}

```

```

59     }
60     dfs1(1, 0, 0);
61     dfs2(1, 0);
62     printf("%lld", ans);
63     return 0;
64 }

```

## 基本概念

这一章节主要用作英文题面阅读理解和单词/符号翻译，并提供简单图论性质(下划线)

**起点** (tail) , **终点** (head) 是有向边(又称弧 arc)  $u \rightarrow v$ 。点数是阶 (order)。无向图里，边  $e$  和点  $v$  **关联** (incident) 是  $e$  的一个**端点** (endpoint) 为  $v$ ，点的领域记作  $N(v)$ ，即相邻点

度为  $d(v) = 1$  的节点称为**叶节点或悬挂点** (leaf/pendant vertex), 按度数奇偶称**奇/偶点** (odd/even vertex), **奇点个数一定是偶数**，**支配点** (universal vertex) 是度为  $|V| - 1$  的点。最小度记作  $\delta(G)$ ，最大度记作  $\Delta(G)$ ，**入度** (in-degree) 记作  $\delta^-(v)$ ，**出度** (out-degree) 记作  $\delta^+(v)$

无**自环** (loop)、**重边** (multiple edge) 的图是**简单图** (simple graph)，否则是**多重图** (multigraph)。具有至少两顶点的简单无向图一定存在度相同的节点(抽屉原理)。有向图  $u \rightarrow v, v \rightarrow u$  不是重边。序列能作为图/简单图的度数列称为序列是可图化/可简单图化的。

**途径**(walk)是若干点连接起来的边的集合，**迹**(trail)是边互不相同的途径。**路径**(path)/**简单路径**(simple path)，**回路**(circuit)是除首尾点，其余点互不相同的迹。边的数量(或边权和)是长度(该段落定义视具体题目题面)

若点集和边集都是某图子集，那么是**子图** (subgraph)  $H \subseteq G$ 。**真子图** 称为 proper-subgraph。若子图只删点( $\forall u, v \in V_H, (u, v) \in E_G \rightarrow (u, v) \in E_H$ )，是**导出子图/诱导子图** (induced subgraph)，只由点集决定，记作  $G[V']$ 。若顶点与母图一样，那么称为**生成子图/支撑子图** (spanning subgraph)。原图或空图称为**平凡子图** (trivial subgraph)。 $k$ -**正则图** ( $k$ -regular graph) 是每个点度均为  $k$  的无向图。若无向图生成子图  $F$  是  $k$ -正则图，那么  $F$  是原图的  $k$ -因子 ( $k$ -factor)。若有向图导出子图的每个点  $v$  满足  $(v, u) \in E$  (在原图边集)且  $u$  在导出子图，那么称为**闭合子图** (closed subgraph)。从图删去  $V'$  的点记作  $G[V \setminus V']$ 。删边同理。删点可记作  $G - V$ 。

**连通**(connected) 是无向图任两点可达。**强连通**是任意两点连通的有向图，**弱连通**是有向图转无向图后成为连通图的图。有**连通块(连通分量)**概念及其强/弱连通分量 (SCC)概念。**连通图点割集** (vertex cut/separating set)大小为 1 叫做**割点** / **割顶** (cut vertex)。若  $|V| \geq k + 1$  且不存在  $k - 1$  大小点割集，图是  $k$ -**点连通的** ( $k$ -vertex-connected)，其最大  $k$  叫**点连通度** (vertex connectivity) 记作  $\kappa(G)$  (非完全图是最小点割集大小，完全图是  $n - 1$ )。使得本可达的  $u, v$  不连通的最小点割集(不考虑其他点连通性)大小是局部点连通度 (local connectivity) 记作  $\kappa(u, v)$ 。同理可以定义边割集和桥， $k$ -**边连通的**，**边连通度**  $\lambda(G)$  和局部边连通度  $\lambda(u, v)$ 。**点双连通** (biconnected) 是除单边双点图外的 2-**点连通** (是没有割点的连通图)。边双连通等于 2-**边连通**。类似有(极大)点/边双连通分量。Whitney 定理: 任意图  $G$   $\kappa(G) \leq \lambda(G) \leq \delta(G)$

一般边数接近点数平方叫稀疏图 (sparse graph)，否则稠密图 (dense graph)。**补图** (complement graph)  $\bar{G}$  是  $(u, v) \in E(\bar{G})$  当且仅当  $(u, v) \notin E(G)$  补图是无向简单图。**反图** (transpose graph) 是有向图每条边反向。**完全图** (complete graph) 是无向简单图，任两点有边，记作  $K_n$ 。有向图任两点有两条互为反向的边记作**有向完全图** (complete digraph)。有向简单图两点间有一条单向边是**竞赛图** (tournament graph)。边集是空叫**零图** (null graph)，记作  $N_n$ 。无向简单图所有边构成一个圈称为**环图/圈图** (cycle graph)， $n \geq 3$  记作  $C_n$ ，充要条件是为 2-正则连通图。若无向简单图满足一个点是支配点，其他点无边相连，称为**星图/菊花图** (star graph),  $n \geq 1, n + 1$  阶星图记作  $S_n$ 。无向简单图满足一个点是支配点，其他点构成一个圈是**轮图** (wheel graph)  $n + 1 (n \geq 3)$  阶轮图是  $W_n$ 。无向简单图

所有边构成简单路径，称为**链**(chain/path graph)，记作 $P_n$ 。无向连通无环图是**树**。无向连通、恰含一环是**基环树**(pseudotree)。有向弱连通，入度均1图是基环外向树，出度均为1是基环内向树。多棵树组成**森林**(基环同理)。基环内向森林：functional graph。无向连通图每条边最多在一个环内是**仙人掌**cactus，仙人掌组成**沙漠**。**二分图**bipartite graph，任何两个不在同一部分的点都有连边是**完全二分图**(complete bipartite graph/biclique)记作 $K_{n,m}$ 。

若 $E' \subseteq E$ 且 $E'$ 任两条不同的边没有公共端点，且不是自环，那么 $E'$ 是一个**匹配**(matching)/**边独立集**(independent edge set)。若一个点是匹配中某边的端点，称为**被匹配的**(matched)/**饱和的**(saturated)，否则是**不被匹配的**(unmatched)。边数最多的匹配是一张图的**最大匹配**(maximum-cardinality matching)，记作 $v(G)$ 。权重和最大的匹配是**最大权匹配**(maximum-weight matching)。匹配加入任意边后不再是匹配，称为**极大匹配**(maximal matching)。最大的极大匹配就是最大匹配，任何最大匹配都是极大匹配。极大匹配一定是边支配集，但边支配集不一定是匹配。最小极大匹配和最小边支配集大小相等，但最小边支配集不一定是匹配。若一个匹配里所有点都是被匹配的，称为**完美匹配**/完备匹配/完全匹配(perfect matching)。只有一个点不被匹配是准完美匹配(near-perfect matching)。对匹配 $M$ ，若一条路径以非匹配点为起点，每相邻两条边的一条在匹配中另一条不在，称为**交替路径**(alternating path)，在非匹配点终止的交替路径称为**增广路径**(augmenting path)。托特定理： $n$ 阶无向图 $G$ 有完美匹配当且仅当 $\forall V' \subseteq V(G), p^{\text{奇}}(G - V') \leq |V'|$  (奇数阶连通分支数)，任何无桥3-正则图都有完美匹配。

若 $V' \subseteq V$ 且 $\forall e \in E$ 满足 $e$ 至少一个端点在 $V'$ 中，称 $V'$ 为一个**点覆盖**(vertex cover)。点覆盖集必为支配集，但极小点覆盖集不一定是极小支配集。一个点集是点覆盖的充要条件是其补集是独立集，因此最小点覆盖的补集是最大独立集。一张图的任何一个匹配的大小都不超过其任何一个点覆盖的大小。完全二分图 $K_{n,m}$ 的最大匹配和最小点覆盖大小都为 $\min(n, m)$ 。

若 $E' \subseteq E$ 且 $\forall v \in V$ 满足 $v$ 与 $E'$ 至少一条边相邻，称为**边覆盖**(edge cover)，最小边覆盖的大小记作 $\rho(G)$ 。对于所有非匹配点，将其一条邻边加入最大匹配中，即得到了一个最小边覆盖。也可以由最小边覆盖求得：对于最小边覆盖中每对有公共点的边删去其中一条。满足 $\rho(G) + v(G) = |V(G)|$ 且 $v(G) \leq \rho(G)$ 特别地，完美匹配一定是一个最小边覆盖，这也是上式取到等号的唯一情况。一张图的任何一个独立集的大小都不超过其任何一个边覆盖的大小。完全二分图 $K_{n,m}$ 的最大独立集和最小边覆盖大小都为 $\max(n, m)$ 。

一张图可以画在一个平面内，无两条边在非端点处相交，是**平面图**(planar graph)，任何子图都不是 $K_5$ 或 $k_{3,3}$ 是为平面图的充要条件。简单连通平面图若 $V \geq 3$ 则 $|E| \leq 3|V| - 6$ 。

若存在双射 $f: V(G) \rightarrow V(H)$ ，满足 $(u, v) \in E(G)$ ，当且仅当 $(f(u), f(v)) \in E(H)$ 称 $f$ 为 $G$ 到 $H$ 的一个**同构**(isomorphism)，图是同构的(isomorphic)，记作 $G \cong H$ ，必须满足点数边数相同，结点度非增序列相同且存在同构导出子图。

图的交、并是点集、边集分别作交、并。图的和/直和(sum/direct sum)是任意构造 $H' \cong H$ 使得 $V(H') \cap V_1 = \emptyset$ ( $H'$ 可以等于 $H$ )，此时与 $G \cup H'$ 同构的任何图叫做 $G$ 和 $H$ 的和/直和/不交并，记作 $G + H$ 或 $G \oplus H$ 。若点集本身不交，则 $G \cup H = G + H$ 。如森林是各树的和。可以理解为，“并”会让两张图中“名字相同”的点、边合并，而“和”则不会。

若 $E' \subseteq E$ 且 $\forall e \in (E \setminus E')$ 存在 $E'$ 中的边与其有公共点，称 $E'$ 是图 $G$ 的一个边支配集(edge dominating set)。若 $V' \subseteq V$ 且 $V'$ 中任意两点都不相邻，则 $V'$ 是一个**独立集**(independent set)。大小记作 $\alpha(G)$ 。若 $V' \subseteq V$ 且 $V'$ 任意两个不同顶点都相邻，则 $V'$ 是一个团(clique)，团的导出子图是完全图。如果一个团在加入任何一个顶点后都不再是一个团，则这个团是一个极大团(Maximal clique)。最大团大小是 $w(G)$ ，满足 $w(G) = \alpha(\bar{G})$ 。

求最大团、最小点覆盖、最大独立集、最小边支配、最小支配集是NP困难的，求完美匹配个数是#P完全的。

网络(流网络)(flow network)是有向图, 边权为容量  $c$  capacity, 有源点source和汇点sink。流量  $f(u, v)$  不超容量限制, 反流量与其相加为零, 源点流出流量等于汇点流入流量。网络流量是源点发出的流量和。剩余容量residual capacity  $c_f(u, v)$  为  $c(u, v) - f(u, v)$ 。剩余容量大于 0 的边构成的子图是残量网络 residual network  $G_f$ 。增广路 augmenting path: 从源点到汇点所有边剩余容量大于零。最小割: 删掉  $X$  条边使得源点汇点不通, 让  $X$  条边加起来的流量总和最小。

## 最短路

最短路存在的条件为无负环。

### floyd

全源最短路, 适用于任何无负环的图。记得初始化为半倍max/极大值, 留意重边

```

1  for (k = 1; k <= n; k++)
2      for (i = 1; i <= n; i++)
3          for (j = 1; j <= n; j++)
4              f[i][j] = min(f[i][j], f[i][k] + f[k][j]);

```

第  $k$  次循环时, 表示只经过前  $k$  个节点时, 最短路的大小。记录路径可以获得更小时设  $pre[i][j]=k$ 。

可以求最长路(输入负, 输出再负一次)或者上述改为 max。

输出方案:

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 using ll = long long;
4 const ll mn = 1e2 + 10;
5 ll n, m, d[mn][mn], nx[mn][mn];
6 signed main()
7 {
8     ios::sync_with_stdio(false), cin.tie(0);
9     cin >> n >> m;
10    memset(d, 0x3f, sizeof d);
11    for (ll i = 1; i <= n; ++i)
12    {
13        d[i][i] = 0, nx[i][i] = i;
14    }
15    for (ll i = 1, u, v, w; i <= m; ++i)
16    {
17        cin >> u >> v >> w;
18        d[u][v] = w, nx[u][v] = v;
19    }
20    for (ll k = 1; k <= n; ++k)
21    {
22        for (ll i = 1; i <= n; ++i)
23        {
24            for (ll j = 1; j <= n; ++j)
25            {
26                if (d[i][j] > d[i][k] + d[k][j])
27                {

```

```

28             d[i][j] = d[i][k] + d[k][j];
29             nx[i][j] = nx[i][k];
30         }
31     }
32 }
33
34 ll q;
35 cin >> q;
36 for (ll u, v; q--;) {
37 {
38     cin >> u >> v;
39     cout << u << ' ';
40     while (u != v)
41     {
42         cout << nx[u][v] << ' ';
43         u = nx[u][v];
44     }
45     cout << '\n';
46 }
47 return 0;
48 }

```

应用：

hdu1599-给定  $n(\leq 100)$  点和  $m(\leq 10^3)$  双向带权路，从任一点出发经过共至少 3 个不同点然后回到该点成一个环，求最小带权环，无环输出 `It's impossible.`

环长等于  $d(a, b) + e(b, c) + e(c, a)$ ，枚举起点  $k$ ，跑 floyd 最短路。

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define sc(x) scanf("%lld", &x)
4 typedef long long ll;
5 #define mn 101
6 ll n, m, f[mn][mn], e[mn][mn], ans, big = 1e9, u, v, w;
7 signed main()
8 {
9     while (EOF != scanf("%lld%lld", &n, &m))
10    {
11        for (ll i = 1; i <= n; ++i)
12            for (ll j = 1; j <= n; ++j)
13                f[i][j] = e[i][j] = big;
14        while (m--)
15        {
16            sc(u), sc(v), sc(w); //防重边
17            e[u][v] = e[v][u] = f[u][v] = f[v][u] = min(e[u][v], w);
18        }
19        ans = big;
20        for (ll k = 1; k <= n; ++k)
21        {
22            for (ll i = 1; i < k; ++i)
23                for (ll j = i + 1; j < k; ++j)
24                    ans = min(ans, f[i][j] + e[j][k] + e[k][i]);
25            for (ll i = 1; i <= n; ++i)

```

```

26         for (ll j = 1; j <= n; ++j)
27             f[i][j] = min(f[i][j], f[i][k] + f[k][j]);
28     }
29     ans == big ? printf("It's impossible.\n") : printf("%lld\n", ans);
30 }
31 return 0;
32 }

```

poj3660-给定  $n(1 \leq n \leq 100)$  点和  $m(1 \leq m \leq 4500)$  关系, 每个关系表示点  $u$  优于点  $v$ , 问根据这些关系能确定多少个点的排名(保证无自环和环, 无矛盾)

floyd跑传递闭包, 得到可达矩阵。如果一个点同时被  $n - 1$  个点可达(无论是自己大于这些点还是这些点小于它), 就表明它的位置唯一确定, floyd 更新代码用:

```
1 | f[i][j] |= (f[i][k] & f[k][j]);
```

如果用 bitset 优化, 复杂度还可以达到  $O(\frac{n^3}{w})$ :

```

1 #include <cstdio>
2 #include <bitset>
3 using namespace std;
4 #define sc(x) scanf("%lld", &x)
5 typedef long long ll;
6 #define mn 105
7 ll n, m, ans, u, v;
8 bitset<mn> f[mn];
9 signed main()
10 {
11     sc(n), sc(m);
12     while (m--)
13         sc(u), sc(v), f[v][u] = 1;
14     for (ll k = 1; k <= n; ++k)
15         for (ll i = 1; i <= n; ++i)
16             if (f[i][k])
17                 f[i] |= f[k];
18     for (ll i = 1; i <= n; ++i)
19     {
20         ll cnt = 0;
21         for (ll j = 1; j <= n; ++j)
22             cnt += (f[i][j] | f[j][i]);
23         ans += cnt == n - 1;
24     }
25     printf("%lld", ans);
26     return 0;
27 }
```

拓展: 如果要求出完整排序, 可以判拓扑排序是否无环且能遍历到所有点, 其完整排序就是拓扑排序序列

## Bellman-Ford

不断尝试对图上每一条边进行松弛。我们每进行一轮循环，就对图上所有的边都尝试进行一次松弛操作，当一次循环中没有成功的松弛操作时，算法停止。复杂度  $O(nm)$ 。

可以求出有负权的图的最短路，并可以对最短路不存在的情况进行判断。最严谨的做法是建立一个超级源点，向图上每个节点连一条权值为 0 的边，然后以超级源点为起点执行 Bellman-Ford 算法。若第  $n$  轮迭代仍有结点的最短路能被更新，则图中有负环

洛谷P3385-多组样例，输入  $w$  为正是无向边，负是有向边 ( $u \rightarrow v$ )，问是否存在从顶点 1 出发可达的负环  $T \leq 10, 1 \leq n \leq 2 \times 10^3, 1 \leq m \leq 3 \times 10^3$

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 const int N = 10010, big = 0xffffffff;
4 int cnt = 0;
5 struct node
6 {
7     int x, y, v;
8 } e[N];
9 void add(int x, int y, int v) { e[++cnt] = {x, y, v}; }
10 int n;
11 bool bellman()
12 {
13     static int d[N];
14     d[1] = 0;
15     for (int i = 2; i <= n; i++)
16         d[i] = big;
17     for (int i = 1; i <= n - 1; i++)
18         for (int j = 1; j <= cnt; j++)
19         {
20             if (d[e[j].x] != big &&
21                 d[e[j].x] + e[j].v < d[e[j].y])
22                 d[e[j].y] = d[e[j].x] + e[j].v;
23         }
24     for (int i = 1; i <= cnt; i++)
25     {
26         if (d[e[i].x] == big || d[e[i].y] == big)
27             continue;
28         if (d[e[i].x] + e[i].v < d[e[i].y])
29             return true; // 负权回路
30     }
31     return false;
32 }
33 signed main()
34 {
35     int t;
36     scanf("%d", &t);
37     while (t--)
38     {
39         memset(e, 0, sizeof(e));
40         cnt = 0;
41         int m;
42         scanf("%d%d", &n, &m);
43         for (int i = 1; i <= m; i++)
44         {
```

```

45         int x, y, v;
46         scanf("%d%d%d", &x, &y, &v);
47         if (v < 0)
48             add(x, y, v);
49         if (v >= 0)
50             add(x, y, v), add(y, x, v);
51     }
52     if (bellman())
53         printf("YES\n");
54     else
55         printf("NO\n");
56 }
57 return 0;
58 }
```

若求最长路，可以设初始值为  $-1$  (或负无穷)，松弛反号(起点不为  $-1$  可以继续)。

## SPFA

队列优化的 SPFA(某些随机图跑得很快)，用途是判负环。最坏情况仍然是  $O(nm)$  (堆/栈优化可以被卡指教级复杂度)。记录最短路经过了多少条边，当经过了至少  $n$  条边时，说明  $s$  点可以抵达一个负环。在没有负权边时最好使用 Dijkstra 算法。下面是负环模板：

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define MAXN 2002
4 #define MAXM 6002
5 #define BIG (1 << 31) - 1
6 #define inita(a, n, v) for (int i = 0; i <= n; i++) a[i] = v
7 int n, m, hd[MAXN], cnt, d[MAXN], cnts[MAXN], ui, vi, wi, ttn;
8 bool vis[MAXN];
9 queue<int> q;
10 struct edge
11 {
12     int to, d, nx;
13 } e[MAXM];
14 inline void adde(int u, int v, int w)
15 {
16     e[++cnt].d = w;
17     e[cnt].to = v;
18     e[cnt].nx = hd[u];
19     hd[u] = cnt;
20 }
21 void spfa()
22 {
23     inita(d, n, BIG);
24     inita(cnts, n, 0);
25     inita(vis, n, false);
26     while (!q.empty())
27     {
28         q.pop();
29         d[1] = 0;
30         vis[1] = true; // 表示当前点是否在队列内
31         q.push(1);
32     }
33 }
```

```

31     int u;
32     while (!q.empty())
33     {
34         u = q.front();
35         vis[u] = false;
36         q.pop();
37         for (int i = hd[u]; i != -1; i = e[i].nx)
38         {
39             if (d[u] + e[i].d < d[e[i].to])
40             {
41                 d[e[i].to] = d[u] + e[i].d;
42                 if (!vis[e[i].to])
43                 {
44                     if (++cnts[e[i].to] >= n)//该单源最短路的长度
45                     {
46                         printf("YES\n");
47                         return;
48                     }
49                     vis[e[i].to] = true;
50                     q.push(e[i].to);
51                 }
52             }
53         }
54     }
55     printf("NO\n");
56     return;
57 }
58 int main()
59 {
60     scanf("%d", &ttn);
61     while (ttn--)
62     {
63         scanf("%d%d", &n, &m);
64         cnt = -1;
65         inita(hd, n, -1);
66         while (m--)
67         {
68             scanf("%d%d%d", &ui, &vi, &wi);
69             adde(ui, vi, wi);
70             if (wi >= 0)
71                 adde(vi, ui, wi);
72         }
73         spfa();
74     }
75     return 0;
76 }

```

求负环并输出方案：

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 using ll = long long;
4 const ll mn = 5e3 + 10, big = 1e18;
5 ll n, m, d[mn], pr[mn];
6 struct edge

```

```

7  {
8      ll u, v, w;
9  } e[mn * 2];
10 bool spfa(ll s)
11 {
12     fill_n(d, n + 3, 1e18);
13     d[s] = 0;
14     for (ll i = 1; i <= n; ++i)
15     {
16         for (ll j = 1; j <= m; ++j)
17         {
18             ll u = e[j].u, v = e[j].v, w = e[j].w;
19             if (d[v] > d[u] + w)
20             {
21                 d[v] = d[u] + w;
22                 pr[v] = j;
23             }
24         }
25     }
26     for (ll j = 1; j <= m; ++j)
27     {
28         ll u = e[j].u, v = e[j].v, w = e[j].w;
29         if (d[v] > d[u] + w)
30         {
31             ll a = u;
32             for (ll i = 1; i <= n; ++i)
33             {
34                 a = e[pr[a]].u;
35             }
36             stack<ll> ans;
37             ans.push(a);
38             for (ll b = e[pr[a]].u; b != a; b = e[pr[b]].u)
39             {
40                 ans.push(b);
41             }
42             cout << "exists\n"
43                 << ans.size() << '\n';
44             while (!ans.empty())
45             {
46                 ll c = ans.top();
47                 ans.pop();
48                 cout << c << ' ';
49             }
50             return false;
51         }
52     }
53     cout << "not exists\n";
54     return true;
55 }
56 signed main()
57 {
58     cin.tie(0)->ios::sync_with_stdio(false);
59     cin >> n >> m;
60     for (ll i = 1, u, v, w; i <= m; ++i)
61     {
62         cin >> u >> v >> w;

```

```

63     e[i] = {u, v, w};
64 }
65 for (ll i = 1; i <= n; ++i)
66 {
67     e[+m] = {0, i, 0};
68 }
69 spfa(0);
70 return 0;
71 }

```

## Dijkstra

非负权图适用。从未确定最短路点集里选最短路最小节点移到已知点集里，把已知点集节点出边松弛。  
稀疏图二叉堆更优，稠密图暴力做法更优。

暴力：(洛谷P3371)  $O(n^2 + m)$

```

1 #include <bits/stdc++.h>
2 #define MAXN 100002
3 #define MAXM 200002
4 #define BIG (1 << 31) - 1
5 using namespace std;
6 int n, m, s, ui, vi, wi, p[MAXN], d[MAXN];
7 vector<pair<int, int>> g[MAXN];
8 bool vis[MAXN];
9 void dijkstra()
10 {
11     for (int i = 1; i <= n; i++)
12         d[i] = BIG;
13     d[s] = 0;
14     int minv, minu;
15     for (int h = 0; h + 1 < n; h++)
16     {
17         minv = BIG;
18         for (int i = 1; i <= n; i++)
19         {
20             if (!vis[i] && d[i] < minv)
21             {
22                 minv = d[i]; //minv不能省略，否则minu可能不为正无穷若初始化min在点范围内
23                 minu = i;
24             }
25         }
26         vis[minu] = true;
27         for (auto i : g[minu])
28         {
29             if (!vis[i.first] && d[minu] + i.second < d[i.first])
30             {
31                 d[i.first] = d[minu] + i.second;
32             }
33         } //如果只求d[t]可以vis[t]时break
34     }
35 }
36 signed main()

```

```

37 {
38     scanf("%d%d%d", &n, &m, &s);
39     while (m--)
40     {
41         scanf("%d%d%d", &ui, &vi, &wi);
42         g[ui].push_back({vi, wi});
43     }
44     dijkstra();
45     for (int i = 1; i <= n; i++)
46         printf("%d ", d[i]);
47     return 0;
48 }
49

```

优先级队列优化: (洛谷P4771)  $O(m \log m)$

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef int ll;
4 #define big 0x3fffffff
5 #define mn 100010
6 ll read() {ll r; scanf("%d", &r); return r; }
7 struct edge
8 {
9     ll to, nx, w;
10 } e[mn << 1];
11 ll hd[mn], cnt, n, m, u, v, w, d[mn], s, vis[mn];
12 void adde(ll &u, ll &v, ll &w)
13 {
14     e[++cnt] = {v, hd[u], w};
15     hd[u] = cnt;
16 }
17 struct node
18 {
19     ll i, d;
20     bool operator<(const node &x) const { return d > x.d; }
21 };
22 priority_queue<node> q;
23 //priority_queue<pair<LL, int>, vector<pair<LL, int>>, greater<pair<LL, int>> q;
24 signed main()
25 {
26     n = read(), m = read(), s = read();
27     for (ll i = 1; i <= m; ++i)
28         u = read(), v = read(), w = read(), adde(u, v, w);
29     memset(d, 127, sizeof d);
30     d[s] = 0;
31     q.push({s, 0});
32     while (!q.empty())
33     {
34         node p = q.top();
35         q.pop();
36         if (vis[p.i])
37             continue;
38         vis[p.i] = true;

```

```

39     for (ll i = hd[p.i]; i; i = e[i].nx)
40     {
41         ll toi = e[i].to;
42         if (d[toi] > d[p.i] + e[i].w)
43         {
44             d[toi] = d[p.i] + e[i].w;
45             q.push({toi, d[toi]});
46         }
47     }
48 }
49 for (ll i = 1; i <= n; ++i)
50     printf("%d ", d[i]);
51 return 0;
52 }
```

如果要输出路径本身，那么每次松弛时让  $v$  指向起点  $u$ ；然后从终点倒序遍历这个指向数组，然后再顺着输出。如果要计数最短路数目，或求最短时点权值最大的路径等，也在松弛时修改。（ $==$ 时不用 `push`）

可以修改松弛条件，来求别的最短路，如路径最大边权最小的路：（也可以最小生成树上DFS）  
(SCNUOJ1458 / lc1631)

$$dis[j] = \min(dis[j], \max(dis[mink], ma[mink][j]))$$

同理最大边权最大的最短路取  $dis = \infty$  ( $dis_0 = 0$ ) 且：(lc1631)

```
1 | if(max(dist[u], w) < dist[v]) dist[v] = max(dist[u], w)
```

非负权图跑  $n$  次可以实现全源最短路，复杂度  $O(nm \log m)$ 。不能求单源最长路。

lc2699-给定  $n$  ( $\leq 100$ ) 点稠密图，有一些边权不确定( $-1$ )，给出一种构造正边权方案，使得指定的两点单源最短路为特定值

解法一：无解的充要条件：①所有边赋 1 无解；②所有边赋无穷无解。

设在上述第一种情况跑出来的  $source$  单源最短路是  $d_{i,0}$ ，当有解时，根据 dijkstra 的特性，只要按照 dijkstra 遍历顺序去修改边，就不会在修改后对已确定的最短路产生影响。

所以在第一种情况下，再跑一遍 dijkstra，此时跑出的结果为  $d_{i,1}$ ，对可修改的边  $(u, v)$ ，设边权改为  $w$ ，则所求路径  $s \rightarrow \dots \rightarrow x \rightarrow y \rightarrow \dots \rightarrow t$  由三部分组成：

1.  $d_{x,1}$  即  $s \rightarrow \dots \rightarrow x$
2.  $x \rightarrow y = w$
3.  $y \rightarrow \dots \rightarrow t$ ，如果  $y$  确实在最短路上，对所有边赋 1 的最短路是  $d_{t,0}$ ，其在后半段的最短路长度可以直接路径相减，即  $d_{t,0} - d_{y,0}$ 。

如果  $y$  不在最短路上，那么这次修改无效，可以直接忽略修改。

要满足题意，即构造  $d_{x,1} + w + d_{t,0} - d_{y,0} = target$ ，即  $w = target - d_{t,0} + d_{y,0} - d_{x,1}$ 。

解法二：设可修改的边的权列表是  $w$ ，为书写方便令  $D = target$ ，则  $w$  的所有可能性不超过  $O(D^2)$  种，分别为：  $[1, 1, \dots, 1], [2, 1, \dots, 1], \dots, [D, 1, \dots, 1], [D, 2, \dots, 1], \dots, [D, D, \dots, D]$  种。

注意到  $[1, D, \dots, 1]$  这样的可能性不计，这是因为只用上述顺序在有解的情况下一定可以构造出，且每次增加总  $\sum w$  为 1 下，最多只增加 1 的答案最短路长度。也就是说不需要指数级的所有可能性都枚举出来，只需要枚举所有可能性的一个特定  $D^2$  子集就一定能覆盖到有解的情况。

复杂度为  $O(2 \log Dn^2)$ 。

## Johnson

复杂度  $O(nm \log m)$ ，适用于所有图，能判负环

洛谷P5905-有向，可能负边权、自环重边、卡SPFA

$n(1 \leq n \leq 3 \times 10^3), m(1 \leq m \leq 6 \times 10^3, |w| \leq 3 \times 10^5)$ ，若负环输出  $-1$ ，否则输出  $n$  行，设最短路为  $dis_{i,j}$ ，输出  $\sum_{j=1}^n j \times dis_{i,j}$ 。不存在路径设  $dis_{i,j} = 10^9$ ， $dis_{i,i} = 0$

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #define repe(i,a,b) for(ll i=a;i<=b;++i)
5 #define MAXN 5006
6 #define BIG 1000000000
7 struct edge { ll to, nx, w; } e[MAXN<<1];
8 struct node
9 {
10     ll d, i;
11     bool operator<(const node& x) const { return d > x.d; }
12     node(ll x, ll y) { d = x, i = y; }
13 };
14 ll hd[MAXN], t[MAXN], cnt, n, m, h[MAXN], dis[MAXN], ui, vi, wi, ans;
15 bool vis[MAXN];
16 inline void adde(ll& u, ll& v, ll& w)
17 {
18     e[++cnt] = { v,hd[u],w };
19     hd[u] = cnt;
20 }
21 inline bool spfa(ll x)
22 {
23     queue<ll> q;
24     memset(h, 63, sizeof h);
25     h[x] = 0, vis[x] = true; q.push(x);
26     while (!q.empty())
27     {
28         ll y = q.front(); q.pop();
29         vis[y] = false;
30         for (ll i = hd[y]; i; i = e[i].nx)
31         {
32             ll z = e[i].to;
33             if (h[z] > h[y] + e[i].w)
34             {
35                 h[z] = h[y] + e[i].w;
36                 if (!vis[z])
37                 {
38                     vis[z] = true, ++t[z];
39                     q.push(z);
40                     if (t[z] > n) return false;
41                 }
42             }
43     }
44 }
```

```

43         }
44     }
45     return true;
46 }
47 inline void dijkstra(11 x)
48 {
49     priority_queue<node> q;
50     repe(i, 1, n) dis[i] = BIG;
51     memset(vis, 0, sizeof vis);
52     dis[x] = 0;
53     q.push(node(0, x));
54     while (!q.empty())
55     {
56         11 u = q.top().i; q.pop();
57         if (vis[u]) continue;
58         vis[u] = true;
59         for (11 i = hd[u]; i; i = e[i].nx)
60         {
61             11 v = e[i].to;
62             if (dis[v] > dis[u] + e[i].w)
63             {
64                 dis[v] = dis[u] + e[i].w;
65                 if (!vis[v]) q.push(node(dis[v], v));
66             }
67         }
68     }
69 }
70 signed main()
71 {
72     scanf("%lld%lld", &n, &m);
73     repe(i, 1, m) scanf("%lld%lld%lld", &ui, &vi, &wi), adde(ui, vi, wi);
74     repe(i, 1, n) adde(ans, i, ans);
75     if (!spfa(0)) return !printf("-1");
76     repe(u, 1, n) for (11 i = hd[u]; i; i = e[i].nx)
77         e[i].w += h[u] - h[e[i].to];
78     repe(i, 1, n)
79     {
80         dijkstra(i);
81         ans = 0;
82         repe(j, 1, n)
83         {
84             if (dis[j] == BIG) ans += j * BIG;
85             else ans += j * (dis[j] + h[j] - h[i]);
86         }
87         printf("%lld\n", ans);
88     }
89     return 0;
90 }

```

## 差分约束

对  $n$  个变量  $x_i$  和  $m$  个不等式  $x_i - x_j \leq c_k, c_k \in R$ , 求一组解  $x$  使得每个不等式都满足, 或者判断无解。

将其变为  $x_i \leq x_j + c_k$ , 联想单源最短路三角不等式, 所以对每个约束条件构造  $j \rightarrow i$  的权为  $c_k$  的有向边, 且建立超级源点  $x_0$  向每个点连权为 0 的边。若负环无解, 否则最短路数组  $-d$  是一组解, 这组解是最小的。注意到若  $x$  是合法解, 则  $x + C$  也是合法解。用 Bellman-Ford / SPFA 即可, 都是  $O(nm)$ 。

构造技巧:

1. 如果有约束条件  $x_i = y_i$ , 等价于两个约束条件  $x_i \leq y_i + 0, y_i \leq x_i + 0$
2. 如果有约束条件  $\frac{x_i}{x_j} \leq c_k$ , 等价于  $\log x_i - \log x_j \leq \log c_k$
3. 反向差分约束, 即全部改为  $x_i \geq x_j + c_k$  连  $j \rightarrow i$  的  $c_k$  才能有超级源点的图, 求最大的跑该不等式的 SPFA 最长路(类似于工程图最晚), 一般用于求最小解
4. 要求  $\max(x_v - x_u)$ , 等价于求  $u$  到  $v$  的最短路; 要求  $\min(x_v - x_u)$  求  $u$  到  $v$  最长路(建的图不一样)

P1260-给定  $n$  个变量  $t$  和  $m$  个不等式  $t_i - t_j \leq b$ 。问能否构造出  $t$  或输出无解。输出构造非负解且至少一个 0。 $n \leq 10^3, m \leq 5 \times 10^3, b \in (-100, 100)$ 。

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 using ll = long long;
4 const ll mn = 5e3 + 10, inf = 1e9;
5 ll n, d[mn], m, vis[mn], cnt[mn];
6 vector<pair<ll, ll>> e[mn];
7 signed main()
8 {
9     cin.tie(0)->ios::sync_with_stdio(false);
10    cin >> n >> m;
11    for (ll i = 1, a, b, c; i <= m; ++i)
12    {
13        cin >> a >> b >> c;
14        e[b].push_back({a, c});
15    }
16    for (ll i = 1; i <= n; ++i)
17    {
18        e[0].push_back({i, 0});
19    }
20    fill_n(d, n + 3, inf);
21    d[0] = 0, vis[0] = true;
22    queue<ll> q;
23    q.push(0);
24    while (!q.empty())
25    {
26        ll u = q.front();
27        vis[u] = false;
28        q.pop();
29        for (auto pr : e[u])
30        {
31            ll v = pr.first, w = pr.second;
32            if (d[v] > d[u] + w)
```

```

33     {
34         d[v] = d[u] + w;
35         if (!vis[v])
36         {
37             if (++cnt[v] >= n + 1)
38             {
39                 printf("NO SOLUTION\n");
40                 return 0;
41             }
42             vis[v] = true;
43             q.push(v);
44         }
45     }
46 }
47 }
48 ll mi = *min_element(d + 1, d + 1 + n);
49 for (ll u = 1; u <= n; ++u)
50 {
51     cout << d[u] - mi << '\n';
52     for (auto pr : e[u])
53     {
54         ll v = pr.first, w = pr.second;
55         assert(d[v] <= d[u] + w);
56     }
57 }
58 return 0;
59 }

```

附：差分约束 P5960 模板题代码 (输入输出与本题类似，但任意解即可)

```

1 struct edge { ll u, v, w; } e[MAXN];
2 ll d[MAXN], n, m, c0, c1, y;
3 signed main()
4 {
5     scanf("%d%d", &n, &m);
6     repe(i, 1, m) scanf("%d%d%d", &c0, &c1, &y), e[i] = { c1, c0, y };
7     repe(i, 1, n) d[i] = (i == 1 ? 0 : 0x3f3f3f3f);
8     rep(i, 1, n) repe(j, 1, m)
9         d[e[j].v] = min(d[e[j].u] + e[j].w, d[e[j].v]);
10    repe(i, 1, m) if (d[e[i].v] > d[e[i].u] + e[i].w)
11        return !printf("NO");
12    repe(i, 1, n) printf("%d ", d[i]);
13    return 0;
14 }

```

## 拓扑排序

有向图存在环则无法进行拓扑排序。通常用的为 Kahn 算法，复杂度  $O(E + V)$

下面程序判断有向图是否有环(含考虑非连通、自环、重边)

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define mn 5010
4 typedef long long ll;
5 struct edge
6 {
7     ll to, nx;
8 } e[500010];
9 ll hd[mn], n, cnt, m, ru[mn];
10 void adde(ll u, ll v)
11 {
12     e[++cnt] = {v, hd[u]};
13     hd[u] = cnt, ++ru[v];
14 }
15 #define sc(x) scanf("%lld", &x)
16 queue<ll> q;
17 vector<ll> vis;
18 signed main()
19 {
20     sc(n), sc(m);
21     for (ll i = 1, u, v; i <= m; ++i)
22     {
23         sc(u), sc(v), adde(u, v);
24     }
25     for (ll i = 1; i <= n; ++i)
26     {
27         if (ru[i] == 0)
28         {
29             q.push(i);
30         }
31     }
32     if (q.empty())
33     {
34         return printf("YES"), 0;
35     }
36     while (!q.empty())
37     {
38         ll u = q.front();
39         q.pop();
40         vis.emplace_back(u);
41         for (ll i = hd[u], v; i; i = e[i].nx)
42         {
43             v = e[i].to;
44             --ru[v];
45             if (!ru[v])
46             {
47                 q.push(v);
48             }
49         }
50     }
51     return printf((ll)vis.size() == n ? "NO" : "YES"), 0;
52 }

```

如果要判无向图的环，把上面初始度(出+入)为 1 的入队，当度剩 1 时继续入队，最后剩 2 度的在一个环内(如下蓝桥-发现环例题：求基环树环上所有节点编号)(也可以DFS做，遍历记录深度和prev，发现浅于自己的就不断走prev)

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 #define sc(x) scanf("%lld", &x)
4 typedef long long ll;
5 #define mn 100010
6 struct edge
7 {
8     ll to, nx;
9     edge(ll a = 0, ll b = 0) : to(a), nx(b) {}
10 } e[mn * 2];
11 ll hd[mn], cnt, n, m, du[mn];
12 void adde(ll u, ll v)
13 {
14     e[++cnt] = edge(v, hd[u]);
15     ++du[v], hd[u] = cnt;
16 }
17 signed main()
18 {
19     sc(n);
20     for (ll i = 1, u, v; i <= n; ++i)
21         sc(u), sc(v), adde(u, v), adde(v, u);
22     queue<ll> q;
23     for (ll i = 1; i <= n; ++i)
24         if (du[i] == 1)
25             q.push(i);
26     while (!q.empty())
27     {
28         ll u = q.front();
29         q.pop();
30         for (ll i = hd[u], v; i; i = e[i].nx)
31         {
32             v = e[i].to;
33             if (--du[v] == 1)
34                 q.push(v);
35         }
36     }
37     for (ll i = 1; i <= n; ++i)
38         if (du[i] == 2)
39             printf("%lld ", i);
40     return 0;
41 }
```

将队列改成最大堆/最小堆可以  $O(E + V \log V)$  实现字典序最大/最小的拓扑排序

拓扑排序可以求(从入度0开始的点的)有向图单源最长路。对其他同样入度 0 的点，先把它们 BFS 一次删掉(预拓扑删掉它们带来的路径)

# 最小生成树

## kruskal

$O(m \log m)$  (在排序上)复杂度, 对稀疏图较好, 反向排序可以最大生成树, 可以判无解

(例题洛谷P3366)

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 #define sc(x) scanf("%lld", &x)
4 typedef long long ll;
5 #define mn 5010
6 #define me 200010
7 struct edge
8 {
9     ll u, v, w;
10    bool operator<(const edge &r) const { return w < r.w; }
11 } e[me];
12 ll fa[mn], n, m, suc, ans;
13 ll findf(ll x)
14 {
15     while (x != fa[x])
16         x = fa[x] = fa[fa[x]];
17     return x;
18 }
19 signed main()
20 {
21     sc(n), sc(m);
22     for (ll i = 1; i <= m; ++i)
23         sc(e[i].u), sc(e[i].v), sc(e[i].w);
24     sort(e + 1, e + 1 + m);
25     for (ll i = 1; i <= n; ++i)
26         fa[i] = i;
27     for (ll i = 1, fu, fv; i <= m && suc <= n - 1; ++i)
28     {
29         fu = findf(e[i].u), fv = findf(e[i].v);
30         if (fu != fv) //sol += u,v
31             fa[fu] = fv, ans += e[i].w, ++suc;
32     }
33     suc == n - 1 ? printf("%lld", ans) : printf("orz");
34     return 0;
35 }
```

部分应用示例:

- (SCNUOJ1124)求最长边与最短边差值最小的最小生成树

每次从第  $i$  短边开始跑 Kruskal, 必然贪心地得到当前第  $i$  短时最小的最长边。复杂度为  $O(m \log m + m^2) = O(m^2)$

- (洛谷P4047)给定  $n$  点聚为  $m$  类, 求两类间最短距离最大值

跑 Kruskal 直到还剩  $m$  个连通块, 此时生成树下一条边长就是答案

## prim

暴力  $O(n^2 + m)$  , 二叉堆优化  $O((n + m) \log n)$  , Fib 堆  $O(n \log n + m)$

暴力: (适合完全图) (洛谷P1265 求坐标轴上点的最小生成树)

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 using ll = long long;
4 using db = double;
5 const ll mn = 5e3 + 10;
6 ll n, x[mn], y[mn], suc, vis[mn];
7 db d[mn], ans;
8 ll sol[mn]; //如果要输出方案
9 signed main()
10 {
11     cin >> n;
12     for (ll i = 1; i <= n; ++i)
13     {
14         cin >> x[i] >> y[i];
15         d[i] = 1e21;
16     }
17     d[1] = 0;
18     for (ll t = 1; t <= n - 1; ++t)
19     {
20         ll a = 0;
21         for (ll j = 1; j <= n; ++j)
22         {
23             if (!vis[j] && (!a || d[j] < d[a]))
24             {
25                 a = j;
26             }
27         }
28         vis[a] = true;
29         for (ll b = 1; b <= n; ++b)
30         {
31             if (!vis[b])
32             {
33                 db dis = sqrt((x[a] - x[b]) * (x[a] - x[b]) + (y[a] - y[b]) *
34 * (y[a] - y[b]));
35                 if (dis < d[b])
36                 {
37                     d[b] = dis;
38                     // sol[b] = a;
39                 }
40             }
41         }
42     }
43     for (ll i = 1; i <= n; ++i)
44     {
45         ans += d[i];
46         // cout << i << ' ' << sol[i] << '\n'; [2,n] 有效
47     }
48     printf("%.2lf", ans);
49     return 0;
50 }
```

二叉堆：

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define sc(x) scanf("%lld", &x)
4 typedef long long ll;
5 #define mn 5010
6 #define me 200010
7 struct
8 {
9     ll to, nx, w;
10 } e[me * 2];
11 ll hd[mn], cnt, d[mn], ans, suc, n, m, vis[mn];
12 void adde(ll u, ll v, ll w)
13 {
14     e[++cnt] = {v, hd[u], w};
15     hd[u] = cnt;
16 }
17 typedef pair<ll, ll> pr; // first:w, second:i
18 priority_queue<pr, vector<pr>, greater<pr>> q;
19 signed main()
20 {
21     sc(n), sc(m);
22     for (ll i = 1, u, v, w; i <= m; ++i)
23         sc(u), sc(v), sc(w), adde(u, v, w), adde(v, u, w);
24     memset(d, 0x3f, sizeof d);
25     d[1] = 0;
26     q.push({0, 1});
27     while (!q.empty() && suc <= n - 1)
28     {
29         ll w = q.top().first, u = q.top().second;
30         q.pop();
31         if (vis[u])
32             continue;
33         ans += w, vis[u] = true, ++suc;
34         for (ll i = hd[u], v; i; i = e[i].nx)
35         {
36             v = e[i].to;
37             if (e[i].w < d[v]) //sol[v]=u;
38                 d[v] = e[i].w, q.push({d[v], v});
39         }
40     }
41     suc == n ? printf("%lld", ans) : printf("orz");
42     return 0;
43 }
```

## Borůvka

可以用于求解边权互不相同的无向图的最小生成森林。权值相同会导致两个连通块互相连的时候出现环，那么此时就需要再按照另一个维度严格排序，常用标号大小排序。即边权相同时，认为编号小的边短。复杂度  $O(m \log n)$ ，每次迭代连通块数量至少减半。

```
1 #include <iostream>
2 #include <cstdio>
3 #include <cstring>
4 using namespace std;
5
6 const int MaxN = 5000 + 5, MaxM = 200000 + 5;
7
8 int N, M;
9 int U[MaxM], V[MaxM], W[MaxM];
10 bool used[MaxM];
11 int par[MaxN], Best[MaxN];
12
13 void init() {
14     scanf("%d %d", &N, &M);
15     for (int i = 1; i <= M; ++i)
16         scanf("%d %d %d", &U[i], &V[i], &W[i]);
17 }
18
19 void init_dsu() {
20     for (int i = 1; i <= N; ++i)
21         par[i] = i;
22 }
23
24 int get_par(int x) {
25     if (x == par[x]) return x;
26     else return par[x] = get_par(par[x]);
27 }
28
29 inline bool Better(int x, int y) {
30     if (y == 0) return true;
31     if (W[x] != W[y]) return W[x] < W[y];
32     return x < y;
33 }
34
35 void Boruvka() {
36     init_dsu();
37
38     int merged = 0, sum = 0;
39
40     bool update = true;
41     while (update) {
42         update = false;
43         memset(Best, 0, sizeof Best);
44
45         for (int i = 1; i <= M; ++i) {
46             if (used[i] == true) continue;
47             int p = get_par(U[i]), q = get_par(V[i]);
48             if (p == q) continue;
49
50             if (W[p] > W[q]) swap(p, q);
51
52             if (W[p] < Best[i]) {
53                 Best[i] = W[p];
54                 update = true;
55             }
56         }
57
58         for (int i = 1; i <= M; ++i) {
59             if (used[i] == true) continue;
60             if (Best[i] == 0) continue;
61
62             int p = get_par(U[i]), q = get_par(V[i]);
63             if (p == q) continue;
64
65             if (W[p] < Best[i]) {
66                 Best[i] = W[p];
67                 update = true;
68             }
69         }
70
71         for (int i = 1; i <= M; ++i) {
72             if (used[i] == true) continue;
73             if (Best[i] == 0) continue;
74
75             int p = get_par(U[i]), q = get_par(V[i]);
76             if (p == q) continue;
77
78             if (W[p] < Best[i]) {
79                 Best[i] = W[p];
80                 update = true;
81             }
82         }
83
84         for (int i = 1; i <= M; ++i) {
85             if (used[i] == true) continue;
86             if (Best[i] == 0) continue;
87
88             int p = get_par(U[i]), q = get_par(V[i]);
89             if (p == q) continue;
90
91             if (W[p] < Best[i]) {
92                 Best[i] = W[p];
93                 update = true;
94             }
95         }
96
97         for (int i = 1; i <= M; ++i) {
98             if (used[i] == true) continue;
99             if (Best[i] == 0) continue;
100
101            int p = get_par(U[i]), q = get_par(V[i]);
102            if (p == q) continue;
103
104            if (W[p] < Best[i]) {
105                Best[i] = W[p];
106                update = true;
107            }
108        }
109
110        for (int i = 1; i <= M; ++i) {
111            if (used[i] == true) continue;
112            if (Best[i] == 0) continue;
113
114            int p = get_par(U[i]), q = get_par(V[i]);
115            if (p == q) continue;
116
117            if (W[p] < Best[i]) {
118                Best[i] = W[p];
119                update = true;
120            }
121        }
122
123        for (int i = 1; i <= M; ++i) {
124            if (used[i] == true) continue;
125            if (Best[i] == 0) continue;
126
127            int p = get_par(U[i]), q = get_par(V[i]);
128            if (p == q) continue;
129
130            if (W[p] < Best[i]) {
131                Best[i] = W[p];
132                update = true;
133            }
134        }
135
136        for (int i = 1; i <= M; ++i) {
137            if (used[i] == true) continue;
138            if (Best[i] == 0) continue;
139
140            int p = get_par(U[i]), q = get_par(V[i]);
141            if (p == q) continue;
142
143            if (W[p] < Best[i]) {
144                Best[i] = W[p];
145                update = true;
146            }
147        }
148
149        for (int i = 1; i <= M; ++i) {
150            if (used[i] == true) continue;
151            if (Best[i] == 0) continue;
152
153            int p = get_par(U[i]), q = get_par(V[i]);
154            if (p == q) continue;
155
156            if (W[p] < Best[i]) {
157                Best[i] = W[p];
158                update = true;
159            }
160        }
161
162        for (int i = 1; i <= M; ++i) {
163            if (used[i] == true) continue;
164            if (Best[i] == 0) continue;
165
166            int p = get_par(U[i]), q = get_par(V[i]);
167            if (p == q) continue;
168
169            if (W[p] < Best[i]) {
170                Best[i] = W[p];
171                update = true;
172            }
173        }
174
175        for (int i = 1; i <= M; ++i) {
176            if (used[i] == true) continue;
177            if (Best[i] == 0) continue;
178
179            int p = get_par(U[i]), q = get_par(V[i]);
180            if (p == q) continue;
181
182            if (W[p] < Best[i]) {
183                Best[i] = W[p];
184                update = true;
185            }
186        }
187
188        for (int i = 1; i <= M; ++i) {
189            if (used[i] == true) continue;
190            if (Best[i] == 0) continue;
191
192            int p = get_par(U[i]), q = get_par(V[i]);
193            if (p == q) continue;
194
195            if (W[p] < Best[i]) {
196                Best[i] = W[p];
197                update = true;
198            }
199        }
200
201        for (int i = 1; i <= M; ++i) {
202            if (used[i] == true) continue;
203            if (Best[i] == 0) continue;
204
205            int p = get_par(U[i]), q = get_par(V[i]);
206            if (p == q) continue;
207
208            if (W[p] < Best[i]) {
209                Best[i] = W[p];
210                update = true;
211            }
212        }
213
214        for (int i = 1; i <= M; ++i) {
215            if (used[i] == true) continue;
216            if (Best[i] == 0) continue;
217
218            int p = get_par(U[i]), q = get_par(V[i]);
219            if (p == q) continue;
220
221            if (W[p] < Best[i]) {
222                Best[i] = W[p];
223                update = true;
224            }
225        }
226
227        for (int i = 1; i <= M; ++i) {
228            if (used[i] == true) continue;
229            if (Best[i] == 0) continue;
230
231            int p = get_par(U[i]), q = get_par(V[i]);
232            if (p == q) continue;
233
234            if (W[p] < Best[i]) {
235                Best[i] = W[p];
236                update = true;
237            }
238        }
239
240        for (int i = 1; i <= M; ++i) {
241            if (used[i] == true) continue;
242            if (Best[i] == 0) continue;
243
244            int p = get_par(U[i]), q = get_par(V[i]);
245            if (p == q) continue;
246
247            if (W[p] < Best[i]) {
248                Best[i] = W[p];
249                update = true;
250            }
251        }
252
253        for (int i = 1; i <= M; ++i) {
254            if (used[i] == true) continue;
255            if (Best[i] == 0) continue;
256
257            int p = get_par(U[i]), q = get_par(V[i]);
258            if (p == q) continue;
259
260            if (W[p] < Best[i]) {
261                Best[i] = W[p];
262                update = true;
263            }
264        }
265
266        for (int i = 1; i <= M; ++i) {
267            if (used[i] == true) continue;
268            if (Best[i] == 0) continue;
269
270            int p = get_par(U[i]), q = get_par(V[i]);
271            if (p == q) continue;
272
273            if (W[p] < Best[i]) {
274                Best[i] = W[p];
275                update = true;
276            }
277        }
278
279        for (int i = 1; i <= M; ++i) {
280            if (used[i] == true) continue;
281            if (Best[i] == 0) continue;
282
283            int p = get_par(U[i]), q = get_par(V[i]);
284            if (p == q) continue;
285
286            if (W[p] < Best[i]) {
287                Best[i] = W[p];
288                update = true;
289            }
290        }
291
292        for (int i = 1; i <= M; ++i) {
293            if (used[i] == true) continue;
294            if (Best[i] == 0) continue;
295
296            int p = get_par(U[i]), q = get_par(V[i]);
297            if (p == q) continue;
298
299            if (W[p] < Best[i]) {
300                Best[i] = W[p];
301                update = true;
302            }
303        }
304
305        for (int i = 1; i <= M; ++i) {
306            if (used[i] == true) continue;
307            if (Best[i] == 0) continue;
308
309            int p = get_par(U[i]), q = get_par(V[i]);
310            if (p == q) continue;
311
312            if (W[p] < Best[i]) {
313                Best[i] = W[p];
314                update = true;
315            }
316        }
317
318        for (int i = 1; i <= M; ++i) {
319            if (used[i] == true) continue;
320            if (Best[i] == 0) continue;
321
322            int p = get_par(U[i]), q = get_par(V[i]);
323            if (p == q) continue;
324
325            if (W[p] < Best[i]) {
326                Best[i] = W[p];
327                update = true;
328            }
329        }
330
331        for (int i = 1; i <= M; ++i) {
332            if (used[i] == true) continue;
333            if (Best[i] == 0) continue;
334
335            int p = get_par(U[i]), q = get_par(V[i]);
336            if (p == q) continue;
337
338            if (W[p] < Best[i]) {
339                Best[i] = W[p];
340                update = true;
341            }
342        }
343
344        for (int i = 1; i <= M; ++i) {
345            if (used[i] == true) continue;
346            if (Best[i] == 0) continue;
347
348            int p = get_par(U[i]), q = get_par(V[i]);
349            if (p == q) continue;
350
351            if (W[p] < Best[i]) {
352                Best[i] = W[p];
353                update = true;
354            }
355        }
356
357        for (int i = 1; i <= M; ++i) {
358            if (used[i] == true) continue;
359            if (Best[i] == 0) continue;
360
361            int p = get_par(U[i]), q = get_par(V[i]);
362            if (p == q) continue;
363
364            if (W[p] < Best[i]) {
365                Best[i] = W[p];
366                update = true;
367            }
368        }
369
370        for (int i = 1; i <= M; ++i) {
371            if (used[i] == true) continue;
372            if (Best[i] == 0) continue;
373
374            int p = get_par(U[i]), q = get_par(V[i]);
375            if (p == q) continue;
376
377            if (W[p] < Best[i]) {
378                Best[i] = W[p];
379                update = true;
380            }
381        }
382
383        for (int i = 1; i <= M; ++i) {
384            if (used[i] == true) continue;
385            if (Best[i] == 0) continue;
386
387            int p = get_par(U[i]), q = get_par(V[i]);
388            if (p == q) continue;
389
390            if (W[p] < Best[i]) {
391                Best[i] = W[p];
392                update = true;
393            }
394        }
395
396        for (int i = 1; i <= M; ++i) {
397            if (used[i] == true) continue;
398            if (Best[i] == 0) continue;
399
400            int p = get_par(U[i]), q = get_par(V[i]);
401            if (p == q) continue;
402
403            if (W[p] < Best[i]) {
404                Best[i] = W[p];
405                update = true;
406            }
407        }
408
409        for (int i = 1; i <= M; ++i) {
410            if (used[i] == true) continue;
411            if (Best[i] == 0) continue;
412
413            int p = get_par(U[i]), q = get_par(V[i]);
414            if (p == q) continue;
415
416            if (W[p] < Best[i]) {
417                Best[i] = W[p];
418                update = true;
419            }
420        }
421
422        for (int i = 1; i <= M; ++i) {
423            if (used[i] == true) continue;
424            if (Best[i] == 0) continue;
425
426            int p = get_par(U[i]), q = get_par(V[i]);
427            if (p == q) continue;
428
429            if (W[p] < Best[i]) {
430                Best[i] = W[p];
431                update = true;
432            }
433        }
434
435        for (int i = 1; i <= M; ++i) {
436            if (used[i] == true) continue;
437            if (Best[i] == 0) continue;
438
439            int p = get_par(U[i]), q = get_par(V[i]);
440            if (p == q) continue;
441
442            if (W[p] < Best[i]) {
443                Best[i] = W[p];
444                update = true;
445            }
446        }
447
448        for (int i = 1; i <= M; ++i) {
449            if (used[i] == true) continue;
450            if (Best[i] == 0) continue;
451
452            int p = get_par(U[i]), q = get_par(V[i]);
453            if (p == q) continue;
454
455            if (W[p] < Best[i]) {
456                Best[i] = W[p];
457                update = true;
458            }
459        }
460
461        for (int i = 1; i <= M; ++i) {
462            if (used[i] == true) continue;
463            if (Best[i] == 0) continue;
464
465            int p = get_par(U[i]), q = get_par(V[i]);
466            if (p == q) continue;
467
468            if (W[p] < Best[i]) {
469                Best[i] = W[p];
470                update = true;
471            }
472        }
473
474        for (int i = 1; i <= M; ++i) {
475            if (used[i] == true) continue;
476            if (Best[i] == 0) continue;
477
478            int p = get_par(U[i]), q = get_par(V[i]);
479            if (p == q) continue;
480
481            if (W[p] < Best[i]) {
482                Best[i] = W[p];
483                update = true;
484            }
485        }
486
487        for (int i = 1; i <= M; ++i) {
488            if (used[i] == true) continue;
489            if (Best[i] == 0) continue;
490
491            int p = get_par(U[i]), q = get_par(V[i]);
492            if (p == q) continue;
493
494            if (W[p] < Best[i]) {
495                Best[i] = W[p];
496                update = true;
497            }
498        }
499
500        for (int i = 1; i <= M; ++i) {
501            if (used[i] == true) continue;
502            if (Best[i] == 0) continue;
503
504            int p = get_par(U[i]), q = get_par(V[i]);
505            if (p == q) continue;
506
507            if (W[p] < Best[i]) {
508                Best[i] = W[p];
509                update = true;
510            }
511        }
512
513        for (int i = 1; i <= M; ++i) {
514            if (used[i] == true) continue;
515            if (Best[i] == 0) continue;
516
517            int p = get_par(U[i]), q = get_par(V[i]);
518            if (p == q) continue;
519
520            if (W[p] < Best[i]) {
521                Best[i] = W[p];
522                update = true;
523            }
524        }
525
526        for (int i = 1; i <= M; ++i) {
527            if (used[i] == true) continue;
528            if (Best[i] == 0) continue;
529
530            int p = get_par(U[i]), q = get_par(V[i]);
531            if (p == q) continue;
532
533            if (W[p] < Best[i]) {
534                Best[i] = W[p];
535                update = true;
536            }
537        }
538
539        for (int i = 1; i <= M; ++i) {
540            if (used[i] == true) continue;
541            if (Best[i] == 0) continue;
542
543            int p = get_par(U[i]), q = get_par(V[i]);
544            if (p == q) continue;
545
546            if (W[p] < Best[i]) {
547                Best[i] = W[p];
548                update = true;
549            }
550        }
551
552        for (int i = 1; i <= M; ++i) {
553            if (used[i] == true) continue;
554            if (Best[i] == 0) continue;
555
556            int p = get_par(U[i]), q = get_par(V[i]);
557            if (p == q) continue;
558
559            if (W[p] < Best[i]) {
560                Best[i] = W[p];
561                update = true;
562            }
563        }
564
565        for (int i = 1; i <= M; ++i) {
566            if (used[i] == true) continue;
567            if (Best[i] == 0) continue;
568
569            int p = get_par(U[i]), q = get_par(V[i]);
570            if (p == q) continue;
571
572            if (W[p] < Best[i]) {
573                Best[i] = W[p];
574                update = true;
575            }
576        }
577
578        for (int i = 1; i <= M; ++i) {
579            if (used[i] == true) continue;
580            if (Best[i] == 0) continue;
581
582            int p = get_par(U[i]), q = get_par(V[i]);
583            if (p == q) continue;
584
585            if (W[p] < Best[i]) {
586                Best[i] = W[p];
587                update = true;
588            }
589        }
590
591        for (int i = 1; i <= M; ++i) {
592            if (used[i] == true) continue;
593            if (Best[i] == 0) continue;
594
595            int p = get_par(U[i]), q = get_par(V[i]);
596            if (p == q) continue;
597
598            if (W[p] < Best[i]) {
599                Best[i] = W[p];
600                update = true;
601            }
602        }
603
604        for (int i = 1; i <= M; ++i) {
605            if (used[i] == true) continue;
606            if (Best[i] == 0) continue;
607
608            int p = get_par(U[i]), q = get_par(V[i]);
609            if (p == q) continue;
610
611            if (W[p] < Best[i]) {
612                Best[i] = W[p];
613                update = true;
614            }
615        }
616
617        for (int i = 1; i <= M; ++i) {
618            if (used[i] == true) continue;
619            if (Best[i] == 0) continue;
620
621            int p = get_par(U[i]), q = get_par(V[i]);
622            if (p == q) continue;
623
624            if (W[p] < Best[i]) {
625                Best[i] = W[p];
626                update = true;
627            }
628        }
629
630        for (int i = 1; i <= M; ++i) {
631            if (used[i] == true) continue;
632            if (Best[i] == 0) continue;
633
634            int p = get_par(U[i]), q = get_par(V[i]);
635            if (p == q) continue;
636
637            if (W[p] < Best[i]) {
638                Best[i] = W[p];
639                update = true;
640            }
641        }
642
643        for (int i = 1; i <= M; ++i) {
644            if (used[i] == true) continue;
645            if (Best[i] == 0) continue;
646
647            int p = get_par(U[i]), q = get_par(V[i]);
648            if (p == q) continue;
649
650            if (W[p] < Best[i]) {
651                Best[i] = W[p];
652                update = true;
653            }
654        }
655
656        for (int i = 1; i <= M; ++i) {
657            if (used[i] == true) continue;
658            if (Best[i] == 0) continue;
659
660            int p = get_par(U[i]), q = get_par(V[i]);
661            if (p == q) continue;
662
663            if (W[p] < Best[i]) {
664                Best[i] = W[p];
665                update = true;
666            }
667        }
668
669        for (int i = 1; i <= M; ++i) {
670            if (used[i] == true) continue;
671            if (Best[i] == 0) continue;
672
673            int p = get_par(U[i]), q = get_par(V[i]);
674            if (p == q) continue;
675
676            if (W[p] < Best[i]) {
677                Best[i] = W[p];
678                update = true;
679            }
680        }
681
682        for (int i = 1; i <= M; ++i) {
683            if (used[i] == true) continue;
684            if (Best[i] == 0) continue;
685
686            int p = get_par(U[i]), q = get_par(V[i]);
687            if (p == q) continue;
688
689            if (W[p] < Best[i]) {
690                Best[i] = W[p];
691                update = true;
692            }
693        }
694
695        for (int i = 1; i <= M; ++i) {
696            if (used[i] == true) continue;
697            if (Best[i] == 0) continue;
698
699            int p = get_par(U[i]), q = get_par(V[i]);
700            if (p == q) continue;
701
702            if (W[p] < Best[i]) {
703                Best[i] = W[p];
704                update = true;
705            }
706        }
707
708        for (int i = 1; i <= M; ++i) {
709            if (used[i] == true) continue;
710            if (Best[i] == 0) continue;
711
712            int p = get_par(U[i]), q = get_par(V[i]);
713            if (p == q) continue;
714
715            if (W[p] < Best[i]) {
716                Best[i] = W[p];
717                update = true;
718            }
719        }
720
721        for (int i = 1; i <= M; ++i) {
722            if (used[i] == true) continue;
723            if (Best[i] == 0) continue;
724
725            int p = get_par(U[i]), q = get_par(V[i]);
726            if (p == q) continue;
727
728            if (W[p] < Best[i]) {
729                Best[i] = W[p];
730                update = true;
731            }
732        }
733
734        for (int i = 1; i <= M; ++i) {
735            if (used[i] == true) continue;
736            if (Best[i] == 0) continue;
737
738            int p = get_par(U[i]), q = get_par(V[i]);
739            if (p == q) continue;
740
741            if (W[p] < Best[i]) {
742                Best[i] = W[p];
743                update = true;
744            }
745        }
746
747        for (int i = 1; i <= M; ++i) {
748            if (used[i] == true) continue;
749            if (Best[i] == 0) continue;
750
751            int p = get_par(U[i]), q = get_par(V[i]);
752            if (p == q) continue;
753
754            if (W[p] < Best[i]) {
755                Best[i] = W[p];
756                update = true;
757            }
758        }
759
760        for (int i = 1; i <= M; ++i) {
761            if (used[i] == true) continue;
762            if (Best[i] == 0) continue;
763
764            int p = get_par(U[i]), q = get_par(V[i]);
765            if (p == q) continue;
766
767            if (W[p] < Best[i]) {
768                Best[i] = W[p];
769                update = true;
770            }
771        }
772
773        for (int i = 1; i <= M; ++i) {
774            if (used[i] == true) continue;
775            if (Best[i] == 0) continue;
776
777            int p = get_par(U[i]), q = get_par(V[i]);
778            if (p == q) continue;
779
780            if (W[p] < Best[i]) {
781                Best[i] = W[p];
782                update = true;
783            }
784        }
785
786        for (int i = 1; i <= M; ++i) {
787            if (used[i] == true) continue;
788            if (Best[i] == 0) continue;
789
790            int p = get_par(U[i]), q = get_par(V[i]);
791            if (p == q) continue;
792
793            if (W[p] < Best[i]) {
794                Best[i] = W[p];
795                update = true;
796            }
797        }
798
799        for (int i = 1; i <= M; ++i) {
800            if (used[i] == true) continue;
801            if (Best[i] == 0) continue;
802
803            int p = get_par(U[i]), q = get_par(V[i]);
804            if (p == q) continue;
805
806            if (W[p] < Best[i]) {
807                Best[i] = W[p];
808                update = true;
809            }
810        }
811
812        for (int i = 1; i <= M; ++i) {
813            if (used[i] == true) continue;
814            if (Best[i] == 0) continue;
815
816            int p = get_par(U[i]), q = get_par(V[i]);
817            if (p == q) continue;
818
819            if (W[p] < Best[i]) {
820                Best[i] = W[p];
821                update = true;
822            }
823        }
824
825        for (int i = 1; i <= M; ++i) {
826            if (used[i] == true) continue;
827            if (Best[i] == 0) continue;
828
829            int p = get_par(U[i]), q = get_par(V[i]);
830            if (p == q) continue;
831
832            if (W[p] < Best[i]) {
833                Best[i] = W[p];
834                update = true;
835            }
836        }
837
838        for (int i = 1; i <= M; ++i) {
839            if (used[i] == true) continue;
840            if (Best[i] == 0) continue;
841
842            int p = get_par(U[i]), q = get_par(V[i]);
843            if (p == q) continue;
844
845            if (W[p] < Best[i]) {
846                Best[i] = W[p];
847                update = true;
848            }
849        }
850
851        for (int i = 1; i <= M; ++i) {
852            if (used[i] == true) continue;
853            if (Best[i] == 0) continue;
854
855            int p = get_par(U[i]), q = get_par(V[i]);
856            if (p == q) continue;
857
858            if (W[p] < Best[i]) {
859                Best[i] = W[p];
860                update = true;
861            }
862        }
863
864        for (int i = 1; i <= M; ++i) {
865            if (used[i] == true) continue;
866            if (Best[i] == 0) continue;
867
868            int p = get_par(U[i]), q = get_par(V[i]);
869            if (p == q) continue;
870
871            if (W[p] < Best[i]) {
872                Best[i] = W[p];
873                update = true;
874            }
875        }
876
877        for (int i = 1; i <= M; ++i) {
878            if (used[i] == true) continue;
879            if (Best[i] == 0) continue;
880
881            int p = get_par(U[i]), q = get_par(V[i]);
882            if (p == q) continue;
883
884            if (W[p] < Best[i]) {
885                Best[i] = W[p];
886                update = true;
887            }
888        }
889
890        for (int i = 1; i <= M; ++i) {
891            if (used[i] == true) continue;
892            if (Best[i] == 0) continue;
893
894            int p = get_par(U[i]), q = get_par(V[i]);
895            if (p == q) continue;
896
897            if (W[p] < Best[i]) {
898                Best[i] = W[p];
899                update = true;
900            }
901        }
902
903        for (int i = 1; i <= M; ++i) {
904            if (used[i] == true) continue;
905            if (Best[i] == 0) continue;
906
907            int p = get_par(U[i]), q = get_par(V[i]);
908            if (p == q) continue;
909
910            if (W[p] < Best[i]) {
911                Best[i] = W[p];
912                update = true;
913            }
914        }
915
916        for (int i = 1; i <= M; ++i) {
917            if (used[i] == true) continue;
918            if (Best[i] == 0) continue;
919
920            int p = get_par(U[i]), q = get_par(V[i]);
921            if (p == q) continue;
922
923            if (W[p] < Best[i]) {
924                Best[i] = W[p];
925                update = true;
926            }
927        }
928
929        for (int i = 1; i <= M; ++i) {
930            if (used[i] == true) continue;
931            if (Best[i] == 0) continue;
932
933            int p = get_par(U[i]), q = get_par(V[i]);
934            if (p == q) continue;
935
936            if (W[p] < Best[i]) {
937                Best[i] = W[p];
938                update = true;
939            }
940        }
941
942        for (int i = 1; i <= M; ++i) {
943            if (used[i] == true) continue;
944            if (Best[i] == 0) continue;
945
946            int p = get_par(U[i]), q = get_par(V[i]);
947            if (p == q) continue;
948
949            if (W[p] < Best[i]) {
950                Best[i] = W[p];
951                update = true;
952            }
953        }
954
955        for (int i = 1; i <= M; ++i) {
956            if (used[i] == true) continue;
957            if (Best[i] == 0) continue;
958
959            int p = get_par(U[i]), q = get_par(V[i]);
960            if (p == q) continue;
961
962            if (W[p] < Best[i]) {
963                Best[i] = W[p];
964                update = true;
965            }
966        }
967
968        for (int i = 1; i <= M; ++i) {
969            if (used[i] == true) continue;
970            if (Best[i] == 0) continue;
971
972            int p = get_par(U[i]), q = get_par(V[i]);
973            if (p == q) continue;
974
975            if (W[p] < Best[i]) {
976                Best[i] = W[p];
977                update = true;
978            }
979        }
980
981        for (int i = 1; i <= M; ++i) {
982            if (used[i] == true) continue;
983            if (Best[i] == 0) continue;
984
985            int p = get_par(U[i]), q = get_par(V[i]);
986            if (p == q) continue;
987
988            if (W[p] < Best[i]) {
989                Best[i] = W[p];
990                update = true;
991            }
992        }
993
994        for (int i = 1; i <= M; ++i) {
995            if (used[i] == true) continue;
996            if (Best[i] == 0) continue;
997
998            int p = get_par(U[i]), q = get_par(V[i]);
999            if (p == q) continue;
1000
1001            if (W[p] < Best[i]) {
1002                Best[i] = W[p];
1003                update = true;
1004            }
1005        }
1006
1007        for (int i = 1; i <= M; ++i) {
1008            if (used[i] == true) continue;
1009            if (Best[i] == 0) continue;
1010
1011            int p = get_par(U[i]), q = get_par(V[i]);
1012            if (p == q) continue;
1013
1014            if (W[p] < Best[i]) {
1015                Best[i] = W[p];
1016                update = true;
1017            }
1018        }
1019
1020        for (int i = 1; i <= M; ++i) {
1021            if (used[i] == true) continue;
1022            if (Best[i] == 0) continue;
1023
1024            int p = get_par(U[i]), q = get_par(V[i]);
1025            if (p == q) continue;
1026
1027            if (W[p] < Best[i]) {

```

```

50         if (Better(i, Best[p]) == true) Best[p] = i;
51         if (Better(i, Best[q]) == true) Best[q] = i;
52     }
53
54     for (int i = 1; i <= N; ++i)
55         if (Best[i] != 0 && used[Best[i]] == false) {
56             update = true;
57             merged++; sum += w[Best[i]];
58             used[Best[i]] = true;
59             par[get_par(U[Best[i]])] = get_par(V[Best[i]]);
60         }
61     }
62
63     if (merged == N - 1) printf("%d\n", sum);
64     else puts("orz");
65 }
66
67 int main() {
68     init();
69     Boruvka();
70     return 0;
71 }

```

## 严格次小生成树

洛谷P4180-可能含自环,  $n \leq 10^5, m \leq 3 \times 10^5$ , 保证有解, 边权  $[0, 10^9]$ , 求严格次小生成树权值和

复杂度为  $O(m \log m)$ , 解法是 Kruskal + LCA + 倍增

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #define sc(x) scanf("%lld", &x)
5 #define mn 400010
6 #define ml 19
7 #define big 0x7fffffffffffff
8 struct edge
9 {
10     ll u, v, w, nx;
11     bool operator<(const edge &x) const { return w < x.w; }
12 } e[mn << 1], a[mn << 1];
13 ll cnt, hd[mn], fa[mn][ml], mx[mn][ml], nd[mn][ml], dep[mn], n, m, ffa[mn];
14 bool vis[mn << 1];
15 void adde(ll &u, ll &v, ll &w)
16 {
17     e[++cnt] = {u, v, w, hd[u]};
18     hd[u] = cnt;
19 }
20 void dfs(ll u, ll f)
21 {
22     fa[u][0] = f;
23     for (ll i = hd[u]; i; i = e[i].nx)
24     {

```

```

25     ll v = e[i].v;
26     if (v == f)
27     {
28         continue;
29     }
30     dep[v] = dep[u] + 1;
31     mx[v][0] = e[i].w;
32     nd[v][0] = -big;
33     dfs(v, u);
34 }
35 }
36 void cal()
37 {
38     for (ll i = 1; i < m1; ++i)
39     {
40         for (ll j = 1; j <= n; ++j)
41         {
42             fa[j][i] = fa[fa[j][i - 1]][i - 1];
43             mx[j][i] = max(mx[j][i - 1], mx[fa[j][i - 1]][i - 1]);
44             nd[j][i] = max(nd[j][i - 1], nd[fa[j][i - 1]][i - 1]);
45             if (mx[j][i - 1] != mx[fa[j][i - 1]][i - 1])
46             {
47                 nd[j][i] = max(nd[j][i], min(mx[j][i - 1], mx[fa[j][i - 1]][i - 1]));
48             }
49         }
50     }
51 }
52 ll lca(ll x, ll y)
53 {
54     if (dep[x] < dep[y])
55     {
56         swap(x, y);
57     }
58     for (ll i = m1 - 1; i >= 0; --i)
59     {
60         if (dep[fa[x][i]] >= dep[y])
61         {
62             x = fa[x][i];
63         }
64     }
65     if (x == y)
66     {
67         return x;
68     }
69     for (ll i = m1 - 1; i >= 0; --i)
70     {
71         if (fa[x][i] ^ fa[y][i])
72         {
73             x = fa[x][i], y = fa[y][i];
74         }
75     }
76     return fa[x][0];
77 }
78 ll qmax(ll u, ll v, ll mxv)
79 {

```

```

80     ll ans = -big;
81     for (ll i = m1 - 1; i >= 0; --i)
82     {
83         if (dep[fa[u][i]] >= dep[v])
84         {
85             if (mxv != mx[u][i])
86             {
87                 ans = max(ans, mx[u][i]);
88             }
89             else
90             {
91                 ans = max(ans, nd[u][i]);
92             }
93             u = fa[u][i];
94         }
95     }
96     return ans;
97 }
98 ll finds(ll x)
99 {
100     while (x != ffa[x])
101     {
102         x = ffa[x] = ffa[ffa[x]];
103     }
104     return x;
105 }
106 signed main()
107 {
108     sc(n), sc(m);
109     for (ll i = 1; i <= m; ++i)
110     {
111         sc(a[i].u), sc(a[i].v), sc(a[i].w);
112     }
113     sort(a + 1, a + 1 + m);
114     for (ll i = 1; i <= n; ++i)
115     {
116         ffa[i] = i;
117     }
118     ll suc = 0, res = big;
119     for (ll i = 1; i <= m; ++i)
120     {
121         ll fu = finds(a[i].u), fv = finds(a[i].v);
122         if (fu != fv)
123         {
124             suc += a[i].w;
125             ffa[fu] = fv, vis[i] = true;
126             adde(a[i].u, a[i].v, a[i].w), adde(a[i].v, a[i].u, a[i].w);
127         }
128     }
129     nd[1][0] = -big, dep[1] = 1;
130     dfs(1, -1), cal();
131     for (ll i = 1, u, v, w, ca, mxu, mxv; i <= m; ++i)
132     {
133         if (!vis[i])
134         {
135             u = a[i].u, v = a[i].v, w = a[i].w;

```

```

136         ca = lca(u, v);
137         mxu = qmax(u, ca, w), mxv = qmax(v, ca, w);
138         res = min(res, suc - max(mxu, mxv) + w);
139     }
140 }
141 printf("%lld", res);
142 return 0;
143 }

```

注：下述代码的 85-92 行改为 `ans = max(ans, mx[u][i]);` 即求非严格的次小生成树的代码。非严格次小生成树与最小生成树权值是否相同可以判断最小生成树是否唯一。（POJ1679）（对较小数据，可以暴力：）

```

1 #include<cmath>
2 #include<cstring>
3 #include<cstdio>
4 #include<algorithm>
5 using namespace std;
6 const int qq=110;
7 const int MAX=1e8+5;
8 int n,m;
9 int vis[qq];
10 int dis[qq];
11 int map[qq][qq];
12 int maxn[qq][qq]; //记录最小生成树中点i到点j的最大距离、
13 int use[qq][qq]; //记录每一条边是否被使用过、
14 int pre[qq];
15 int Prim()
16 {
17     memset(vis,0,sizeof(vis));
18     memset(use,0,sizeof(use));
19     memset(maxn,0,sizeof(maxn));
20     for(int i=1; i<=n; ++i) dis[i]=map[1][i],pre[i]=1;
21     vis[1]=1;
22     dis[1]=0;
23     pre[1]=-1;
24     int ans=0,minx,k;
25     for(int i=1; i<n; ++i){
26         minx=MAX;
27         for(int j=1; j<=n; ++j)
28             if(!vis[j] && dis[j]<minx)
29                 minx=dis[k=j];
30         if(minx==MAX) return -1;
31         ans+=minx;
32         use[k][pre[k]]=use[pre[k]][k]=1;
33         vis[k]=1;
34         for(int j=1; j<=n; ++j){
35             if(vis[j]) maxn[j][k]=maxn[k][j]=max(dis[k],maxn[j]
36 [pre[k]]);
37             if(!vis[j] && map[k][j]<dis[j])
38                 pre[j]=k,dis[j]=map[k][j];
39         }
40     }
41     return ans;
42 }

```

```

42 int main()
43 {
44     int t; scanf("%d", &t);
45     while(t--){
46         scanf("%d%d", &n, &m);
47         int u, v, w;
48         for(int i=0; i<=n; ++i)
49             for(int j=0; j<=n; ++j)
50                 if(i!=j) map[i][j]=MAX;
51                 else map[i][j]=0;
52         for(int i=0; i<m; ++i){
53             scanf("%d%d%d", &u, &v, &w);
54             map[u][v]=map[v][u]=w;
55         }
56         int ans=Prim();
57         if(ans==1){
58             printf("Not Unique!\n");
59             continue;
60         }
61         int Min=MAX;
62         for(int j, i=1; i<=n; ++i)
63             for(j=i+1; j<=n; ++j)
64                 if(!use[i][j] && map[i][j] != MAX)
65                     Min=min(Min, ans-maxn[i][j]+map[i][j]);
66         if(Min==ans) printf("Not Unique!\n");
67         else printf("%d\n", ans);
68     }
69     return 0;
70 }

```

## 连通性

### 连通分量

无向图求连通分量：并查集 / DFS  $O(n + m)$

有向图求强联通分量：Tarjan 算法  $O(n + m)$

求无向图割点： $u$  为割点当①  $u$  非根节点且  $\exists low_v \geq dfn_u$  ②  $u$  为根节点且儿子不少于 2 个

求无向图桥：修改  $low$ ，限定非树边不能是子到父的反向边时，如果  $p$  是  $q$  的父节点，并且  $low_q \geq dfn_p$ ，那么  $p \leftrightarrow q$  是桥

有向图环缩点重建图：跑完 tarjan 后枚举边，不在同一强连通分量的在缩点图上连新边

可以解决：求有向图可重复走边/点的最长路 (缩点图跑拓扑排序求最长路)

### 强连通分量

洛谷P2863-给定有向图，求点数大于 1 的强连通分量的个数

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef int ll;

```

```

4  const ll MAXN = 2e4 + 10, MAXM = 2e5 + 10;
5  struct edge
6  {
7      ll to, nx;
8  } e[MAXN];
9  ll hd[MAXN], cnt, n, m, dfn[MAXN], low[MAXN], co[MAXN], num[MAXN], conum,
10 st;
11 bool vis[MAXN];
12 stack<ll> s;
13 void adde(ll &u, ll &v)
14 {
15     e[++cnt] = {v, hd[u]};
16     hd[u] = cnt;
17 }
18 void paint(ll &x)
19 {
20     s.pop();
21     co[x] = conum; //点x属于第conum个分量
22     ++num[conum]; //该分量有多少点
23     vis[x] = false; //出栈
24 }
25 void tarjan(ll x)
26 {
27     dfn[x] = low[x] = ++st;
28     s.push(x);
29     vis[x] = true;
30     for (ll i = hd[x]; i; i = e[i].nx)
31     {
32         ll toi = e[i].to;
33         if (!dfn[toi]) //小心别写成x
34         {
35             tarjan(toi); //小心别写成x
36             low[x] = min(low[x], low[toi]);
37         }
38         else if (vis[toi])
39             low[x] = min(low[x], dfn[toi]);
40     }
41     if (low[x] == dfn[x])
42     {
43         ++conum;
44         while (s.top() != x)
45             paint(s.top());
46         paint(x);
47     }
48 }
49 signed main()
50 {
51     ll ui, vi, ans = 0;
52     scanf("%d%d", &n, &m);
53     while (m--)
54         scanf("%d%d", &ui, &vi), adde(ui, vi);
55     for (ll i = 1; i <= n; ++i)
56         if (!dfn[i])
57             tarjan(i);
58     for (ll i = 1; i <= conum; ++i)
59         if (num[i] > 1)

```

```

59         ++ans;
60     printf("%d", ans);
61     return 0;
62 }

```

## 割点

洛谷P3388-无向图(不保证连通)求割点数和按编号顺序输出每个割点

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef int ll;
4 #define MAXN 20002
5 #define MAXM 100002
6 struct edge
7 {
8     ll to, nx;
9 } e[MAXM << 1];
10 ll n, m, idx, cnt, tot, hd[MAXN], dfn[MAXN], low[MAXN], ui, vi;
11 bool cut[MAXN];
12 inline void adde(ll &x, ll &y)
13 {
14     e[++cnt] = {y, hd[x]};
15     hd[x] = cnt;
16 }
17 void tarjan(ll u, ll fa)
18 {
19     dfn[u] = low[u] = ++idx;
20     ll child = 0;
21     for (ll i = hd[u]; i; i = e[i].nx)
22     {
23         ll toi = e[i].to;
24         if (!dfn[toi]) //尚未访问
25         {
26             tarjan(toi, fa);
27             low[u] = min(low[u], low[toi]);
28             if (low[toi] >= dfn[u] && u != fa)
29                 cut[u] = true;
30             if (u == fa)
31                 ++child;
32         }
33         low[u] = min(low[u], dfn[toi]);
34     }
35     if (child >= 2) // u == fa
36         cut[u] = true;
37 }
38 signed main()
39 {
40     scanf("%d%d", &n, &m);
41     while(m--)
42         scanf("%d%d", &ui, &vi), adde(ui, vi), adde(vi, ui);
43     for(ll i = 1; i <= n; ++i) if (!dfn[i]) tarjan(i, i);
44     for(ll i = 1; i <= n; ++i) if (cut[i]) ++tot;
45     printf("%d\n", tot);

```

```

46     for(11 i = 1; i <= n; ++i) if (cut[i]) printf("%d ", i);
47     return 0;
48 }

```

## 桥

UVA796-求有向图桥数量并按字典序输出每个桥的端点

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define sc(x) scanf("%lld", &x)
4 typedef long long ll;
5 #define mn 100010
6 #define mm 2000010
7 struct edge
8 {
9     ll to, nx;
10 } e[mm * 2];
11 ll hd[mn], cnt, n, ans, dfn[mn], low[mn], st;
12 vector<pair<ll, ll>> res;
13 void adde(ll u, ll v)
14 {
15     e[++cnt] = {v, hd[u]};
16     hd[u] = cnt;
17 }
18 void tarjan(ll u, ll fa)
19 {
20     dfn[u] = low[u] = ++st;
21     for (ll i = hd[u], v; i; i = e[i].nx)
22     {
23         v = e[i].to;
24         if (v != fa)
25         {
26             if (!dfn[v])
27             {
28                 tarjan(v, u);
29                 low[u] = min(low[u], low[v]);
30                 if (low[v] > dfn[u])
31                     ++ans, res.emplace_back(min(u, v), max(u, v));
32             }
33         }
34         low[u] = min(low[u], dfn[v]);
35     }
36 }
37
38 signed main()
39 {
40     while (EOF != scanf("%lld", &n))
41     {
42         cnt = ans = st = 0, res.clear();
43         for (ll i = 1, m, u, v; i <= n; ++i)
44         {
45             hd[i] = 0, dfn[i] = low[i] = 0;
46             scanf("%lld (%lld)", &u, &m), ++u;

```

```

47         while (m--)
48             scanf("%lld", &v), ++v, adde(u, v);
49     }
50     for (ll i = 1; i <= n; ++i)
51         if (!dfn[i])
52             tarjan(i, i);
53     sort(res.begin(), res.end());
54     printf("%lld critical links\n", ans);
55     for (auto &i : res)
56         printf("%lld - %lld\n", i.first - 1, i.second - 1);
57     putchar('\n');
58 }
59
60 }
```

## 点双连通分量

P8435  $n(\leq 5 \times 10^5)$  点  $m(\leq 2 \times 10^6)$  边无向图，找出点双数量，输出每个点双的点  
点双定义见下文圆方树

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 using ll = long long;
4 const ll mn = 5e5 + 10;
5 ll n, m, dfn[mn], low[mn], st, stk[mn], stop, cn;
6 vector<ll> c[mn], g[mn];
7 void tarjan(ll u, ll fa)
8 {
9     ll son = 0;
10    dfn[u] = low[u] = ++st;
11    stk[++stop] = u;
12    for (auto v : g[u])
13    {
14        if (!dfn[v])
15        {
16            ++son;
17            tarjan(v, u);
18            low[u] = min(low[u], low[v]);
19            if (low[v] >= dfn[u])
20            {
21                c[++cn].push_back(u);
22                for (ll w = 0; w != v;)
23                {
24                    w = stk[stop--];
25                    c[cn].push_back(w);
26                }
27            }
28        }
29    else
30    {
31        low[u] = min(low[u], dfn[v]);
32    }
33 }
```

```

34     if (!fa && !son)
35     {
36         c[++cn].push_back(u);
37     }
38 }
39 signed main()
{
40     cin.tie(0)->ios::sync_with_stdio(false);
41     cin >> n >> m;
42     for (ll i = 1, u, v; i <= m; ++i)
43     {
44         cin >> u >> v;
45         g[u].push_back(v), g[v].push_back(u);
46     }
47     for (ll i = 1; i <= n; ++i)
48     {
49         if (!dfn[i])
50         {
51             stop = 0, tarjan(i, 0);
52         }
53     }
54     cout << cn << '\n';
55     for (ll i = 1; i <= cn; ++i)
56     {
57         cout << c[i].size() << ' ';
58         for (auto u : c[i])
59         {
60             cout << u << ' ';
61         }
62         cout << '\n';
63     }
64     return 0;
65 }
66 }
67

```

## 边双连通分量

定义为无向图上没有桥的极大连通图。

牛客暑假7-L 给定带互异边权  $[-10^9, 10^9]$  的无向连通图  $3 \leq n \leq m \leq 10^5$ , 求环上边权最大最小值差最大的差值, 并以任意顺序时针输出该环

找环  $O(n)$ , 就是在边双上找最大和最小。然后用网络流输出方案。

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define sc(x) scanf("%lld", &x)
4 typedef long long ll;
5 const ll mn = 1e5 + 10;
6 ll n, m, hd[mn], cnt, w[mn], eu[mn], ev[mn];
7 struct edge
{

```

```

9      11 to, nx, idx, w = 0, isbridge = false;
10 } e[mn * 2], e2[mn * 2];
11 11 st, dfn[mn], low[mn], dcnt, vis[mn];
12 vector<11> dcc[mn]; //边双
13 11 hd2[mn], cnt2, h[mn], d[mn], cur[mn];
14 signed main()
15 {
16     sc(n), sc(m);
17     auto adde = [&](11 u, 11 v, 11 w, 11 idx)
18     {
19         e[++cnt] = {v, hd[u], idx};
20         hd[u] = cnt;
21         ::w[idx] = w, eu[idx] = u, ev[idx] = v;
22     };
23     for (11 i = 1, u, v, w; i <= m; ++i)
24     {
25         sc(u), sc(v), sc(w), adde(u, v, w, i), adde(v, u, w, i);
26     }
27     auto tarjan = [&](auto self, 11 u, 11 fa) -> void
28     {
29         dfn[u] = low[u] = ++st;
30         for (11 i = hd[u]; i; i = e[i].nx)
31         {
32             11 v = e[i].to, idx = e[i].idx;
33             if (!dfn[v])
34             {
35                 self(self, v, u);
36                 low[u] = min(low[u], low[v]);
37                 if (dfn[u] < low[v])
38                 {
39                     e[idx].isbridge = true;
40                 }
41             }
42             else if (v != fa)
43             {
44                 low[u] = min(low[u], dfn[v]);
45             }
46         }
47     };
48     tarjan(tarjan, 1, 0);
49     auto dfs = [&](auto self, 11 u) -> void
50     {
51         vis[u] = true;
52         for (11 i = hd[u]; i; i = e[i].nx)
53         {
54             11 v = e[i].to, idx = e[i].idx;
55             if (!e[idx].isbridge)
56             {
57                 dcc[dcnt].push_back(idx);
58                 if (!vis[v])
59                 {
60                     self(self, v);
61                 }
62             }
63         }
64     };

```

```

65     for (ll i = 1; i <= n; ++i)
66     {
67         if (!vis[i])
68         {
69             ++dcnt;
70             dfs(dfs, i);
71         }
72     }
73     ll res = -1, id = 0;
74     for (ll i = 1; i <= dcnt; ++i)
75     {
76         sort(dcc[i].begin(), dcc[i].end());
77         dcc[i].erase(unique(dcc[i].begin(), dcc[i].end()), dcc[i].end());
78         sort(dcc[i].begin(), dcc[i].end(), [&](ll u, ll v)
79             { return w[u] < w[v]; });
80         if (dcc[i].size() >= 2)
81         {
82             ll x = w[dcc[i].back()] - w[dcc[i].front()];
83             if (x > res)
84             {
85                 res = x, id = i;
86             }
87         }
88     }
89     printf("%lld\n", res);
90
91     //网络流是有向图, 所以要重建图
92     memset(hd2, -1, sizeof hd2);
93     ll s = n + 1, t = n + 2, mxi = dcc[id].back(), mii = dcc[id].front();
94     auto adde2 = [&](ll u, ll v, ll w = 1)
95     {
96         e2[cnt2] = {v, hd2[u], 0, w};
97         hd2[u] = cnt2++;
98     };
99     adde2(s, eu[mxi]), adde2(eu[mxi], s, 0); //注意顺序
100    adde2(s, ev[mxi]), adde2(ev[mxi], s, 0);
101    adde2(eu[mii], t), adde2(t, eu[mii], 0);
102    adde2(ev[mii], t), adde2(t, ev[mii], 0);
103    for (auto i : dcc[id])
104    {
105        if (i != mxi && i != mii)
106        {
107            adde2(eu[i], ev[i]), adde2(ev[i], eu[i]);
108        }
109    }
110
111    auto dinic_dfs = [&](auto self, ll u, ll limit) -> ll
112    {
113        // printf("<%lld %lld\n", u, limit);
114        if (u == t)
115        {
116            return limit;
117        }
118        ll flow = 0;
119        for (ll i = cur[u]; i != -1 && flow < limit; i = e2[i].nx)
120        {

```

```

121     cur[u] = i; //输出路径
122     ll v = e2[i].to;
123     if (d[v] == d[u] + 1 && e2[i].w)
124     {
125         ll dlt = self(self, v, min(e2[i].w, limit - flow));
126         if (!dlt)
127         {
128             d[v] = -1;
129         }
130         e2[i].w -= dlt;
131         e2[i].idx += dlt;
132         e2[i ^ 1].w += dlt;
133         e2[i ^ 1].idx -= dlt;
134         flow += dlt;
135     }
136 }
137     return flow;
138 };
139 auto dinic_bfs = [&]() -> bool
140 {
141     memset(d, -1, sizeof d);
142     queue<ll> q;
143     q.push(s);
144     d[s] = 0, cur[s] = hd2[s];
145     while (!q.empty())
146     {
147         ll u = q.front();
148         q.pop();
149         for (ll i = hd2[u]; i != -1; i = e2[i].nx)
150         {
151             ll v = e2[i].to;
152             // printf("%lld %lld\n", u, v);
153             if (d[v] == -1 && e2[i].w)
154             {
155                 d[v] = d[u] + 1, cur[v] = hd2[v];
156                 if (v == t)
157                 {
158                     return true;
159                 }
160                 q.push(v);
161             }
162         }
163     }
164     return false;
165 };
166 auto dinic = [&]()
167 {
168     ll res = 0;
169     while (dinic_bfs())
170     {
171         res += dinic_dfs(dinic_dfs, s, 1e18);
172     }
173     return res;
174 };
175 dinic();
176

```

```

177     vector<ll> res1, res2;
178     auto dfs2 = [&](auto self, ll u, vector<ll> &res) -> void
179     {
180         res.push_back(u);
181         for (ll i = hd2[u]; i != -1; i = e2[i].nx)
182         {
183             ll v = e2[i].to;
184             if (e2[i].idx > 0)
185             {
186                 e2[i].idx--;
187                 self(self, v, res);
188                 break;
189             }
190         }
191     };
192     dfs2(dfs2, s, res1);
193     dfs2(dfs2, s, res2);
194     reverse(res2.begin(), res2.end());
195     printf("%lld\n", res1.size() + res2.size() - 4);
196     auto print = [&](vector<ll> &res)
197     {
198         for (auto i : res)
199         {
200             if (i != s && i != t)
201             {
202                 printf("%lld ", i);
203             }
204         }
205     };
206     print(res1), print(res2);
207     return 0;
208 }

```

## 圆方树

圆方树 (Block forest 或 Round-square tree) 是一种将图变成树的方法从而把一般图上的某些问题转化到树上考虑

在圆方树中，原来的每个点对应一个 **圆点**，每一个点双对应一个 **方点**。

所以共有  $n + c$  个点，其中  $n$  是原图点数， $c$  是原图点双连通分量的个数。

而对于每一个点双连通分量，它对应的方点向这个点双连通分量中的每个点连边。

每个点双形成一个“菊花图”，多个“菊花图”通过原图中的割点连接在一起（因为点双的分隔点是割点）

如果原图连通，则“圆方树”才是一棵树，如果原图有  $k$  个连通分量，则它的圆方树也会形成  $k$  棵树形成的森林。如果原图中某个连通分量只有一个点，则需要具体情况具体分析，我们在后续讨论中不考虑孤立点。

点双是不存在割点的分量，一个点可能属于多个点双。使用 tarjan 算法求点双及每个点双包含的点。那么每个点双上的点都是圆点，连向一个新增的方点

## 2-SAT

洛谷P4782 有  $n$  个布尔值变量  $x$  和  $m$  个要满足的条件  $x_i$  为  $a$  或  $x_j$  为  $b$ ，构造一组解或报告无解  $1 \leq n, m \leq 10^6$

对每个变量  $x$ ，存两个点  $x, \neg x$  分别表示  $x$  取 1/0，编号为  $i, i + n$ 。在建图后，同一强连通分量内变量值一定相等。若  $x, \neg x$  在同一 SCC，那么一定无解。当  $x$  所在 SCC 的拓扑序在  $\neg x$  所在 SCC 拓扑序之后的话令  $x$  为真。复杂度  $O(n + m)$

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 #define sc(x) scanf("%lld", &x)
4 typedef long long ll;
5 #define mn 2000010
6 ll n, m, hd[mn], cnt, st, low[mn], dfn[mn], ins[mn], scc, c[mn];
7 stack<ll> s;
8 struct edge
9 {
10     ll to, nx;
11 } e[mn * 2];
12 void tarjan(ll u)
13 {
14     low[u] = dfn[u] = ++st, ins[u] = true;
15     s.push(u);
16     for (ll i = hd[u], v; i; i = e[i].nx)
17     {
18         v = e[i].to;
19         if (!dfn[v])
20         {
21             tarjan(v);
22             low[u] = min(low[u], low[v]);
23         }
24         else if (ins[v])
25         {
26             low[u] = min(low[u], dfn[v]);
27         }
28     }
29     if (low[u] == dfn[u])
30     {
31         ++scc;
32         do
33         {
34             c[u] = scc;
35             u = s.top();
36             s.pop();
37             ins[u] = false;
38         } while (low[u] != dfn[u]);
39     }
40 }
41 signed main()
42 {
43     sc(n), sc(m);
44     auto adde = [&](ll u, ll v)
45     { e[++cnt] = {v, hd[u]}, hd[u] = cnt; };
46     for (ll i = 1, a, va, b, vb; i <= m; ++i)
```

```

47  {
48      sc(a), sc(va), sc(b), sc(vb);
49      if (va && vb)
50      {
51          adde(a + n, b), adde(b + n, a);
52      }
53      else if (!va && vb)
54      {
55          adde(a, b), adde(b + n, a + n);
56      }
57      else if (va && !vb)
58      {
59          adde(a + n, b + n), adde(b, a);
60      }
61      else if (!va && !vb)
62      {
63          adde(a, b + n), adde(b, a + n);
64      }
65      // i.e. adde(a+n*va,b+n*(vb^1)),adde(b+n*vb,a+n*(va^1))
66  }
67  for (ll i = 1; i <= 2 * n; ++i)
68  {
69      if (!dfn[i])
70      {
71          tarjan(i);
72      }
73  }
74  for (ll i = 1; i <= n; ++i)
75  {
76      if (c[i] == c[i + n])
77      {
78          printf("IMPOSSIBLE");
79          return 0;
80      }
81  }
82  printf("POSSIBLE\n");
83  for (ll i = 1; i <= n; ++i)
84  { // tarjan逆拓扑序
85      printf("%d ", c[i] < c[i + n]);
86  }
87  return 0;
88 }

```

## 匹配问题

### 二分图

二分图性质:

1. 不存在长为奇数的环 (也是判定方法, 直接点 DFS 染二色即可判断)
2. 最小顶点覆盖数等于最大匹配数  $v(G)$

### 3. 最大独立集数 $\alpha(G) = n - v(G)$

转二分图：把每个点拆成两个点，分别加入二分图的两个点集，原图中一条由a到b的边在二分图中是一条由第一个点集中的第a个点到第二个点集中的第b个点

POJ1422-多组测试，给定  $n$  点  $m$  边；可以用多少条路径覆盖全部边使得每个点仅被一条路径访问

DAG 最小不可相交路径覆盖 =  $n - v(G)$

对每条边按上文拆分，求得最大匹配  $v(G)$ ，答案为  $n - v(G)$

POJ2594-给定有向图(可能零图)，求至少多少条路径可以遍历全部点

DAG 的最小可相交路径覆盖就先 floyd 求传递闭包，在闭包新图跑最大匹配，答案依然为  $n - v(G)$

## 最大匹配

匈牙利算法又名 KM 算法，邻接矩阵复杂度为  $O(n^3)$ ，邻接表为  $O(nm)$ ，可以求二分图最大匹配以及二分图最大权匹配(有完美时可求出完美)

也可以使用网络最大流 Dinic 算法以  $O(\sqrt{nm})$  求出

DFS 写法求二分图最大匹配数及其方案 (洛谷P2756-左部点权  $[1, m]$ ，右部  $[m + 1, n]$  若干条边)：

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 typedef double db;
5 #define sc(x) scanf("%lld", &x)
6 #define mn 105
7 vector<ll> e[mn];
8 ll m, n, ui, vi, t, mch[mn], vis[mn], r;
9 bool dfs(ll u, ll st)
10 {
11     if (vis[u] == st)
12     {
13         return false;
14     }
15     vis[u] = st;
16     for (auto &v : e[u])
17     {
18         if (!mch[v] || dfs(mch[v], st))
19         {
20             mch[v] = u;
21             return true;
22         }
23     }
24     return false;
25 }
26 signed main()
27 {
28     sc(m), sc(n);
29     while (true)
30     {
31         sc(ui), sc(vi);
32         if (ui == -1 && vi == -1)
```

```

33     {
34         break;
35     }
36     e[ui].emplace_back(vi);
37 }
38 for (ll i = 1; i <= m; ++i)
39 {
40     if (dfs(i, i))
41     {
42         ++r;
43     }
44 }
45 printf("%lld\n", r);
46 for (ll i = m + 1; i <= n; ++i)
47 {
48     if (mch[i])
49     {
50         printf("%lld %lld\n", mch[i], i);
51     }
52 }
53 return 0;
54 }

```

## 完美匹配

洛谷P6577-题面类似  $n_l = n_r$  , 求完美匹配(无重边, 保证有完美匹配)

BFS 版本: (复杂度  $O(n^3)$ )

对于允许非完美匹配的题目, 我们允许选取虚边, 也就是将虚边权设为0即可

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #define sc(x) scanf("%lld", &x)
5 #define mn 504
6 #define big 0xfffffffffffffff
7 ll n, m, e[mn][mn], mb[mn], vb[mn], ka[mn], kb[mn], p[mn], c[mn];
8 ll qf, qb, q[mn], u, v, w;
9 void bfs(ll u)
10 {
11     ll a, v = 0, v1 = 0, d;
12     fill(p + 1, p + n + 1, 0), fill(c + 1, c + n + 1, big), mb[v] = u;
13     do
14     {
15         a = mb[v], d = big, vb[v] = 1;
16         for (ll b = 1; b <= n; ++b)
17         {
18             if (!vb[b])
19             {
20                 if (c[b] > ka[a] + kb[b] - e[a][b])
21                 {
22                     c[b] = ka[a] + kb[b] - e[a][b], p[b] = v;
23                 }
24                 if (c[b] < d)
25                 {

```

```

26             d = c[b], v1 = b;
27         }
28     }
29 }
30 for (ll b = 0; b <= n; ++b)
31 {
32     if (vb[b])
33     {
34         ka[mb[b]] -= d, kb[b] += d;
35     }
36     else
37     {
38         c[b] -= d;
39     }
40 }
41 v = v1;
42 } while (mb[v]);
43 while (v)
44 {
45     mb[v] = mb[p[v]], v = p[v];
46 }
47 }
48 ll km()
49 { // memset mb,ka,kb->0
50     for (ll a = 1; a <= n; ++a)
51     {
52         fill(vb, vb + n + 1, 0);
53         bfs(a);
54     }
55     ll res = 0;
56     for (ll b = 1; b <= n; ++b)
57     {
58         res += e[mb[b]][b];
59     }
60     return res;
61 }
62 signed main()
63 {
64     sc(n), sc(m);
65     memset(e, -127, sizeof e); // -big 暴毙
66     while (m--)
67     {
68         sc(u), sc(v), sc(w);
69         e[u][v] = max(e[u][v], w);
70     }
71     printf("%lld\n", km());
72     for (ll i = 1; i <= n; ++i)
73     {
74         printf("%lld ", mb[i]);
75     }
76     return 0;
77 }

```

## 最大权匹配

UOJ80-左部点数  $n_l$ ，右部  $n_r$ ，编号依次 1 开始， $m$  条加权边，求最大权匹配以及第  $i$  个左部点匹配哪个右部点（0 代表无匹配）

BFS写法

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 #define Maxn 410
4 #define Maxm 160010
5 #define LL long long
6 const LL INF = 1LL << 62;
7 int N, n, m;
8 int match[Maxn], op[Maxn];
9 LL eg[Maxn][Maxn], lx[Maxn], ly[Maxn], slack[Maxn];
10 int visx[Maxn], visy[Maxn], pre[Maxn];
11 void mh(int y)
12 {
13     for (int x, z; y > 0; y = z)
14     {
15         x = pre[y];
16         z = op[x];
17         op[x] = y;
18         match[y] = x;
19     }
20 }
21
22 int nt;
23 void ffind(int st)
24 {
25     for (int i = 1; i <= N; i++)
26         slack[i] = INF;
27     queue<int> q;
28     nt++;
29     q.push(st);
30     visx[st] = nt;
31     while (1)
32     {
33         while (!q.empty())
34         {
35             int x = q.front();
36             q.pop();
37             for (int y = 1; y <= N; y++)
38                 if (visy[y] != nt)
39                 {
40                     if (lx[x] + ly[y] == eg[x][y])
41                     {
42                         pre[y] = x;
43                         if (!match[y])
44                         {
45                             mh(y);
46                             return;
47                         }
48                         q.push(match[y]);
49                         visx[match[y]] = nt;
50                     }
51                 }
52         }
53     }
54 }
```

```

50             visy[y] = nt;
51         }
52         else if (slack[y] > lx[x] + ly[y] - eg[x][y])
53             slack[y] = lx[x] + ly[y] - eg[x][y], pre[y] = x;
54     }
55 }
56 LL delta = INF;
57 for (int y = 1; y <= N; y++)
58     if (visy[y] != nt)
59         delta = min(delta, slack[y]);
60 for (int i = 1; i <= N; i++)
61 {
62     if (visx[i] == nt)
63         lx[i] -= delta;
64     if (visy[i] == nt)
65         ly[i] += delta;
66     else
67         slack[i] -= delta;
68 }
69 for (int i = 1; i <= N; i++)
70     if (visy[i] != nt && slack[i] == 0)
71     {
72         if (!match[i])
73         {
74             mh(i);
75             return;
76         }
77         q.push(match[i]);
78         visx[match[i]] = nt;
79         visy[i] = nt;
80     }
81 }
82 }
83 void KM()
84 {
85     for (int i = 1; i <= N; i++)
86         lx[i] = ly[i] = 0, op[i] = match[i] = 0;
87     for (int i = 1; i <= N; i++)
88         for (int j = 1; j <= N; j++)
89             lx[i] = max(lx[i], eg[i][j]);
90     for (int i = 1; i <= N; i++)
91         ffind(i);
92 }
93 void output()
94 {
95     LL ans = 0;
96     for (int i = 1; i <= N; i++)
97         ans += lx[i] + ly[i];
98     printf("%lld\n", ans);
99     for (int i = 1; i <= n; i++)
100         if (eg[i][op[i]] != 0)
101             printf("%d ", op[i]);
102         else
103             printf("0 ");
104 }
105 void init()

```

```

106 {
107     int l;
108     scanf("%d%d%d", &n, &m, &l);
109     for (int i = 1; i <= l; i++)
110     {
111         int x, y;
112         LL c;
113         scanf("%d%d%lld", &x, &y, &c);
114         eg[x][y] = max(eg[x][y], c);
115     }
116     N = max(n, m);
117 }
118 int main()
119 {
120     init();
121     KM();
122     output();
123     return 0;
124 }
```

## 最小边覆盖

lc1595-求最小边覆盖(用权和最小的边集覆盖所有的点)

最小权边覆盖可以转化为最小权匹配。边覆盖=匹配+额外边。

额外边是加了之后只多覆盖一个点的边，对不在匹配的每个点，选择最小的边，不会影响其他不在匹配的点。对邻接矩阵，二分图左边第  $i$  个点的最优额外边是  $\min e_{i,:}$ ，右边第  $j$  个点的最优额外边是  $\min e_{:,j}$ 。即分别为行列最小值。

显然一种较差的解是每个点都直接选择最小的额外边，称为基本解。在基本解的前提下，引入匹配，每加入一条匹配边  $(u, v)$ ，首先去掉了两边的额外边，再加上这条边的边权，即  $e_{i,j} - \min e_{i,:} - \min e_{:,j}$ 。在费用网络流建立一个流量为 1，费用为上述值的边。然后源点跟左边、右边跟汇点都连流量为 1 费用为 0 的边，即求朴素的最小权匹配。

使用 MCFN，复杂度为  $O(\min(n, m)^2 \max(n, m)) = O(n^3)$ 。

一个包装良好的复杂长板子：

前置依赖：

```

1 namespace details {
2     inline unsigned int bsf32(uint32_t n) noexcept {
3 #if defined __GNUC__
4         return __builtin_ctz(n);
5 #elif defined __MSC_VER
6         unsigned long ans;
7         _BitScanForward(&ans, n);
8         return ans;
9 #elif
10         static constexpr unsigned char table[32] = {
11             0, 1, 28, 2, 29, 14, 24, 3, 30, 22, 20, 15, 25, 17, 4, 8,
12             31, 27, 13, 23, 21, 19, 16, 7, 26, 12, 18, 6, 11, 5, 10,
13             9,
14         };
15         return table[((n & -n) * 0x077CB531) >> 57];
16 #endif
17 }
```

```

16     }
17
18     inline unsigned int bsf64(uint64_t n) noexcept {
19     #if defined __GNUC__
20         return __builtin_ctzll(n);
21     #elif defined _MSC_VER
22         unsigned long ans;
23         _BitScanForward64(&ans, n);
24         return ans;
25     #elif
26         static constexpr unsigned char table[64] = {
27             0, 1, 56, 2, 57, 49, 28, 3, 61, 58, 42, 50, 38, 29, 17, 4,
28             62, 47, 59, 36, 45, 43, 51, 22, 53, 39, 33, 30, 24, 18,
29             12, 5,
30             63, 55, 48, 27, 60, 41, 37, 16, 46, 35, 44, 21, 52, 32,
31             23, 11,
32             54, 26, 40, 15, 34, 20, 31, 10, 25, 14, 19, 9, 13, 8, 7,
33             6,
34         };
35         return table[((n & -n) * 0x03f79d71b4ca8b09) >> 58];
36     #endif
37     }
38
39     inline unsigned int bsr32(uint32_t n) noexcept {
40     #if defined __GNUC__
41         return 31 - __builtin_clz(n);
42     #elif defined _MSC_VER
43         unsigned long ans;
44         _BitScanReverse(&ans, n);
45         return ans;
46     #elif
47         float t = n;
48         uint32_t ans;
49         memcpy(&ans, &t, sizeof(float));
50         return (ans >> 23 & 255) - 127;
51     #endif
52     }
53
54     inline unsigned int bsr64(uint64_t n) noexcept {
55     #if defined __GNUC__
56         return 63 - __builtin_clzll(n);
57     #elif defined _MSC_VER
58         unsigned long ans;
59         _BitScanReverse64(&ans, n);
60         return ans;
61     #elif
62         float t = n;
63         uint32_t ans;
64         memcpy(&ans, &t, sizeof(float));
65         return (ans >> 23 & 255) - 127;
66     #endif
67     }
68
69     inline unsigned int popcnt32(uint32_t n) noexcept {
70     #if defined __GNUC__
71         return __builtin_popcount(n);
72     #endif
73     }
74
75     inline unsigned int popcnt64(uint64_t n) noexcept {
76     #if defined __GNUC__
77         return __builtin_popcountll(n);
78     #endif
79     }
80
81     inline unsigned int popcnt128(uint128_t n) noexcept {
82     #if defined __GNUC__
83         return __builtin_popcountll(n);
84     #endif
85     }
86
87     inline unsigned int popcnt256(uint256_t n) noexcept {
88     #if defined __GNUC__
89         return __builtin_popcountll(n);
90     #endif
91     }
92
93     inline unsigned int popcnt512(uint512_t n) noexcept {
94     #if defined __GNUC__
95         return __builtin_popcountll(n);
96     #endif
97     }
98
99     inline unsigned int popcnt1024(uint1024_t n) noexcept {
100    #if defined __GNUC__
101        return __builtin_popcountll(n);
102    #endif
103    }
104
105    inline unsigned int popcnt2048(uint2048_t n) noexcept {
106        #if defined __GNUC__
107            return __builtin_popcountll(n);
108        #endif
109        }
110
111    inline unsigned int popcnt4096(uint4096_t n) noexcept {
112        #if defined __GNUC__
113            return __builtin_popcountll(n);
114        #endif
115        }
116
117    inline unsigned int popcnt8192(uint8192_t n) noexcept {
118        #if defined __GNUC__
119            return __builtin_popcountll(n);
120        #endif
121        }
122
123    inline unsigned int popcnt16384(uint16384_t n) noexcept {
124        #if defined __GNUC__
125            return __builtin_popcountll(n);
126        #endif
127        }
128
129    inline unsigned int popcnt32768(uint32768_t n) noexcept {
130        #if defined __GNUC__
131            return __builtin_popcountll(n);
132        #endif
133        }
134
135    inline unsigned int popcnt65536(uint65536_t n) noexcept {
136        #if defined __GNUC__
137            return __builtin_popcountll(n);
138        #endif
139        }
140
141    inline unsigned int popcnt131072(uint131072_t n) noexcept {
142        #if defined __GNUC__
143            return __builtin_popcountll(n);
144        #endif
145        }
146
147    inline unsigned int popcnt262144(uint262144_t n) noexcept {
148        #if defined __GNUC__
149            return __builtin_popcountll(n);
150        #endif
151        }
152
153    inline unsigned int popcnt524288(uint524288_t n) noexcept {
154        #if defined __GNUC__
155            return __builtin_popcountll(n);
156        #endif
157        }
158
159    inline unsigned int popcnt1048576(uint1048576_t n) noexcept {
160        #if defined __GNUC__
161            return __builtin_popcountll(n);
162        #endif
163        }
164
165    inline unsigned int popcnt2097152(uint2097152_t n) noexcept {
166        #if defined __GNUC__
167            return __builtin_popcountll(n);
168        #endif
169        }
170
171    inline unsigned int popcnt4194304(uint4194304_t n) noexcept {
172        #if defined __GNUC__
173            return __builtin_popcountll(n);
174        #endif
175        }
176
177    inline unsigned int popcnt8388608(uint8388608_t n) noexcept {
178        #if defined __GNUC__
179            return __builtin_popcountll(n);
180        #endif
181        }
182
183    inline unsigned int popcnt16777216(uint16777216_t n) noexcept {
184        #if defined __GNUC__
185            return __builtin_popcountll(n);
186        #endif
187        }
188
189    inline unsigned int popcnt33554432(uint33554432_t n) noexcept {
190        #if defined __GNUC__
191            return __builtin_popcountll(n);
192        #endif
193        }
194
195    inline unsigned int popcnt67108864(uint67108864_t n) noexcept {
196        #if defined __GNUC__
197            return __builtin_popcountll(n);
198        #endif
199        }
200
201    inline unsigned int popcnt134217728(uint134217728_t n) noexcept {
202        #if defined __GNUC__
203            return __builtin_popcountll(n);
204        #endif
205        }
206
207    inline unsigned int popcnt268435456(uint268435456_t n) noexcept {
208        #if defined __GNUC__
209            return __builtin_popcountll(n);
210        #endif
211        }
212
213    inline unsigned int popcnt536870912(uint536870912_t n) noexcept {
214        #if defined __GNUC__
215            return __builtin_popcountll(n);
216        #endif
217        }
218
219    inline unsigned int popcnt1073741824(uint1073741824_t n) noexcept {
220        #if defined __GNUC__
221            return __builtin_popcountll(n);
222        #endif
223        }
224
225    inline unsigned int popcnt2147483648(uint2147483648_t n) noexcept {
226        #if defined __GNUC__
227            return __builtin_popcountll(n);
228        #endif
229        }
230
231    inline unsigned int popcnt4294967296(uint4294967296_t n) noexcept {
232        #if defined __GNUC__
233            return __builtin_popcountll(n);
234        #endif
235        }
236
237    inline unsigned int popcnt8589934592(uint8589934592_t n) noexcept {
238        #if defined __GNUC__
239            return __builtin_popcountll(n);
240        #endif
241        }
242
243    inline unsigned int popcnt17179869184(uint17179869184_t n) noexcept {
244        #if defined __GNUC__
245            return __builtin_popcountll(n);
246        #endif
247        }
248
249    inline unsigned int popcnt34359738368(uint34359738368_t n) noexcept {
250        #if defined __GNUC__
251            return __builtin_popcountll(n);
252        #endif
253        }
254
255    inline unsigned int popcnt68719476736(uint68719476736_t n) noexcept {
256        #if defined __GNUC__
257            return __builtin_popcountll(n);
258        #endif
259        }
260
261    inline unsigned int popcnt137438953472(uint137438953472_t n) noexcept {
262        #if defined __GNUC__
263            return __builtin_popcountll(n);
264        #endif
265        }
266
267    inline unsigned int popcnt274877906944(uint274877906944_t n) noexcept {
268        #if defined __GNUC__
269            return __builtin_popcountll(n);
270        #endif
271        }
272
273    inline unsigned int popcnt549755813888(uint549755813888_t n) noexcept {
274        #if defined __GNUC__
275            return __builtin_popcountll(n);
276        #endif
277        }
278
279    inline unsigned int popcnt1099511627776(uint1099511627776_t n) noexcept {
280        #if defined __GNUC__
281            return __builtin_popcountll(n);
282        #endif
283        }
284
285    inline unsigned int popcnt219902325552(uint219902325552_t n) noexcept {
286        #if defined __GNUC__
287            return __builtin_popcountll(n);
288        #endif
289        }
290
291    inline unsigned int popcnt439804651104(uint439804651104_t n) noexcept {
292        #if defined __GNUC__
293            return __builtin_popcountll(n);
294        #endif
295        }
296
297    inline unsigned int popcnt879609302208(uint879609302208_t n) noexcept {
298        #if defined __GNUC__
299            return __builtin_popcountll(n);
300        #endif
301        }
302
303    inline unsigned int popcnt1759218604416(uint1759218604416_t n) noexcept {
304        #if defined __GNUC__
305            return __builtin_popcountll(n);
306        #endif
307        }
308
309    inline unsigned int popcnt3518437208832(uint3518437208832_t n) noexcept {
310        #if defined __GNUC__
311            return __builtin_popcountll(n);
312        #endif
313        }
314
315    inline unsigned int popcnt7036874417664(uint7036874417664_t n) noexcept {
316        #if defined __GNUC__
317            return __builtin_popcountll(n);
318        #endif
319        }
320
321    inline unsigned int popcnt14073748835328(uint14073748835328_t n) noexcept {
322        #if defined __GNUC__
323            return __builtin_popcountll(n);
324        #endif
325        }
326
327    inline unsigned int popcnt28147497670656(uint28147497670656_t n) noexcept {
328        #if defined __GNUC__
329            return __builtin_popcountll(n);
330        #endif
331        }
332
333    inline unsigned int popcnt56294995341312(uint56294995341312_t n) noexcept {
334        #if defined __GNUC__
335            return __builtin_popcountll(n);
336        #endif
337        }
338
339    inline unsigned int popcnt112589990682624(uint112589990682624_t n) noexcept {
340        #if defined __GNUC__
341            return __builtin_popcountll(n);
342        #endif
343        }
344
345    inline unsigned int popcnt225179981365248(uint225179981365248_t n) noexcept {
346        #if defined __GNUC__
347            return __builtin_popcountll(n);
348        #endif
349        }
350
351    inline unsigned int popcnt450359962730496(uint450359962730496_t n) noexcept {
352        #if defined __GNUC__
353            return __builtin_popcountll(n);
354        #endif
355        }
356
357    inline unsigned int popcnt900719925460992(uint900719925460992_t n) noexcept {
358        #if defined __GNUC__
359            return __builtin_popcountll(n);
360        #endif
361        }
362
363    inline unsigned int popcnt1801439850921984(uint1801439850921984_t n) noexcept {
364        #if defined __GNUC__
365            return __builtin_popcountll(n);
366        #endif
367        }
368
369    inline unsigned int popcnt3602879701843968(uint3602879701843968_t n) noexcept {
370        #if defined __GNUC__
371            return __builtin_popcountll(n);
372        #endif
373        }
374
375    inline unsigned int popcnt7205759403687936(uint7205759403687936_t n) noexcept {
376        #if defined __GNUC__
377            return __builtin_popcountll(n);
378        #endif
379        }
380
381    inline unsigned int popcnt14411518807375872(uint14411518807375872_t n) noexcept {
382        #if defined __GNUC__
383            return __builtin_popcountll(n);
384        #endif
385        }
386
387    inline unsigned int popcnt28823037614751744(uint28823037614751744_t n) noexcept {
388        #if defined __GNUC__
389            return __builtin_popcountll(n);
390        #endif
391        }
392
393    inline unsigned int popcnt57646075229503488(uint57646075229503488_t n) noexcept {
394        #if defined __GNUC__
395            return __builtin_popcountll(n);
396        #endif
397        }
398
399    inline unsigned int popcnt115292150459006976(uint115292150459006976_t n) noexcept {
400        #if defined __GNUC__
401            return __builtin_popcountll(n);
402        #endif
403        }
404
405    inline unsigned int popcnt230584300918013952(uint230584300918013952_t n) noexcept {
406        #if defined __GNUC__
407            return __builtin_popcountll(n);
408        #endif
409        }
410
411    inline unsigned int popcnt461168601836027904(uint461168601836027904_t n) noexcept {
412        #if defined __GNUC__
413            return __builtin_popcountll(n);
414        #endif
415        }
416
417    inline unsigned int popcnt922337203672055808(uint922337203672055808_t n) noexcept {
418        #if defined __GNUC__
419            return __builtin_popcountll(n);
420        #endif
421        }
422
423    inline unsigned int popcnt1844674407344111616(uint1844674407344111616_t n) noexcept {
424        #if defined __GNUC__
425            return __builtin_popcountll(n);
426        #endif
427        }
428
429    inline unsigned int popcnt3689348814688223232(uint3689348814688223232_t n) noexcept {
430        #if defined __GNUC__
431            return __builtin_popcountll(n);
432        #endif
433        }
434
435    inline unsigned int popcnt7378697629376446464(uint7378697629376446464_t n) noexcept {
436        #if defined __GNUC__
437            return __builtin_popcountll(n);
438        #endif
439        }
440
441    inline unsigned int popcnt14757395258752892928(uint14757395258752892928_t n) noexcept {
442        #if defined __GNUC__
443            return __builtin_popcountll(n);
444        #endif
445        }
446
447    inline unsigned int popcnt29514790517505785856(uint29514790517505785856_t n) noexcept {
448        #if defined __GNUC__
449            return __builtin_popcountll(n);
450        #endif
451        }
452
453    inline unsigned int popcnt59029581035011571712(uint59029581035011571712_t n) noexcept {
454        #if defined __GNUC__
455            return __builtin_popcountll(n);
456        #endif
457        }
458
459    inline unsigned int popcnt11805916207002314344(uint11805916207002314344_t n) noexcept {
460        #if defined __GNUC__
461            return __builtin_popcountll(n);
462        #endif
463        }
464
465    inline unsigned int popcnt23611832414004628688(uint23611832414004628688_t n) noexcept {
466        #if defined __GNUC__
467            return __builtin_popcountll(n);
468        #endif
469        }
470
471    inline unsigned int popcnt47223664828009257376(uint47223664828009257376_t n) noexcept {
472        #if defined __GNUC__
473            return __builtin_popcountll(n);
474        #endif
475        }
476
477    inline unsigned int popcnt94447329656018514752(uint94447329656018514752_t n) noexcept {
478        #if defined __GNUC__
479            return __builtin_popcountll(n);
480        #endif
481        }
482
483    inline unsigned int popcnt18889465931203702904(uint18889465931203702904_t n) noexcept {
484        #if defined __GNUC__
485            return __builtin_popcountll(n);
486        #endif
487        }
488
489    inline unsigned int popcnt37778931862407405808(uint37778931862407405808_t n) noexcept {
490        #if defined __GNUC__
491            return __builtin_popcountll(n);
492        #endif
493        }
494
495    inline unsigned int popcnt75557863724814811616(uint75557863724814811616_t n) noexcept {
496        #if defined __GNUC__
497            return __builtin_popcountll(n);
498        #endif
499        }
500
501    inline unsigned int popcnt1511157274496296232(uint1511157274496296232_t n) noexcept {
502        #if defined __GNUC__
503            return __builtin_popcountll(n);
504        #endif
505        }
506
507    inline unsigned int popcnt3022314548992592464(uint3022314548992592464_t n) noexcept {
508        #if defined __GNUC__
509            return __builtin_popcountll(n);
510        #endif
511        }
512
513    inline unsigned int <b
```

```

69 #elif defined __MSC_VER
70     return __popcnt(n);
71 #elif
72     n -= ((n >> 1) & 0x55555555);
73     n = (n & 0x33333333) + ((n >> 2) & 0x33333333);
74     return (((n >> 4) + n) & 0x0f0f0f0f) * 0x01010101) >> 24;
75 #endif
76 }
77
78     inline unsigned int popcnt64(uint64_t n) noexcept {
79 #if defined __GNUC__
80     return __builtin_popcountll(n);
81 #elif defined __MSC_VER
82     return __popcnt64(n);
83 #elif
84     n -= ((n >> 1) & 0x5555555555555555);
85     n = (n & 0x3333333333333333) + ((n >> 2) &
86     0x3333333333333333);
87     return (((n >> 4) + n) & 0x0f0f0f0f0f0f0f) *
88     0x0101010101010101) >> 56;
89 #endif
90 }
91
92     inline unsigned int parity32(uint32_t n) noexcept {
93 #if defined __GNUC__
94     return __builtin_parity(n);
95 #elif defined __MSC_VER
96     return __popcnt(n) & 1;
97 #elif
98     n ^= n >> 1;
99     n ^= n >> 2;
100    n = (n & 0x11111111) * 0x11111111;
101    return (n >> 28) & 1;
102 #endif
103 }
104
105     inline unsigned int parity64(uint64_t n) noexcept {
106 #if defined __GNUC__
107     return __builtin_parityll(n);
108 #elif defined __MSC_VER
109     return __popcnt64(n) & 1;
110 #elif
111     n ^= n >> 1;
112     n ^= n >> 2;
113     n = (n & 0x1111111111111111) * 0x1111111111111111;
114     return (n >> 60) & 1;
115 #endif
116 }
117
118     template<typename I, bool B = 32 < std::numeric_limits<I>::digits>
119     struct _BitFuncImpl {
120         inline static unsigned int bsf(I n) noexcept { return
bsf64(n); }
121         inline static unsigned int bsr(I n) noexcept { return
bsr64(n); }

```

```

121     inline static unsigned int popcnt(I n) noexcept { return
122     popcnt64(n); }
123     inline static unsigned int parity(I n) noexcept { return
124     parity64(n); }
125
126     template<typename I>
127     struct _BitFuncImpl<I, false> {
128         inline static unsigned int bsf(I n) noexcept { return
129         bsf32(n); }
130         inline static unsigned int bsr(I n) noexcept { return
131         bsr32(n); }
132         inline static unsigned int popcnt(I n) noexcept { return
133         popcnt32(n); }
134         inline static unsigned int parity(I n) noexcept { return
135         parity32(n); }
136
137         template<typename I>
138         inline unsigned int bsf(I n) noexcept { return
139         _BitFuncImpl<I>::bsf(n); }
140         template<typename I>
141         inline unsigned int bsr(I n) noexcept { return
142         _BitFuncImpl<I>::bsr(n); }
143         template<typename I>
144         inline unsigned int popcnt(I n) noexcept { return
145         _BitFuncImpl<I>::popcnt(n); }
146         template<typename I>
147         inline unsigned int parity(I n) noexcept { return
148         _BitFuncImpl<I>::parity(n); }
149     }
150
151     template<typename E>
152     class Network {
153     public:
154         using edge_type = E;
155         using size_type = unsigned;
156
157         private:
158         size_type n, m;
159         size_type* pos;
160         std::unique_ptr<edge_type[]> edges;
161
162         public:
163         Network(std::unique_ptr<edge_type[]> edges,
164                 std::unique_ptr<size_type[]> spos, size_type n, size_type m)
165             : n(n), m(m), pos(spos.release() + 1), edges(std::move(edges))
166         {
167             pos[-1] = 1;
168         }
169
170         Network(const Network& other) : n(other.n), m(other.m),
171             pos(other.pos), edges(new edge_type[m]) {
172             std::copy_n(other.edges.get(), m, edges.get());
173             ++pos[-1];
174         }

```

```

165
166 ~Network() { if (--pos[-1] == 0) delete[] (pos - 1); }
167
168 inline size_type order() const noexcept { return n; }
169 inline size_type size() const noexcept { return m; }
170 inline const size_type* raw_pos() const noexcept { return pos; }
171 inline const edge_type* raw_edges() const noexcept { return
172 edges.get(); }
173 inline edge_type* raw_edges() noexcept { return edges.get(); }
174 };
175
176 template<typename F = int, typename C = int>
177 class MCFN {
178 public:
179     using size_type = unsigned;
180
181 private:
182     struct RawEdge {
183         size_type from, to;
184         F cap;
185         C cost;
186     };
187
188     struct Edge {
189         friend class MCFN;
190         size_type _rev, _to;
191         F _cap;
192         C _cost;
193         Edge(size_type rev, size_type to, F cap, C cost) : _rev(rev),
194             _to(to), _cap(cap), _cost(cost) {}
195     public:
196         using flow_type = F;
197         using cost_type = C;
198         Edge() = default;
199         inline size_type rev() const noexcept { return _rev; }
200         inline size_type to() const noexcept { return _to; }
201         inline flow_type residual() const noexcept { return _cap; }
202         inline bool full() const noexcept { return _cap == 0; }
203         inline void push(flow_type d) noexcept { _cap -= d; }
204         inline cost_type cost() const noexcept { return _cost; }
205     };
206
207     std::vector<RawEdge> edges;
208     std::vector<size_type> deg;
209
210 public:
211     MCFN() noexcept = default;
212
213     MCFN(size_type n) : deg(n, 0) {}
214
215     void reserve(size_type n, size_type m) {
216         edges.reserve(m);
217         deg.reserve(n);
218     }

```

```

219 inline size_type add_vertex() {
220     const size_type ans = deg.size();
221     deg.push_back(0);
222     return ans;
223 }
224
225 inline void add_edge(size_type from, size_type to, F cap, C cost) {
226     if (cap <= 0) return;
227     edges.emplace_back(RawEdge{from, to, cap, cost});
228     ++deg[from];
229     ++deg[to];
230 }
231
232 Network<Edge> prepare() const {
233     using size_type = typename Network<Edge>::size_type;
234     const size_type n = deg.size(), m = edges.size();
235     std::unique_ptr<Edge[]> sorted_edges(new Edge[2 * m]);
236     std::unique_ptr<size_type[]> spos(new size_type[n + 2]);
237     const auto pos = spos.get() + 1;
238     *std::partial_sum(deg.begin(), deg.end(), pos) = 2 * m;
239     for (size_type i = m; i-- > 0;) {
240         const RawEdge e = edges[i];
241         const size_type u = e.from, v = e.to;
242         const size_type rp = --pos[v], p = --pos[u];
243         sorted_edges[p] = Edge(rp, v, e.cap, e.cost);
244         sorted_edges[rp] = Edge(p, u, 0, -e.cost);
245     }
246     return Network<Edge>(std::move(sorted_edges), std::move(spos), n, 2
247     * m);
248 }

```

网络流算法：

```

1 template<typename F = int, typename C = int, typename G>
2 auto ssp(G&& g, std::size_t s, std::size_t t, bool nneg = false)
3 -> std::pair<
4     typename std::common_type<F, typename
5         std::remove_reference<G>::type::edge_type::flow_type>::type,
6     typename std::common_type<C, typename
7         std::remove_reference<G>::type::edge_type::cost_type>::type>
8     {
9         using graph_type = typename std::remove_reference<G>::type;
10        using size_type = typename graph_type::size_type;
11        using edge_type = typename graph_type::edge_type;
12        using flow_type = typename std::common_type<F, typename
13            edge_type::flow_type>::type;
14        using cost_type = typename std::common_type<C, typename
15            edge_type::cost_type>::type;
16        static constexpr size_type BIN_LEN =
17            std::numeric_limits<cost_type>::digits;
18        static constexpr cost_type INF =
19            std::numeric_limits<cost_type>::max();
20        struct ShortestPath {
21            size_type pre;

```

```

16     cost_type pot, dis;
17 };
18 struct ListNode {
19     size_type prev, next;
20 };
21 const size_type n = g.order(), m = g.size();
22 const bool use_heap = m / n <= n / std::min<size_type>(3 *
23     details::bsr(n + 7) + 15, 64);
24 const auto edges = g.raw_edges();
25 const auto pos = g.raw_pos();
26 std::vector<ShortestPath> sp(n);
27 std::vector<bool> vis;
28 const auto init = [&] {
29     vis.assign(n, false);
30     std::vector<size_type> q(n);
31     size_type l = 0, r = 1;
32     for (auto& e : sp)
33         e.pot = INF;
34     sp[s] = {0, 0};
35     vis[s] = true;
36     q[0] = s;
37     while (l != r) {
38         const size_type u = q[l];
39         l = l + 1 != n ? l + 1 : 0;
40         vis[u] = false;
41         const size_type pu = sp[u].pre;
42         const cost_type du = sp[u].pot;
43         const auto start_edge = edges + pos[u];
44         const auto end_edge = edges + pos[u + 1];
45         for (auto i = start_edge; i != end_edge; ++i) {
46             const edge_type e = *i;
47             if (e.full()) continue;
48             const size_type v = e.to();
49             const cost_type dv = sp[v].pot;
50             const cost_type ndv = du + e.cost();
51             if (ndv < dv) {
52                 sp[v].pre = pu + 1;
53                 sp[v].pot = ndv;
54                 if (sp[v].pre >= n) throw std::invalid_argument("graph
contains negative cycles.");
55                 if (!vis[v]) {
56                     q[r] = v;
57                     vis[v] = true;
58                     r = r + 1 != n ? r + 1 : 0;
59                 }
60             }
61         }
62     };
63     if (!nneg) init();
64     const cost_type pt = sp[t].pot;
65     flow_type flow = 0;
66     cost_type cost = 0;
67     const auto augment = [&] {
68         flow_type f = std::numeric_limits<flow_type>::max();
69         for (size_type i = t; i != s;) {

```

```

70     const edge_type e = edges[sp[i].pre];
71     const flow_type r = edges[e.rev()].residual();
72     if (r < f) f = r;
73     i = e.to();
74 }
75 for (size_type i = t; i != s;) {
76     auto& e = edges[sp[i].pre];
77     edges[e.rev()].push(f);
78     e.push(-f);
79     i = e.to();
80 }
81 flow += f;
82 const cost_type dt = sp[t].dis;
83 for (auto& e : sp)
84     if (e.dis < dt)
85         e.pot -= dt - e.dis;
86 cost += cost_type(f) * (pt - sp[s].pot);
87 };
88 const auto flow_with_heap = [&] {
89     std::vector<ListNode> list(n + BIN_LEN + 1);
90     const auto dijkstra_with_heap = [&] {
91         const auto radix_distance = [] (cost_type lhs, cost_type rhs) {
92             return lhs != rhs ? details::bsr(lhs ^ rhs) + 1 : 0;
93         };
94         const auto buk = list.data() + n;
95         const auto insert_list = [&] (size_type idx, size_type v) {
96             const size_type rear = buk[idx].prev;
97             list[rear].next = v;
98             list[v].prev = rear;
99             buk[idx].prev = v;
100            list[v].next = idx + n;
101        };
102        cost_type top = 0;
103        for (size_type i = 0; i <= n + BIN_LEN; ++i)
104            list[i] = {i, i};
105        list[s] = {n, n};
106        buk[0] = {size_type(s), size_type(s)};
107        for (auto& e : sp)
108            e.dis = INF;
109        sp[s].dis = 0;
110        while (true) {
111            size_type u = buk[0].next;
112            if (u == n) {
113                size_type idx = 1;
114                for (; idx <= BIN_LEN; ++idx)
115                    if (buk[idx].next < n)
116                        break;
117                    if (idx > BIN_LEN) break;
118                    const size_type head = buk[idx].next;
119                    top = sp[head].dis;
120                    for (size_type i = head; (i = list[i].next) < n;)
121                        if (sp[i].dis < top)
122                            top = sp[i].dis;
123                    for (size_type i = head; i < n;)
124                        const size_type j = list[i].next;
125                        insert_list(radix_distance(top, sp[i].dis), i);

```

```

126             i = j;
127         }
128         buk[idx] = {idx + n, idx + n};
129         u = buk[0].next;
130     }
131     if (u == size_type(t)) return true;
132     const size_type next = list[u].next;
133     list[next].prev = n;
134     buk[0].next = next;
135     const cost_type pu = sp[u].pot;
136     const cost_type du = sp[u].dis;
137     const auto start_edge = edges + pos[u];
138     const auto end_edge = edges + pos[u + 1];
139     for (auto i = start_edge; i != end_edge; ++i) {
140         const edge_type e = *i;
141         if (e.full()) continue;
142         const size_type v = e.to();
143         const cost_type pv = sp[v].pot;
144         const cost_type dv = sp[v].dis;
145         const cost_type ndv = e.cost() + pu - pv + du;
146         if (ndv < dv) {
147             const size_type new_idx = radix_distance(top, ndv);
148             if (new_idx != radix_distance(top, dv)) {
149                 const size_type prev = list[v].prev;
150                 const size_type next = list[v].next;
151                 list[prev].next = next;
152                 list[next].prev = prev;
153                 insert_list(new_idx, v);
154             }
155             sp[v].dis = ndv;
156             sp[v].pre = e.rev();
157         }
158     }
159 }
160 return false;
161 };
162 while (dijkstra_with_heap())
163     augment();
164 };
165 const auto flow_without_heap = [&] {
166     std::vector<size_type> min_buk;
167     const auto dijkstra_without_heap = [&] {
168         vis.assign(n, false);
169         min_buk = {size_type(s)};
170         for (auto& e : sp)
171             e.dis = INF;
172         sp[s].dis = 0;
173         while (true) {
174             size_type u;
175             cost_type du;
176             if (!min_buk.empty()) {
177                 u = min_buk.back();
178                 min_buk.pop_back();
179                 if (vis[u]) continue;
180                 du = sp[u].dis;
181             } else {

```

```

182         du = INF;
183         for (size_type i = 0; i < n; ++i)
184             if (!vis[i] && sp[i].dis < du)
185                 du = sp[u = i].dis;
186             if (du == INF) break;
187         }
188         if (u == size_type(t)) return true;
189         vis[u] = true;
190         const cost_type pu = sp[u].pot;
191         const auto start_edge = edges + pos[u];
192         const auto end_edge = edges + pos[u + 1];
193         for (auto i = start_edge; i != end_edge; ++i) {
194             const edge_type e = *i;
195             if (e.full()) continue;
196             const size_type v = e.to();
197             const cost_type pv = sp[v].pot;
198             const cost_type dv = sp[v].dis;
199             const cost_type ndv = e.cost() + pu - pv + du;
200             if (ndv < dv) {
201                 sp[v].dis = ndv;
202                 sp[v].pre = e.rev();
203                 if (ndv == du) min_buk.push_back(v);
204             }
205         }
206     }
207     return false;
208 };
209 while (dijkstra_without_heap())
210     augment();
211 };
212 if (use_heap)
213     flow_with_heap();
214 else
215     flow_without_heap();
216 return std::make_pair(flow, cost);
217 }

```

本题代码：

```

1 class Solution {
2 public:
3     int connectTwoGroups(const vector<vector<int>>& cost) {
4         const int m = cost.size(), n = cost[0].size();
5         vector<int> row_min(m, INT_MAX);
6         vector<int> col_min(n, INT_MAX);
7         for (int i = 0; i < m; ++i) {
8             for (int j = 0; j < n; ++j) {
9                 row_min[i] = min(row_min[i], cost[i][j]);
10                col_min[j] = min(col_min[j], cost[i][j]);
11            }
12        }
13        MCFN<int, int> mcfn(m + n + 2);
14        const int s = m + n, t = s + 1;
15        for (int i = 0; i < m; ++i) {
16            for (int j = 0; j < n; ++j)

```

```

17         mcfn.add_edge(i, m + j, 1, min(cost[i][j] - row_min[i] -
18             col_min[j], 0));
19         mcfn.add_edge(s, i, 1, 0);
20     }
21     for (int i = 0; i < n; ++i)
22         mcfn.add_edge(m + i, t, 1, 0);
23     return reduce(row_min.begin(), row_min.end(), 0) +
24         reduce(col_min.begin(), col_min.end(), 0) + ssp(mcfn.prepare(), s,
25             t).second;
26 }
27 };

```

## 网络流

### 最大流

洛谷P3376为例

#### EK 算法

$O(nm^2)$

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #define me 5010
5 #define mn 205
6 ll n, m, s, t, u, v;
7 ll w, ans, dis[mn];
8 ll tot = 1, vis[mn], pre[mn], head[mn], flag[mn][mn];
9 struct node
10 {
11     ll to, net;
12     ll val;
13 } e[me * 2];
14 void add(ll u, ll v, ll w)
15 {
16     e[++tot] = {v, head[u], w};
17     head[u] = tot;
18 }
19 ll bfs()
20 { // bfs寻找增广路
21     for (ll i = 1; i <= n; i++)
22         vis[i] = 0;
23     queue<ll> q;
24     q.push(s);
25     vis[s] = 1;
26     dis[s] = 0x7fffffff;
27     while (!q.empty())
28     {
29         ll x = q.front();
30         q.pop();
31         for (ll i = head[x]; i; i = e[i].net)
32         {
33             if (e[i].val == 0)

```

```

34         continue; //我们只关心剩余流量>0的边
35         ll v = e[i].to;
36         if (vis[v] == 1)
37             continue; //这一条增广路没有访问过
38         dis[v] = min(dis[x], e[i].val);
39         pre[v] = i; //记录前驱, 方便修改边权
40         q.push(v);
41         vis[v] = 1;
42         if (v == t)
43             return 1; //找到了一条增广路
44     }
45 }
46 return 0;
47 }
48 void update()
49 { //更新所经过边的正向边权以及反向边权
50     ll x = t;
51     while (x != s)
52     {
53         ll v = pre[x];
54         e[v].val -= dis[t];
55         e[v ^ 1].val += dis[t];
56         x = e[v ^ 1].to;
57     }
58     ans += dis[t]; //累加每一条增广路径的最小流量值
59 }
60 signed main()
61 {
62     scanf("%lld%lld%lld%lld", &n, &m, &s, &t);
63     for (ll i = 1; i <= m; i++)
64     {
65         scanf("%lld%lld%lld", &u, &v, &w);
66         if (flag[u][v] == 0)
67             { //处理重边的操作 (加上这个模板题就可以用EK算法过了)
68                 add(u, v, w), add(v, u, 0);
69                 flag[u][v] = tot;
70             }
71         else
72         {
73             e[flag[u][v] - 1].val += w;
74         }
75     }
76     while (bfs() != 0)
77     { //直到网络中不存在增广路
78         update();
79     }
80     printf("%lld", ans);
81     return 0;
82 }

```

## Dinic算法

最坏  $O(n^2m)$ ，二分图  $O(\sqrt{nm})$ ，以下代码可解决自环

若输出方案，

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #define mn 202
5 #define me 10002
6 ll n, m, s, t, hd[mn], cnt = 1, ui, vi, wi, res, tot, dep[mn];
7 ll q[mn], lf, rf;
8 struct edge
9 {
10     ll to, v, nx;
11 } e[me];
12 bool vis[mn];
13 void adde(ll u, ll v, ll w)
14 {
15     e[++cnt] = {v, w, hd[u]};
16     hd[u] = cnt;
17 }
18 bool bfs()
19 {
20     fill_n(dep, n + 2, 0); // 总点数，与题意可能不一样
21     q[lf = rf = 1] = s;
22     dep[s] = 1;
23     while (lf <= rf)
24     {
25         ll u = q[lf++];
26         for (ll i = hd[u]; i; i = e[i].nx)
27         {
28             ll v = e[i].to;
29             if (e[i].v && !dep[v])
30                 dep[v] = dep[u] + 1, q[++rf] = v;
31         }
32     }
33     return dep[t];
34 }
35 ll dfs(ll u, ll flow)
36 {
37     if (u == t)
38         return flow;
39     ll out = 0;
40     for (ll i = hd[u]; i; i = e[i].nx)
41     {
42         ll v = e[i].to;
43         if (e[i].v && dep[v] == dep[u] + 1)
44         {
45             ll res = dfs(v, min(e[i].v, flow));
46             e[i].v -= res;
47             e[i ^ 1].v += res;
48             out += res;
49             flow -= res;
50         }
51     }
52 }
```

```

51     }
52     return !out ? (dep[u] = 0) : out;
53 }
54 signed main()
55 {
56     scanf("%lld%lld%lld%lld", &n, &m, &s, &t);
57     for (ll i = 1; i <= m; ++i)
58     {
59         scanf("%lld%lld%lld", &ui, &vi, &wi);
60         adde(ui, vi, wi), adde(vi, ui, 0);
61     }
62     while (bfs())
63         tot += dfs(s, 1e18);
64     printf("%lld", tot);
65     return 0;
66 }

```

## 最小费用最大流

洛谷P3381-输入边及其流量限制、单位流量费用；输出最大流以及在最大流量时的最小费用(无负费用) ( $1 \leq w, c \leq 10^3, 1 \leq n \leq 5 \times 10^3, 1 \leq m \leq 5 \times 10^4$ )

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef int ll;
4 #define MAXN 100002
5 bool vis[MAXN];
6 ll n, m, s, t, x, y, z, f, dis[MAXN], pre[MAXN], last[MAXN], flow[MAXN];
7 ll mxflow, micost, cnt, hd[MAXN];
8 struct edge
9 {
10     ll to, nx, f, d;
11 } e[MAXN];
12 queue<ll> q;
13 void adde(ll &u, ll &v, ll f, ll d)
14 {
15     e[++cnt] = {v, hd[u], f, d};
16     hd[u] = cnt;
17 }
18 bool spfa(ll s, ll t)
19 {
20     memset(dis, 0x7f, sizeof dis);
21     memset(flow, 0x7f, sizeof flow);
22     memset(vis, 0, sizeof vis);
23     q.push(s);
24     vis[s] = true, dis[s] = 0, pre[t] = -1;
25     while (!q.empty())
26     {
27         ll no = q.front();
28         q.pop();
29         vis[no] = false;
30         for (ll i = hd[no]; i != -1; i = e[i].nx)
31         {

```

```

32         ll toi = e[i].to;
33         if (e[i].f > 0 && dis[toi] > dis[no] + e[i].d)
34         {
35             dis[toi] = dis[no] + e[i].d;
36             pre[toi] = no;
37             last[toi] = i;
38             flow[toi] = min(e[i].f, flow[no]);
39             if (!vis[toi])
40                 vis[toi] = true, q.push(toi);
41         }
42     }
43 }
44 return pre[t] != -1;
45 }
46 void mcmf()
47 {
48     while (spfa(s, t))
49     {
50         ll no = t;
51         mxflow += flow[t];
52         micost += flow[t] * dis[t];
53         while (no != s)
54         {
55             e[last[no]].f -= flow[t];
56             e[last[no] ^ 1].f += flow[t];
57             no = pre[no];
58         }
59     }
60 }
61 signed main()
62 {
63     memset(hd, -1, sizeof hd);
64     cnt = -1; //因为有异或操作, 从0开始, 否则TLE
65     scanf("%d%d%d%d", &n, &m, &s, &t);
66     for (ll i = 1; i <= m; ++i)
67     {
68         scanf("%d%d%d%d", &x, &y, &z, &f);
69         adde(x, y, z, f);
70         adde(y, x, 0, -f);
71     }
72     mcmf();
73     printf("%d %d", mxflow, micost);
74     return 0;
75 }

```

牛客2022暑假集训10-E 有  $n$  ( $1 \leq n \leq 400$ ) 人  $m$  ( $1 \leq m \leq 400$ ) 稿。若  $a_{ij} = 1$  则第  $i$  人可读第  $j$  稿。每人只能读一篇稿。问是否存在分配方案使得每人都能读稿(不能输出  $-1$ )，在此前提下输出使得最大化每篇稿被读次数的方案。

最小费用最大流。不考虑费用时显然源点向每个人连一条流量为 1 的边，每个人向他能读的稿连流量为 1 的边，而每篇稿子向汇点连流量为入度的边。显然，题意无解即有人没读到，即流量不为  $n$ 。所有有解都是没人读一篇稿，接下来就是稿的分配了，要使得每稿尽可能多人。但这样建图只能最大化流量，即使得总流量最大，而总流量最大不能保证最小流量最大。为此，将流向汇点的流量为入度的边拆分为若干条流量为 1，代价递增的边。这样可以在总流量最大的情况下，让分配更加地平均。

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 using ll = long long;
4
5 template <const int mn, class ty = int>
6 struct MinCostMaxFlow
7 {
8     const ty inf = numeric_limits<ty>::max();
9     struct edge
10    {
11         int v;
12         ty cap, cost;
13         int rev;
14     };
15     vector<edge> g[mn];
16     ty sum{}, dis[mn]{};
17     int n, s, t, aug[mn]{};
18     ty flow{}, cost{};
19     bool augment()
20    {
21         fill(dis, dis + n + 1, inf);
22         priority_queue<pair<int, int>> q;
23         dis[t] = 0;
24         q.push({dis[t], t});
25         for (int u, v; !q.empty() && q.top().first != inf;)
26         {
27             u = q.top().second;
28             q.pop();
29             for (auto e : g[u])
30             {
31                 v = e.v;
32                 if (g[v][e.rev].cap && dis[v] > dis[u] - e.cost)
33                 {
34                     dis[v] = dis[u] - e.cost;
35                     q.push({dis[v], v});
36                 }
37             }
38         }
39         sum += dis[s];
40         for (int u = 1; u <= n; ++u)
41         {
42             for (auto &e : g[u])
43             {
44                 e.cost += dis[e.v] - dis[u];
45             }
46         }
47         return dis[s] != inf;
48     }
49
50     ty dfs(int u, ty lim)
51     {
52         if (u == t)
53         {
54             cost += lim * sum;
55             return lim;
56         }

```

```

57     ty f = 0;
58     aug[u] = 1;
59     for (auto &e : g[u])
60     {
61         int v = e.v;
62         if (!e.cost && e.cap && !aug[v])
63         {
64             ty t = dfs(v, min(lim - f, e.cap));
65             e.cap -= t;
66             g[v][e.rev].cap += t;
67             f += t;
68             if (f == lim)
69             {
70                 break;
71             }
72         }
73     }
74     if (f == lim)
75     {
76         aug[u] = 0;
77     }
78     return f;
79 }
80
81 void add(int u, int v, ty cap, ty cost)
82 {
83     g[u].push_back({v, cap, cost, (int)g[v].size()});
84     g[v].push_back({u, 0, -cost, (int)g[u].size() - 1});
85 }
86
87 pair<ty, ty> solve(int _n, int _s, int _t)
88 {
89     n = _n, s = _s, t = _t, flow = cost = 0;
90     do
91     {
92         ty f;
93         do
94         {
95             memset(aug, 0, (n + 1) << 2);
96             f = dfs(s, inf);
97             flow += f;
98             } while (f > 0);
99         } while (augment());
100     return {flow, cost};
101 }
102 };
103
104 const ll mn = 405;
105 MinCostMaxFlow<mn * 2, int> d;
106 signed main()
107 {
108     cin.tie(0)->ios::sync_with_stdio(false);
109     int n, m;
110     char a[mn];
111     cin >> n >> m;
112     int s = n + m + 1, t = s + 1;

```

```

113     for (ll i = 1; i <= n; ++i)
114     {
115         cin >> (a + 1);
116         d.add(s, i, 1, 0);
117         for (ll j = 1; j <= m; ++j)
118         {
119             if (a[j] == '1')
120             {
121                 d.add(i, n + j, 1, 0);
122             }
123         }
124     }
125     for (ll i = 1; i <= m; ++i)
126     {
127         for (ll j = 1; j <= n; ++j)
128         {
129             d.add(n + i, t, 1, j);
130         }
131     }
132     auto f = d.solve(t, s, t);
133     if (f.first != n)
134     {
135         cout << -1;
136         return 0;
137     }
138     for (ll i = 1; i <= n; ++i)
139     {
140         for (auto e : d.g[i])
141         {
142             if (!e.cap && e.v > n)
143             {
144                 cout << (e.v - n) << ' ';
145                 break;
146             }
147         }
148     }
149     return 0;
150 }
```

## 最小割

SCNUOJ1975-给定  $n, m (2 \leq n \leq 10^n, 1 \leq m \leq 10^5)$  的有向简单连通图, 保证点 1 对  $n$  可达且任意  $1 < x < n$  出入度均不为零。求删掉最少多少边使得 1,  $n$  不连通并输出方案。

割将原点和汇点分在两个不同点集的点划分方案, 容量为分属两点集的边的数目。本题显然是最小割模板。

最大流最小割定理: 能够从源点到达汇点的最大流量等于从网络中移除导致网络流中断的边集的最小容量和。

Dinic  $O(n\sqrt{m})$ , 残余网络搜索有流量的可达边集及其反集, 连接两集合的边可删。

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 const int N = 1e5 + 5, M = 2e5 + 5;
```

```

5  int n, m, s, t, tot = 1, lnk[N], ter[M], nxt[M], val[M], dep[N], cur[N];
6
7  void add(int u, int v, int w)
8  {
9      ter[++tot] = v, nxt[tot] = lnk[u], lnk[u] = tot, val[tot] = w;
10 }
11
12 void addedge(int u, int v, int w) { add(u, v, w), add(v, u, 0); }
13
14 int bfs(int s, int t)
15 {
16     memset(dep, 0, sizeof(dep));
17     memcpy(cur, lnk, sizeof(lnk));
18     std::queue<int> q;
19     q.push(s), dep[s] = 1;
20     while (!q.empty())
21     {
22         int u = q.front();
23         q.pop();
24         for (int i = lnk[u]; i; i = nxt[i])
25         {
26             int v = ter[i];
27             if (val[i] && !dep[v])
28                 q.push(v), dep[v] = dep[u] + 1;
29         }
30     }
31     return dep[t];
32 }
33
34 int dfs(int u, int t, int flow)
35 {
36     if (u == t)
37         return flow;
38     int ans = 0;
39     for (int &i = cur[u]; i && ans < flow; i = nxt[i])
40     {
41         int v = ter[i];
42         if (val[i] && dep[v] == dep[u] + 1)
43         {
44             int x = dfs(v, t, std::min(val[i], flow - ans));
45             if (x)
46                 val[i] -= x, val[i ^ 1] += x, ans += x;
47         }
48     }
49     if (ans < flow)
50         dep[u] = -1;
51     return ans;
52 }
53
54 int dinic(int s, int t)
55 {
56     int ans = 0;
57     while (bfs(s, t))
58     {
59         int x;
60         while ((x = dfs(s, t, 1 << 30)))

```

```

61         ans += x;
62     }
63     return ans;
64 }
65
66 int vis[N];
67 void dfs(int u)
68 {
69     vis[u] = 1;
70     for (int i = lnk[u]; i; i = nxt[i])
71     {
72         int v = ter[i];
73         if (!vis[v] && val[i])
74             dfs(v);
75     }
76 }
77
78 int main()
79 {
80     scanf("%d%d", &n, &m);
81     s = 1, t = n;
82     while (m--)
83     {
84         int u, v, w = 1;
85         scanf("%d%d", &u, &v);
86         addedge(u, v, w);
87     }
88     printf("%d\n", dinic(s, t));
89     dfs(s);
90     for (int u = 1; u <= n; u++)
91     {
92         for (int i = lnk[u]; i; i = nxt[i])
93         {
94             int v = ter[i];
95             if (vis[u] == 1 && vis[v] == 0 && val[i] == 0)
96             {
97                 printf("%d %d\n", u, v);
98             }
99         }
100    }
101    return 0;
102 }

```

## 杂项

### 欧拉图

欧拉路径：经过每条边一次且仅一次的路径(起点不为终点)

欧拉回路：经过每条边一次且仅一次的回路

有向图欧拉路径：恰好存在一个点出度比入度多一(起点)，一个点入度比出度多一(终点)，其他点入度与出度相同

有向图欧拉回路：所有点入度=出度，起点和终点任选

无向图欧拉路径：恰好存在 2 点度数是奇数(分别是起点终点)，其他点度数是偶数

无向图欧拉回路：所有点度数是偶数(起点终点任选)

若存在欧拉回路也一定存在欧拉路径

此外，还应该判定图是否是连通图(满足上面条件也有非连通图)显然可以用并查集来做。

具有欧拉回路的无向或有向图是欧拉图；具有欧拉通路而不具有回路的是半欧拉图。(直观理解回路即任意点开始可以一笔画完图；通路只有特定点开始可以)

洛谷 P7771 输出  $n, m$  有向图的最小字典序欧拉路径，若不存在输出 No  
 $1 \leq n \leq 10^5, 1 \leq m \leq 2 \times 10^5$

下面代码模板没有判定联通，要判并查集/tarjan即可

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 #define mn 100010
4 typedef long long ll;
5 ll n, m, ru[mn], cu[mn], s = 1, fail, vis[mn], bgs, eds;
6 vector<ll> e[mn];
7 stack<ll> ans;
8 void dfs(ll u)
9 {
10     for (ll i = vis[u]; i < e[u].size(); i = vis[u])
11     { //不i++是可能被后续dfs更新过
12         vis[u] = i + 1;
13         dfs(e[u][i]);
14     }
15     ans.push(u);
16 }
17 signed main()
18 {
19     scanf("%lld%lld", &n, &m);
20     for (ll i = 0, u, v; i < m; ++i)
21     {
22         scanf("%lld%lld", &u, &v);
23         ++cu[u], ++ru[v];
24         e[u].emplace_back(v);
25     }
26     for (ll i = 1; i <= n; ++i)
27     {
28         if (cu[i] == ru[i] + 1)
29         {
30             s = i;
31             ++bgs;
32         }
33         else if (ru[i] == cu[i] + 1)
34         {
35             ++eds;
36         }
37         if (ru[i] != cu[i])
38         {
39             ++fail;
40         }
41     }
42 }
```

```

40     }
41     sort(e[i].begin(), e[i].end());
42 }
43 if (!(fail == 0 || (fail == 2 && bgs == 1 && eds == 1)))
44 {
45     printf("No");
46     return 0;
47 }
48 dfs(s);
49 while (!ans.empty())
50 {
51     printf("%lld ", ans.top());
52     ans.pop();
53 }
54 return 0;
55 }

```

P2731洛谷-给定无向图边数  $m$ ，点在  $[1, 500]$  编号，必然有解且联通，输出字典序最小欧拉路

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #define mn 512
5 ll m, s = 1, g[mn][mn], du[mn], n;
6 stack<ll> ans;
7 void dfs(ll u)
8 {
9     for (ll v = 1; v <= n; ++v)
10    {
11        if (g[u][v] > 0)
12        {
13            --g[u][v], --g[v][u];
14            dfs(v);
15        }
16    }
17    ans.push(u);
18 }
19 signed main()
20 {
21     scanf("%lld", &m);
22     for (ll i = 0, u, v; i < m; ++i)
23    {
24        scanf("%lld%lld", &u, &v);
25        ++du[u], ++du[v], ++g[u][v], ++g[v][u];
26        n = max(n, max(u, v));
27    }
28    for (ll i = 1; i <= n; ++i)
29    {
30        if (du[i] & 1)
31        {
32            s = i;
33            break;
34        }
35    }
36    dfs(s);

```

```

37     while (!ans.empty())
38     {
39         printf("%lld\n", ans.top());
40         ans.pop();
41     }
42     return 0;
43 }
```

## 哈密顿图

哈密顿通路：经过图中所有顶点一次且仅一次的通路

哈密顿回路：经过图中所有顶点一次且仅一次的回路

具有哈密顿回路的图为哈密顿图

具有哈密顿通路，不具有哈密顿回路的图为半哈密顿图

设  $p(x)$  是图  $x$  的连通分支数，对哈密顿图任意非空子集  $V_1$ ，均有  $p(G - V_1) \leq |V_1|$ 。对半哈密顿图有  $p(G - V_1) \leq |V_1| + 1$

完全图  $K_{2m+1}(m \geq 1)$  含  $m$  条边不重的哈密顿回路，且加起来含完全图中所有边。完全图  $K_{2m}(m \geq 2)$  含  $m - 1$  条边不重的哈密顿回路，删掉全部回路后所得图含  $m$  条互不相邻边。

充分条件：对无向简单图  $G(|V| \geq 2)$ ，若任意不相邻顶点  $v_i, v_j$  度数满足  $d(v_i) + d(v_j) \geq n - 1$  则存在哈密顿通路。若  $|V| \geq 3$  且满足  $d(v_i) + d(v_j) \geq n$  则存在哈密顿回路。若任意顶点均有

$d(v_i) \geq \frac{n}{2}$  则存在哈密顿回路。

$n \geq 2$  阶竞赛图具有哈密顿通路，含竞赛图子图也具有哈密顿通路。强联通竞赛图有哈密顿回路。含强联通竞赛图作为子图具有哈密顿回路。

已经证明“判断一个图是否存在哈密顿回路”是NPC。

## 竞赛图

兰道定理：判定竞赛图。设比分序列是竞赛图出度排序后的序列  $s$ ，序列能是竞赛图当且仅当

$$\forall 1 \leq k \leq n, \sum_{i=1}^k s_i \geq \binom{k}{2}$$

且  $k = n$  时必须取等。注意并不是说  $s$  每个还原图都是竞赛图，只是证明存在至少一种正确。

竞赛图如果存在环，至少是三元环。

## 动态规划

### 模板

### 背包问题

## 01背包：

```
1 for (int i = 1; i <= n; i++) //w容量 v价值
2     for (int l = w; l >= w[i]; l--) f[l] = max(f[l], f[l - w[i]] + v[i]);
```

方案数：(洛谷P1164)

```
1 dp[0]=1;
2 for(int i=0;i<n;++i)
3     for(int j=m-a[i];j>=0;--j)
4         dp[j+a[i]] += dp[j];
```

## 完全背包：(可以选取无限次) (将01背包第二维修改顺序即可)

```
1 for (int l = w[i]; l <= w; l++)
```

**多重背包：**(每种物品有  $k_i$  个) 将  $k_i$  拆分为  $2^i$ ，然后用 01 背包解决

**二维费用背包：**(有两个重量维度) (加一层循环, 加一维数组枚举, 同 01 背包解决)

**分组背包：**(每组内物品冲突, 只能选组内一个)

```
1 for (int k = 1; k <= ts; k++)           // 循环每一组
2     for (int i = m; i >= 0; i--)           // 循环背包容量
3         for (int j = 1; j <= cnt[k]; j++) // 循环该组的每一个物品
4             if (i >= w[t[k][j]])
5                 dp[i] = max(dp[i],
6                             dp[i - w[t[k][j]]] + c[t[k][j]]); // 像0-1背包一样状态转移
```

**依赖背包：**将依赖关系转化为分组关系, 用分组背包求

## 字符DP

atcoder-contest122-d 求有多少长为  $n$ ，只含字符 `ACGT` 且满足 ①无子串 `AGC` ②交换相邻两位仍无子串 `AGC`；对  $10^9 + 7$  取模

设  $dp[h][i][j][k]$  是长为  $h$  且最后一、二、三个字符分别为  $i, j, k$  的字符串的答案  $O(n4^4)$

```
1 #include <iostream>
2 #define maxn 105
3 using namespace std;
4 typedef long long ll;
5
6 ll dp[maxn][4][4][4];
7 const ll MOD = 1e9 + 7;
8 bool check(int j, int k, int v)
9 {
10     return !(j == 0 && k == 2 && v == 1);
11 }
12 bool check2(int j, int k, int v)
```

```

13 {
14     // 0->a, 1->c, 2->g, 3->t
15     if (!((j == 0 && k == 2 && v == 1) || (j == 0 && k == 1 && v == 2) || (j
16 == 2 && k == 0 && v == 1)))
17         return true;
18     return false;
19 }
20
21 int main()
22 {
23     int n;
24     cin >> n;
25
26     for (int j = 0; j < 4; j++)
27     {
28         for (int k = 0; k < 4; k++)
29         {
30             for (int v = 0; v < 4; v++)
31             {
32                 if (check2(j, k, v))
33                     dp[3][j][k][v] = 1;
34             }
35         }
36
37         for (int i = 4; i <= n; i++)
38         {
39             for (int j = 0; j < 4; j++)
40             {
41                 for (int k = 0; k < 4; k++)
42                 {
43                     for (int v = 0; v < 4; v++)
44                     {
45                         for (int u = 0; u < 4; u++)
46                             if (check2(j, k, v) && check(u, k, v) && check(u, j,
47 v) && check(u, j, k))//最后一个check没有也行; 因为这种情况下原dp[i-1][u][j][k]就是0
48                                 dp[i][j][k][v] = (dp[i][j][k][v] + dp[i - 1][u]
49 [j][k]) % MOD;
50                     }
51                 }
52             }
53
54             ans = 0;
55             for (int i = 0; i < 4; i++)
56                 for (int j = 0; j < 4; j++)
57                     for (int k = 0; k < 4; k++)
58                         ans = (ans + dp[n][i][j][k]) % MOD;
59
60             cout << ans << endl;
61             return 0;
62 }

```

## 树上DP

$n$  点有点权树, 选当前节点就不能选全部直接子节点, 求选择方案使得点权和最大, 输出最大值(洛谷 P1352)

设  $dp[i][j]$ ,  $j = 0$  表示不选  $i$  时  $i$  子树最大值;  $j = 1$  表示选  $i$  时  $i$  子树最大值

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 #define MAXN 6002
4 struct edge
5 {
6     int to, nx;
7 } e[MAXN];
8 int n, ui, vi, hd[MAXN], cnt, dp[MAXN][2], v[MAXN], root;
9 bool fa[MAXN];
10 inline void adde(int &u, int &v)
11 {
12     e[++cnt].to = v;
13     e[cnt].nx = hd[u];
14     hd[u] = cnt;
15 }
16 void dfs(int h)
17 {
18     dp[h][0] = 0;
19     dp[h][1] = v[h];
20     for (int i = hd[h]; i; i = e[i].nx)
21     {
22         int toi = e[i].to;
23         dfs(toi);
24         dp[h][0] += max(dp[toi][0], dp[toi][1]);
25         dp[h][1] += dp[toi][0];
26     }
27 }
28 signed main()
29 {
30     scanf("%d", &n);
31     for (int i = 1; i <= n; ++i)
32         scanf("%d", &v[i]);
33     for (int i = 1; i < n; ++i)
34     {
35         scanf("%d%d", &vi, &ui);
36         adde(ui, vi);
37         fa[vi] = true;
38     }
39     for (int i = 1; i <= n; ++i)
40         if (!fa[i])
41         {
42             root = i;
43             break;
44         }
45     dfs(root);
46     printf("%d", max(dp[root][0], dp[root][1]));
47     return 0;
48 }
```

## 区间DP

区间 DP 的复杂度是  $O(n^3)$

洛谷P1880-(a,b)合并获得值a+b, 若干石子成环(数大于等于0)的最大最小值

```
1 #include <bits/stdc++.h>
2 #define MAXN 203
3 const int BIG = 0xffffffff;
4 using namespace std;
5 int mi, mx, n, dpmi[MAXN][MAXN], dpmx[MAXN][MAXN], s[MAXN], a[MAXN], kd;
6 signed main()
7 {
8     scanf("%d", &n);
9     for (int i = 1; i <= n; ++i)
10    {
11        scanf("%d", &a[i]);
12        a[i + n] = a[i];
13    }
14    for (int i = 1; i < n + n; ++i)
15    {
16        s[i] = s[i - 1] + a[i];
17    }
18    for (int h = 1; h < n; ++h)
19    {
20        for (int i = 1, j = i + h; i < n + n && j < n + n; i++, j = i + h)
21        {
22            dpmi[i][j] = BIG;
23            kd = s[j] - s[i - 1];
24            for (int k = i; k < j; ++k)
25            {
26                dpmi[i][j] = min(dpmi[i][j], dpmi[i][k] + dpmi[k + 1][j] +
27 kd);
28                dpmx[i][j] = max(dpmx[i][j], dpmx[i][k] + dpmx[k + 1][j] +
29 kd);
30            }
31            mi = BIG;
32            for (int i = 1; i <= n; ++i)
33            {
34                mx = max(mx, dpmx[i][i + n - 1]);
35                mi = min(mi, dpmi[i][i + n - 1]);
36            }
37            printf("%d\n%d", mi, mx);
38        }
39    }
}
```

P1063洛谷-如果前一颗能量珠的头标记为m, 尾标记为r, 后一颗能量珠的头标记为r, 尾标记为n, 则聚合后释放的能量为 $m \times r \times n$ , 新产生的珠子的头标记为m, 尾标记为n。给出成环的n和各个标记, 求最大能量

```
1 #include <bits/stdc++.h>
```

```

2 #define MAXN 205
3 #define pr pair<int, int>
4 using namespace std;
5 int n, e, a[MAXN][MAXN], b[MAXN], ans;//a为[i,j]内答案
6 signed main()
7 {
8     scanf("%d", &n);
9     for (int i = 1; i <= n; ++i)
10    {
11        scanf("%d", &b[i]);
12        b[i + n] = b[i];
13    }
14    for (int i = 2; i <= n + 1; ++i)//区间长
15    {
16        for (int lf = 1, rf = lf + i - 1; rf <= n + n; ++lf, ++rf)
17        {
18            for (int k = lf + 1; k <= rf - 1; ++k)
19            {
20                a[lf][rf] = max(a[lf][rf], a[lf][k] + a[k][rf] + b[lf] *
b[k] * b[rf]);
21            }
22        }
23    }
24    for (int i = 1; i <= n; ++i) ans = max(ans, a[i][i + n]);
25    printf("%d", ans);
26    return 0;
27 }

```

## 状压DP

枚举每个二进制状态，并降序枚举每个状态的子集状态，复杂度为  $O(3^n)$

```

1 for (int m = 0; m < (1 << n); ++m)// 降序遍历 m 的非空子集
2     for (int s = m; s; s = (s - 1) & m) // s 是 m 的一个非空子集

```

洛谷P1896-在  $n \times n$  格子放  $k$  人，每人周围八方向不得有人，求方案数

$$1 \leq n \leq 9, 0 \leq k \leq n^2$$

设每一行的状态为  $j$ ，即  $n$  个 0/1 组合的  $2^n$  个状态，设  $dp[i][j][k]$  表示只考虑前  $i$  行，第  $i$  行的分布为  $j$ ，已经放置了  $k$  个人的方案数；先暴力处理一行内所有可能的情况(某状态对应多少个人)

复杂度  $O(2^n + n2^{2n})$

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #define repe(i,a,b) for(ll i=a;i<=b;++i)
5 #define MAXN 514//2^n
6 #define N 12//dp的第三维度(从左往右数)定11或以下会WA
7 ll sit[MAXN], gs[MAXN], cnt, n, k, dp[N][MAXN][N], ans;
8 void dfs(ll h, ll sum, ll po)
9 { //现在是状态h,有sum个人,现在处理第po位

```

```

10  if (po >= n)//处理完了
11  { //一行内所有可能的状态(编号)数量为cnt;cnt<2^n,优化后续DP
12      sit[++cnt] = h;//该编号对应二进制状态h
13      gs[cnt] = sum;//该编号有多少个人(1数目)
14      return;
15  }
16  dfs(h, sum, po + 1); //该位选与不选
17  dfs(h + (1 << (int)po), sum + 1, po + 2);
18 }
19 signed main()
20 {
21     scanf("%lld%lld", &n, &k);
22     dfs(0, 0, 0);
23     repe(i, 1, cnt) dp[1][i][gs[i]] = 1;
24     repe(i, 2, n) repe(j, 1, cnt) repe(a, 1, cnt)
25     { //上下//左上右下//右上左下有人冲突
26         if (sit[j] & sit[a] || (sit[j] << 1) & sit[a] || sit[j] & (sit[a] << 1)) continue;
27         for (ll s=k; s>=gs[j]; --s) dp[i][j][s] += dp[i - 1][a][s - gs[j]];
28     }
29     repe(i, 1, cnt) ans += dp[n][i][k];
30     printf("%lld", ans);
31     return 0;
32 }

```

## 数位DP

lc233-问  $[1, n]$  ( $n \leq 10^9$ ) 的整数出现了多少个 1

$f$  是从左到右构造好了  $i$  个位, 造好里已经出现了  $cnt_1$  个 1, 当前位是否被  $n$  限制。 $dp$  是已造好  $i$  个位, 出现了  $j$  个 1 的答案。 $O(\log^2 n)$ 。

```

1 const int M = 15;
2 int countDigitOne(int n) {
3     auto s = to_string(n);
4     int m = s.length();
5     int dp[M][M];
6     memset(dp, -1, sizeof(dp));
7     function<int(int, int, bool)> f = [&](int i, int cnt1, bool is_limit) ->
8     int {
9         if (i == m) return cnt1;
10        if (!is_limit && dp[i][cnt1] >= 0) return dp[i][cnt1];
11        int res = 0;
12        for (int d = 0, up = is_limit ? s[i] - '0' : 9; d <= up; ++d) // 枚举
13            // 要填入的数字 d
14            res += f(i + 1, cnt1 + (d == 1), is_limit && d == up);
15        if (!is_limit) dp[i][cnt1] = res;
16        return res;
17    };
18    return f(0, 0, true);
19 }

```

牛客2022寒假第五场D-求区间  $[l, r]$  内满足如下条件的数字个数①相邻两数位和是素数 ②至少有一个数位1 ③无前导0  $1 \leq l \leq r \leq 10^{10}$

设  $dp[i][j][k]$  表示从个位开始数的  $i$  位，最大的位数字是  $j$ ， $k$  表示是否出现过 1，相邻数位和都是数组的方案数，即  $i$  位数区间  $[j0 \cdots 0, j9 \cdots 9]$  范围

根据定义，初始化为  $dp[1][1][1] = 1$ ， $dp[1][j][0] = 1 (0 \leq j \leq 9, j \neq 1)$ ，表示十个个位数的情况

递推方程为若  $j + j'$  是素数，那么  $k' = k \cup k' \cup j == 1 \cup j' == 1$  时可以加上去(具体看代码)

特别地，设  $dd[i]$  表示区间  $[1, 10^{i-1} - 1]$  的答案，则：

$$dd[i] = dd[i - 1] + \sum_{j=1}^9 dp[i - 1][j][1]$$

对  $[1, r]$  的区间内的答案计数，对  $r = \overline{a_k \cdots a_2 a_1}$ ，先用  $dd[k]$  计数前  $k$  位数的答案，然后讨论  $k$  位数答案，对第  $i$  位， $> i$  的位都取  $r$  值即  $a_h (h > i)$ ，第  $i$  位进行枚举  $[0, a_i)$ ，得到一个范围内的答案；注意若  $> i$  时出现过 1，则  $dp$  第三维度任取，否则只能取 1

答案为  $[1, r]$  内答案减去  $[1, l - 1]$  内答案

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 typedef double db;
5 #define sc(x) scanf("%lld", &x)
6 ll dp[12][12][2]; //后i位,最高位为j,是否出现过1,相邻数位和为素数的方案数
7 ll prime[] = {2, 3, 5, 7, 11, 13, 17}, ps = 7;
8 ll l, r, res;
9 ll dd[12]; // [1, 10^(i-1)-1]内的答案
10 ll ispr(ll x)
11 {
12     for (ll i = 0; i < ps; ++i)
13     {
14         if (x == prime[i])
15         {
16             return 1;
17         }
18     }
19     return 0;
20 }
21 ll ask(ll rf)
22 {
23     res = 0;
24     if (rf <= 9 && rf >= 1)
25     {
26         return 1;
27     }
28     ll bs = 0;
29     ll b[12] = {};
30     for (ll i = rf; i; i /= 10)
31     {
32         ++bs;
33         b[bs] = i % 10;
34     }
}
```

```

35     ll ever1 = 0, ever2 = 0;
36     for (ll i = bs; i >= 1; --i)
37     {
38         ll je = b[i] + (i == 1);
39         if (je > 0 && i > 1 && !ever2)
40         {
41             res += dd[i];
42             ever2 = 1;
43         }
44         ll jb = 1 - (i != bs);
45         for (ll j = jb; j < je; ++j)
46         {
47             ll k = (i > 1) ? 1 : ((j == 1) ? 1 : !ever1);
48             if (i != bs && !ispr(b[i + 1] + j))
49             {
50                 continue;
51             }
52             res += dp[i][j][k];
53             if (ever1)
54             {
55                 res += dp[i][j][k ^ 1];
56             }
57         }
58         ever1 |= (b[i] == 1);
59         if (i != bs && !ispr(b[i] + b[i + 1]))
60         {
61             break;
62         }
63     }
64     return res;
65 }
66 ll bfcheck(ll lf, ll rf)
67 {
68     ll cnt = 0;
69     for (ll h = lf; h <= rf; ++h)
70     {
71         ll has1 = 0;
72         ll bs = 0;
73         ll b[12] = {};
74         for (ll j = h; j; j /= 10)
75         {
76             ++bs;
77             b[bs] = j % 10;
78             has1 |= (b[bs] == 1);
79         }
80         if (bs == 1)
81         {
82             cnt += (h == 1);
83             continue;
84         }
85         ll ok = 1;
86         for (ll i = 1; i < bs; ++i)
87         {
88             ll isprime = ispr(b[i] + b[i + 1]);
89             if (isprime == 0)
90             {

```

```

91         ok = 0;
92         break;
93     }
94 }
95 if (ok && has1)
96 {
97     ++cnt;
98 }
99 }
100 return cnt;
101 }
102 signed main()
103 {
104     dp[1][1][1] = 1;
105     for (ll i = 2; i <= 9; ++i)
106     {
107         dp[1][i][0] = 1; //没有相邻, 条件1恒成立
108     }
109     dp[1][0][0] = 1;
110     for (ll i = 2; i <= 11; ++i)
111     {
112         for (ll x = 0; x <= 9; ++x)
113         {
114             ll now0 = (x == 1);
115             for (ll y = 0; y <= 9; ++y)
116             {
117                 for (ll pr = 0; pr <= 1; ++pr)
118                 {
119                     ll has0 = now0 || ((y == 1) || pr);
120                     ll s = x + y;
121                     ll isprime = 0;
122                     for (ll j = 0; j < ps; ++j)
123                     {
124                         if (s == prime[j])
125                         {
126                             isprime = 1;
127                             break;
128                         }
129                     }
130                     if (!isprime)
131                     {
132                         continue;
133                     }
134                     dp[i][x][has0] += dp[i - 1][y][pr];
135                 }
136             }
137         }
138     }
139     for (ll i = 2; i <= 11; ++i)
140     {
141         dd[i] = dd[i - 1];
142         for (ll j = 1; j <= 9; ++j)
143         {
144             dd[i] += dp[i - 1][j][1];
145         }
146     }

```

```

147     sc(l), sc(r);
148     ll ar = ask(r), al = ask(l - 1);
149     printf("%lld", ar - al);
150     return 0;
151 }

```

## 杂项

### 最长公共子序列

LCS, 如abcbdbab,bdcaba的最长子序列是bcba

```

1 int c[N+1][N+1];//c[i][j]代表Xi与Yj的LCS长度
2 int lcs(string X, string Y)
3 {
4     int m = X.size();
5     int n = Y.size();
6     int maxl = 0;
7     X = ' ' + X;
8     Y = ' ' + Y;
9     for(int i=1;i<=m;i++) c[i][0] = 0;
10    for(int j=1;j<=n;j++) c[0][j] = 0;
11    for(int i=1;i<=m;i++)
12    {
13        for(int j=1;j<=n;j++)
14        {
15            if(X[i]==Y[j]) c[i][j] = c[i-1][j-1] + 1;
16            else c[i][j] = max(c[i-1][j], c[i][j-1]);
17            maxl = max(maxl, c[i][j]);
18        }
19    }
20    return maxl;
21 }

```

### 编辑距离

lc72-有字符串x,y; 每次只能进行如下编辑的一种：删除/插入/修改一个字符，求最小编辑次数，使x=y。

定义 $dp[i][j]$ 表示将长为i的x子串编辑为长为j的y子串的编辑距离。

```

1 int minDistance(string s, string t) {
2     int n = s.length(), m = t.length(), f[2][m + 1];
3     for (int j = 0; j <= m; ++j) f[0][j] = j;
4     for (int i = 0; i < n; ++i) {
5         f[(i + 1) % 2][0] = i + 1;
6         for (int j = 0; j < m; ++j)
7             f[(i + 1) % 2][j + 1] = s[i] == t[j] ? f[i % 2][j] : min(min(f[i
8                 % 2][j + 1], f[(i + 1) % 2][j]), f[i % 2][j]) + 1;
9     }
10    return f[n % 2][m];
}

```

## 约瑟夫问题

$n$  人从 1 开始编号排列成环，每次数到第  $q$  个人杀掉，求最后活着的人编号

当  $q = 2$  时， $n = 2^k, J = 1$ ，否则  $J(2^k + t) = 2t + 1$  (做掉  $t$  人后，变成了一个  $2^k$  环，在这个  $2^k$  环上，最后活下来的是环的第一个人)

一般情况(设编号从 0 开始):  $J_{n,q} = (J_{n-1,q} + q) \bmod n$ ，初始状态为  $J_{1,q} = 0$

## 字符串

### 字符串哈希

$$h(x) = \begin{cases} h(1) = s_1 \\ h(i) = h(i-1) \times p + s_i \quad , i > 1 \end{cases} \bmod 2^{64}$$

$$h(x) = \sum_{i=1}^x s_i \cdot p^{x-i} = s_1 p^{x-1} + s_2 p^{x-2} + \dots + s_x p^0 \bmod 2^{64}$$

对子串  $S[l..r]$ ，有  $h(S[l..r]) = h(S[1..r]) - h(S[1..l-1]) \times p^{r-l+1}$

以字符串匹配为例 SCNUOJ1713 求  $s$  里  $t$  的每个出现位置下标

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 typedef unsigned long long ull;
5 #define mn 2000010
6 char s[mn], t[mn];
7 ull p = 131, pw[mn], h[mn], ht, n, m, cnt;
8 signed main()
{
10    pw[0] = 1;
11    for (ll i = 1; i < mn; ++i)
12    {
13        pw[i] = pw[i - 1] * p;
14    }
15    scanf("%s%s", s, t);
16    n = strlen(s), m = strlen(t);
17    for (ull i = 0; i < n; ++i)

```

```

18     {
19         h[i + 1] = h[i] * p + s[i];
20     }
21     for (ull i = 0; i < m; ++i)
22     {
23         ht = ht * p + t[i];
24     }
25     for (ull lf = 1, rf = m; rf <= n; ++lf, ++rf)
26     {
27         if (h[rf] - h[lf - 1] * pw[rf - lf + 1] == ht)
28         {
29             printf("%lld ", lf), ++cnt;
30         }
31     }
32     if (!cnt)
33     {
34         printf("-1"); // not found
35     }
36     return 0;
37 }

```

## KMP

$\pi(x)$  是长为  $x$  的子串最长真前缀与真后缀相等的长度；无解为 0

以字符串匹配为例 SCNUOJ1713

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #define mn 2000010
5 char s[mn], t[mn];
6 ll kmp[mn], ns, nt, cnt;
7 signed main()
8 {
9     scanf("%s%s", s + 1, t + 1);
10    ns = strlen(s + 1), nt = strlen(t + 1);
11    for (ll i = 2, j = 0; i <= nt; ++i)
12    {
13        while (j > 0 && t[j + 1] != t[i])
14        {
15            j = kmp[j];
16        }
17        if (t[j + 1] == t[i])
18        {
19            ++j;
20        }
21        kmp[i] = j;
22    }
23    for (ll i = 1, j = 0; i <= ns; ++i)
24    {
25        while (j > 0 && t[j + 1] != s[i])
26        {

```

```

27         j = kmp[j];
28     }
29     if (t[j + 1] == s[i])
30     {
31         ++j;
32     }
33     if (j == nt)
34     {
35         ++cnt;
36         printf("%lld ", i - j + 1);
37         j = kmp[j];
38     }
39 }
40 if (cnt == 0)
41 {
42     printf("-1");
43 }
44 return 0;
45 }

```

应用：

- 定义  $\exists p \in N, \forall i \in [1, |S| - p]$  都有  $S[i] = S[i + p]$ ，那么  $p$  是  $S$  的周期，则最短周期为  $|S| - \pi(|S|)$  (SCNUOJ1716)  
定义  $S$  的 border 是前缀和后缀相等的一段前缀。所有 border 长度为  $\pi(|S|), \pi(\pi(S)), \dots$ 。周期数等于 border 数且串长-周期长度=border长度。
- 串  $S$  的每个真前缀在串  $S$  出现次数: (SCNUOJ1717)

```

1 fill(ans + 1, ans + 1 + n, 1);
2 for (ll i = n; i >= 1; --i)
3     ans[kmp[i]] += ans[i];

```

串  $S$  的每个真前缀在串  $T$  出现次数: (SCNUOJ1718)

重构  $S = S + \# + T$ ，继续执行上述过程 ( $n' = n + m + 1$ )，取  $ans_{1..n}$

- 将  $s$  拼接最小长度字符串使其成为回文串 (lc214)  
设  $s$  为模式串， $s'$  (逆)为原串，用 KMP 在  $s'$  里匹配出最长的  $s$  前缀，就是最长的回文串。也可以 manacher 或字符串哈希做
- 环上长为  $n$  的回文串数目(多边形对称轴) (p3454)  
二倍长链  $s$  并求单倍长逆  $t'$ ，求  $s$  中  $t'$  的数目

## manacher

洛谷P3805-只由小写英文字符组成，求：最长回文子串长度

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define mn 22000010
4 char s[mn], ch;
5 int p[mn], n, ans, r, c;
6 signed main()

```

```

7  {
8      s[0] = '^', s[++n] = '#';
9      while (EOF != (ch = getchar()))
10         s[++n] = ch, s[++n] = '#';
11     s[n + 1] = '?'; // 多次询问清尾
12     for (int i = 1; i <= n; ++i)
13     {
14         if (i <= r)
15             p[i] = min(p[c * 2 - i], r - i + 1);
16         while (s[i + p[i]] == s[i - p[i]])
17             ++p[i];
18         if (i + p[i] > r)
19             r = p[i] + i - 1, c = i;
20         ans = max(ans, p[i]);
21     }
22     printf("%d", ans - 1);
23     return 0;
24 }

```

回文子串数目:  $\sum \lfloor \frac{p_i}{2} \rfloor$ 。

最长回文前缀:  $i - p_i + 1 = 1$  时更新最值为  $2p_i - 1$ 。

长为  $2k$  的回文串数目(p3454): `for(int i=2*k+1; i<=4*k; i++) res+=(p[i]>=n);`

## 字典树

主要是01字典树相关的一些总结。

对多个数  $a$  找一个数  $a_i$  使得所有  $a$  里  $a_i$  异或指定数  $k$  得到的值  $r$  最大:

对这些数  $a$  建立01trie, 然后以  $k$  遍历 trie, 贪心选择与  $k$  二进制数位相反的路径

```

1  ll res = 0, u = rt; //取字典树根节点
2  for(ll i = 11; i >= 0; --i)
3  {
4      ll c = (k >> i) & 1; //k的第i位是c
5      if (ch[u][c ^ 1]) //存在该位相反的数a,异或得1<<i
6          u = ch[u][c ^ 1], res += (1 << i);
7      else u = ch[u][c]; //不存在
8  } //伪代码

```

## AC自动机

失配指针: 状态  $u$  的失配指针  $fail$  指向另一个状态  $v$ , 满足  $v$  是  $u$  的最长后缀。复杂度  $O(Cn)$

洛谷P3808-求有多少个模式串在主串出现过(模式串和主串长度  $\leq 10^6$ )

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  #define sc(x) scanf("%lld", &x)

```

```

5 #define mn 1000010
6 int tr[mn][26], cnt, e[mn], fail[mn];
7 ll n, ans;
8 char s[mn];
9 queue<int> q;
10 signed main()
11 {
12     sc(n);
13     for (ll i = 1; i <= n; ++i)
14     {
15         scanf("%s", s + 1);
16         int u = 0;
17         for (int i = 1; s[i]; ++i)
18         {
19             int a = s[i] - 'a';
20             if (!tr[u][a])
21             {
22                 tr[u][a] = ++cnt;
23             }
24             u = tr[u][a];
25         }
26         ++e[u];
27     }
28     for (ll i = 0; i < 26; ++i)
29     {
30         if (tr[0][i])
31         {
32             q.push(tr[0][i]);
33         }
34     }
35     while (!q.empty())
36     {
37         int u = q.front();
38         q.pop();
39         for (int i = 0; i < 26; ++i)
40         {
41             if (tr[u][i])
42             {
43                 fail[tr[u][i]] = tr[fail[u]][i];
44                 q.push(tr[u][i]);
45             }
46             else
47             {
48                 tr[u][i] = tr[fail[u]][i];
49             }
50         }
51     }
52     scanf("%s", s + 1);
53     int u = 0;
54     for (int i = 1; s[i]; ++i)
55     {
56         int a = s[i] - 'a';
57         u = tr[u][a];
58         for (int j = u; j && e[j] != -1; j = fail[j])
59         {
60             ans += e[j], e[j] = -1;

```

```

61     }
62     }
63     printf("%lld", ans);
64     return 0;
65 }

```

洛谷P3796-求所有出现次数最多的字符串(每个模式串长度不超过 70 )

思路：对模式串AC自动机，对主串遍历时每个遍历到的节点，不断跳 fail，跳到的点对应的模式串出现次数都加一，复杂度  $O(70n + Cn)$

洛谷P5357-求每个模式串的出现次数，模式串总长度不超过  $2 \times 10^5$

思路：在P3796基础上，对模式串  $T$  去个重。建自动机时，把每个真实连接的 fail 指针(而不是失配 fail)指向的节点增加一个入度，使用拓扑排序跳 fail，复杂度  $O(|S| + 26|T|)$

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #define sc(x) scanf("%lld", &x)
5 #define mn 200010
6 #define alp 26
7 ll n, cnt, ss;
8 ll tr[mn][alp], idx[mn], nti[mn], fail[mn], in[mn], ans[mn], vis[mn];
9 char t[mn], s[mn * 10];
10 signed main()
11 {
12     sc(n);
13     for (ll i = 1, ts; i <= n; ++i)
14     {
15         scanf("%s", t + 1);
16         ts = strlen(t + 1);
17         ll p = 0;
18         for (ll j = 1, ti; j <= ts; ++j)
19         {
20             ti = t[j] - 'a';
21             if (tr[p][ti] == 0)
22             {
23                 tr[p][ti] = ++cnt;
24             }
25             p = tr[p][ti];
26         }
27         if (idx[p] == 0)
28         {
29             idx[p] = i;
30         }
31         nti[i] = idx[p];
32     }
33     queue<ll> q;
34     for (ll i = 0; i < alp; ++i)
35     {
36         if (tr[0][i])
37         {
38             q.push(tr[0][i]);
39         }
40     }

```

```

41     while (!q.empty())
42     {
43         ll u = q.front();
44         q.pop();
45         for (ll i = 0; i < alp; ++i)
46         {
47             if (tr[u][i])
48             {
49                 fail[tr[u][i]] = tr[fail[u]][i];
50                 ++in[fail[tr[u][i]]];
51                 q.push(tr[u][i]);
52             }
53             else
54             {
55                 tr[u][i] = tr[fail[u]][i];
56             }
57         }
58     }
59     scanf("%s", s + 1);
60     ss = strlen(s + 1);
61     for (ll i = 1, p = 0; i <= ss; ++i)
62     {
63         p = tr[p][s[i] - 'a'];
64         ++ans[p];
65     }
66     for (ll i = 0; i <= cnt; ++i)
67     {
68         if (in[i] == 0)
69         {
70             q.push(i);
71         }
72     }
73     while (!q.empty())
74     {
75         ll u = q.front();
76         q.pop();
77         vis[idx[u]] = ans[u];
78         ll v = fail[u];
79         ans[v] += ans[u];
80         if (--in[v] == 0)
81         {
82             q.push(v);
83         }
84     }
85     for (ll i = 1; i <= n; ++i)
86     {
87         printf("%lld\n", vis[nti[i]]);
88     }
89     return 0;
90 }

```

# 后缀数组

## 基础

下标从 1 开始, 后缀  $i$  表示以第  $i$  个字符开头的后缀。后缀间大小比较依据是字典序

数组  $sa[i]$  表示将所有后缀排序后第  $i$  小后缀的编号

数组  $rk[i]$  表示后缀  $i$  的排名

满足性质:  $sa[rk[i]] == rk[sa[i]] == i$

定义  $LCP(x, y)$  为后缀  $x, y$  的最长公共前缀

定义  $height[i]$ : 为  $LCP(sa[i], sa[i - 1])$ , 规定  $height[1] = 0$

定义:  $H[i]$ : 为  $height[rk[i]]$

性质:  $H[i] \geq H[i - 1] - 1$

洛谷P3809  $|S| \leq 10^6$ ; 复杂度  $O(n \log n)$

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #define mn 1000010
5 char s[mn];
6 ll n, sa[mn], rk[mn], oldrk[mn << 1], id[mn], px[mn], cnt[mn];
7 bool cmp(ll x, ll y, ll w)
8 {
9     return oldrk[x] == oldrk[y] && oldrk[x + w] == oldrk[y + w];
10 }
11 ll m = 300, i, p, w; //ASCII范围<300为计数排序值域, p是新值域
12 signed main()
13 {
14     scanf("%s", s + 1), n = strlen(s + 1);
15     for (ll i = 1; i <= n; ++i)
16     { //暂时以第一关键字为排序序, 有很多同名并列
17         ++cnt[rk[i]] = s[i];
18     }
19     for (ll i = 1; i <= m; ++i)
20     {
21         cnt[i] += cnt[i - 1];
22     }
23     for (ll i = n; i >= 1; --i)
24     {
25         sa[cnt[rk[i]]--] = i;
26     }
27     for (w = 1;; w <= 1, m = p)
28     {
29         for (p = 0, i = n; i > n - w; --i)
30         { //第二关键字排序: 无穷小区域
31             id[++p] = i;
32         }
33         for (ll i = 1; i <= n; ++i)
34         {
35             if (sa[i] > w)
36             {
```

```

37         id[++p] = sa[i] - w;
38     }
39 }
40 memset(cnt, 0, sizeof cnt);
41 for (ll i = 1; i <= n; ++i) //第一关键字排序
42 {
43     ++cnt[px[i] = rk[id[i]]];
44 }
45 for (ll i = 1; i <= m; ++i)
46 {
47     cnt[i] += cnt[i - 1];
48 }
49 for (ll i = n; i >= 1; --i)
50 {
51     sa[cnt[px[i]]--] = id[i];
52 }
53 memcpy(oldrk, rk, sizeof rk);
54 for (p = 0, i = 1; i <= n; ++i)
55 {
56     rk[sa[i]] = cmp(sa[i], sa[i - 1], w) ? p : ++p;
57 }
58 if (p == n)
59 {
60     for (ll i = 1; i <= n; ++i)
61     {
62         sa[rk[i]] = i;
63     }
64     break;
65 }
66 }
67 for (ll i = 1; i <= n; ++i)
68 {
69     printf("%lld ", sa[i]);
70 }
71 return 0;
72 }

```

$O(n)$  求 height 数组:

```

1 for (i = 1, k = 0; i <= n; ++i) {
2     if (rk[i] == 0) continue;
3     if (k) --k;
4     while (s[i + k] == s[sa[rk[i] - 1] + k]) ++k;
5     height[rk[i]] = k;
6 }

```

应用:

- 字符串匹配: 模式串  $T$  若出现必然是主串  $S$  后缀的前缀, 二分  $sa$ , 复杂度为  $O(|T| \log |S|)$ , 可以一并求出出现次数和位置
- 字典序最大子串: 必然是一个后缀, 对全体后缀跑后缀数组即可
- 求两字符串的最长公共子串: 将其拼接起来, 求 height 数组最值(取得 height 的两项要求一个在左原串一个在右)

- 求循环同构串里字典序最小的 (最小表示法算法的第二解法) (可以离散化)
- 两子串的最长公共前缀  $lcp(sa[i], sa[j]) = \min(height[i + 1..j])$
- 比较两个子串的大小关系:  $A = S[a..b], B = S[c..d]$  , 若  
 $lcp(a, c) \geq \min(|A|, |B|)$ ,  $A < B \Leftrightarrow |A| < |B|$  , 否则  $A < B \Leftrightarrow rk[a] < rk[c]$
- 本质不同的子串的数目  $\frac{n(n + 1)}{2} - \sum_{i=2}^n height[i]$
- 出现至少  $k$  次的子串的最大长度: 出现至少  $k$  次意味着后缀排序后有至少连续  $k$  个后缀的 LCP 是这个子串。所以, 求出每相邻  $k-1$  个  $height$  的最小值, 再求这些最小值的最大值就是答案。可以使用单调队列  $O(n)$  解决
- 是否有某字符串在文本串中至少不重叠地出现了两次: 可以二分目标串的长度  $|s|$  , 将  $h$  数组划分成若干个连续 LCP 大于等于  $|s|$  的段, 利用 RMQ 对每个段求其中出现的数中最大和最小的下标, 若这两个下标的距离满足条件, 则一定有长度为  $|s|$  的字符串不重叠地出现了两次
- 连续的若干个相同子串: 枚举连续串的长度  $|s|$  , 按照  $|s|$  对整个串进行分块, 对相邻两块的块首进行 LCP 与 LCS 查询

## LCP

例: 求 LCP 和 LCS(两前缀的最长公共后缀)

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 using ll = long long;
4 const ll mn = 1e6 + 10;
5 struct suffixArray
6 {
7     ll n, m; // m是字符集(char)大小
8     ll sa[mn], rk[mn * 2], hei[mn], ct[mn], s2[mn], r1[mn];
9     char *s;
10    void calcsa()
11    {
12        for (ll i = 0; i <= m; ++i)
13        {
14            ct[i] = 0;
15        }
16        for (ll i = n + 1; i <= n * 2; ++i)
17        {
18            rk[i] = 0;
19        }
20        for (ll i = 1; i <= n; ++i)
21        {
22            rk[i] = s[i];
23            ct[rk[i]]++;
24        }
25        for (ll i = 1; i <= m; ++i)
26        {
27            ct[i] += ct[i - 1];
28        }
29        for (ll i = n; i >= 1; --i)
30        {
31            sa[ct[rk[i]]--] = i;
32        }

```

```

33     ll mx = m;
34     for (ll j = 1, k = 0; k < n; j *= 2, mx = k)
35     {
36         ll p = 0;
37         for (ll i = n - j + 1; i <= n; ++i)
38         {
39             s2[++p] = i;
40         }
41         for (ll i = 1; i <= n; ++i)
42         {
43             if (sa[i] > j)
44             {
45                 s2[++p] = sa[i] - j;
46             }
47         }
48         for (ll i = 0; i <= mx; ++i)
49         {
50             ct[i] = 0;
51         }
52         for (ll i = 1; i <= n; ++i)
53         {
54             ct[rk[s2[i]]]++;
55         }
56         for (ll i = 1; i <= mx; ++i)
57         {
58             ct[i] += ct[i - 1];
59         }
60         for (ll i = n; i >= 1; --i)
61         {
62             sa[ct[rk[s2[i]]]--] = s2[i];
63         }
64         r1[sa[1]] = k = 1;
65         for (ll i = 2; i <= n; ++i)
66         {
67             if (rk[sa[i - 1]] == rk[sa[i]] && rk[sa[i - 1] + j] ==
rk[sa[i] + j])
68             {
69                 r1[sa[i]] = k;
70             }
71             else
72             {
73                 r1[sa[i]] = ++k;
74             }
75         }
76         for (ll i = 1; i <= n; ++i)
77         {
78             rk[i] = r1[i];
79         }
80     }
81 }
82 void calcHeight()
83 {
84     memset(hei, 0, sizeof hei);
85     for (ll i = 1, j; i <= n; ++i)
86     {
87         if (rk[i] == 1)

```

```

88         {
89             continue;
90         }
91         j = max(hei[rk[i] - 1] - 1, 0LL);
92         while (s[i + j] == s[sa[rk[i] - 1] + j])
93         {
94             ++j;
95         }
96         hei[rk[i]] = j;
97     }
98 }
99 ll t1[mn][20], lg2[mn];
100 void initLCP()
101 {
102     for (ll i = 2; i <= n; ++i)
103     {
104         lg2[i] = lg2[i / 2] + 1;
105     }
106     for (ll i = 1; i <= n; ++i)
107     {
108         t1[i][0] = hei[i];
109     }
110     for (ll j = 1; j <= 20; ++j)
111     {
112         for (ll i = 1; i + (1 << j) - 1 <= n; ++i)
113         {
114             t1[i][j] = min(t1[i][j - 1], t1[i + (1 << (j - 1))][j - 1]);
115         }
116     }
117 }
118 void build(char *t)
119 {
120     s = t, n = strlen(s + 1), m = 'z';
121     calcSA();
122     calcHeight();
123     initLCP();
124 }
125 ll lcp(ll x, ll y)
126 {
127     ll l = rk[x], r = rk[y];
128     if (l > r)
129     {
130         swap(l, r);
131     }
132     ++l;
133     ll k = lg2[r - l + 1]; // or __lg(r-l+1);
134     return min(t1[l][k], t1[r - (1 << k) + 1][k]);
135 }
136 } s1, s2;
137 char s[mn];
138 ll n, t, l, r;
139 signed main()
140 {
141     cin.tie(0)->iostream::sync_with_stdio(false);
142     cin >> s + 1;

```

```

143     n = strlen(s + 1);
144     s1.build(s);
145     reverse(s + 1, s + 1 + n);
146     s2.build(s);
147     for (cin >> t; t; --t)
148     {
149         cin >> l >> r;
150         cout << s1.lcp(l, r) << ' ' << s2.lcp(n - l + 1, n - r + 1) <<
151         '\n';
152     }
153     return 0;
154 }
```

用单调栈线性求  $\sum_{1 \leq i < j \leq n} lcp(i, j)$ , 意义: 任选两个不同位置子串, 子串相同的方案数。一种实现:

```

1  ll lc[mn], rc[mn];
2  ll calcLCPsum()
3  {
4      stack<ll> a;
5      ll sum = 0;
6      for (ll i = 1; i <= n + 1; ++i)
7      {
8          while (!a.empty() && hei[a.top()] > hei[i])
9          {
10             ll j = a.top();
11             a.pop();
12             rc[j] = i - j;
13         }
14         a.push(i);
15     }
16     stack<ll> b;
17     for (ll i = n; i >= 0; --i)
18     {
19         while (!b.empty() && hei[b.top()] >= hei[i])
20         {
21             ll j = b.top();
22             b.pop();
23             lc[j] = j - i;
24         }
25         b.push(i);
26     }
27     for (ll i = 1; i <= n; ++i)
28     {
29         sum += hei[i] * lc[i] * rc[i];
30     }
31     return sum;
32 }
```

时空复杂度  $O(n)$

例：牛客2022暑假3H，长为  $n(n \leq 10^5)$  的  $A$  串对应权值序列  $v(|v_i| \leq 10^9)$ ，有  $k$  个查询，每次问长为  $m(m, k \leq 10^5, mk \leq 10^6)$  的  $B$  串所有子串能匹配到所有  $A$  的子串区间里，区间和最大值

后缀数组求高度数组  $lcp$ ，即第  $i$  大后缀与第  $i + 1$  大后缀的最长公共前缀，在其上面建立 ST 表，可求任意两后缀的 LCP 为  $query_{\min}(rank_x, rank_y - 1)$ ，假设  $rank_x < rank_y$ 。

对  $B_i$  的每个位置  $j(1 \leq j \leq m)$ ，求  $j$  开始的最长子串，且在  $A$  出现过，即找到最长  $len$ ，满足  $B[j..j + len - 1]$  是  $A$  子串。再找  $j$  为左端点，长在  $len$  内所有区间权值和最大值，即  $\max(\sum_{k=j}^l v_k)(j \leq l \leq j + len - 1)$ 。这部分对数组  $v$  前缀和数组建 ST 表，求  $[j, j + len - 1]$  区间最大值。

将  $A$  和所有  $B_i$  以  $\$$  或其他不在字符集的分割符连起来得到  $S$ ，对  $S$  跑后缀数组，维护每个  $B_i$  在  $S$  的起始下标，第  $i$  个字符对应原来哪个串。(注意  $A, B$  间用别的分割符)

按字典序遍历所有后缀，可以知道当前后缀对应哪个  $B_i$  或  $A$ ，若当前后缀对应  $B_i$ ，找到离它最近的属于  $A$  的后缀，求  $lcp$  即上文的  $len$ 。

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define sc(x) scanf("%lld", &x)
4 typedef long long ll;
5 const ll mn = 1.2e6 + 5;
6 char s[mn];
7 // begin SA-IS
8 ll sa[mn], rk[mn], lcp[mn];
9 ll str[mn * 2], ty[mn * 2], p[mn], cnt[mn], cur[mn];
10 #define pushs(x) sa[cur[str[x]]--] = x
11 #define pushl(x) sa[cur[str[x]]++] = x
12 void sais(ll n, ll m, ll *str, ll *ty, ll *p)
13 {
14     ll n1 = ty[n - 1] = 0, ch = rk[0] = -1, *s1 = str + n;
15     for (ll i = n - 2; ~i; --i)
16     {
17         ty[i] = str[i] == str[i + 1] ? ty[i + 1] : str[i] > str[i + 1];
18     }
19     for (ll i = 1; i < n; ++i)
20     {
21         rk[i] = ty[i - 1] && !ty[i] ? (p[n1] = i, n1++) : -1;
22     }
23
24     auto induce_sort = [&](ll *v)
25     {
26         fill_n(sa, n, -1);
27         fill_n(cnt, m, 0);
28         for (ll i = 0; i < n; ++i)
29         {
30             cnt[str[i]]++;
31         }
32         for (ll i = 1; i < m; ++i)
33         {
34             cnt[i] += cnt[i - 1];
35         }
36         for (ll i = 0; i < n; ++i)
37         {
38             ll j = sa[i];
39             if (j < m)
40             {
41                 v[i] = str[j];
42             }
43             else
44             {
45                 v[i] = str[j - m];
46             }
47         }
48     };
49     induce_sort(v);
50     for (ll i = 0; i < n; ++i)
51     {
52         ll j = sa[i];
53         if (j < m)
54         {
55             str[i] = str[j];
56         }
57         else
58         {
59             str[i] = str[j - m];
60         }
61     }
62     for (ll i = 0; i < n; ++i)
63     {
64         ll j = sa[i];
65         if (j < m)
66         {
67             ty[i] = str[j];
68         }
69         else
70         {
71             ty[i] = str[j - m];
72         }
73     }
74     for (ll i = 0; i < n; ++i)
75     {
76         ll j = sa[i];
77         if (j < m)
78         {
79             p[i] = str[j];
80         }
81         else
82         {
83             p[i] = str[j - m];
84         }
85     }
86     for (ll i = 0; i < n; ++i)
87     {
88         ll j = sa[i];
89         if (j < m)
90         {
91             rk[i] = ty[i];
92         }
93         else
94         {
95             rk[i] = ty[i - m];
96         }
97     }
98     for (ll i = 0; i < n; ++i)
99     {
100        ll j = sa[i];
101        if (j < m)
102        {
103            lcp[i] = str[j];
104        }
105        else
106        {
107            lcp[i] = str[j - m];
108        }
109    }
110 }

```

```

35      }
36      for (ll i = 0; i < m; ++i)
37      {
38          cur[i] = cnt[i] - 1;
39      }
40      for (ll i = n1 - 1; ~i; --i)
41      {
42          pushs(v[i]);
43      }
44      for (ll i = 1; i < m; ++i)
45      {
46          cur[i] = cnt[i - 1];
47      }
48      for (ll i = 0; i < n; ++i)
49      {
50          if (sa[i] > 0 && ty[sa[i] - 1])
51          {
52              pushl(sa[i] - 1);
53          }
54      }
55      for (ll i = 0; i < m; ++i)
56      {
57          cur[i] = cnt[i] - 1;
58      }
59      for (ll i = n - 1; ~i; --i)
60      {
61          if (sa[i] > 0 && !ty[sa[i] - 1])
62          {
63              pushs(sa[i] - 1);
64          }
65      }
66  };
67  induce_sort(p);
68  for (ll i = 0, x, y; i < n; ++i)
69  {
70      if (~(x = rk[sa[i]]))
71      {
72          if (ch < 1 || p[x + 1] - p[x] != p[y + 1] - p[y])
73          {
74              ++ch;
75          }
76      }
77      else
78      {
79          for (ll j = p[x], k = p[y]; j <= p[x + 1]; ++j, ++k)
80          {
81              if ((str[j] << 1 | ty[j]) != (str[k] << 1 | ty[k]))
82              {
83                  ++ch;
84                  break;
85              }
86          }
87          s1[y = x] = ch;
88      }
89  }
90  if (ch + 1 < n1)

```

```

91     {
92         sais(n1, ch + 1, s1, ty + n, p + n1);
93     }
94     else
95     {
96         for (ll i = 0; i < n1; ++i)
97         {
98             sa[s1[i]] = i;
99         }
100    }
101    for (ll i = 0; i < n1; ++i)
102    {
103        s1[i] = p[sa[i]];
104    }
105    induce_sort(s1);
106 }
107 ll cti(ll n)
108 {
109     ll m = *max_element(s, s + n);
110     fill_n(rk, m + 1, 0);
111     for (ll i = 0; i < n; ++i)
112     {
113         rk[s[i]] = 1;
114     }
115     for (ll i = 0; i < m; ++i)
116     {
117         rk[i + 1] += rk[i];
118     }
119     for (ll i = 0; i < n; ++i)
120     {
121         str[i] = rk[s[i]] - 1;
122     }
123     return rk[m];
124 }
125 void make_sa(ll n)
126 {
127     // s[n] 一定要比 s 中所有字符 ascii 值小, s[n+1] 倒无所谓
128     s[n] = '!', s[n + 1] = '\0';
129     ll m = cti(++n);
130     sais(n, m, str, ty, p);
131     for (ll i = 0; i < n; ++i)
132     {
133         rk[sa[i]] = i;
134     }
135     for (ll i = 0, h = lcp[0] = 0; i < n - 1; ++i)
136     {
137         ll j = sa[rk[i] - 1];
138         while (i + h < n && j + h < n && s[i + h] == s[j + h])
139         {
140             ++h;
141         }
142         lcp[rk[i] - 1] = h;
143         if (lcp[rk[i] - 1])
144         {
145             --h;
146         }

```

```

147     }
148     s[n] = '\0';
149 }
150 // end SA-IS
151 const ll mm = 1e5 + 5, mlg = 20;
152 ll st[mm][mlg], lg[mm], prf[mm];
153 void build_st(ll n)
154 {
155     for (ll i = 1; i <= n; ++i)
156     {
157         st[i][0] = prf[i];
158     }
159     for (ll k = 1, len = 2; len <= n; len <= 1, ++k)
160     {
161         for (ll i = 1; i + len - 1 <= n; ++i)
162         {
163             st[i][k] = max(st[i][k - 1], st[i + len / 2][k - 1]);
164         }
165     }
166 }
167 ll query(ll x, ll y)
168 {
169     ll k = lg[y - x + 1];
170     return max(st[x][k], st[y - (1 << k) + 1][k]);
171 }
172 ll v[mm], bg_pos[mm], ans[mm], maps[mn], n, m, k;
173 signed main()
174 {
175     lg[1] = 0;
176     for (ll i = 2; i < mm; ++i)
177     {
178         lg[i] = lg[i >> 1] + 1;
179     }
180     sc(n), sc(m), sc(k), scanf("%s", s);
181     for (ll i = 1; i <= m; ++i)
182     {
183         sc(v[i]);
184     }
185
186     ll tot = n - 1;
187     for (ll i = 1; i <= k; ++i)
188     {
189         ++tot;
190         s[tot] = '$', maps[tot] = -1;
191         bg_pos[i] = tot + 1;
192         scanf("%s", s + tot + 1);
193         for (ll j = tot + 1; j <= tot + m; ++j)
194         {
195             maps[j] = i;
196         }
197         tot += m;
198     }
199     s[n] = '#';
200     ++tot;
201     make_sa(tot);
202 }
```

```

203     // printf("%s\n", s);
204     // for (ll i = 0; i <= tot; ++i)
205     //{
206     //     printf("%lld %lld %lld %s\n", i, sa[i], lcp[i], s + sa[i]);
207     //}
208
209     for (ll i = 1; i <= m; ++i)
210     {
211         prf[i] = prf[i - 1] + v[i];
212     }
213     build_st(m);
214
215     ll mi = 0;
216     for (ll i = 1; i <= tot; ++i)
217     {
218         ll j = maps[sa[i]];
219         if (j == 0)
220         {
221             mi = lcp[i];
222         }
223         else
224         {
225             if (j > 0 && mi > 0) //不是分隔符, 有公共
226             {
227                 ll idx = sa[i] - bg_pos[j] + 1; // B_j起始下标
228                 ll mx = query(idx, idx + mi - 1);
229                 ans[j] = max(ans[j], mx - prf[idx - 1]);
230             }
231             mi = min(mi, lcp[i]);
232         }
233     }
234
235     mi = 0;
236     for (ll i = tot; i; --i)
237     {
238         ll j = maps[sa[i]];
239         if (j == 0)
240         {
241             mi = lcp[i - 1];
242         }
243         else
244         {
245             if (j > 0 && mi > 0) //不是分隔符, 有公共
246             {
247                 ll idx = sa[i] - bg_pos[j] + 1; // B_j起始下标
248                 ll mx = query(idx, idx + mi - 1);
249                 ans[j] = max(ans[j], mx - prf[idx - 1]);
250             }
251             mi = min(mi, lcp[i - 1]);
252         }
253     }
254     for (ll i = 1; i <= k; ++i)
255     {
256         printf("%lld\n", ans[i]);
257     }
258     return 0;

```

## 后缀自动机

构造时空复杂度  $O(n)$ ，在线算法

洛谷P3804-求出现次数不为 1 的子串的出现次数乘长度的最大值

```

1 #include<cstdio>
2 #include<algorithm>
3 using namespace std;const int N=3*1e6+10;typedef long long ll;
4 char mde[N];int n1;ll res;
5 struct suffixautomation
6 {
7     int mp[N][30];int fa[N];int ed;int ct;int len[N];int siz[N];
8     suffixautomation(){ed=ct=1;}
9     int v[N];int x[N];int al[N];int cnt;
10    inline void add(int u,int v){v[++cnt]=v;x[cnt]=al[u];al[u]=cnt;}
11    inline void ins(int c)
12    {
13        int p=ed;siz[ed+=ct]=1;len[ed]=n1;//先初始化size和len
14        for(;p&&mp[p][c]==0;p=fa[p])mp[p][c]=ed;//然后顺着parent树的路径向上找
15        if(p==0){fa[ed]=1;return;}int q=mp[p][c];//case1
16        if(len[p]+1==len[q]){fa[ed]=q;return;}//case2
17        len[++ct]=len[p]+1;//case 3
18        for(int i=1;i<=26;i++)mp[ct][i]=mp[q][i];
19        fa[ct]=fa[q];fa[q]=ct;fa[ed]=ct;
20        for(int i=p;mp[i][c]==q;i=fa[i])mp[i][c]=ct;
21    }
22    inline void bt(){for(int i=2;i<=ct;i++)add(fa[i],i);} //暴力建树
23    void dfs(int u)//dfs
24    {
25        for(int i=al[u];i;i=x[i])dfs(v[i]);siz[u]+=siz[v[i]];
26        if(siz[u]!=1){res=max(res,(ll)siz[u]*len[u]);}
27    }
28 }sam;
29 int main()
30 {
31     scanf("%s",mde+1);
32     for(n1=1;mde[n1]!='\0';n1++)sam.ins(mde[n1]-'a'+1);
33     sam.bt();sam.dfs(1);printf("%lld",res);return 0;
34 }
```

## FFT字符串匹配

字符串  $A$  与  $B$  以第  $x$  位结束的连续  $m$  位完全匹配可以表示为：(从 0 开始下标)

$$P(x) = \sum_{i=0}^{m-1} [A(i) - B(x - (m - 1) + i)]^2$$

设  $S(x) = A(m - x - 1)$ ，则：

$$\begin{aligned} P(x) &= \sum_{i=0}^{m-1} [S(m - i - 1) - B(x - m + i + 1)]^2 \\ &= \sum_{i=0}^{m-1} S(i)^2 + \sum_{i=x-m+1}^x B(i)^2 - 2 \sum_{i+j=x} S(i)B(j) \end{aligned}$$

设通配符的ASCII为0，则：

$$C(x, y) = [A(x) - B(y)]^2 A(x)B(y)$$

完全匹配函数即：

$$\begin{aligned} P(x) &= \sum_{i=0}^{m-1} C(i, x - m + i + 1) \\ &= \sum_{i+j=x} S(i)^3 B(j) + \sum_{i+j=x} S(i)B(j)^3 - 2 \sum_{i+j=x} S(i)^2 B(j)^2 \end{aligned}$$

带幂的可以先预处理每一项的系数做幂运算。伪代码如下：

```
1 void FFT_match(char *s1, char *s2, int m, int n)
2 {
3     reverse(ss1, ss1+m);
4     for(int i=0; i<m; i++) A[i]=(s1[i]!='*')?(s1[i]-'a'+1):0;
5     for(int i=0; i<n; i++) B[i]=(s2[i]!='*')?(s2[i]-'a'+1):0;
6
7     for(int i=0; i<len; i++) a[i]=Comp(A[i]*A[i]*A[i], 0), b[i]=Comp(B[i], 0);
8     FFT(a, len, 1); FFT(b, len, 1);
9     for(int i=0; i<len; i++) P[i]=P[i]+a[i]*b[i];
10
11    for(int i=0; i<len; i++) a[i]=Comp(A[i], 0), b[i]=Comp(B[i]*B[i]*B[i], 0);
12    FFT(a, len, 1); FFT(b, len, 1);
13    for(int i=0; i<len; i++) P[i]=P[i]+a[i]*b[i];
14
15    for(int i=0; i<len; i++) a[i]=Comp(A[i]*A[i], 0), b[i]=Comp(B[i]*B[i], 0);
16    FFT(a, len, 1); FFT(b, len, 1);
17    for(int i=0; i<len; i++) P[i]=P[i]-a[i]*b[i]*Comp(2, 0);
18
19    FFT(P, len, -1);
20    for(int i=m-1; i<n; i++) if(fabs(P[i].r)<=1e-7) printf("%d ", i-m+2);
21 }
```

给定带通配符\*的字符串  $a, b$ ,  $|a| = m, |b| = n$ , 下标从1开始, 求  $a$  的首位作为  $b$  的第  $k$  位时能够匹配的所有解和解数目。 $1 \leq m \leq n \leq 3 \times 10^5$

对  $b$ , 设  $P(i)$ , 一共需要计算  $m - n + 1$  个  $P$ 。事实上, 卷积的时候, 由于计算的是所有  $i \times j$ , 然后对于某一次只取  $i + j = C$  的, 那么可以对计算出来的结果一次FFT计算后分别累加到不同的  $P$  上。

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 #define db double
4 const db pi = acos(-1);
5 #define MAX 1100000
6 char sa[MAX], sb[MAX];
```

```

7  int m, n, N, top, a[MAX], b[MAX];
8  int rev[MAX], sta[MAX];
9  #define il inline
10 #define ll int
11 #define re
12 il ll read()
13 {
14     re char p = 0;
15     re ll r = 0, o = 0;
16     for (; p < '0' || p > '9'; o |= p == '-' , p = getchar())
17         ;
18     for (; p >= '0' && p <= '9'; r = (r << 1) + (r << 3) + (p ^ 48), p =
getchar())
19         ;
20     return o ? (~r) + 1 : r;
21 }
22 struct cpx
23 {
24     double x, y;
25     cpx() {}
26     cpx(double xx, double yy) { x = xx, y = yy; }
27     cpx(int xx, int yy) { x = xx, y = yy; }
28     friend cpx operator*(cpx a, cpx b)
29     {
30         return cpx(a.x * b.x - a.y * b.y, a.x * b.y + a.y * b.x);
31     }
32     friend cpx operator/(cpx a, double b)
33     {
34         return cpx(a.x / b, a.y / b);
35     }
36     friend cpx operator+(cpx a, cpx b)
37     {
38         return cpx(a.x + b.x, a.y + b.y);
39     }
40     friend cpx operator-(cpx a, cpx b)
41     {
42         return cpx(a.x - b.x, a.y - b.y);
43     }
44     friend cpx operator*(cpx a, double b)
45     {
46         return cpx(a.x * b, a.y * b);
47     }
48 };
49 cpx A[MAX], B[MAX], C[MAX];
50 void fft(cpx *a, int f)
51 {
52     for (int i = 0; i < N; i++)
53         if (i < rev[i])
54             swap(a[i], a[rev[i]]);
55     for (int i = 1; i < N; i <= i)
56     {
57         cpx wn(cos(pi / i), f * sin(pi / i));
58         for (int j = 0; j < N; j += (i << 1))
59         {
60             cpx w(1, 0);
61             for (int k = 0; k < i; k++)

```

```

62         {
63             cpx x = a[j + k], y = w * a[j + k + i];
64             a[j + k] = x + y;
65             a[j + k + i] = x - y;
66             w = w * wn;
67         }
68     }
69 }
70 if (f == -1)
71     for (int i = 0; i < n; i++)
72         a[i] = a[i] / n;
73 }
74 int main()
75 {
76     scanf("%d%d%s%s", &m, &n, sa, sb);
77     reverse(sa, sa + m);
78     int l = 0;
79     for (N = 1; N < 2 * n; N <= l)
80     {
81         l++;
82     }
83     for (int i = 0; i < N; i++)
84     {
85         rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (l - 1));
86     }
87     for (int i = 0; i < m; i++)
88     {
89         if (sa[i] != '*')
90         {
91             a[i] = sa[i] - 'a' + 1;
92         }
93     }
94     for (int i = 0; i < n; i++)
95     {
96         if (sb[i] != '*')
97         {
98             b[i] = sb[i] - 'a' + 1;
99         }
100    }
101    for (int i = 0; i < m; i++)
102    {
103        A[i] = cpx(a[i] * a[i] * a[i], 0);
104    }
105    for (int i = 0; i < n; i++)
106    {
107        B[i] = cpx(b[i], 0);
108    }
109    fft(A, 1);
110    fft(B, 1);
111    for (int i = 0; i < N; i++)
112    {
113        C[i] = C[i] + A[i] * B[i];
114    }
115
116    for (int i = 0; i < N; i++)
117    {

```

```

118     A[i] = B[i] = cpx(0, 0);
119 }
120 for (int i = 0; i < m; i++)
121 {
122     A[i] = cpx(a[i] * a[i], 0);
123 }
124 for (int i = 0; i < n; i++)
125 {
126     B[i] = cpx(b[i] * b[i], 0);
127 }
128 fft(A, 1);
129 fft(B, 1);
130 for (int i = 0; i < N; i++)
131 {
132     C[i] = C[i] - A[i] * B[i] * 2.0;
133 }
134
135 for (int i = 0; i < N; i++)
136 {
137     A[i] = B[i] = cpx(0, 0);
138 }
139 for (int i = 0; i < m; i++)
140 {
141     A[i] = cpx(a[i], 0);
142 }
143 for (int i = 0; i < n; i++)
144 {
145     B[i] = cpx(b[i] * b[i] * b[i], 0);
146 }
147 fft(A, 1);
148 fft(B, 1);
149 for (int i = 0; i < N; i++)
150 {
151     C[i] = C[i] + A[i] * B[i];
152 }
153
154 fft(C, -1);
155 for (int i = m - 1; i < n; i++)
156 {
157     if (fabs(C[i].x) < 0.5)
158     {
159         sta[top++] = i - m + 2;
160     }
161 }
162 printf("%d\n", top);
163 for (int i = 0; i < top; i++)
164 {
165     printf("%d ", sta[i]);
166 }
167 return 0;
168 }

```

## 子序列自动机

对  $S$  每个字符，升序记录它出现的位置，设字符  $i$  出现的位置的集合为  $pos_i$ 。

判断  $T$  是否是  $S$  的子序列，每次二分找  $T_i$  在上一次匹配  $S_j$  后的最近下标，如果一直能找到则是子序列。复杂度取  $T \log |S|$ 。

P5826-字符集大小  $10^5$ ,  $|S| \leq 10^5$ ,  $\sum |T| \leq 10^6$ 。上述问题。

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef int ll;
4 #define mn 100002
5 ll n, q, m, type, ii;
6 vector<ll> v[mn];
7 signed main()
8 {
9     scanf("%d%d%d%d", &type, &n, &q, &m);
10    for(ll i=1;i<=n;++i) scanf("%d", &ii), v[ii].push_back(i);
11    while (q--)
12    {
13        ll len, at = 0, x;
14        bool suc = true;
15        scanf("%d", &len);
16        while (len--)
17        {
18            scanf("%d", &x);
19            if (!suc) continue;
20            auto it = lower_bound(v[x].begin(), v[x].end(), at + 1);
21            if (it == v[x].end()) suc = false;
22            else at = *it;
23        }
24        printf(suc ? "Yes\n" : "No\n");
25    }
26    return 0;
27 }
```

第二种建图方式： $nxt_{i,j}$  表示  $i$  位置后的第一个  $j$  的位置，0 是根节点。

```
1 for(LL i=n;i>=1;--i){
2     for(LL j=1;j<=a;++j) nxt[i-1][j]=nxt[i][j];
3     nxt[i-1][s[i]]=i;
4 }
```

当字符集较大时，可套用可持久化，在叶子节点放一个 id，表示出边。

将  $T$  输入自动机，可以检验是否是子序列。

求本质不同子序列个数：记忆化 DFS，设  $x$  的 DFN 子树有  $f_x$  个子序列，答案为  $f_0$ ，转移为  $f_u = 1 + \sum f_v$ 。非空就  $f_0 - 1$ 。

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 using ll = long long;
4 const ll mod = 1e9 + 7;
```

```

5  vector<ll> nxt[28], f;
6  ll dfs(int u)
7  {
8      if (f[u])
9          return f[u];
10     ll ans = 1;
11     for (auto i = 1; i <= 26; ++i)
12     {
13         if (nxt[i][u])
14         {
15             (ans += dfs(nxt[i][u])) %= mod;
16         }
17     }
18     return f[u] = ans;
19 }
20 int distinctSubseqII(string s)
21 {
22     ll n = s.size();
23     for (ll j = 1; j <= 26; ++j)
24         nxt[j].resize(n + 2, 0);
25     for (ll i = n; i >= 1; --i)
26     {
27         for (ll j = 1; j <= 26; ++j)
28             nxt[j][i - 1] = nxt[j][i];
29         nxt[s[i - 1] - 'a' + 1][i - 1] = i;
30     }
31     f.resize(n + 2, 0);
32     return (dfs(0) - 1 + mod) % mod;
33 }

```

求两串公共子序列个数，类比且一下：

```

1  LL Dfs(LL x,LL y){
2      if(f[x][y]) return f[x][y];
3      for(LL i=1;i<=a;++i)
4          if(nxt1[x][i]&&nxt2[y][i])
5              f[x][y]+=Dfs(nxt1[x][i],nxt2[y][i]);
6      return ++f[x][y];
7  }

```

回文子序列

```

1  LL Dfs(LL x,LL y){
2      if(f[x][y]) return f[x][y];
3      for(LL i=1;i<=a;++i)
4          if(nxt1[x][i]&&nxt2[y][i]){
5              if(nxt1[x][i]+nxt2[y][i]>n+1) continue;
6              if(nxt1[x][i]+nxt2[y][i]<n+1) f[x][y]++;
7              f[x][y]=(f[x][y]+Dfs(nxt1[x][i],nxt2[y][i]))%mod;
8          }
9      return ++f[x][y];
10 }

```

## 最小表示法

可以用 SAM, SA, Lyndon 分解等做

求循环同构串里字典序最小的串, 复杂度  $O(n)$

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 ll n, ans, a[300002], i, j = 1, k;
5 signed main()
6 {
7     scanf("%lld", &n);
8     for(ll i=0;i<n;++i) scanf("%lld", a + i);
9     while (i < n && j < n && k < n)
10    {
11        if (a[(i + k) % n] == a[(j + k) % n]) ++k;
12        else
13        {
14            if (a[(i + k) % n] > a[(j + k) % n]) i += k + 1;
15            else j += k + 1;
16            if (i == j) ++i;
17            k = 0;
18        }
19    }
20    ans = min(i, j);
21    for(ll i=0;i<n;++i) printf("%lld ", a[(i + ans) % n]);
22    return 0;
23 }
```

## Lyndon分解

SAM 也可以做 Lyndon 分解

对于一个字符串, 若其本身就是其最小后缀, 则称它为 Lyndon 串, 即若一个串的字典序比他所有后缀的字典序小。

任意一个字符串都可以被唯一的分解成若干个字典序非严格递减的 Lyndon 串。对于长度为  $n$  的字符串  $s$ , 存在唯一的若干个 Lyndon 串  $t_{1\dots m}$ , 满足  $s = t_1 + t_2 + \dots + t_m$  且  $t_1 \geq t_2 \geq \dots \geq t_m$ 。

Duval 算法可以在  $O(n)$  的时间求出一个字符串的 Lyndon 分解

洛谷P6114-求所有Lyndon串右端点的异或和

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef int ll;
4 ll n, ans, i = 1, j, k;
5 char s[500002];
6 signed main()
7 {
8     scanf("%s", s + 1);
9     n = strlen(s + 1);
10    while (i <= n)
```

```

11  {
12      j = i, k = i + 1;
13      while (k <= n && s[j] <= s[k]) j = s[j] == s[k++] ? j + 1 : i;
14      while (i <= j) i += k - j, ans ^= i - 1;
15  }
16  printf("%d", ans);
17  return 0;
18 }

```

改成 `, ans = i <= n ? i : ans` 可以求最小表示法的起点

## 杂项

### 排序

归并、计数、基数是稳定排序。

#### 计数排序

参考代码: (设值域是  $[1, k]$  的整数)

```

1 for (int i = 1; i <= n; ++i)
2 {
3     ++c[a[i]]; // c 初始值均 0; a 数组使用下标范围 [1, n] 存值
4 }
5 for (int i = 1; i <= k; ++i) // k 是 a 的最大值
6 {
7     for (int j = 1; j <= c[i]; ++j)
8     {
9         a[++p] = i; // p 初始值为 0
10    }
11 }

```

前缀和写法:

```

1 for (int i = 1; i <= n; ++i)
2 {
3     ++cnt[a[i]];
4 }
5 for (int i = 1; i <= k; ++i)
6 {
7     cnt[i] += cnt[i - 1];
8 }
9 for (int i = n; i >= 1; --i) //顺着i到n遍历也行
10 {
11     b[cnt[a[i]]--] = a[i];
12 }

```

## 基数排序

以下代码以  $1 \leq n \leq 8 \times 10^6$ ，值域在  $2^{31}$ ， $q$  次询问排序后位置为例(下标从 1 开始) (SCNUOJ 1577)

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef int ll;
4 #define il inline
5 il ll read()
6 {
7     char p = 0;
8     ll r = 0, o = 0;
9     for (; p < '0' || p > '9'; o |= p == '-' , p = getchar())
10    ;
11     for (; p >= '0' && p <= '9'; r = (r << 1) + (r << 3) + (p ^ 48), p = getchar())
12    ;
13     return o ? (~r) + 1 : r;
14 }
15 #define sc(x) x = read()
16 #define mn 8000010
17 ll n, a[mn], q, x, b[mn], cnt[1 << 8];
18 void radixsort(ll n)
19 {
20     ll mask = (1 << 8) - 1;
21     ll *x = a, *y = b;
22     for (ll i = 0; i < 32; i += 8) //每八位每八位拆分, int拆为四个关键字
23     {
24         for (ll j = 0; j != (1 << 8); ++j) // [0,255]
25         {
26             cnt[j] = 0; //计数排序归零
27         }
28         for (ll j = 0; j != n; ++j)
29         {
30             ++cnt[x[j] >> i & mask]; //当前的八个位, 注意位运算优先级
31         }
32         for (ll sum = 0, j = 0; j != (1 << 8); ++j)
33         { //计数排序变式, cnt代表同值元素段的开头下标;即减去前缀和前自己
34             sum += cnt[j], cnt[j] = sum - cnt[j];
35         }
36         for (ll j = 0; j != n; ++j)
37         { //计数排序变式
38             y[cnt[x[j] >> i & mask]++] = x[j];
39         }
40         swap(x, y); // y是排序好的,x是未排序的;换过来, 继续操作
41     }
42 }
43 signed main()
44 {
45     sc(n);
46     for (ll i = 0; i < n; ++i)
47     {
48         sc(a[i]);
49     }
```

```

50     radixsort(n);
51     sc(q);
52     while (q--)
53     {
54         sc(x);
55         printf("%d\n", a[x - 1]);
56     }
57     return 0;
58 }

```

## 归并求逆序对

参考代码: (洛谷P1908)

```

1 #include <bits/stdc++.h> //洛谷p1908
2 using namespace std;
3 #define mn 500010
4 typedef long long ll;
5 ll n, a[mn], b[mn], ans;
6 void mergesort(ll lf, ll rf)
7 {
8     if (lf < rf)
9     {
10         ll cf = (lf + rf) >> 1; // left-face; center-face; right-face
11         mergesort(lf, cf);
12         mergesort(cf + 1, rf);
13         ll i = lf, j = cf + 1, ie = rf, ie = cf, k = 0; //左[i,ie],右[j,je]
14         while (i <= ie && j <= ie)
15         {
16             if (a[i] <= a[j])
17             {
18                 b[k++] = a[i++];
19             }
20             else
21             {
22                 ans += ie - i + 1;
23                 b[k++] = a[j++];
24             }
25         }
26         while (i <= ie)
27         {
28             b[k++] = a[i++];
29         }
30         while (j <= ie)
31         {
32             b[k++] = a[j++];
33         }
34         for (ll h = 0; h < k; ++h)
35         {
36             a[lf + h] = b[h];
37         }
38     }
39 }

```

```

40 signed main()
41 {
42     scanf("%lld", &n);
43     for (ll i = 1; i <= n; ++i)
44     {
45         scanf("%lld", &a[i]);
46     }
47     mergesort(1, n);
48     printf("%lld", ans);
49     return 0;
50 }

```

例题蓝桥-小朋友排队:将小朋友从低到高排序,每次只能交换相邻两个,交换使得他们不高兴程度增加,增加幅度是上次他增加的数量+1(一开始的上次是0),求排序后他们的不高兴程度之和的最小值。

总不高兴程度最小时,一个小朋友被交换次数等于他前边比他高的人数+后边比他低的人数,由此可以套用归并排序求逆序对的公式并加以改变:

```

1 #include <bits/stdc++.h>
2 #define mn 100002
3 typedef long long ll;
4 struct pupil
5 {
6     ll v,m;
7 } a[mn],b[mn];
8 ll ans,n;
9 void merges(ll lf, ll rf)
10 {
11     if(lf==rf) return;
12     ll cf=lf+rf>>1;
13     merges(lf,cf);
14     merges(cf+1,rf);
15     ll p=lf,q=cf+1,t=lf;
16     while(p<=cf&&q<=rf)
17     {
18         if(a[p].v>a[q].v) a[q].m+=cf+1-p,b[t++]=a[q++];
19         else a[p].m+=q-1-cf,b[t++]=a[p++];
20     }
21     while(q<=rf) b[t++]=a[q++]; //+=写成=暴死
22     while(p<=cf) a[p].m+=q-1-cf,b[t++]=a[p++];
23     for(int i=lf;i<=rf;++i) a[i]=b[i];
24 }
25 signed main()
26 {
27     scanf("%lld",&n);
28     for(int i=0;i<n;++i) scanf("%lld",&a[i].v);
29     merges(0,n-1);
30     for(int i=0;i<n;++i) ans+=(a[i].m+1)*a[i].m>>1;
31     printf("%lld",ans);
32     return 0;
33 }

```

归并排序求其他内容:

- lc315-每个元素右边比它小的元素数

归并合并  $a_i \leq a_j$  时,  $j$  是第一个  $\leq$  的, 故  $[mid, j)$  全满足, 对当前  $i$  的原下标累加贡献。用离散化权值树状数组也行。

- lc493-翻转对-求二倍逆序对(前者两倍严格大于后者)的数目

归并合并前, 双指针累加符合的区间数, 算完了再合并当前的。

- lc327-区间和在  $[l, u]$  的区间个数

归并合并前, 双指针累加符合的区间数, 算完了再合并当前的。

## 二分

整数三分公式: (以求U型区间最小值/单调区间最小值为例)

```

1  while (lf < rf)
2  {
3      ll tri = (rf - lf) / 3;
4      ll lc = lf + tri, rc = rf - tri;
5      db lv = calc(lc), rv = calc(rc);
6      ans = min({ans, lv, rv});
7      if (lv < rv)
8      {
9          rf = rc - 1;
10     }
11    else
12    {
13        lf = lc + 1;
14    }
15 }
```

此三分无法处理  $lv==rv$ , 可以通过离散化去除连续段。

## 最长单调序列

LIS (longest increase sequence)

单调栈上二分 复杂度  $O(n \log n)$

Dilworth定理: 偏序集的最少反链划分数等于最长链的长度

即: 求一个序列最少能分成多少个最长不上升序列, 可以求最长上升序列的长度即为答案。其他情况类推。

```

1  #include <bits/stdc++.h>
2  #define MAXN 100002
3  #define y1 dy
4  using namespace std;
5  int n, a[MAXN], uf[MAXN], df[MAXN], ufl, dfl, x[MAXN], y[MAXN], x1, y1;
6  int main()
7  {
8      while(EOF!=scanf("%d", &a[++n])); //这样的n比真实+1
9      uf[++ufl]=df[++dfl]=a[1];
```

```

10    x[++x1]=y[++y1]=a[1];
11    for(int i=2;i<n;i++)
12    {
13        if(uf[uf1]>=a[i])uf[++uf1]=a[i];//最长不上升
14        else *upper_bound(uf+1,uf+1+uf1,a[i],greater<int>())=a[i];
15
16        if(df[df1]<=a[i])df[++df1]=a[i];//最长上升,与上面互为定律
17        else *lower_bound(df+1,df+1+df1,a[i])=a[i];
18
19        if(x[x1]>=a[i])x[++x1]=a[i];//最长下降
20        else *upper_bound(x+1,x+1+x1,a[i],greater<int>())=a[i];
21
22        if(y[y1]<=a[i])y[++y1]=a[i];//最长不下降,与上面互为定律
23        else *lower_bound(y+1,y+1+y1,a[i])=a[i];
24    }
25    printf("\n%d\n%d\n%d\n%d",uf1,df1,x1,y1);
26    return 0;
27 }

```

规律：

- 对降的(严格降、非严格降(即不升)), 一律用 `upper_bound` + `greater<>()` 求第一个小于它的, 将这个位置改成更大, 得更优解
- 对升的, 一律用 `lower_bound` 求第一个大于它的, 将这个位置改成更小, 得更优解

模板题

P1020 导弹拦截 输入至多  $10^5$  个数, 求最长不下降序列的长度和可以划分为多少个最长不下降序列(不是不下降子串)

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #define sc(x) scanf("%lld", &x)
5 #define mn 100010
6 ll n, a[mn], dpnd[mn], dpu[mn], ans, x, nnd, nu;
7 signed main()
8 {
9     while (EOF != scanf("%lld", &x)) a[++n] = x;
10    dpnd[++nnd] = dpu[++nu] = a[1];
11    for (ll i = 2; i <= n; ++i)
12    {
13        if (a[i] <= dpnd[nnd]) dpnd[++nnd] = a[i];
14        else *upper_bound(dpnd + 1, dpnd + 1 + nnd, a[i], greater<ll>()) =
a[i];
15        if (a[i] > dpu[nu]) dpu[++nu] = a[i];
16        else *lower_bound(dpu + 1, dpu + 1 + nu, a[i]) = a[i];
17    }
18    printf("%lld\n%lld", nnd, nu);
19    return 0;
20 }

```

例题 P2782 友好城市 南北岸有  $n$  个不同城市, 在不同坐标, 北岸每个城市与南岸一个城市相连, 连线不能相交, 求最多能留多少条连线  $1 \leq n \leq 2 \times 10^5, 0 \leq x \leq 10^6$

以一个岸为基准结构体排序，然后求另一个岸的最长单调上升子序列长度即可

### lc354-求长宽同时严格单调上升的最长子序列

先按  $w$  升序排序，再第二关键字将  $h$  降序排序，从而避免了相同下存在递增。再第二关键字将  $h$  降序排序，从而避免了相同下存在递增。

## 最长公共排列

LCS(longest common sequence)(见DP)

假设元素互不重复，将一个序列重定义为它的每个元素在另一个序列里出现的位置，即设  $h[a[i]] = i$ ，令  $b[i] = h[b[i]]$ 。那么，只需要求该序列的最长上升子序列即可。

### 模板题 P1439 最长公共子排列 ( $n \leq 10^5$ )

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #define sc(x) scanf("%lld", &x)
5 #define mn 100010
6 ll a[mn], b[mn], n, dp[mn], m, h[mn];
7 signed main()
8 {
9     sc(n);
10    for (ll i = 1; i <= n; ++i)
11    {
12        sc(a[i]);
13        h[a[i]] = i;
14    }
15    for (ll i = 1; i <= n; ++i)
16    {
17        sc(b[i]);
18        b[i] = h[b[i]];
19    }
20    dp[++m] = b[1];
21    for (ll i = 2; i <= n; ++i)
22    {
23        if (b[i] > dp[m])
24        {
25            dp[++m] = b[i];
26        }
27        else
28        {
29            *lower_bound(dp + 1, dp + 1 + m, b[i]) = b[i];
30        }
31    }
32    printf("%lld", m);
33    return 0;
34 }
```

## 二分答案

### 最小中位数矩阵

(SCNUOJ1464)有 $n \times n$ 矩阵 $a$ , 找一个 $k \times k$ 子阵, 使得在所有 $k$ 阶子阵里, 这个子阵的中位数(第 $\lfloor \frac{k^2}{2} + 1 \rfloor$ 大数)最小, 输出中位数。 $1 \leq k \leq n \leq 800, 0 \leq a_{i,j} \leq 10^9$

这题不能暴力枚举(或者用二维双指针法), 无法解决 $O(1)$ 找中位数所以都会TLE。复杂度是 $n^3 \log n$ 。

考虑二分, 设答案为二分对象, 显然这个值越大它的排位越大, 所以可以检验排位, 从一个方向逐渐逼近中位排位, 即 $\lfloor \frac{k^2}{2} + 1 \rfloor$ , 每次检验二分值, 使得答案逼近于唯一的真实答案。可以证明二分到的一定是有存在的最小的中位数。

```
1 #include <bits/stdc++.h> //https://oj.socoding.cn/p/1464
2 using namespace std;
3 #define sc(x) scanf("%lld", &x)
4 typedef long long ll;
5 #define repe(i, a, b) for (ll i = a; i <= b; ++i)
6 #define mn 804
7 ll n, k, a[mn][mn], s[mn][mn], lf, rf = 1e9, cf, res, t;
8 bool check()
9 { //利用前缀和快速检验中位数
10     repe(i, 1, n) repe(j, 1, n) s[i][j] = s[i - 1][j] + s[i][j - 1] - s[i - 1][j - 1] + (a[i][j] >= cf);
11     repe(i, k, n) repe(j, k, n)
12     {
13         t = s[i][j] + s[i - k][j - k] - s[i][j - k] - s[i - k][j];
14         if (t < k * k / 2 + 1)
15             return false;
16     }
17     return true;
18 }
19 signed main()
20 {
21     sc(n), sc(k);
22     repe(i, 1, n) repe(j, 1, n) sc(a[i][j]);
23     while (lf <= rf)
24     {
25         cf = (lf + rf) >> 1; //假设存在一个子矩阵的中位数是cf
26         if (check()) //如果以cf为中位数不少于k*k/2+1个元素大于它
27             lf = cf + 1, res = cf; //那么中位数大于等于cf
28         else
29             rf = cf - 1; //否则中位数小于cf
30     }
31     printf("%d", res);
32     return 0;
33 }
```

### 最大子串平均值

例题洛谷P1419-长为n的连续序列, 定义平均值为区间和除以区间长度, 求长度在 $[s,t]$ 之间的子序列的最大平均值 $1 \leq n \leq 10^5, 1 \leq s \leq t \leq n, -10^4 \leq a_i \leq 10^4$

答案满足单调性, 设区间 $[l, r]$ , 则其平均值 $x$ 的算式为:

$$\frac{\sum_{i=l}^r a_i}{r-l+1} = x, \text{ 即 } \sum_{i=l}^r a_i = x \times (r-l+1)$$

当某个区间平均值优于已算出的平均值x时，有：

$$\frac{\sum_{i=l}^r a_i}{r-l+1} \geq x, \text{ 即 } \sum_{i=l}^r a_i \geq x \times (r-l+1)$$

$$\text{即 } \sum_{i=l}^r a_i \geq \sum_{i=l}^r x, \text{ 即 } \sum_{i=l}^r (a_i - x) \geq 0$$

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef int ll;
4 #define MAXN 100002
5 ll n, s, t, q[MAXN];
6 double c[MAXN], lf = -1e4, rf = 1e4, cf, a[MAXN];
7 inline bool ck(double &v)
8 {
9     int l = 1, r = 0;
10    for (ll i = 1; i <= n; ++i)
11        c[i] = c[i - 1] + a[i] - v;
12    for (ll i = 1; i <= n; ++i)
13    {
14        if (i >= s) //单调队列法
15        {
16            while (r >= l && c[i - s] < c[q[r]])
17                --r;
18            q[++r] = i - s;
19        }
20        if (l <= r && q[l] < i - t)
21            ++l;
22        if (l <= r && c[i] >= c[q[l]])
23            return true;
24    }
25    return false;
26 }
27 signed main()
28 {
29     scanf("%d%d%d", &n, &s, &t);
30     for (ll i = 1; i <= n; ++i)
31         scanf("%lf", &a[i]);
32     while (rf - lf > 1e-5)
33     {
34         cf = (lf + rf) / 2;
35         if (ck(cf))
36             lf = cf;
37         else
38             rf = cf;
39     }
40     printf("%.3lf", lf);
41     return 0;
42 }
```

## 其他

### 点权最小最短路

洛谷P1462 无向加权图(可能有自环/重边), 而且点加权, 从1走到n, 在路径长不超过b的前提下, 使得经过点集点权的最大值最小, 求这个最小值  
点数 $n \leq 10^4$ , 边数 $m \leq 5 \times 10^4$ , 边权、点权、 $b \leq 10^9$

二分最小值, 把大于最小值的点都删了, 然后看看能不能有不超过  $b$  的最短路

### 环转链最短距离最大值

(SCNUOJ1479)有 $n$ 个点, 编号 $[1, n]$ , 第 $i$ 个点和下一个编号的点连第 $i$ 条无向边, 成环。有 $m$ 个点对 $(a, b)$ , 现在删掉一条边, 求删掉边后这 $m$ 个点对的两两最短路的最大值最少是多少。  
 $2 \leq n \leq 2 \times 10^5$ ,  $1 \leq m \leq 2 \times 10^5$ ,  $1 \leq a_i, b_i \leq n$ ,  $1 \leq i \leq m$ , 有多组测试, 保证  
 $\sum n, \sum m \leq 2 \times 10^5$

显然对一个点对 $(a, b)$ ,  $a \leq b$ , 最短路要么是 $b - a$ , 要么是 $n - (b - a)$ 。二分最小值, 然后把所有超过这个值的段区间标记(差分+前缀和)

```
1 #define sc(x) x = read()
2 #define mn 200010
3 ll n, m, e[mn][2], u, v, lf, rf, cf, res, s[mn];
4 bool check(ll k)
5 {
6     memset(s, 0, sizeof s);
7     for (ll i = 0; i < m; ++i)
8     {
9         //只能在[a,b)这一段删, 不然会走[a,b)使得答案>k
10        //所以不能走这一段的补集
11        if (e[i][1] - e[i][0] > k)
12        {
13            ++s[1];
14            --s[e[i][0]];
15            ++s[e[i][1]];
16        }
17        if (n - (e[i][1] - e[i][0]) > k)
18        {
19            ++s[e[i][0]];
20            --s[e[i][1]];
21        }
22    }
23    for (ll i = 1; i <= n; ++i)
24    {
25        s[i] += s[i - 1];
26        if (!s[i])
27        {
28            return true;
29        }
30    }
31    return false;
32 }
33 signed main()
34 {
35     while (EOF != scanf("%d%d", &n, &m))
```

```

36     {
37         for (ll i = 0; i < m; ++i)
38         {
39             sc(u), sc(v); //输入不保证a<=b所以要这样
40             e[i][0] = min(u, v), e[i][1] = max(u, v);
41         }
42         lf = 0, rf = n - 1;
43         while (lf <= rf)
44         {
45             cf = (lf + rf) >> 1;
46             if (check(cf)) //如果有答案, 那么找一下更小的
47             {
48                 rf = cf - 1;
49             }
50             else
51             {
52                 lf = cf + 1;
53             }
54         }
55         printf("%d\n", rf + 1);
56     }
57     return 0;
58 }

```

## 整体二分

主体思路：把多个离线查询一起解决

需要满足以下性质：

1. 询问的答案可以二分
2. 修改对判定答案的贡献相互独立，修改之间不影响效果
3. 修改若对判定答案有贡献，则贡献为一个确定的与判定标准无关的值
4. 贡献满足交换律、结合律、具有可加性
5. 题目允许离线算法

时间复杂度  $O(T \log n)$ 。当使用别的结构(如树状数组)时，叠加复杂度

例题洛谷P1527-给定  $n \times n$  ( $1 \leq n \leq 500$ ) 矩阵，有  $q$  ( $1 \leq q \leq 6 \times 10^4$ ) 次询问，每次求矩阵左上角  $(x_1, y_1)$  和右下角  $(x_2, y_2)$  的子矩阵的第  $k$  小数(不去重) ( $1 \leq a_{i,j} \leq 10^9$ )

整体二分+离散化+二维树状数组。复杂度为  $O((n^2 + q) \log^3 n)$

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define sc(x) scanf("%lld", &x)
4 typedef long long ll;
5 #define mn 510
6 #define mm 60010
7 ll n, t, n2, b[mn][mn];
8 struct matrix
9 {
10     ll i, j, v;

```

```

11     bool operator<(const matrix &r) const { return v < r.v; }
12 } a[mn * mn];
13 struct queries
14 {
15     ll ax, ay, bx, by, k;
16 } q[mm];
17 ll qid[mm], tmp1[mm], tmp2[mm], ans[mm], now[mm];
18 ll lowbit(ll x) { return x & -x; }
19 void add(ll x, ll y, ll v)
20 {
21     for (ll i = x; i <= n; i += lowbit(i))
22         for (ll j = y; j <= n; j += lowbit(j))
23             b[i][j] += v;
24 }
25 ll get(ll x, ll y)
26 {
27     ll res = 0;
28     for (ll i = x; i; i -= lowbit(i))
29         for (ll j = y; j; j -= lowbit(j))
30             res += b[i][j];
31     return res;
32 }
33 ll gets(queries p)
34 {
35     return get(p.bx, p.by) - get(p.ax - 1, p.by) - get(p.bx, p.ay - 1) +
36     get(p.ax - 1, p.ay - 1);
37 }
38 void solve(ll lf, ll rf, ll lc, ll rc)
39 {
40     if (lc > rc)
41         return;
42     if (lf == rf)
43     {
44         for (ll i = lc; i <= rc; ++i)
45             ans[qid[i]] = a[lf].v;
46         return;
47     }
48     ll cf = (lf + rf) >> 1;
49     for (ll i = lf; i <= cf; ++i)
50         add(a[i].i, a[i].j, 1);
51     ll cnt1 = 0, cnt2 = 0;
52     for (ll i = lc, u; i <= rc; ++i)
53     {
54         u = qid[i];
55         ll s = now[u] + gets(q[u]);
56         if (s >= q[u].k)
57             tmp1[++cnt1] = u;
58         else
59             tmp2[++cnt2] = u, now[u] = s; // 本质为u-s
60         ll j = lc - 1;
61         for (ll i = 1; i <= cnt1; ++i)
62             qid[++j] = tmp1[i];
63         for (ll i = 1; i <= cnt2; ++i)
64             qid[++j] = tmp2[i];
65         for (ll i = lf; i <= cf; ++i)

```

```

66         add(a[i].i, a[i].j, -1);
67         solve(lf, cf, lc, lc + cnt1 - 1);
68         solve(cf + 1, rf, lc + cnt1, rc);
69     }
70     signed main()
71 {
72     sc(n), sc(t);
73     for (ll i = 1; i <= n; ++i)
74     {
75         for (ll j = 1, v; j <= n; ++j)
76         {
77             sc(v);
78             a[++n2] = {i, j, v};
79         }
80     }
81     sort(a + 1, a + 1 + n2);
82     for (ll i = 1; i <= t; ++i)
83     {
84         sc(q[i].ax), sc(q[i].ay), sc(q[i].bx), sc(q[i].by), sc(q[i].k);
85         qid[i] = i;
86     }
87     solve(1, n2, 1, t);
88     for (ll i = 1; i <= t; ++i)
89         printf("%lld\n", ans[i]);
90     return 0;
91 }

```

洛谷P2617例题-给定长为  $n$  的数列  $a$ ，支持  $m$  次两种操作：

$1 \leq n, m \leq 10^5, 0 \leq a_i, y \leq 10^9$

- `Q l r k` 查询区间第  $k$  小
- `C x y` 将  $a_x$  改为  $y$

因为涉及修改，所以不要重排序操作。那么在修改前的询问是不受后续修改的影响的。初始修改放在操作的最前面。下面代码没有使用到离散化。

还有一个小要点，就是对于脏数据的处理。脏数据也就是会对计算结果产生影响的数据。对于本题而言，值大于等于  $mb$  的操作在  $[lb, mb)$  中是没有影响的，但是在  $[mb, ub)$  的操作中，值小于  $mb$  的操作会产生影响，但是显然如果减去一个  $k$ ，那么就可以消除这个影响。

```

1 #include <iostream>
2 #include <cstdio>
3 #include <vector>
4 using namespace std;
5 struct op
6 {
7     int type;
8     // type==0 Change i means position; j means ispositive; k means the
9     // number after change
10    // type==1 Query i means left; r means right; k means kth-number
11    int i, j, k, id;
12};
13 int n, m, a[10050], ans[10050], f[50050];

```

```

13  vector<op> q;
14  int lowbit(int x)
15  {
16      return x & -x;
17  }
18  void add(int x, int k)
19  {
20      while (x <= n)
21      {
22          f[x] += k;
23          x += lowbit(x);
24      }
25  }
26  int query(int k)
27  {
28      int ans = 0;
29      while (k > 0)
30      {
31          ans += f[k];
32          k -= lowbit(k);
33      }
34      return ans;
35  }
36  void solve(int lb, int ub, vector<op> &q)
37  {
38      vector<op> Left, Right;
39      int mb = (lb + ub) >> 1;
40      if (ub - lb == 1)
41      {
42          for (int i = 0; i < q.size(); i++)
43          {
44              if (q[i].type == 1)
45                  ans[q[i].id] = lb;
46          }
47          return;
48      }
49      else if (q.empty())
50          return;
51      for (int i = 0; i < q.size(); i++)
52      {
53          op tmp = q[i];
54          if (tmp.type == 0)
55          {
56              if (tmp.k < mb)
57              {
58                  add(tmp.i, tmp.j); // i:pos j:num
59                  Left.push_back(tmp);
60              }
61              else
62                  Right.push_back(tmp);
63          }
64          else
65          {
66              int kth = query(tmp.j) - query(tmp.i - 1);
67              if (kth >= tmp.k)
68                  Left.push_back(tmp);

```

```

69         else
70         {
71             tmp.k -= kth;
72             Right.push_back(tmp);
73         }
74     }
75 }
76 for (int x = 0; x < Left.size(); x++)
77 {
78     if (Left[x].type == 0)
79         add(Left[x].i, -Left[x].j);
80     solve(lb, mb, Left);
81     solve(mb, ub, Right);
82 }
83 int main()
84 {
85     scanf("%d%d", &n, &m);
86     for (int i = 1; i <= n; i++)
87     {
88         scanf("%d", &a[i]);
89         op tmp = {0, i, 1, a[i]};
90         q.push_back(tmp);
91     }
92     for (int x = 0; x < m; x++)
93     {
94         char cmd;
95         int i;
96         scanf("%s%d", &cmd, &i);
97         if (cmd == 'C')
98         {
99             int t;
100            scanf("%d", &t);
101            op tmp = {0, i, -1, a[i], 0};
102            q.push_back(tmp);
103            a[i] = t;
104            tmp = {0, i, 1, t, 0};
105            q.push_back(tmp);
106        }
107     }
108     else
109     {
110         int j, k;
111         scanf("%d%d", &j, &k);
112         op tmp = {1, i, j, k, x};
113         q.push_back(tmp);
114     }
115     for (int i = 0; i < m; i++)
116     {
117         ans[i] = -1;
118         solve(0, 1e9, q);
119         if (ans[i] != -1)
120             printf("%d\n", ans[i]);
121     }
122 }

```

## wqs二分

王钦石二分(Alien Trick)。

lc188/SCNUOI1069-有  $n$  ( $1 \leq n \leq 10^5$ ) 天, 当日股票价格  $p_i$  ( $1 \leq p_i \leq 10^9$ ), 每天可以买入或卖出股票, 但最多持有一股, 问最大收益。

设恰好完成  $k$  次买卖时, 最大收益是  $g_k$ 。设增量(/导数)为  $g'_k = g_k - g_{k-1}$  那么可以得到一个结论, 增量  $g'_k$  一定单调不降(不严格单调递增)。当  $k$  不断增大到最大  $n$  ( $n$  次交易即都当天买当天卖等价于 0 次交易)时,  $g_k$  会降到 0, 即  $g_0 = g_n = 0$ , 一阶导  $g'_k$  单调递减, 不难判断  $g$  呈倒 U 型。

任务是求出  $g$  数列(/离散函数)的最大值  $\max g$ 。用几何思路, 考虑作直线与  $g$  图像相切, 设枚举了斜率为  $c (c > 0)$  的直线, 过所有点  $g_i (1 \leq i \leq n)$  作  $n$  条直线。

由于  $c > 0$ , 所以  $y$  截距最大的直线一定是相切的。设有斜率为  $c$  过点  $(k, g_k)$  的直线, 不难计算得截距是  $b = g_k - kc$ 。回到要求的问题上, 可以将截距抽象成进行了  $k$  次买卖, 净利润为  $g_k$ , 但每次卖出需要付手续费  $c$ 。抽象为已知手续费  $c$ , 且不限交易次数, 求出最大的含手续费利润  $b$ 。

因为不限交易次数, 考虑贪心, 即只要能赚的钱不低于手续费就买卖一次。由此可以  $O(n)$  计算出对固定的斜率  $c$  得到的最大切线截距  $b = g_k - kc$  及其  $k$  值( $sellcnt$ )。对答案  $\max g = b + gk$  而言,  $b$  相等时要尽可能大的  $k$ , 所以在上述贪心时, 在利润最大化的同时, 应当再最大化交易次数。

注意到斜率一定不会超过增量的最大值即  $\max p$ , 故二分上界取  $\max p$  即可。分母为 1, 故折线斜率为整数, 即二分只需要在整数范围内即可。如果题给的  $k$  在  $g$  单调递减时取得(即最高点  $k' < k, g_{k'} > g_k$ ), 也就是说不限次数地贪心得到的  $sellcnt < k$ , 那么此时也可以直接不限次数贪心求出  $\max g$ , 只要  $p_i > p_{i-1}$  就可以前一天买今天卖。

时间复杂度是  $O(n \log \max p)$ 。

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 using ll = long long;
4 signed main()
5 {
6     ll k, n;
7     cin >> n >> k;
8     vector<ll> p(n);
9     for (int i = 0; i < n; ++i)
10    {
11        cin >> p[i];
12    }
13    ll lf = 1, rf = *max_element(p.begin(), p.end());
14    ll ans = -1;
15    while (lf <= rf)
16    {
17        ll cf = (lf + rf) >> 1; // 斜率
18        ll buycnt = 0, sellcnt = 0;
19        ll buyp = -p[0], sellp = 0;
20        for (int i = 1; i < n; ++i) // 贪心
21        {
22            if (sellp - p[i] >= buyp)
23            {
24                buyp = sellp - p[i];
25                buycnt = sellcnt;
26            }
27            if (buyp + p[i] - cf >= sellp)
```

```

28         {
29             sellp = buyp + p[i] - cf;
30             sellcnt = buycnt + 1;
31         }
32     }
33     // cout << cf << ' ' << sellp << ' ' << sellcnt << '\n';
34     if (sellcnt >= k)
35     {
36         ans = sellp + k * cf; // 注意不是+sellcnt, 是+k
37         lf = cf + 1;
38     }
39     else
40     {
41         rf = cf - 1;
42     }
43 }
44 if (ans == -1)
45 {
46     ans = 0;
47     for (int i = 1; i < n; ++i)
48     {
49         ans += max(p[i] - p[i - 1], 0LL);
50     }
51 }
52 cout << ans;
53 return 0;
54 }
55 /*
56 8 3
57 1 100 1 4 1 4 1 4
58 */

```

## 前缀和/差分

树上前缀和/差分及其例题见图论-树-LCA应用。

高维前缀和可以考虑压缩数组，叠前缀和时可以容斥原理，也可以逐维叠加，即以每个维度进行一次一维前缀和，其他维度不变。

对  $a = (1, 1, \dots)$  的  $k$  阶前缀和  $s_n$  是  $\frac{n(n+1)\cdots(n+k-1)}{k!} = C_{n+k-1}^k$ 。

对任意  $a$  的  $y$  阶前缀和  $s_{y,x} = \sum_{i=1}^x C_{y+x-i-1}^{y-1} a_i = \sum_{i=1}^x C_{x-i}^{y-1} a_i$ ，可以  $O(x)$  求。要求一段前缀和的和，可以求它更高一阶的两边界值。

### 常见应用：

一维前缀和/差分：

- (SCNUOJ1410猜数)有一个目标数字，猜了若干次，反馈结果为比目标大//小//相等，问结果里最多多少个是对的  
把每个猜的结果转化为坐标轴的一段猜对目标数字取值区间，用差分累加所有区间，然后离散化枚举每个区间边界，被多少个区间覆盖猜对了多少次， $O(n \log n)$
- (蓝桥-最大子阵)求矩阵的最大非空子矩阵的元素和

先计算每列一维前缀和，然后枚举子矩阵最下行  $i$ ，枚举行跨度  $j(\leq i)$ ，枚举列  $k$ ，逐列累加跨  $j$  行后的新元素和，如果更大加入答案，如果当前和小于 0 舍弃全部，否则继续累加，复杂度  $O(n^2m)$

```

1 for(int i=1;i<=n;++i) for(int j=1;j<=m;++j)
2     scanf("%d",&v),s[i][j]=s[i-1][j]+v;//当前行前缀和
3 for(int i=1;i<=n;++i) for(int j=1;j<=i;++j) //矩阵右下角为(i,j)
4     for(int k=1,v=0;k<=m;++k) //使用O(m)计算二维前缀和
5     {
6         v+=s[i][k]-s[j-1][k];//当前矩阵前缀和
7         mx=max(mx,v);
8         v=max(v,0); //负数特判
9     }

```

- (蓝桥-k倍区间) 求有多少个连续子序列的和是  $k$  的倍数

是  $k$  的倍数  $\rightarrow$  前缀和模  $k$  得 0  $\rightarrow s[r] \equiv s[l-1] \pmod k$ ，即求多少个相等前缀和对，即(解法一)  $\sum_{i=1}^k C_2^{s[i]}$  ( $O(n+k)$ ) 或(解法二)每次求得新相等时与原本相等数累加。注意特判本有  $s[0] = 1$ 。若  $k$  过大，可以选择离散化

```

1 scanf("%d%d",&n,&k);
2 for(int i=1;i<=n;++i)
3     scanf("%d",&v),(cnt+=v)%=k,sum+=vis[cnt]++;
4 printf("%lld",sum+vis[0]);

```

- (SCNUOJ1193子段异或) 求异或和为 0 的连续子段数

异或为 0  $\rightarrow$  前缀异或满足  $s[r] = s[l-1]$ ，与上题类似， $O(n \log a)$  (离散化)

```

1 for(scanf("%lld",&n),++m[0];n--n) //m:map
2     scanf("%lld",&v),ans+=m[c^=v]++;
3     return printf("%lld",ans)&0;

```

- (SCNUOJ1433周游山区)  $n$  个加油站成环，离顺时针下一个加油站距离  $d_i$ ，可加油  $p_i$ ，问从每个加油站出发(本站油量为起始油量)能否顺或逆时针走完一圈

走完一圈后油量可以为 0，但没走完之前不可以为 0；但是如果考虑子问题，那么只需要考虑大于等于 0，在顺时针时，从  $i$  出发，可以走回  $i$  的条件为同时满足以下等式：(需要注意  $d_0 = 0$ )， $O(n)$

$$\begin{cases} (p_i - d_i) \geq 0 \\ (p_i - d_i) + (p_{i+1} - d_{i+1}) \geq 0 \\ \dots \\ (p_i - d_i) + (p_{i+1} - d_{i+1}) + \dots + (p_{i+n-1} - d_{i+n-1}) \geq 0 \end{cases}$$

设前缀和  $s_n = \sum_{i=1}^n p_i - d_i$ ，即需要同时满足：

$$\begin{cases} s_i - s_{i-1} \geq 0 \\ s_{i+1} - s_{i-1} \geq 0 \\ \dots \\ s_{i+n-1} - s_{i-1} \geq 0 \end{cases}$$

即在  $[i, i + n - 1]$  内， $s_{\min} - s_{i-1} \geq 0$ ，逆时针同理：

$$(s_i - s_j)_{\min} \geq 0 \quad j \in [i - n + 1, i] \Rightarrow s_i - s_{\max} \geq 0$$

可以用单调队列维护一个最小值(逆则最大)。注意单调区间长度不可以大于等于  $n$

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #define mn 2000002
5 #define nemp hd<=ed
6 ll q[mn], hd, ed=-1, s[mn], p[mn], d[mn], n, n2;
7 bool ok[mn];
8 signed main()
9 {
10     scanf("%lld", &n); n2=n<<1;
11     for(ll i=1; i<=n; ++i) scanf("%lld%lld", p+i, d+i), s[i]=s[i+n]=p[i]-d[i];
12     for(ll i=1; i<=n2; ++i) s[i]+=s[i-1];
13     for(ll i=n2; i; --i)
14     {
15         while(nemp&&q[hd]-i>=n) ++hd;
16         while(nemp&&s[q[ed]]>=s[i]) --ed;
17         q[++ed]=i;
18         if(i<=n&&s[q[hd]]-s[i-1]>=0) ok[i]=true;
19     } hd=0, ed=-1; d[0]=d[n];
20     for(ll i=1; i<=n; ++i) s[i]=s[i+n]=p[i]-d[i-1];
21     for(ll i=1; i<=n2; ++i) s[i]+=s[i-1];
22     for(ll i=1; i<=n2; ++i)
23     {
24         while(nemp&&i-q[hd]>=n) ++hd;
25         if(i>n&&s[i]-s[q[hd]]>=0) ok[i-n]=true;
26         while(nemp&&s[q[ed]]<=s[i]) --ed;
27         q[++ed]=i;
28     }
29     for(ll i=1; i<=n; ++i) printf(ok[i]?"YES\n": "NO\n");
30     return 0;
31 }

```

二维前缀和/差分：

- (SCNUOJ1226守卫农田)矩阵内有若干个子矩形被覆盖，若干次询问某个子矩阵是否都被覆盖  
覆盖过程可以二维差分；得出布尔值化的原数组后再求一个二维前缀和，查询时就能  $O(1)$  比较子矩阵面积与被覆盖大小  $O(n^2)$

高维前缀和/差分：

- (蓝桥-三体攻击)有立方体舰队阵列，每次攻击使一个子立方体全员受到定值伤害，求首次出现死亡时是第几次攻击

二分答案+三维差分。把攻击做成差分数组，二分攻击次数，判断是否出现负数即可  $O(n^3 \log t)$

前缀和/差分思想：

- (洛谷P2926)给定  $n$  个值域在  $w$  的可重复数，求每个数有多少数能被它整除  
先计数每个数出现了几次，然后对每个因子其所有倍数叠加一遍， $O(n \log n)$

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef int ll;
4 #define MAXN 1000002
5 ll ans[MAXN], n, a[MAXN], mx, vis[MAXN];

```

```

6  signed main()
7  {
8      scanf("%d", &n);
9      for (ll i = 1; i <= n; ++i)
10         scanf("%d", &a[i]), ++vis[a[i]], mx = max(a[i], mx);
11     for (ll i = 1; i <= mx; ++i)
12     {
13         if (!vis[i])
14             continue;
15         for (ll j = 1; j * i <= mx; ++j)
16             if (vis[i * j])
17                 ans[i * j] += vis[i];
18             --ans[i];
19     }
20     for (ll i = 1; i <= n; ++i)
21         printf("%d\n", ans[a[i]]);
22     return 0;
23 }

```

## 搜索

调整 BFS 的顺序，可以达到最短路径字典序最小等要求。对未知有限步数，有时可以设一个大数作为搜索深度边界。遇事不决剪枝记忆化。

一些数字类/字典序/倍数因数的题目也可以搜索每个位数来逐个求解所有可能的值。

### IDA\*

启发式搜索不保证绝对准确性，建立在其基础之上的A\*和IDA\*也不保证绝对的准确性。因为估价函数的估计值并不一定是答案。

其中IDA\*是每次限定搜索深度(逐次增加)的DFS来模拟BFS。好处是空间开销小。不加A\*的该方法为迭代加深搜索

洛谷P1379 把  $3 \times 3$  棋盘 1 ~ 8 数字 (0 是空格) 转化为 123804765 状态，问最小步数

这题也可以BFS+记忆化，双向BFS，康托展开来做

```

1 #include<iostream>
2 #include<string>
3 #include<map>
4 #include<cmath>
5 #include<algorithm>
6 #include<queue>
7 #include<cstring>
8 #include<cstdio>
9 using namespace std;
10 typedef long long lt;
11 int read()
12 {
13     int f=1,x=0;
14     char ss=getchar();
15     while(ss<'0'||ss>'9') {if(ss=='-') f=-1; ss=getchar();}
16     while(ss>='0'&&ss<='9') {x=x*10+ss-'0'; ss=getchar();}
17     return f*x;

```

```

18 }
19 char ss[15];
20 int ans[4][4]=
21 {{0,0,0,0},
22 {0,1,2,3},
23 {0,8,0,4},
24 {0,7,6,5}};
25 int a[5][5],k,judge;
26 int nxtx[]={0,1,-1,0};//特殊构造
27 int nxty[]={1,0,0,-1};//pre+i==3时两两互逆,即pre和i互为相反步
28 int check()
29 {
30     for(int i=1;i<=3;++i)
31         for(int j=1;j<=3;++j)
32             if(ans[i][j]!=a[i][j])return 0;
33     return 1;
34 }
35 int test(int step)//预测还要走多少步,启发假设有一个不等就走多一步
36 {
37     int cnt=0;
38     for(int i=1;i<=3;++i)
39         for(int j=1;j<=3;++j)
40             if(ans[i][j]!=a[i][j]) { if(++cnt+step>k) return 0; }
41     return 1;
42 }
43 void A_star(int step,int x,int y,int pre)//pre防止下一次走回上一步
44 {
45     if(step==k){ if(check())judge=1; return; } //达到当前限制的最大深度
46     if(judge) return;
47     for(int i=0;i<4;++i)
48     {
49         int nx=x+nxtx[i],ny=y+nxty[i];
50         if(nx<1||nx>3||ny<1||ny>3||pre+i==3) continue; //加入了上述最优性剪枝
51         swap(a[x][y],a[nx][ny]);
52         if(test(step)&&!judge) A_star(step+1,nx,ny,i); //A*估价合法再向下搜索
53         swap(a[x][y],a[nx][ny]);
54     }
55 }
56 int main()
57 {
58     int x,y;
59     scanf("%s",&ss);
60     for(int i=0;i<9;++i)
61     {
62         a[i/3+1][i%3+1]=ss[i]-'0';
63         if(ss[i]-'0'==0)x=i/3+1,y=i%3+1;
64     }
65     if(check()){printf("0");return 0;} //特判不用移动
66     while(++k)//枚举最大深度
67     {
68         A_star(0,x,y,-1);
69         if(judge){printf("%d",k);break;}
70     }
71     return 0;
72 }

```

## 双向搜索

对BFS而言，将规模 $a^n$ 缩减为 $2a^{\frac{n}{2}}$

从起点开始和从终点开始的点分别标注为1和2，那么相加为3必然相遇。

while条件为两队列均不为空还是for和dx,dy搜索。代码类似BFS，很好写

## 折半搜索

对DFS而言，第一次搜索前半部分，第二次搜索后半部分，最后再组合两部分，将搜索的复杂度指数降低一半，并将空间复杂度提升到时间复杂度等同

洛谷P4799-有n场比赛，可用m元，比赛入场需要门票钱x，请问他有多少种参加方案。

$$1 \leq n \leq 40, 1 \leq m \leq 10^{18}, 1 \leq x \leq 10^{16}$$

第一次搜索前一半比赛并将小于m的全部花费记录。第二次搜索后一半比赛并同样记录，得到的记录数目为 $2 \times 2^{\frac{n}{2}}$ ，时间复杂度也是如此(01背包型搜索)。最后合并，取余额为 $m - a_i$ ，即减去第一次搜索得到的一个花销和，然后在第二次二分搜索(排序后)里寻找大于等于余额的下标，下标即方案数。这里运用了前缀和思维的计数，使用 $O(len \log len)$ 的复杂度完成了合并。总时间复杂度为 $O(2^{\frac{n}{2}+1} \log 2^{\frac{n}{2}}) = O(n2^{\frac{n}{2}})$ ，空间复杂度为 $O(2^{\frac{n}{2}+1})$

```
1 #include <bits/stdc++.h>
2 typedef long long ll;
3 using namespace std;
4 #define mn (1<<21)+5
5 ll n,m,x[62],ac,bc,as[mn],bs[mn],cf,ans;
6 void dfs(ll lf, ll& rf, ll v, ll& cnt, ll* arr)
7 {
8     if(v>m) return;
9     if(lf==rf){arr[cnt++]=v; return;}
10    dfs(lf+1,rf,v+x[lf],cnt,arr);
11    dfs(lf+1,rf,v,cnt,arr);
12 }
13 signed main()
14 {
15     scanf("%lld%lld",&n,&m); cf=n>>1;
16     for(int i=0;i<n;++i) scanf("%lld",x+i);
17     dfs(0,cf,0,ac,as);
18     dfs(cf,n,0,bc,bs);
19     sort(bs,bs+bc);
20     for(int i=0;i<ac;++i)
21         ans+=upper_bound(bs,bs+bc,m-as[i])-bs;
22     return printf("%lld",ans),0;
23 }
```

洛谷P3067-给定n个数x，选出一些数划分为两堆和相等的数，求方案数。(

$$2 \leq n \leq 20, 1 \leq x \leq 10^8$$

显然不会爆int。由于划分的两堆数数目不定，所以需要使用搜索，单纯搜索时，有三种分支，即当前数不加入，当前数放左堆，当前数放右堆，是三进制背包DFS，复杂度为 $O(3^n)$ 不通过。

考虑折半搜索。并考虑利用状态压缩、vis数组去重和map离散化。第一次搜索前面一半的数，设sum为左堆-右堆的差值，第二次设sum为右堆-左堆的差值，如果第二次的sum等于第一次的，证明合起来之后，这两堆的和相等。最后统计有效状态即可。

时间复杂度为 $O(3^{\frac{n}{2}})$ ，空间复杂度为 $O(2^n)$ ，空间复杂度来自于状态压缩(选了什么数)。

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 #define mn (1<<20)+5
4 int n, half, mc, ans, a[21];
5 bool memo[mn];
6 vector<int> d[mn]; //有多少种nowstate可以拼出情况mc (mc即nowsum)
7 map<int, int> m; //m[v] -> mc 求和值与情况mc一一对应
8 void dfs1(int i, int nowsum, int nowstate)
9 {
10     if(i==half)
11     {
12         if(!m.count(nowsum)) m[nowsum]=++mc; //和为nowsum时对应一个独一无二的mc
13         d[m[nowsum]].push_back(nowstate);
14         return;
15     }
16     dfs1(i+1, nowsum, nowstate); //两边都不选
17     dfs1(i+1, nowsum+a[i], nowstate|(1<<i)); //左边选了
18     dfs1(i+1, nowsum-a[i], nowstate|(1<<i)); //右边选了
19 }
20 void dfs2(int i, int nowsum, int nowstate) //这次+ -的意义跟上次相反
21 {
22     if(i==n)
23     {
24         if(m.count(nowsum)) //如果可以拼出nowsum
25         {
26             int x=m[nowsum];
27             for(int i=0, ds=d[x].size(); i<ds; ++i) memo[d[x]
28 [i]|nowstate]=true; //起到了去重
29         }
30         return;
31     }
32     dfs2(i+1, nowsum, nowstate);
33     dfs2(i+1, nowsum+a[i], nowstate|(1<<i)); //位运算(状态压缩)里i从0开始
34     dfs2(i+1, nowsum-a[i], nowstate|(1<<i));
35 }
36 signed main()
37 {
38     scanf("%d", &n);
39     half=n>>1;
40     for(int i=0; i<n; ++i) scanf("%d", a+i);
41     dfs1(0, 0, 0);
42     dfs2(half, 0, 0);
43     for(int i=1, n2=1<<n; i<=n2; ++i) ans+=memo[i];
44     return printf("%d", ans), 0;
45 }
```

## 随机化

$$(1 - \frac{1}{n})^n \leq \frac{1}{e}, \forall n \geq 1$$

意义：有  $n$  个相互独立的坏事件，且设它们的发生概率都是  $1 - \frac{1}{n}$ ，那么它们全部都发生的概率至多是  $\frac{1}{e} \approx 0.3679$

假设可以枚举到的全体解是  $U$ ， $|U| = n$ ，其中正解是  $S$ ，如果每次随机枚举一个解，期望正确率是  $1 - \epsilon$  (即错误率是  $\epsilon$ )，那么单次枚举有  $\frac{|S|}{n}$  概率得到正解，枚举  $n(-\log \frac{\epsilon}{|S|})$  次可以把全体  $S$  元素都枚举一次

### 爬山算法 / 模拟退火：

设枚举次数是  $m$ ，那么  $tC^m \approx t_0$  即  $m \approx \log(\frac{t_0}{t}, C)$ ，即复杂度为  $O\left(\log(\frac{t_0}{t}, C)\right)$

通常降温系数  $C \in [0.985, 0.999]$ ,  $t_0 = 10^{-15}$ ， $t$  一般看定义域，例如  $t = 10^4$

规定熵值越低解越优。通常可以跑多次。一种卡时方法：

```
1 | while ((double)clock() / CLOCKS_PER_SEC < MAX_TIME) simulateAnneal();
```

(SCNUOJ1679) 求满足  $x + y = k$  下  $\sqrt{p - a + x^2} + \sqrt{p - b + y^2}$  最小的解  $(x, y)$

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 typedef double db;
5 ll p, a, b, q;
6 db k, t = 0.5, d, minx = 0.5, nowx = 0.5;
7 db f(db x) // assume k=1
8 {
9     return sqrt(p - a + x * x) + sqrt(p - b + (k - x) * (k - x));
10 }
11 signed main()
12 {
13     scanf("%lld%lld%lld%lld", &p, &a, &b, &q);
14     while (q--)
15     {
16         scanf("%lf", &k);
17         minx = nowx = k / 2;
18         t = k / 2;
19         while (t > 1e-6) // t是温度参数
20         {
21             db x1 = nowx + t;
22             db x2 = nowx - t;
23             db newx = f(x1) < f(x2) ? x1 : x2;
24             nowx = newx;
25             if (f(newx) < f(minx))
26             {
27                 minx = newx;
28             }
29         }
30     }
31 }
```

```

27         minx = newx;
28     }
29     t *= 0.9; // 降温系数
30 }
31 printf("%lf %lf\n", minx, (k - minx));
32 }
33 return 0;
34 }

```

(SCNUOJ1730) 给定常数  $a, b, c, d, e, l, r$  , 求下列函数在定义域  $[l, r]$  的最大值:

$$f(x) = \sin\left(\frac{x}{a}\right) + \sin\left(\frac{x}{b}\right) + \sin\left(\frac{x}{c}\right) + \sin\left(\frac{x}{d}\right) + \sin\left(\frac{x}{e}\right)$$

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define sc(x) scanf("%lld", &x)
4 typedef long long ll;
5 typedef double db;
6 ll a, b, c, d, e, l, r;
7 db x, t, now, mx, ans = -6;
8 db f(db x)
9 {
10     return sin(x / a) + sin(x / b) + sin(x / c) + sin(x / d) + sin(x / e);
11 }
12 void solve()
13 {
14     static ll rd = 1; //也可以用random_device生成种子
15     mt19937 mt(rd + time(0));
16     rd *= 2;
17     uniform_real_distribution<db> dist0(0, 1);
18     now = (l + r) / 2, t = r - l;
19     while (t > 1e-6)
20     {
21         uniform_real_distribution<db> dist(-t, t);
22         db newx = now + dist(mt);
23         newx = max(1. * l, min(1. * r, newx));
24         db fnow = f(now), fnew = f(newx);
25         db dt = (-fnew) - (-fnow);
26         if (fnow < fnew || exp(-dt / t) > dist0(mt))
27         {
28             now = newx;
29         }
30         if (ans < fnew)
31         {
32             ans = fnew;
33         }
34         t *= 0.999;
35     }
36     for (ll i = 0; i < 1000; ++i) //在终温随机多次
37     { //优化效果不明显
38         uniform_real_distribution<db> dist(-t, t);
39         db newx = now + dist(mt);
40         newx = max(1. * l, min(1. * r, newx));

```

```

41     db fnow = f(now), fnew = f(newx);
42     if (fnow < fnew)
43     {
44         now = newx;
45     }
46     if (ans < fnew)
47     {
48         ans = fnew;
49     }
50 }
51 }
52 signed main()
53 {
54     sc(a), sc(b), sc(c), sc(d), sc(e), sc(l), sc(r);
55     for (ll i = 0; i < 20; ++i) //重要: 多次退火
56     {
57         solve();
58     }
59     printf("%lf", ans);
60     return 0;
61 }

```

洛谷P2503-将  $n$  个数分成  $m$  组, 组内求和得到  $m$  个数, 求最小方差  $m \leq n \leq 20, 2 \leq m \leq 6$

对某个序列, 每次将当前数贪心放进  $m$  组里和最小的一组, 可以证明是对确定序列得到最小方差的; 然后用模拟退火不断打乱序列即可, 对每个位置退火 1 次, 共退火  $n$  次

Ic1157-有数组  $n$  ( $\leq 2 \times 10^4$ ) 和  $q$  ( $\leq 10^4$ ) 次询问, 每次问子区间  $[l, r]$  的出现频率不低于  $t$  ( $t$  至少区间一半长) 的元素是什么。

因为出现次数至少一半, 随机枚举 20 次, 几乎不可能取不到是该数字。每次枚举, 通过 lower, upper bound 对该元素的出现下标有序数组搜索得到出现频次。

解法二: 使用 Boyer-Moore 投票算法  $O(n)$  在已知存在众数的情况下求出众数, 利用投票算法套线段树维护信息。

## 莫队

### 普通

洛谷P3901-静态  $n$  长数列  $a$  ( $1 \leq n, a_i \leq 10^5$ ), 有  $m$  ( $1 \leq m \leq 10^5$ ) 次询问, 问区间内是否每个数互不相同

复杂度  $O(n\sqrt{m})$ , 三种正确的循环位置: `l--, r--, r++, l++` , `l--, r++, r--, l++` , `l--, r++, l++, r--` (前两步先扩大区间(`l--, r++`), 后两步缩小区间(`l++, r--`))

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define sc(x) scanf("%lld", &x)
4 typedef long long ll;
5 #define mn 100010

```

```

6  ll n, m, x, bin[mn], ans[mn], sq, a[mn], sum;
7  struct query
8  {
9      ll l, r, i;
10     bool operator<(const query &x) const
11     {
12         if (l / sq != x.l / sq)
13             return l < x.l;
14         if (l / sq & 1)
15             return r < x.r;
16         return r > x.r;
17     }
18 } q[mn];
19 void add(ll i)
20 {
21     sum += bin[a[i]]++ == 0;
22 }
23 void del(ll i)
24 {
25     sum -= --bin[a[i]] == 0;
26 }
27 signed main()
28 {
29     sc(n), sc(m), sq = sqrt(n);
30     for (ll i = 1; i <= n; ++i)
31     {
32         sc(a[i]);
33     }
34     for (ll i = 1; i <= m; ++i)
35     {
36         sc(q[i].l), sc(q[i].r), q[i].i = i;
37     }
38     sort(q + 1, q + 1 + m);
39     for (ll i = 1, l = 1, r = 0; i <= m; ++i)
40     {
41         while (l > q[i].l)
42             add(--l);
43         while (r < q[i].r)
44             add(++r);
45         while (l < q[i].l)
46             del(l++);
47         while (r > q[i].r)
48             del(r--);
49         ans[q[i].i] = sum == (q[i].r - q[i].l + 1);
50     }
51     for (ll i = 1; i <= m; ++i)
52     {
53         printf("%s\n", ans[i] ? "Yes" : "No");
54     }
55     return 0;
56 }

```

## 带修

洛谷1903 有  $n$  长序列  $a(a_i \leq 10^6)$ ，有  $m(n, m \leq 133333)$  次操作： $Q l r$  表示询问区间  $[l, r]$  内不同数值数； $R p c$  表示将  $a_p$  修改为  $c$ 。

复杂度  $O(n^{\frac{5}{3}})$

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 #define sc(x) scanf("%lld", &x)
4 #define ll long long
5 #define mn 133340
6 ll n, m, sq, a[mn], a0[mn], ans[mn], m1, m2, sum, bin[1000010];
7 struct query1
8 {
9     ll l, r, i, c;
10    bool operator<(const query1 &x) const
11    {
12        if (l / sq == x.l / sq)
13        {
14            return r / sq == x.r / sq ? i < x.i : r < x.r;
15        }
16        return l < x.l;
17    }
18} q1[mn];
19 struct query2
20 {
21     ll i, f, t;
22} q2[mn];
23 void add(ll i)
24 {
25     sum += bin[i]++;
26 }
27 void del(ll i)
28 {
29     sum -= --bin[i];
30 }
31 ll l = 1, r;
32 void addt(ll t)
33 {
34     if (l <= q2[t].i && q2[t].i <= r)
35         del(q2[t].f), add(q2[t].t);
36     a0[q2[t].i] = q2[t].t;
37 }
38 void delt(ll t)
39 {
40     if (l <= q2[t].i && q2[t].i <= r)
41         del(q2[t].t), add(q2[t].f);
42     a0[q2[t].i] = q2[t].f;
43 }
44 signed main()
45 {
46     sc(n), sc(m), sq = pow(n, 2. / 3);
47     for (ll i = 1; i <= n; ++i)
48     {
```

```

49         sc(a[i]), a0[i] = a[i];
50     }
51     char ch[5] = {};
52     for (ll i = 1, x, y; i <= m; ++i)
53     {
54         scanf("%s%lld%lld", ch, &x, &y);
55         if (ch[0] == 'Q')
56         {
57             ++m1, q1[m1] = {x, y, m1, m2 + 1};
58         }
59         else
60         {
61             ++m2, q2[m2] = {x, a[x], y}, a[x] = y;
62         }
63     }
64     sort(q1 + 1, q1 + 1 + m1);
65     for (ll i = 1, t = 1; i <= m1; ++i)
66     {
67         while (t < q1[i].c)
68             addt(t++);
69         while (t > q1[i].c)
70             delt(--t);
71         while (l > q1[i].l)
72             add(a0[--l]);
73         while (r < q1[i].r)
74             add(a0[++r]);
75         while (l < q1[i].l)
76             del(a0[l++]);
77         while (r > q1[i].r)
78             del(a0[r--]);
79         ans[q1[i].i] = sum;
80     }
81     for (ll i = 1; i <= m1; ++i)
82     {
83         printf("%lld\n", ans[i]);
84     }
85     return 0;
86 }

```

## 树上

子树:

例题洛谷U41492:  $n(n \leq 10^5)$  个节点的树, 每个节点有不同的颜色( $10^5$  种), 问每个节点为根的子树中, 所有节点有多少种不同的颜色;  $m(m \leq 10^5)$  次询问, 每次问一棵子树

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define sc(x) scanf("%lld", &x)
4 typedef long long ll;
5 #define mn 100010
6 ll n, m, sq, ans[mn], bin[mn], sum, a[mn], hd[mn], cnt;
7 ll dfn[mn], st, mx[mn], id[mn];
8 struct query

```

```

9  {
10     ll l, r, i;
11     bool operator<(const query &x) const
12     {
13         if (l / sq != x.l / sq)
14             return l < x.l;
15         return l / sq & 1 ? r < x.r : r > x.r;
16     }
17 } q[mn];
18 struct edge
19 {
20     ll to, nx;
21 } e[mn * 2];
22 signed main()
23 {
24     sc(n), sq = sqrt(n);
25     auto adde = [&](ll u, ll v)
26     { e[++cnt] = {v, hd[u]}, hd[u] = cnt; };
27     for (ll i = 1, u, v; i < n; ++i)
28     {
29         sc(u), sc(v), adde(u, v), adde(v, u);
30     }
31     for (ll i = 1; i <= n; ++i)
32     {
33         sc(a[i]);
34     }
35     auto dfs = [&](auto self, ll u, ll fa) -> ll
36     {
37         dfn[u] = ++st, mx[u] = st, id[st] = a[u];
38         for (ll i = hd[u], v; i; i = e[i].nx)
39         {
40             if ((v = e[i].to) != fa)
41             {
42                 mx[u] = max(mx[u], self(self, v, u));
43             }
44         }
45         return mx[u];
46     };
47     dfs(dfs, 1, 0);
48
49     sc(m);
50     for (ll i = 1, x; i <= m; ++i)
51     {
52         sc(x), q[i].l = dfn[x], q[i].r = mx[x], q[i].i = i;
53     }
54     sort(q + 1, q + 1 + m);
55     auto add = [&](ll i)
56     { sum += bin[id[i]]++ == 0; };
57     auto del = [&](ll i)
58     { sum -= --bin[id[i]] == 0; };
59     for (ll i = 1, l = 1, r = 0; i <= n; ++i)
60     {
61         while (l > q[i].l)
62             add(--l);
63         while (r < q[i].r)
64             add(++r);

```

```

65     while (l < q[i].l)
66         del(l++);
67     while (r > q[i].r)
68         del(r--);
69     ans[q[i].i] = sum;
70 }
71 for (ll i = 1; i <= m; ++i)
72 {
73     printf("%lld\n", ans[i]);
74 }
75 return 0;
76 }

```

链:

SP10707 有  $n \leq 4 \times 10^4$  节点树, 每个点一种颜色  $\leq 2 \times 10^9$ , 有  $m \leq 10^5$  次询问, 问  $u, v$  路径上不同颜色数

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define sc(x) scanf("%lld", &x)
4 typedef long long ll;
5 #define mn 40010
6 #define mm 100010
7 #define mlg 17
8 ll n, m, sq;
9 ll a[mn], b[mn], bs;
10 ll hd[mn], cnt;
11 struct edge
12 {
13     ll to, nx;
14 } e[mn * 2];
15 ll ocnt, ord[mn * 2], st[mn], nd[mn], fa[mn][mlg], dep[mn], lg[mn];
16 struct query
17 {
18     ll l, r, i, c;
19     bool operator<(const query &x) const
20     {
21         if (l / sq != x.l / sq)
22             return l < x.l;
23         return l / sq & 1 ? r < x.r : r > x.r;
24     }
25 } q[mm];
26 ll bin[mn * 2], sum, vis[mn], ans[mm];
27 signed main()
28 {
29     sc(n), sc(m), sq = sqrt(n);
30
31     for (ll i = 1; i <= n; ++i)
32     {
33         sc(a[i]), b[i] = a[i];
34     }
35     sort(b + 1, b + 1 + n);
36     bs = unique(b + 1, b + 1 + n) - (b + 1);

```

```

37     for (ll i = 1; i <= n; ++i)
38     {
39         a[i] = lower_bound(b + 1, b + 1 + bs, a[i]) - b;
40     }
41
42     auto adde = [&](ll u, ll v)
43     {e[++cnt] = {v, hd[u]}; hd[u] = cnt; };
44     for (ll i = 1, u, v; i < n; ++i)
45     {
46         sc(u), sc(v), adde(u, v), adde(v, u);
47     }
48     for (ll i = 1; i <= n; ++i)
49     {
50         lg[i] = lg[i / 2] + 1;
51     }
52     auto dfs = [&](auto self, ll u) -> void
53     {
54         ord[++ocnt] = u;
55         st[u] = ocnt;
56         for (ll i = hd[u], v; i; i = e[i].nx)
57         {
58             if ((v = e[i].to) != fa[u][0])
59             {
60                 dep[v] = dep[u] + 1;
61                 fa[v][0] = u;
62                 for (ll i = 1; i <= lg[dep[v]]; ++i)
63                 {
64                     fa[v][i] = fa[fa[v][i - 1]][i - 1];
65                 }
66                 self(self, v);
67             }
68         }
69         ord[++ocnt] = u;
70         nd[u] = ocnt;
71     };
72     dfs(dfs, 1);
73
74     auto lca = [&](ll u, ll v)
75     {
76         if (dep[u] < dep[v])
77         {
78             swap(u, v);
79         }
80         while (dep[u] > dep[v])
81         {
82             u = fa[u][lg[dep[u] - dep[v]] - 1];
83         }
84         if (u == v)
85         {
86             return u;
87         }
88         for (ll i = lg[dep[u]] - 1; i >= 0; --i)
89         {
90             if (fa[u][i] != fa[v][i])
91             {
92                 u = fa[u][i], v = fa[v][i];

```

```

93         }
94     }
95     return fa[u][0];
96 };
97
98 for (ll i = 1, l, r, c; i <= m; ++i)
99 {
100     sc(l), sc(r), c = lca(l, r), q[i].i = i;
101     if (st[l] > st[r])
102     {
103         swap(l, r);
104     }
105     if (l == c)
106     {
107         q[i].l = st[l], q[i].r = st[r];
108     }
109     else
110     {
111         q[i].l = nd[l], q[i].r = st[r], q[i].c = c;
112     }
113 }
114
115 sort(q + 1, q + 1 + m);
116 auto work = [&](ll i)
117 {
118     if (vis[i])
119     {
120         sum -= --bin[a[i]] == 0;
121     }
122     else
123     {
124         sum += bin[a[i]]++ == 0;
125     }
126     vis[i] ^= 1;
127 };
128 for (ll i = 1, l = 1, r = 0; i <= m; ++i)
129 {
130     while (l < q[i].l)
131         work(ord[l++]);
132     while (l > q[i].l)
133         work(ord[--l]);
134     while (r < q[i].r)
135         work(ord[++r]);
136     while (r > q[i].r)
137         work(ord[r--]);
138     if (q[i].c)
139         work(q[i].c);
140     ans[q[i].i] = sum;
141     if (q[i].c)
142         work(q[i].c);
143 }
144 for (ll i = 1; i <= m; ++i)
145 {
146     printf("%lld\n", ans[i]);
147 }
148 return 0;

```

## 回滚

AT1219 给定长为  $n$  的数组  $a$  ( $1 \leq a_i \leq 10^9$ ) 和  $q$  ( $1 \leq n, q \leq 10^5$ ) 次询问，每次问  $[l, r]$  内最大重要度。数字  $x$  重要度等于  $x$  乘以它的出现次数

复杂度  $O(n\sqrt{m})$ ，按顺序处理询问：

- 如果询问左端点所属块  $b$  和上一个询问左端点所属块的不同，那么将莫队区间的左端点初始化为  $b$  的右端点加 1，将莫队区间的右端点初始化为  $b$  的右端点；
- 如果询问的左右端点所属的块相同，那么直接扫描区间回答询问；
- 如果询问的左右端点所属的块不同：
  - 如果询问的右端点大于莫队区间的右端点，那么不断扩展右端点直至莫队区间的右端点等于询问的右端点；
  - 不断扩展莫队区间的左端点直至莫队区间的左端点等于询问的左端点；
  - 回答询问；
  - 撤销莫队区间左端点的改动，使莫队区间的左端点回滚到  $b$  的右端点加 1。

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define sc(x) scanf("%lld", &x)
4 typedef long long ll;
5 #define mn 100010
6 ll n, x[mn], t[mn], m, k, ans[mn], bin[mn], cnt[mn];
7 ll sq, bn, bel[mn], lf[mn], rf[mn];
8 struct query
9 {
10     ll l, r, i;
11     bool operator<(const query &x) const
12     { //一定要这么排序
13         return bel[l] == bel[x.l] ? r < x.r : bel[l] < bel[x.l];
14     }
15 } q[mn];
16 void del(ll v) { --bin[v]; }
17 void add(ll v, ll &tmp) { tmp = max(tmp, (++bin[v]) * t[v]); }
18 signed main()
19 {
20     sc(n), sc(m), sq = sqrt(n), bn = n / sq;
21     for (ll i = 1; i <= n; ++i)
22     {
23         sc(x[i]), t[i] = x[i];
24     }
25     for (ll i = 1; i <= m; ++i)
26     {
27         sc(q[i].l), sc(q[i].r), q[i].i = i;
28     }
29     for (ll i = 1; i <= bn; ++i) //一定要这么分块，不能/sq，不然会炸
30         lf[i] = (i - 1) * sq + 1, rf[i] = i * sq;
31     if (rf[bn] < n)
32         ++bn, lf[bn] = rf[bn - 1] + 1, rf[bn] = n;

```

```

33     for (ll i = 1; i <= bn; ++i)
34         for (ll j = lf[i]; j <= rf[i]; ++j)
35             bel[j] = i;
36     sort(q + 1, q + 1 + m);
37     sort(t + 1, t + 1 + n);
38     k = unique(t + 1, t + 1 + n) - (t + 1);
39     for (ll i = 1; i <= n; ++i)
40     { // x[i]是输入的a[i]排在第x[i]位,离散化压[1,1e9]到[1,1e5]
41         x[i] = lower_bound(t + 1, t + 1 + k, x[i]) - t;
42     }
43     ll l = 1, r = 0, la = 0, sum = 0, l2 = 0, sum2 = 0;
44     for (ll i = 1; i <= m; ++i)
45     {
46         if (bel[q[i].l] == bel[q[i].r])
47         { // 询问的左右端点同属于一个块则暴力扫描回答
48             for (ll j = q[i].l; j <= q[i].r; ++j)
49             {
50                 ++cnt[x[j]]; //入桶
51             }
52             for (ll j = q[i].l; j <= q[i].r; ++j)
53             { //按题意计算
54                 ans[q[i].i] = max(ans[q[i].i], t[x[j]] * cnt[x[j]]);
55             }
56             for (ll j = q[i].l; j <= q[i].r; ++j)
57             {
58                 --cnt[x[j]]; //清理记忆
59             }
60             continue;
61         }
62         if (bel[q[i].l] != la) // 访问到了新的块则重新初始化莫队区间
63         {
64             while (r > rf[bel[q[i].l]])
65                 del(x[r--]);
66             while (l < rf[bel[q[i].l]] + 1)
67                 del(x[l++]);
68             sum = 0, la = bel[q[i].l];
69         }
70         while (r < q[i].r) // 扩展右端点
71             add(x[++r], sum);
72         l2 = l, sum2 = sum;
73         while (l2 > q[i].l) // 扩展左端点
74             add(x[--l2], sum2);
75         ans[q[i].i] = sum2;
76         while (l2 < l) // 回滚
77             del(x[l2++]);
78     }
79     for (ll i = 1; i <= m; ++i)
80     {
81         printf("%lld\n", ans[i]);
82     }
83     return 0;
84 }

```

洛谷5906 给定长为  $n$  的序列  $a$  ( $1 \leq a_i \leq 2 \times 10^9$ ) 和  $q$  ( $1 \leq n, q \leq 2 \times 10^5$ ) 个询问，每次问  $[l, r]$  内相同数的最远间隔距离，特别的如果没有相同数输出 0

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 #define sc(x) scanf("%lld", &x)
4 typedef long long ll;
5 #define mn 200010
6 ll n, x[mn], t[mn], m, k, ans[mn];
7 ll lc[mn], rc[mn], cnt[mn], cls[mn], cln;
8 ll sq, bn, bel[mn], lf[mn], rf[mn];
9 struct query
10 {
11     ll l, r, i;
12     bool operator<(const query &x) const
13     { //一定要这么排序
14         return bel[l] == bel[x.l] ? r < x.r : bel[l] < bel[x.l];
15     }
16 } q[mn];
17 signed main()
18 {
19     sc(n), sq = sqrt(n), bn = n / sq;
20     for (ll i = 1; i <= n; ++i)
21     {
22         sc(x[i]), t[i] = x[i];
23     }
24     sc(m);
25     for (ll i = 1; i <= m; ++i)
26     {
27         sc(q[i].l), sc(q[i].r), q[i].i = i;
28     }
29     for (ll i = 1; i <= bn; ++i) //一定要这么分块，不能/sq，不然会炸
30         lf[i] = (i - 1) * sq + 1, rf[i] = i * sq;
31     if (rf[bn] < n)
32         ++bn, lf[bn] = rf[bn - 1] + 1, rf[bn] = n;
33     for (ll i = 1; i <= bn; ++i)
34         for (ll j = lf[i]; j <= rf[i]; ++j)
35             bel[j] = i;
36     sort(q + 1, q + 1 + m);
37     sort(t + 1, t + 1 + n);
38     k = unique(t + 1, t + 1 + n) - (t + 1);
39     for (ll i = 1; i <= n; ++i)
40     { // x[i]是输入的a[i]排在第x[i]位,离散化压[1,1e9]到[1,1e5]
41         x[i] = lower_bound(t + 1, t + 1 + k, x[i]) - t;
42     }
43     for (ll i = 1, j = 1; j <= bn; ++j)
44     {
45         ll br = rf[j], l = br + 1, r = l - 1, sum = 0, sum2 = 0, l2 = 0;
46         cln = 0;
47         for (; bel[q[i].l] == j; ++i)
48         {
49             if (bel[q[i].l] == bel[q[i].r])
50             { // 询问的左右端点同属于一个块则暴力扫描回答
51                 for (ll j = q[i].l; j <= q[i].r; ++j)
52                 {
```

```

53             cnt[x[j]] = 0;
54         }
55         for (ll j = q[i].l; j <= q[i].r; ++j)
56         { //按题意计算
57             if (!cnt[x[j]])
58             {
59                 cnt[x[j]] = j;
60             }
61             else
62             {
63                 ans[q[i].i] = max(ans[q[i].i], j - cnt[x[j]]);
64             }
65         }
66         continue;
67     }
68     while (r < q[i].r) // 扩展右端点
69     {
70         ++r;
71         rc[x[r]] = r;
72         if (!lc[x[r]])
73         {
74             lc[x[r]] = r;
75             cls[++clsn] = x[r];
76         }
77         sum = max(sum, r - lc[x[r]]);
78     }
79     l2 = 1, sum2 = sum;
80     while (l2 > q[i].l) // 扩展左端点
81     {
82         --l2;
83         if (rc[x[l2]])
84         {
85             sum2 = max(sum2, rc[x[l2]] - l2);
86         }
87         else
88         {
89             rc[x[l2]] = l2;
90         }
91     }
92     ans[q[i].i] = sum2;
93     while (l2 < 1) // 回滚
94     {
95         if (rc[x[l2]] == l2)
96         {
97             rc[x[l2]] = 0;
98         }
99         ++l2;
100    }
101 }
102 for (ll k = 1; k <= clsn; ++k)
103 {
104     lc[cls[k]] = rc[cls[k]] = 0;
105 }
106 }
107 for (ll i = 1; i <= m; ++i)
108 {

```

```

109         printf("%lld\n", ans[i]);
110     }
111     return 0;
112 }

```

## 高精度

### C++

以下给出高精度之间的、带负数的加减乘除模运算 (洛谷P1932):

其中高精度间乘法复杂度是  $O(n^2)$ ，高精度间除法使用二分答案，复杂度大概是  $O(n \log n)$

```

1 // luogu-judger-enable-o2
2 #include <cstring>
3 #include <cstdio>
4 #include <algorithm>
5 #include <cassert>
6
7 typedef int i32;
8 typedef unsigned u32;
9 typedef unsigned long long u64;
10
11 struct BigInt
12 {
13     const static u32 exp = 9;
14     const static u32 mod = 1000000000;
15
16     static i32 abs_comp(const BigInt &lhs, const BigInt &rhs)
17     {
18         if (lhs.len != rhs.len)
19             return lhs.len < rhs.len ? -1 : 1;
20         for (u32 i = lhs.len - 1; ~i; --i)
21             if (lhs.val[i] != rhs.val[i])
22                 return lhs.val[i] < rhs.val[i] ? -1 : 1;
23         return 0;
24     }
25
26     u32 *val, len, sgn;
27
28     BigInt(u32 *val = nullptr, u32 len = 0, u32 sgn = 0) : val(val),
29     len(len), sgn(sgn) {}
30
31     // copy_to cannot guarantee val[x] == 0 for x >= len
32     // other function should set (the position they assume to be zero) as
33     // zero
34     void copy_to(BigInt &dst) const
35     {
36         dst.len = len, dst.sgn = sgn;
37         memcpy(dst.val, val, sizeof(u32) * len);
38     }
39
40     void print() const
41     {
42         if (sgn == -1)
43             printf("-");
44         for (u32 i = len - 1; ~i; --i)
45             printf("%03d", val[i]);
46     }
47
48     void add(BigInt &rhs)
49     {
50         if (abs_comp(*this, rhs) == -1)
51             sub(rhs);
52         else
53             add(*this, rhs);
54     }
55
56     void sub(BigInt &rhs)
57     {
58         if (abs_comp(*this, rhs) == 1)
59             sub(rhs);
60         else
61             add(rhs, *this);
62     }
63
64     void mul(BigInt &rhs)
65     {
66         if (abs_comp(*this, rhs) == 1)
67             sub(rhs);
68         else
69             add(rhs, *this);
70     }
71
72     void div(BigInt &rhs)
73     {
74         if (abs_comp(*this, rhs) == 1)
75             sub(rhs);
76         else
77             add(rhs, *this);
78     }
79
80     void mod(BigInt &rhs)
81     {
82         if (abs_comp(*this, rhs) == 1)
83             sub(rhs);
84         else
85             add(rhs, *this);
86     }
87
88     void abs(BigInt &rhs)
89     {
90         if (abs_comp(*this, rhs) == 1)
91             sub(rhs);
92         else
93             add(rhs, *this);
94     }
95
96     void neg(BigInt &rhs)
97     {
98         if (abs_comp(*this, rhs) == 1)
99             sub(rhs);
100        else
101            add(rhs, *this);
102    }
103
104    void print() const
105    {
106        if (sgn == -1)
107            printf("-");
108        for (u32 i = len - 1; ~i; --i)
109            printf("%03d", val[i]);
110    }
111
112    void print() const
113    {
114        if (sgn == -1)
115            printf("-");
116        for (u32 i = len - 1; ~i; --i)
117            printf("%03d", val[i]);
118    }
119
120    void print() const
121    {
122        if (sgn == -1)
123            printf("-");
124        for (u32 i = len - 1; ~i; --i)
125            printf("%03d", val[i]);
126    }
127
128    void print() const
129    {
130        if (sgn == -1)
131            printf("-");
132        for (u32 i = len - 1; ~i; --i)
133            printf("%03d", val[i]);
134    }
135
136    void print() const
137    {
138        if (sgn == -1)
139            printf("-");
140        for (u32 i = len - 1; ~i; --i)
141            printf("%03d", val[i]);
142    }
143
144    void print() const
145    {
146        if (sgn == -1)
147            printf("-");
148        for (u32 i = len - 1; ~i; --i)
149            printf("%03d", val[i]);
150    }
151
152    void print() const
153    {
154        if (sgn == -1)
155            printf("-");
156        for (u32 i = len - 1; ~i; --i)
157            printf("%03d", val[i]);
158    }
159
160    void print() const
161    {
162        if (sgn == -1)
163            printf("-");
164        for (u32 i = len - 1; ~i; --i)
165            printf("%03d", val[i]);
166    }
167
168    void print() const
169    {
170        if (sgn == -1)
171            printf("-");
172        for (u32 i = len - 1; ~i; --i)
173            printf("%03d", val[i]);
174    }
175
176    void print() const
177    {
178        if (sgn == -1)
179            printf("-");
180        for (u32 i = len - 1; ~i; --i)
181            printf("%03d", val[i]);
182    }
183
184    void print() const
185    {
186        if (sgn == -1)
187            printf("-");
188        for (u32 i = len - 1; ~i; --i)
189            printf("%03d", val[i]);
190    }
191
192    void print() const
193    {
194        if (sgn == -1)
195            printf("-");
196        for (u32 i = len - 1; ~i; --i)
197            printf("%03d", val[i]);
198    }
199
200    void print() const
201    {
202        if (sgn == -1)
203            printf("-");
204        for (u32 i = len - 1; ~i; --i)
205            printf("%03d", val[i]);
206    }
207
208    void print() const
209    {
210        if (sgn == -1)
211            printf("-");
212        for (u32 i = len - 1; ~i; --i)
213            printf("%03d", val[i]);
214    }
215
216    void print() const
217    {
218        if (sgn == -1)
219            printf("-");
220        for (u32 i = len - 1; ~i; --i)
221            printf("%03d", val[i]);
222    }
223
224    void print() const
225    {
226        if (sgn == -1)
227            printf("-");
228        for (u32 i = len - 1; ~i; --i)
229            printf("%03d", val[i]);
230    }
231
232    void print() const
233    {
234        if (sgn == -1)
235            printf("-");
236        for (u32 i = len - 1; ~i; --i)
237            printf("%03d", val[i]);
238    }
239
240    void print() const
241    {
242        if (sgn == -1)
243            printf("-");
244        for (u32 i = len - 1; ~i; --i)
245            printf("%03d", val[i]);
246    }
247
248    void print() const
249    {
250        if (sgn == -1)
251            printf("-");
252        for (u32 i = len - 1; ~i; --i)
253            printf("%03d", val[i]);
254    }
255
256    void print() const
257    {
258        if (sgn == -1)
259            printf("-");
260        for (u32 i = len - 1; ~i; --i)
261            printf("%03d", val[i]);
262    }
263
264    void print() const
265    {
266        if (sgn == -1)
267            printf("-");
268        for (u32 i = len - 1; ~i; --i)
269            printf("%03d", val[i]);
270    }
271
272    void print() const
273    {
274        if (sgn == -1)
275            printf("-");
276        for (u32 i = len - 1; ~i; --i)
277            printf("%03d", val[i]);
278    }
279
280    void print() const
281    {
282        if (sgn == -1)
283            printf("-");
284        for (u32 i = len - 1; ~i; --i)
285            printf("%03d", val[i]);
286    }
287
288    void print() const
289    {
290        if (sgn == -1)
291            printf("-");
292        for (u32 i = len - 1; ~i; --i)
293            printf("%03d", val[i]);
294    }
295
296    void print() const
297    {
298        if (sgn == -1)
299            printf("-");
300        for (u32 i = len - 1; ~i; --i)
301            printf("%03d", val[i]);
302    }
303
304    void print() const
305    {
306        if (sgn == -1)
307            printf("-");
308        for (u32 i = len - 1; ~i; --i)
309            printf("%03d", val[i]);
310    }
311
312    void print() const
313    {
314        if (sgn == -1)
315            printf("-");
316        for (u32 i = len - 1; ~i; --i)
317            printf("%03d", val[i]);
318    }
319
320    void print() const
321    {
322        if (sgn == -1)
323            printf("-");
324        for (u32 i = len - 1; ~i; --i)
325            printf("%03d", val[i]);
326    }
327
328    void print() const
329    {
330        if (sgn == -1)
331            printf("-");
332        for (u32 i = len - 1; ~i; --i)
333            printf("%03d", val[i]);
334    }
335
336    void print() const
337    {
338        if (sgn == -1)
339            printf("-");
340        for (u32 i = len - 1; ~i; --i)
341            printf("%03d", val[i]);
342    }
343
344    void print() const
345    {
346        if (sgn == -1)
347            printf("-");
348        for (u32 i = len - 1; ~i; --i)
349            printf("%03d", val[i]);
350    }
351
352    void print() const
353    {
354        if (sgn == -1)
355            printf("-");
356        for (u32 i = len - 1; ~i; --i)
357            printf("%03d", val[i]);
358    }
359
360    void print() const
361    {
362        if (sgn == -1)
363            printf("-");
364        for (u32 i = len - 1; ~i; --i)
365            printf("%03d", val[i]);
366    }
367
368    void print() const
369    {
370        if (sgn == -1)
371            printf("-");
372        for (u32 i = len - 1; ~i; --i)
373            printf("%03d", val[i]);
374    }
375
376    void print() const
377    {
378        if (sgn == -1)
379            printf("-");
380        for (u32 i = len - 1; ~i; --i)
381            printf("%03d", val[i]);
382    }
383
384    void print() const
385    {
386        if (sgn == -1)
387            printf("-");
388        for (u32 i = len - 1; ~i; --i)
389            printf("%03d", val[i]);
390    }
391
392    void print() const
393    {
394        if (sgn == -1)
395            printf("-");
396        for (u32 i = len - 1; ~i; --i)
397            printf("%03d", val[i]);
398    }
399
400    void print() const
401    {
402        if (sgn == -1)
403            printf("-");
404        for (u32 i = len - 1; ~i; --i)
405            printf("%03d", val[i]);
406    }
407
408    void print() const
409    {
410        if (sgn == -1)
411            printf("-");
412        for (u32 i = len - 1; ~i; --i)
413            printf("%03d", val[i]);
414    }
415
416    void print() const
417    {
418        if (sgn == -1)
419            printf("-");
420        for (u32 i = len - 1; ~i; --i)
421            printf("%03d", val[i]);
422    }
423
424    void print() const
425    {
426        if (sgn == -1)
427            printf("-");
428        for (u32 i = len - 1; ~i; --i)
429            printf("%03d", val[i]);
430    }
431
432    void print() const
433    {
434        if (sgn == -1)
435            printf("-");
436        for (u32 i = len - 1; ~i; --i)
437            printf("%03d", val[i]);
438    }
439
440    void print() const
441    {
442        if (sgn == -1)
443            printf("-");
444        for (u32 i = len - 1; ~i; --i)
445            printf("%03d", val[i]);
446    }
447
448    void print() const
449    {
450        if (sgn == -1)
451            printf("-");
452        for (u32 i = len - 1; ~i; --i)
453            printf("%03d", val[i]);
454    }
455
456    void print() const
457    {
458        if (sgn == -1)
459            printf("-");
460        for (u32 i = len - 1; ~i; --i)
461            printf("%03d", val[i]);
462    }
463
464    void print() const
465    {
466        if (sgn == -1)
467            printf("-");
468        for (u32 i = len - 1; ~i; --i)
469            printf("%03d", val[i]);
470    }
471
472    void print() const
473    {
474        if (sgn == -1)
475            printf("-");
476        for (u32 i = len - 1; ~i; --i)
477            printf("%03d", val[i]);
478    }
479
480    void print() const
481    {
482        if (sgn == -1)
483            printf("-");
484        for (u32 i = len - 1; ~i; --i)
485            printf("%03d", val[i]);
486    }
487
488    void print() const
489    {
490        if (sgn == -1)
491            printf("-");
492        for (u32 i = len - 1; ~i; --i)
493            printf("%03d", val[i]);
494    }
495
496    void print() const
497    {
498        if (sgn == -1)
499            printf("-");
500        for (u32 i = len - 1; ~i; --i)
501            printf("%03d", val[i]);
502    }
503
504    void print() const
505    {
506        if (sgn == -1)
507            printf("-");
508        for (u32 i = len - 1; ~i; --i)
509            printf("%03d", val[i]);
510    }
511
512    void print() const
513    {
514        if (sgn == -1)
515            printf("-");
516        for (u32 i = len - 1; ~i; --i)
517            printf("%03d", val[i]);
518    }
519
520    void print() const
521    {
522        if (sgn == -1)
523            printf("-");
524        for (u32 i = len - 1; ~i; --i)
525            printf("%03d", val[i]);
526    }
527
528    void print() const
529    {
530        if (sgn == -1)
531            printf("-");
532        for (u32 i = len - 1; ~i; --i)
533            printf("%03d", val[i]);
534    }
535
536    void print() const
537    {
538        if (sgn == -1)
539            printf("-");
540        for (u32 i = len - 1; ~i; --i)
541            printf("%03d", val[i]);
542    }
543
544    void print() const
545    {
546        if (sgn == -1)
547            printf("-");
548        for (u32 i = len - 1; ~i; --i)
549            printf("%03d", val[i]);
550    }
551
552    void print() const
553    {
554        if (sgn == -1)
555            printf("-");
556        for (u32 i = len - 1; ~i; --i)
557            printf("%03d", val[i]);
558    }
559
560    void print() const
561    {
562        if (sgn == -1)
563            printf("-");
564        for (u32 i = len - 1; ~i; --i)
565            printf("%03d", val[i]);
566    }
567
568    void print() const
569    {
570        if (sgn == -1)
571            printf("-");
572        for (u32 i = len - 1; ~i; --i)
573            printf("%03d", val[i]);
574    }
575
576    void print() const
577    {
578        if (sgn == -1)
579            printf("-");
580        for (u32 i = len - 1; ~i; --i)
581            printf("%03d", val[i]);
582    }
583
584    void print() const
585    {
586        if (sgn == -1)
587            printf("-");
588        for (u32 i = len - 1; ~i; --i)
589            printf("%03d", val[i]);
590    }
591
592    void print() const
593    {
594        if (sgn == -1)
595            printf("-");
596        for (u32 i = len - 1; ~i; --i)
597            printf("%03d", val[i]);
598    }
599
600    void print() const
601    {
602        if (sgn == -1)
603            printf("-");
604        for (u32 i = len - 1; ~i; --i)
605            printf("%03d", val[i]);
606    }
607
608    void print() const
609    {
610        if (sgn == -1)
611            printf("-");
612        for (u32 i = len - 1; ~i; --i)
613            printf("%03d", val[i]);
614    }
615
616    void print() const
617    {
618        if (sgn == -1)
619            printf("-");
620        for (u32 i = len - 1; ~i; --i)
621            printf("%03d", val[i]);
622    }
623
624    void print() const
625    {
626        if (sgn == -1)
627            printf("-");
628        for (u32 i = len - 1; ~i; --i)
629            printf("%03d", val[i]);
630    }
631
632    void print() const
633    {
634        if (sgn == -1)
635            printf("-");
636        for (u32 i = len - 1; ~i; --i)
637            printf("%03d", val[i]);
638    }
639
640    void print() const
641    {
642        if (sgn == -1)
643            printf("-");
644        for (u32 i = len - 1; ~i; --i)
645            printf("%03d", val[i]);
646    }
647
648    void print() const
649    {
650        if (sgn == -1)
651            printf("-");
652        for (u32 i = len - 1; ~i; --i)
653            printf("%03d", val[i]);
654    }
655
656    void print() const
657    {
658        if (sgn == -1)
659            printf("-");
660        for (u32 i = len - 1; ~i; --i)
661            printf("%03d", val[i]);
662    }
663
664    void print() const
665    {
666        if (sgn == -1)
667            printf("-");
668        for (u32 i = len - 1; ~i; --i)
669            printf("%03d", val[i]);
670    }
671
672    void print() const
673    {
674        if (sgn == -1)
675            printf("-");
676        for (u32 i = len - 1; ~i; --i)
677            printf("%03d", val[i]);
678    }
679
680    void print() const
681    {
682        if (sgn == -1)
683            printf("-");
684        for (u32 i = len - 1; ~i; --i)
685            printf("%03d", val[i]);
686    }
687
688    void print() const
689    {
690        if (sgn == -1)
691            printf("-");
692        for (u32 i = len - 1; ~i; --i)
693            printf("%03d", val[i]);
694    }
695
696    void print() const
697    {
698        if (sgn == -1)
699            printf("-");
700        for (u32 i = len - 1; ~i; --i)
701            printf("%03d", val[i]);
702    }
703
704    void print() const
705    {
706        if (sgn == -1)
707            printf("-");
708        for (u32 i = len - 1; ~i; --i)
709            printf("%03d", val[i]);
710    }
711
712    void print() const
713    {
714        if (sgn == -1)
715            printf("-");
716        for (u32 i = len - 1; ~i; --i)
717            printf("%03d", val[i]);
718    }
719
720    void print() const
721    {
722        if (sgn == -1)
723            printf("-");
724        for (u32 i = len - 1; ~i; --i)
725            printf("%03d", val[i]);
726    }
727
728    void print() const
729    {
730        if (sgn == -1)
731            printf("-");
732        for (u32 i = len - 1; ~i; --i)
733            printf("%03d", val[i]);
734    }
735
736    void print() const
737    {
738        if (sgn == -1)
739            printf("-");
740        for (u32 i = len - 1; ~i; --i)
741            printf("%03d", val[i]);
742    }
743
744    void print() const
745    {
746        if (sgn == -1)
747            printf("-");
748        for (u32 i = len - 1; ~i; --i)
749            printf("%03d", val[i]);
750    }
751
752    void print() const
753    {
754        if (sgn == -1)
755            printf("-");
756        for (u32 i = len - 1; ~i; --i)
757            printf("%03d", val[i]);
758    }
759
760    void print() const
761    {
762        if (sgn == -1)
763            printf("-");
764        for (u32 i = len - 1; ~i; --i)
765            printf("%03d", val[i]);
766    }
767
768    void print() const
769    {
770        if (sgn == -1)
771            printf("-");
772        for (u32 i = len - 1; ~i; --i)
773            printf("%03d", val[i]);
774    }
775
776    void print() const
777    {
778        if (sgn == -1)
779            printf("-");
780        for (u32 i = len - 1; ~i; --i)
781            printf("%03d", val[i]);
782    }
783
784    void print() const
785    {
786        if (sgn == -1)
787            printf("-");
788        for (u32 i = len - 1; ~i; --i)
789            printf("%03d", val[i]);
790    }
791
792    void print() const
793    {
794        if (sgn == -1)
795            printf("-");
796        for (u32 i = len - 1; ~i; --i)
797            printf("%03d", val[i]);
798    }
799
800    void print() const
801    {
802        if (sgn == -1)
803            printf("-");
804        for (u32 i = len - 1; ~i; --i)
805            printf("%03d", val[i]);
806    }
807
808    void print() const
809    {
810        if (sgn == -1)
811            printf("-");
812        for (u32 i = len - 1; ~i; --i)
813            printf("%03d", val[i]);
814    }
815
816    void print() const
817    {
818        if (sgn == -1)
819            printf("-");
820        for (u32 i = len - 1; ~i; --i)
821            printf("%03d", val[i]);
822    }
823
824    void print() const
825    {
826        if (sgn == -1)
827            printf("-");
828        for (u32 i = len - 1; ~i; --i)
829            printf("%03d", val[i]);
830    }
831
832    void print() const
833    {
834        if (sgn == -1)
835            printf("-");
836        for (u32 i = len - 1; ~i; --i)
837            printf("%03d", val[i]);
838    }
839
840    void print() const
841    {
842        if (sgn == -1)
843            printf("-");
844        for (u32 i = len - 1; ~i; --i)
845            printf("%03d", val[i]);
846    }
847
848    void print() const
849    {
850        if (sgn == -1)
851            printf("-");
852        for (u32 i = len - 1; ~i; --i)
853            printf("%03d", val[i]);
854    }
855
856    void print() const
857    {
858        if (sgn == -1)
859            printf("-");
860        for (u32 i = len - 1; ~i; --i)
861            printf("%03d", val[i]);
862    }
863
864    void print() const
865    {
866        if (sgn == -1)
867            printf("-");
868        for (u32 i = len - 1; ~i; --i)
869            printf("%03d", val[i]);
870    }
871
872    void print() const
873    {
874        if (sgn == -1)
875            printf("-");
876        for (u32 i = len - 1; ~i; --i)
877            printf("%03d", val[i]);
878    }
879
880    void print() const
881    {
882        if (sgn == -1)
883            printf("-");
884        for (u32 i = len - 1; ~i; --i)
885            printf("%03d", val[i]);
886    }
887
888    void print() const
889    {
890        if (sgn == -1)
891            printf("-");
892        for (u32 i = len - 1; ~i; --i)
893            printf("%03d", val[i]);
894    }
895
896    void print() const
897    {
898        if (sgn == -1)
899            printf("-");
900        for (u32 i = len - 1; ~i; --i)
901            printf("%03d", val[i]);
902    }
903
904    void print() const
905    {
906        if (sgn == -1)
907            printf("-");
908        for (u32 i = len - 1; ~i; --i)
909            printf("%03d", val[i]);
910    }
911
912    void print() const
913    {
914        if (sgn == -1)
915            printf("-");
916        for (u32 i = len - 1; ~i; --i)
917            printf("%03d", val[i]);
918    }
919
920    void print() const
921    {
922        if (sgn == -1)
923            printf("-");
924        for (u32 i = len - 1; ~i; --i)
925            printf("%03d", val[i]);
926    }
927
928    void print() const
929    {
930        if (sgn == -1)
931            printf("-");
932        for (u32 i = len - 1; ~i; --i)
933            printf("%03d", val[i]);
934    }
935
936    void print() const
937    {
938        if (sgn == -1)
939            printf("-");
940        for (u32 i = len - 1; ~i; --i)
941            printf("%03d", val[i]);
942    }
943
944    void print() const
945    {
946        if (sgn == -1)
947            printf("-");
948        for (u32 i = len - 1; ~i; --i)
949            printf("%03d", val[i]);
950    }
951
952    void print() const
953    {
954        if (sgn == -1)
955            printf("-");
956        for (u32 i = len - 1; ~i; --i)
957            printf("%03d", val[i]);
958    }
959
960    void print() const
961    {
962        if (sgn == -1)
963            printf("-");
964        for (u32 i = len - 1; ~i; --i)
965            printf("%03d", val[i]);
966    }
967
968    void print() const
969    {
970        if (sgn == -1)
971            printf("-");
972        for (u32 i = len - 1; ~i; --i)
973            printf("%03d", val[i]);
974    }
975
976    void print() const
977    {
978        if (sgn == -1)
979            printf("-");
980        for (u32 i = len - 1; ~i; --i)
981            printf("%03d", val[i]);
982    }
983
984    void print() const
985    {
986        if (sgn == -1)
987            printf("-");
988        for (u32 i = len - 1; ~i; --i)
989            printf("%03d", val[i]);
990    }
991
992    void print() const
993    {
994        if (sgn == -1)
995            printf("-");
996        for (u32 i = len - 1; ~i; --i)
997            printf("%03d", val[i]);
998    }
999
1000    void print() const
1001    {
1002        if (sgn == -1)
1003            printf("-");
1004        for (u32 i = len - 1; ~i; --i)
1005            printf("%03d", val[i]);
1006    }
1007
1008    void print() const
1009    {
1010        if (sgn == -1)
1011            printf("-");
1012        for (u32 i = len - 1; ~i; --i)
1013            printf("%03d", val[i]);
1014    }
1015
1016    void print() const
1017    {
1018        if (sgn == -1)
1019            printf("-");
1020        for (u32 i = len - 1; ~i; --i)
1021            printf("%03d", val[i]);
1022    }
1023
1024    void print() const
1025    {
1026        if (sgn == -1)
1027            printf("-");
1028        for (u32 i = len - 1; ~i; --i)
1029            printf("%03d", val[i]);
1030    }
1031
1032    void print() const
1033    {
1034        if (sgn == -1)
1035            printf("-");
1036        for (u32 i = len - 1; ~i; --i)
1037            printf("%03d", val[i]);
1038    }
1039
1040    void print() const
1041    {
1042        if (sgn == -1)
1043            printf("-");
1044        for (u32 i = len - 1; ~i; --i)
1045            printf("%03d", val[i]);
1046    }
1047
1048    void print() const
1049    {
1050        if (sgn == -1)
1051            printf("-");
1052        for (u32 i = len - 1; ~i; --i)
1053            printf("%03d", val[i]);
1054    }
1055
1056    void print() const
1057    {
1058        if (sgn == -1)
1059            printf("-");
1060        for (u32 i = len - 1; ~i; --i)
1061            printf("%03d", val[i]);
1062    }
1063
1064    void print() const
1065    {
1066        if (sgn == -1)
1067            printf("-");
1068        for (u32 i = len - 1; ~i; --i)
1069            printf("%03d", val[i]);
1070    }
1071
1072    void print() const
1073    {
1074        if (sgn == -1)
1075            printf("-");
1076        for (u32 i = len - 1; ~i; --i)
1077            printf("%03d", val[i]);
1078    }
1079
1080    void print() const
1081    {
1082        if (sgn == -1)
1083            printf("-");
1084        for (u32 i = len - 1; ~i; --i)
1085            printf("%03d", val[i]);
1086    }
1087
1088    void print() const
1089    {
1090        if (sgn == -1)
1091            printf("-");
1092        for (u32 i = len - 1; ~i; --i)
1093            printf("%03d", val[i]);
1094    }
1095
1096    void print() const
1097    {
1098        if (sgn == -1)
1099            printf("-");
1100        for (u32 i = len - 1; ~i; --i)
1101            printf("%03d", val[i]);
1102    }
1103
1104    void print() const
1105    {
1106        if (sgn == -1)
1107            printf("-");
1108        for (u32 i = len - 1; ~i; --i)
1109            printf("%03d", val[i]);
1110    }
1111
1112    void print() const
1113    {
1114        if (sgn == -1)
1115            printf("-");
1116        for (u32 i = len - 1; ~i; --i)
1117            printf("%03d", val[i]);
1118    }
1119
1120    void print() const
1121    {
1122        if (sgn == -1)
1123            printf("-");
1124        for (u32 i = len - 1; ~i; --i)
1125            printf("%03d", val[i]);
1126    }
1127
1128    void print() const
1129    {
1130        if (sgn == -1)
1131            printf("-");
1132        for (u32 i = len - 1; ~i; --i)
1133            printf("%03d", val[i]);
1134    }
1135
1136    void print() const
1137    {
1138        if (sgn == -
```

```

38     void trim()
39     {
40         while (len && !val[len - 1])
41             --len;
42         if (len == 0)
43             sgn = 0;
44     }
45
46     void add(BigInt &x)
47     {
48         if (sgn ^ x.sgn)
49             return x.sgn ^= 1, sub(x);
50         val[len = std::max(len, x.len)] = 0;
51         for (u32 i = 0; i < x.len; ++i)
52             if ((val[i] += x.val[i]) >= mod)
53                 val[i] -= mod, ++val[i + 1];
54         for (u32 i = x.len; i < len && val[i] >= mod; ++i)
55             val[i] -= mod, ++val[i + 1];
56         if (val[len])
57             ++len;
58     }
59
60     void sub(BigInt &x)
61     {
62         if (sgn ^ x.sgn)
63             return x.sgn ^= 1, add(x);
64         if (abs_comp(*this, x) < 0)
65             std::swap(*this, x), sgn ^= 1;
66         val[len] = 0;
67         for (u32 i = 0; i < x.len; ++i)
68             if ((val[i] -= x.val[i]) & 0x80000000)
69                 val[i] += mod, --val[i + 1];
70         for (u32 i = x.len; i < len && val[i] & 0x80000000; ++i)
71             val[i] += mod, --val[i + 1];
72         trim();
73     }
74
75     void mul(BigInt &x, u32 *ext_mem)
76     {
77         assert(this != &x);
78         memcpy(ext_mem, val, sizeof(u32) * len);
79         memset(val, 0, sizeof(u32) * (len + x.len));
80         for (u32 i = 0; i < len; ++i)
81             for (u32 j = 0; j < x.len; ++j)
82             {
83                 u64 tmp = (u64)ext_mem[i] * x.val[j] + val[i + j];
84                 val[i + j] = tmp % mod;
85                 val[i + j + 1] += tmp / mod;
86             }
87         len += x.len, sgn ^= x.sgn;
88         trim();
89     }
90
91     void mul(u32 x)
92     {
93         if (x & 0x80000000)

```

```

94         x = -x, sgn ^= 1;
95         u64 carry = 0;
96         for (u32 i = 0; i < len; ++i)
97         {
98             carry += (u64)val[i] * x;
99             val[i] = carry % mod;
100            carry /= mod;
101        }
102        if (carry)
103            val[len++] = carry;
104        trim();
105    }
106
107    void div(BigInt &x, BigInt &remainder, u32 *ext_mem)
108    {
109        assert(this != &x && this != &remainder);
110        copy_to(remainder), memset(val, 0, sizeof(u32) * len);
111        u32 shift = len - x.len;
112        if (shift & 0x80000000)
113            return void(len = sgn = 0);
114        while (~shift)
115        {
116            u32 l = 0, r = mod;
117            BigInt mul_test{ext_mem}, remainder_high{remainder.val + shift,
118 remainder.len - shift};
119            while (l <= r)
120            {
121                u32 mid = (l + r) / 2;
122                x.copy_to(mul_test), mul_test.mul(mid);
123                abs_comp(mul_test, remainder_high) < 0 ? l = mid + 1 : r =
124 mid - 1;
125            }
126            val[shift] = r;
127            x.copy_to(mul_test), mul_test.mul(r);
128            remainder_high.sub(mul_test), remainder.trim();
129            --shift;
130        }
131        sgn ^= x.sgn;
132        trim();
133    }
134
135    void div(u32 x)
136    {
137        if (x & 0x80000000)
138            x = -x, sgn ^= 1;
139        u64 carry = 0;
140        for (u32 i = len - 1; ~i; --i)
141        {
142            carry = carry * mod + val[i];
143            val[i] = carry / x;
144            carry %= x;
145        }
146        trim();
147    }
148    void read(const char *s)

```

```

148     {
149         sgn = len = 0;
150         i32 bound = 0, pos;
151         if (s[0] == '-')
152             sgn = bound = 1;
153         u64 cur = 0, pow = 1;
154         for (pos = strlen(s) - 1; pos + 1 >= exp + bound; pos -= exp,
155             val[len++] = cur, cur = 0, pow = 1)
156             for (i32 i = pos; i + exp > pos; --i)
157                 cur += (s[i] - '0') * pow, pow *= 10;
158             for (cur = 0, pow = 1; pos >= bound; --pos)
159                 cur += (s[pos] - '0') * pow, pow *= 10;
160             if (cur)
161                 val[len++] = cur;
162     }
163
164     void print()
165     {
166         if (len)
167         {
168             if (sgn)
169                 putchar('-');
170             printf("%u", val[len - 1]);
171             for (u32 i = len - 2; ~i; --i)
172                 printf("%0*u", exp, val[i]);
173         }
174         else
175             putchar('0');
176         puts("");
177     }
178
179     const int N = 1e4 + 20;
180     u32 a_[N], b_[N], r_[N], tmp[N * 2];
181     char sa[N], sb[N];
182
183     int main()
184     {
185         scanf("%s%s", sa, sb);
186         {
187             BigInt a{a_}, b{b_};
188             a.read(sa), b.read(sb), a.add(b), a.print();
189         }
190         {
191             BigInt a{a_}, b{b_};
192             a.read(sa), b.read(sb), a.sub(b), a.print();
193         }
194         {
195             BigInt a{a_}, b{b_};
196             a.read(sa), b.read(sb), a.mul(b, tmp), a.print();
197         }
198         {
199             BigInt a{a_}, b{b_}, r{r_};
200             a.read(sa), b.read(sb), a.div(b, r, tmp), a.print();
201             r.print();
202         }

```

## FFT 乘法

复杂度  $O(n \log n)$ ，可以通过  $10^6$  数据量

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 typedef double db;
5 #define sc(x) x = read()
6 #define mn ((1 << 21) + 1)
7 ll n1, n2, rev[mn], ans[mn], k, s = 1, len, n;
8 db pi = acos(-1), v;
9 typedef complex<db> cp;
10 cp a[mn], b[mn];
11 char s1[mn], s2[mn];
12 void fft(cp *a, ll n, ll flag)
13 {
14     for (ll i = 0; i < n; ++i)
15     {
16         if (i < rev[i])
17         {
18             swap(a[i], a[rev[i]]);
19         }
20     }
21     for (ll h = 1; h < n; h <= 1)
22     {
23         cp wn = exp(cp(0, flag * pi / h));
24         for (ll j = 0; j < n; j += h << 1)
25         {
26             cp w(1, 0);
27             for (ll k = j; k < j + h; ++k)
28             {
29                 cp x = a[k], y = w * a[k + h];
30                 a[k] = x + y;
31                 a[k + h] = x - y;
32                 w *= wn;
33             }
34         }
35     }
36     if (flag == -1)
37     {
38         for (ll i = 0; i < n; ++i)
39         {
40             a[i] /= n;
41         }
42     }
43 }
44
45 signed main()
46 {
47     scanf("%s%s", s1, s2);

```

```

48     n1 = strlen(s1), n2 = strlen(s2), n = max(n1, n2);
49     for (ll i = 0; i < n1; ++i)
50     {
51         a[i] = (db)(s1[n1 - i - 1] - '0');
52     }
53     for (ll i = 0; i < n2; ++i)
54     {
55         b[i] = (db)(s2[n2 - i - 1] - '0');
56     }
57     k = 1, s = 2;
58     while ((1 << k) < (n << 1) - 1)
59     {
60         ++k, s <= 1;
61     }
62     // while (s <= n)
63     // {
64     //     s <= 1, ++k;
65     // }
66     for (ll i = 0; i < s; ++i)
67     {
68         rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (k - 1));
69     }
70     fft(a, s, 1), fft(b, s, 1);
71     for (ll i = 0; i <= s; ++i)
72     {
73         a[i] *= b[i];
74     }
75     fft(a, s, -1);
76     for (ll i = 0; i < s; ++i)
77     {
78         ans[i] += (ll)(a[i].real() + 0.5);
79         ans[i + 1] += ans[i] / 10, ans[i] %= 10;
80     }
81     while (!ans[s] && s > -1)
82     {
83         --s;
84     }
85     if (s == -1)
86     {
87         puts("0");
88     }
89     else
90     {
91         for (ll i = s; i >= 0; --i)
92         {
93             printf("%lld", ans[i]);
94         }
95     }
96     return 0;
97 }

```

## Java 高精度

`java.math.BigInteger, java.math.BigDecimal` 高精度运算 JAVA8为例

构造函数只能传字符串。可以直接输出。

`BigInteger` 方法: (注意不会改变lhs本身,需要用新变量存结果)

- add, subtract, multiply, divide
- remainder 取模
- divideAndRemainder 返回数组, 第一个是商, 第二个是余
- pow(指数是int不是BigInteger) (应该不是快速幂)
- negate 相反数
- shiftLeft 左移<<(负数右移) 参数是int
- shiftRight 右移>>(负数左移) 参数是int
- and or 位运算
- compareTo
- equals 参数rhs是Object
- min,max
- isProbablePrime(值) 值越大, 得到的结果越准确
- toString(进制) 可以返回16进制字符串

`BigDecimal` 方法:

- add subtract multiply divide

divide的设置方法处理小数位:

- ROUND\_UP 商的最后一一位大于0时向前进位, 正负号均如此
- ROUND\_DOWN 商的最后一一位忽略
- ROUND\_CEILING 正up负down 故近似值 $\geq$ 实际值
- ROUND\_FLOOR 负up正down 故近似值 $\leq$ 实际值
- ROUND\_HALF\_DOWN 商四舍五入 $\leq 5$ 舍弃否则进位
- ROUND\_HALF\_UP  $< 5$ 舍弃否则进位
- ROUND\_HALF\_EVEN 商倒数第二位奇数halfup否则halfdown

要调用, 三个参数, 第一个是rhs,第二个是商小数点保留位数, 第三个是处理方式, 如

`BigDecimal.ROUND_UP`

```
1 import java.util.Scanner; // 洛谷P1932
2 import java.math.BigInteger;
3
4 public class Main {
5     public static void main(String[] args) {
6         Scanner sc = new Scanner(System.in);
7         BigInteger a = new BigInteger(sc.next());
8         BigInteger b = new BigInteger(sc.next());
9         System.out.println(a.add(b));
10        System.out.println(a.subtract(b));
```

```

11     System.out.println(a.multiply(b));
12     System.out.println(a.divide(b));
13     System.out.println(a.mod(b));
14     sc.close();
15 }
16 }
```

除法必须规定位数(java9后当前函数会报 warning), 如:

```

1 BigDecimal a = new BigDecimal(sc.next());
2 BigDecimal b = new BigDecimal(sc.next());
3 System.out.println(a.divide(b, 2, BigDecimal.ROUND_HALF_DOWN));
```

求高精度整数平方根: 范围约为  $10^{1000}$

```

1 import java.math.BigInteger;
2
3 public class Main {
4     public static void main(String[] args) {
5         System.out.println(isqrtNewton(new BigInteger("99999")));
6     }
7
8     public static BigInteger isqrtNewton(BigInteger n) {
9         BigInteger a = BigInteger.ONE.shiftLeft(n.bitLength() / 2);
10        boolean p_dec = false;
11        for (;;) {
12            BigInteger b = n.divide(a).add(a).shiftRight(1);
13            if (a.compareTo(b) == 0 || a.compareTo(b) < 0 && p_dec)
14                break;
15            p_dec = a.compareTo(b) > 0;
16            a = b;
17        }
18        return a;
19    }
20 }
```

## Python 高精度

以例子说明: (正常使用四则运算, 输入输出和格式化, 设置精度即可)

```

1 from decimal import *
2 getcontext().prec = 20
3 x = Decimal('1')
4 y = Decimal('7.0')
5 print((x / y * y * y).quantize(Decimal('0.00'), ROUND_HALF_DOWN))
6 #第二个参数可以不填
```

精度: (官方文档)

ROUND\_CEILING (towards Infinity),  
 ROUND\_DOWN (towards zero),  
 ROUND\_FLOOR (towards -Infinity),  
 ROUND\_HALF\_DOWN (to nearest with ties going towards zero),  
 ROUND\_HALF\_EVEN (to nearest with ties going to nearest even integer),  
 ROUND\_HALF\_UP (to nearest with ties going away from zero), or  
 ROUND\_UP (away from zero).  
 ROUND\_05UP (away from zero if last digit after rounding towards zero would have been 0 or 5; otherwise towards zero)

## 悬线法

可以应用于满足以下条件的题目：

- 需要在扫描序列时维护单调的信息；
- 可以使用单调栈解决；
- 不需要在单调栈上二分。

P4147-给定只包含 F R 的矩阵，求全为 F 矩阵的最大面积的3倍

举每个 F 的位置作为矩形的最高点，然后找到以该点向上最上的前提下，向左向右最远能扩展多远。设  $l_{i,j}, r_{i,j}, a_{i,j}$  分别表示点  $(i, j)$  向左右上的最远距离。时间  $O(nm)$ ，空间  $O(m)$ 。

```

1 #include <algorithm>
2 #include <cstdio>
3 int m, n, a[1010], l[1010], r[1010], ans;
4
5 int main() {
6     scanf("%d%d", &n, &m);
7     for (int i = 1; i <= n; i++) {
8         for (int j = 1; j <= m; j++) {
9             l[j] = r[j] = j;
10        }
11        char s[3];
12        for (int j = 1; j <= m; j++) {
13            scanf("%s", s);
14            if (s[0] == 'F')
15                a[j]++;
16            else if (s[0] == 'R')
17                a[j] = 0;
18        }
19        for (int j = 1; j <= m; j++)
20            while (l[j] != 1 && a[l[j] - 1] >= a[j]) l[j] = l[l[j] - 1];
21        for (int j = m; j >= 1; j--)
22            while (r[j] != m && a[r[j] + 1] >= a[j]) r[j] = r[r[j] + 1];
23        for (int j = 1; j <= m; j++) ans = std::max(ans, (r[j] - l[j] + 1) *
24            a[j]);
25    }
26    printf("%d", ans * 3);
27    return 0;
}
```

单调栈法：对每行执行一次每个  $i$  对应的最远  $l, r$  可以用单调栈维护。维护单调递增栈，每个元素被退栈之时得到它的右边界。然后再继续来一遍，得到左边界。之后同理操作即可。

# 程序语法

## 常规运算

- `__int128`  $[-2^{127}, 2^{127} - 1]$ , 16 字节, 其中  $2^{127} \approx 8.5 \times 10^{37}$
- `double` 有效位数约 15 位, 8 字节, 范围约为  $[-1.79 \times 10^{308}, 1.79 \times 10^{308}]$
- `long double` 通常有效位数约 20 位, 16 字节, 范围约为  $[-1.2 \times 10^{4932}, 1.2 \times 10^{4932}]$

注意:

- 取模只可以是整型之间的; 负数运算结果等于取绝对值计算, 最后正负号与被除数一致, 如  $(-5) \bmod 3 = -2, 5 \bmod (-3) = 2$  (C99/C++11开始 / JAVA同)  
所有版本 C/C++ 必然满足:  $(a/b) * b + a \% b = a$
- 浮点数除, 被除数为 0, 被除数和除数存在一个负数, 结果为负零

Python : `-17%10=-7, 17%-10=7, -17%-10=-7` (除时取取比实际结果稍小的最大整数)

整数上取整公式:

$$\lceil \frac{a}{b} \rceil = \frac{a + b - 1}{b} = \frac{a - 1}{b} + 1$$

`__gcd(a, b)` 若一方为零, 返回另一方(无论正负); 二者均负返回负, 一正一负比较特殊

## 位运算

异或的性质:

- 交换律、结合律、消去律, 有单位元 0, 自己与自己运算得单位元
- $a \oplus b \leq a + b = (a|b) + (a \& b)$

前半句: 因异或是不进位的加法; 后半句: 因  $a \& b$  是进位部分

注意要点:

- 优先级: `~; +, -; <<, >>; ==, !=; &; ^; |; &&; ||; ?:`
- 移位结果为 `ll` 时应该是 `1LL << k`
- 右移位, 等同于 `round(x/2.0)`, 负数的移位结果不会大于-1

常见应用:

- 取正数 `x` 的从左往右(从零数)第 `i` 位: `(x>>i)&1`
- 对某个正数 `x` 从左往右(从零数)第 `k` 位修改取反: `x^=(1<<k)`
- `c&15` 或 `c^'0'` 优化 数字字符转数值
- `~` 运算仅对 -1 得 0, 可以用于递减到 0 的循环枚举常数优化
- 交换两个数 `a^=b^=a^=b`
- 删掉最低的 1, 其他不变 `x&(x-1)`

内建函数：

- 注：对 `unsigned long long` 每个函数名后面加上 `ll` (传入的是什么类型不影响结果, 影响的是函数名)

### 1. `__builtin_popcount(unsigned int n)`

该函数时判断n的二进制中有多少个1

```
1 | int n = 15; //二进制为1111
2 | cout<<__builtin_popcount(n)<<endl; //输出4
```

### 2. `__builtin_parity(unsigned int n)`

该函数是判断n的二进制中1的个数的奇偶性

```
1 | int n = 15; //二进制为1111
2 | int m = 7; //111
3 | cout<<__builtin_parity(n)<<endl; //偶数个, 输出0
4 | cout<<__builtin_parity(m)<<endl; //奇数个, 输出1
```

### 3. `__builtin_ffs(unsigned int n)`

该函数判断n的二进制末尾最后一个1的位置, 从一开始

```
1 | int n = 1; //1
2 | int m = 8; //1000
3 | cout<<__builtin_ffs(n)<<endl; //输出1
4 | cout<<__builtin_ffs(m)<<endl; //输出4
```

### 4. `__builtin_ctz(unsigned int n)`

该函数判断n的二进制末尾后面0的个数, 当n为0时, 和n的类型有关

```
1 | int n = 1; //1
2 | int m = 8; //1000
3 | cout<<__builtin_ctzll(n)<<endl; //输出0
4 | cout<<__builtin_ctz(m)<<endl; //输出3
```

### 5. `__builtin_clz (unsigned int x)`

返回前导的0的个数。

```
1 | int n = 1; //1
2 | int m = 8; //1000
3 | cout<< 32 - __builtin_clz(n) <<endl; //输出1
4 | cout<< 64 - __builtin_clzll(m) <<endl; //输出4
```

应用: `31 - __builtin_clz(n)` 等效于  $\lfloor \log_2 n \rfloor$

## 指针

```
1 | 指针数组 int *p[3];  
2 | 指向二维数组的指针(第二维为3) int (*p)[3];//即列指针
```

```
1 | #include <stdio.h>  
2 | int fx(int x)  
3 | {  
4 |     return x*x;  
5 | }  
6 | int add(int x, int y)  
7 | {  
8 |     return x*y;  
9 | }  
10 | int h(int (*f)(), int a, int b)  
11 | {  
12 |     return f(a,b);  
13 | }  
14 | signed main()  
15 | {  
16 |     int (*g)();  
17 |     g=fx;  
18 |     printf("%d\n",g(5));  
19 |     printf("%d",h(add,5,-1));  
20 |     return 0;  
21 | }
```

```
1 | 例如: void f(int n, int (*compare)(int a, int b));  
2 | int g(int a, int b);  
3 | 使用时: g(3, f);
```

## I/O

cout 输出小数: `cout<<setprecision(n)`

cout 输出不同进制:

```
1 | cout<<hex<<17<<endl;  
2 | cout<<dec<<18<<endl;  
3 | cout<<oct<<9<<endl;  
4 | cout<<bitset<sizeof(unsigned long) * 8>(1UL << 27)<<endl;
```

## 其他

计时: (单位: 秒)

```
1 | (double)clock() / CLOCKS_PER_SEC
```

decltype

```
1 int x;
2 decltype((x)) z;//z为int & ,da ck
3 decltype(x) y;//y为int
4 typedef decltype(f(x)+g(x)) dt;
5 dt var1;
```

使用了万能头时，不允许使用这些变量名：y1, y0, yn, prev, tm 等

Python日期函数举例：

```
1 from datetime import *
2 d1, d2 = datetime(2022, 5, 8), datetime(2002, 5, 8)
3 print((d1-d2).days) # 天数差
4 print((datetime.today() + timedelta(days=1)).weekday()) #星期[0,6]---日
```

输出版本的语法：(适用于 OMS 等可测试平台)

```
1 printf("%lld", __STDC_VERSION__); //c
1 std::cout << __cplusplus; //c++
1 print(sys.version) # Python
1 System.out.println(System.getProperty("java.version")); //Java
```

## STL

### 函数

#### xx\_bound

bound函数，参数起始迭代器,终止迭代器,值，二分查找 [起,止)

**upper\_bound** 大于 x 的第一个位置

**lower\_bound** 大于等于 x 的第一个位置

对升序序列：

`lower_bound(begin, end, v)` 在  $[begin, end)$  找第一个大于等于  $v$  的值，返回该值的迭代器

`upper_bound(begin, end, v)` 在  $[begin, end)$  找第一个大于  $v$  的值，返回该值的迭代器

若找不到返回  $end$

对降序序列：

`lower_bound(begin, end, v, greater<type>())` 在  $[begin, end)$  找第一个小于等于  $v$  的值, 返回该值的迭代器

`upper_bound(begin, end, v, greater<type>())` 在  $[begin, end)$  找第一个小于  $v$  的值, 返回该值的迭代器

若找不到返回  $end$

以二分搜索向左第一个数为例:

```
1 int d = lower_bound(a, a+n, f)-a;
2 if(f!=a[d]) printf("-1");
3 else printf("%d", d+1);
```

默认对非降序列使用, 如果要对下降序列使用, 需要加greater如下(指针写法):

```
1 *upper_bound(d1+1, d1+1+len1, a[i], greater<int>())=a[i];
```

二分查找值区间  $[a, b]$  长:  $u(, , b) - l(, , a)$

## 字符串

C 风格字符串

常用函数:

- `cin.getline(char* s, int len)` 读整行
- `strcat(char* s1, char* s2)`  $s1 = s1 + s2$
- `strstr(char* s1, char* s2)` 在主串  $s1$  进行模式匹配, 返回首个匹配成功地址或 `NULL`
- `strchr(char* s1, char s2)` 模式匹配
- `strchr(char* s1, char s2)` 模式匹配(从右开始)
- `strtok(char* s1, const char* sep)` 子串分割

```
1 const char *sep = ", "; //必须const
2 char *u = strtok(s, sep); // strtok用法模板
3 while (u)
4 {
5     t.insert(u); //输出分割结果
6     u = strtok(nullptr, sep);
7 }
```

## 正则表达式

```
1 char data[] = "he...ll..o, worl..d!";
2 std::regex reg("\\.");
3 std::cout << std::regex_replace(data, reg, "");
```

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 ll n;
5 string s, t;
6 signed main()
7 {
8     scanf("%lld%c", &n);
9     while (n--)
10    {
11         getline(cin, s);
12         cin >> t;
13         cin.ignore();
14         regex reg("\b" + t + "\b");
15         smatch m;
16         auto lf = s.cbegin(); // begin(), end()参数不匹配
17         auto rf = s.cend();
18         ll ans = 0;
19         for (; regex_search(lf, rf, m, reg); lf = m.suffix().first)
20             ++ans;
21         cout << ans << '\n';
22     }
23     return 0;
24 }

```

## 随机数

```

1 #include <iostream>
2 #include <chrono>
3 #include <random>
4 using namespace std;
5 int main()
6 {
7     //unsigned seed =
8     std::chrono::system_clock::now().time_since_epoch().count();
9     //mt19937 rand_num(seed); 方法一
10    random_device divc;
11    mt19937 rand_num(divc()); //方法二: Linux真随机数
12    uniform_int_distribution<long long> dist(0, 1000000000);
13    cout << dist(rand_num) << endl;
14    return 0;
15 }

```

distribution 范围支持负数和超过 int 的范围(即 long long)。

`uniform_real_distribution<T>` 生成实数范围

随机打乱: `shuffle(first, last, myrand)` , `myrand` 通常用 `mt19937`

## 杂项

**unique** : 排序后使用, 返回去重后下界。

```
1 | 11 m = unique(a + 1, a + 6) - (a + 1);
```

原地去重:

```
1 | nums.erase(unique(nums.begin(), nums.end()), nums.end());
```

**nth\_element** : (a,a+k,a+n,cmp); 使容器a+k所在元素左边都小于它, 右边都大于它(a+k通常移位), 不返回值

```
1 | nth_element(a + 1, a + k, a + n + 1); //使得a[k]归位
```

**partial\_sort** : 保证[lf,cf)有序, [cf,rf]按本来的相对顺序。排序逆序可以如 `greater<11>()` 做参数

```
1 | partial_sort(lf,cf,rf);
```

**stable\_sort** : 用法同 `std::sort` , 区别在于是稳定排序

**advance** : 使迭代器前进n个元素, 适用于所有迭代器, 第一个参数是迭代器引用, 第二个参数是整数距离, 表示前进或后退的单位(正负控制)。

**max\_element** : 返回首个最大元素所在位置迭代器

**inplace\_merge** : (lf,cf,rf,cmp) 对  $[lf, cf)$  和  $[cf, rf)$  就地归并排序

```
1 | inplace_merge(a + lf, a + cf + 1, a + rf + 1);
```

**merge** : (lf1, rf1, lf2, rf2, res, cmp) 归并排序并放到结果迭代器

## 数据结构

### string

构造函数

```
1 | string two(20, '*');
2 | string five(one, 10);
3 | string six(one+1, one+10);
4 | string eight(one, 1, 16);
```

常用方法: (复杂度都是暴力实现的复杂度)

- `append(string s)` 或 `push_back(char c)` 末尾增加, 等效于 `+=`
- `find(char/string s, start=0)` 从 start 下标开始找第一个出现的子串, 返回出现下标或 -1  
此外还有 `find_first_of` , `find_last_of` , `find_first_not_of` , `find_last_not_of`
- `substr(int start[, int len])` 从 start 下标开始截长为 len 的子串(缺省则截到尾)

- `erase(int pos, int n)` 删除 pos 下标开始的 n 个字符
- `insert(int pos, string s)` 在 pos 位置插入字符串 s
- `append(string s, int pos, int n)` 等于 `append(s.substr(pos, n))`
- `back()` 取最后一个字符

常用函数：

- `reverse(首迭代器, 尾迭代器)` 反转传入的字符串
- `getline(cin, string)` 读整行(注意跟 `cin.getline` 区分)
- `to_string(any)` 其它数值类型转 `std::string` (注: `char` 会视为 `int`, 不能传C风格字符串)
- `stoi`, `stol`, `stof`, `stod`, 字符串转 `int`, `long long`, `float`, `double`
- `reduce(首, 尾, v)` 令每个元素为  $a_i = i + v$
- `lexicographical_compare(迭代器a首, 迭代器a尾, 迭代器b首, 迭代器b尾)` 字典序比较两容器(不一定是字符串)
- `replace(首迭代器, 尾迭代器, 待替换字符, 替换成的字符)` 批量替换
- `count(首迭代器, 尾迭代器, 字符)` 字符出现次数统计
- `reduce(首, 尾, 起始值)` 返回容器元素之和加上起始值, C++17

`stringstream`: 从里面读出数据类比 `cin`, 用 `ss >> v` ; 把数据塞进去用 `<<`

## vector

```

1 vector<数据类型>变量名 {元素数};
2 vector<数据类型>变量名 (长度);
3 vector<数据类型>变量名 (长度, 初始值);
4 vector<数据类型>变量名 = 变量名2;
5 vector<数据类型>变量名(变量名2);

```

多维：

```

1 vector<vector<int> > x(10, vector<int>(20));
2 vector< vector< vector<int> > > x(5, vector<vector<int>>(8, vector<int>(10)));
3 vector< vector< vector<int> > > x(5, vector<vector<int>>(8, vector<int>(10,
1)));
4 //1是初始值
5 vector< vector< vector<int> > > vector_3D_5(n, vector_2D_4);
6 vector< vector< vector<int> > > vector_3D_5(n);
7 //仅仅指定一维大小。当然二维也可以这样的。
8 vector<vector<vector<vector<int>>> x(3, vector<vector<vector<int>>>
(5, vector< vector<int>>(7, vector<int>(10, 1))));

```

`resize(size, value)`方法改变大小并初始化, 默认value0

`copy(a.begin(), a.end(), b.begin())`, 注意b要有足够的空间, 可以初始化时给。

`emplace_back()`在容器尾部添加一个元素, 这个元素原地构造, 不需要触发拷贝构造和转移构造。而且调用形式更加简洁, 直接根据参数初始化临时对象的成员。(当然int之类的也能用)

rbegin(),rend()可以进行反向输出。(返回reverse\_iterartor)

```
1 | for(auto p = a.rbegin();p != a.rend();p++) cout<<*p<<endl;
```

front()方法返回首元素的引用, 例如: c.front() = 3;

back()方法同理

insert(p, value)插入 O(n)

insert(p, b1, b2);将[b1,b2)内元素插入到p位置前面

erase(p)删除 也可以用诸如x.end()-1;O(n)

erase(p, q)删除区间[p,q)的元素

如: erase(s.begin(),s.begin()+2)删除头两个元素。

## set

想要元素可变尝试设置结构体/类成员属性为 `mutable`。

逆序 `set<类型, greater<类型>>`

可以`set<类型> var{数组, 数组+len}`来获取数组的set化

`insert(元素值)` 返回 pair, 分别代表迭代器和是否成功插入

`insert(sp, se)`方法插入闭区间

`erase(元素值)`方法。可以删除不存在的值, 将会忽略操作

`erase(迭代器)`方法, 如`x.end()`

对 `multiset` 而言, `erase` 元素值会删掉所有同值, 迭代器则删单个

`find()`有, 返回迭代器, 无, 返回末尾`end()`

并集`set_union(Ab, Ae, Bp, Be, 输出迭代器)//algorithm`

交集`set_intersection`

差`set_difference`

重载 `unset`: 模板添加一个类,重载`==`方法与返回 `size_t` 的()方法

```
1 | #include <bits/stdc++.h>
2 | using namespace std;
3 | unordered_set<int> s = {1, 1, 4, 5, 1, 4};
4 | struct node
5 | {
6 |     int x, y, z;
7 |     bool operator==(const node &r) const
8 |     {
9 |         return x == r.x && y == r.y && z == r.z;
10 |     }
11 |     size_t operator()(const node &r) const
12 |     {
13 |         return r.x * 1e9 + r.y * 1e5 + r.z;
14 |     }
15 | };
16 | int main()
17 | {
18 |     cout << s[0] << endl;
19 |     cout << s[1] << endl;
20 |     cout << s[2] << endl;
21 |     cout << s[3] << endl;
22 |     cout << s[4] << endl;
23 |     cout << s[5] << endl;
24 | }
```

```

14     }
15 };
16 unordered_set<node, node> s2;
17 int main()
18 {
19     printf("%d\n", s.size());
20     s2.insert({1, 2, 3});
21     s2.insert({1, 2, 3});
22     s2.insert({1, 2, 4});
23     printf("%d\n", s2.size());
24     return 0;
25 }

```

## map

count方法：若有查找key，返回1，否则0

erase(key)

pair<key类型,value类型> target = \*T.find(key) //注意区分[]

## bitset

```

1 bitset<位数> var;
2 bitset<d> s1(12);
3 string str="100110";
4 bitset<d> s2(str);
5 //如果是整型，那么填充时会自动转换为二进制，如果是字符串，但是其中出现了 0/1 以外的字符，就会爆炸

```

可以直接当 bool 数组来用。也有方法：

1. 基本的位操作，如： $s| = s << w[i]$ ;
2. .count()计算1的个数
3. .any()是否存在1
4. .none()是否存在1
5. .set()全设为1
6. .reset()全设为0
7. .flip()按位取反
8. .test(i) 第i位是不是1

上述函数可以传一个参数，则只对该位操作

## 其他

priority\_queue

默认大根堆。定义小根堆：(或自行写结构体)

```
1 | priority_queue<int, vector<int>, greater<int> > q;
```

tuple

直接用 {} 构造

取出(如 map 迭代器里的可能要先 second 一下), 然后用 tie(a,b,c)=tuple 变量 的方法获得每个元素

list 双向链表

deque

array, valarray

complex<T> 复数, 支持 (,) 构造, 能传入 abs, arg ([-π, π] 离 x 正半最近的偏移), 能和 T 运算。  
conj 共轭复数, norm 范数, polar(T rho, T theta) 构造对应复数

## pb\_ds

有一种比 unordered\_map 更快的哈希表, 具体参见上文 [线段树-树上二分]。

```
1 | #include <ext/pb_ds/assoc_container.hpp>
2 | #include <ext/pb_ds/tree_policy.hpp>
3 | __gnu_pbds::tree<类型, __gnu_pbds::null_type, less<类型>,
4 | __gnu_pbds::rb_tree_tag, __gnu_pbds::tree_order_statistics_node_update> 变量名;
```

- order\_of\_key(x) 返回值 x 在 set 的下标
- find\_by\_order(x) 返回下标为 x 的迭代器

使用示例: (洛谷P6136)

题意: 输入  $n, m (1 \leq n \leq 10^5, 1 \leq m \leq 10^6)$ , 输入  $a_i (0 \leq a_i < 2^{30})$ , 接下来有 6 种在线操作 ( $x = x \oplus \text{last}$ ):

1. 插入整数  $x (0 \leq x < 2^{30})$
2. 删除整数  $x$  (若有多个相同, 只删一个)
3. 查询整数  $x$  的排名(比它小的数个数 +1)
4. 查询排名为  $x$  的数(不存在时查小于  $x$  的最大数, 保证  $x$  不越当前界)
5. 求  $x$  前驱(小于  $x$  的最大的数)
6. 求  $x$  后继(大于  $x$  的最小的数)

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #include <ext/pb_ds/assoc_container.hpp>
5 #include <ext/pb_ds/tree_policy.hpp>
6 __gnu_pbds::tree<ll, __gnu_pbds::null_type, less<ll>,
7 __gnu_pbds::rb_tree_tag, __gnu_pbds::tree_order_statistics_node_update> tr;
8 ll n, m, lastans, dig, ans;
9 signed main()
10 {
11     scanf("%lld%lld", &n, &m);
12     dig = n + m;
13     for (ll i = 1, a; i <= n; ++i)
14     {
15         scanf("%lld", &a);
16         tr.insert(a * dig + i);
17     }
18     for (ll i = n + 1, opt, x; i <= m + n; ++i)
19     {
20         scanf("%lld%lld", &opt, &x);
21         x ^= lastans;
22         if (opt == 1)
23         {
24             tr.insert((x * dig) + i);
25         }
26         else if (opt == 2)
27         {
28             tr.erase(tr.lower_bound(x * dig));
29         }
30         else
31         {
32             if (opt == 3)
33             {
34                 lastans = tr.order_of_key(x * dig) + 1;
35             }
36             else if (opt == 4)
37             {
38                 lastans = (*tr.find_by_order(x - 1)) / dig;
39             }
40             else if (opt == 5)
41             {
42                 lastans = (*--tr.lower_bound(x * dig)) / dig;
43             }
44             else if (opt == 6)
45             {
46                 lastans = (*tr.upper_bound(x * dig + dig)) / dig;
47             }
48             ans ^= lastans;
49         }
50     }
51     printf("%lld\n", ans);
52     return 0;
53 }

```

# 卡常

## 快读/写

关闭同步流略。好看版快读：

```
1 11 read()
2 {
3     11 num = 0;
4     char c = getchar(), up = c;
5     while (c < '0' || c > '9')
6         up = c, c = getchar();
7     while (c >= '0' && c <= '9')
8         num = (num << 1) + (num << 3) + (c ^ '0'), c = getchar();
9     return up == '-' ? -num : num;
10 }
```

位运算快读：

```
1 11 read()
2 {
3     char p = 0;
4     11 r = 0, o = 0;
5     for (; p < '0' || p > '9'; o |= p == '-' , p = getchar())
6         ;
7     for (; p >= '0' && p <= '9'; r = (r << 1) + (r << 3) + (p ^ 48), p = getchar())
8         ;
9     return o ? (~r) + 1 : r;
10 }
```

快写：

```
1 void write(11 x)
2 {
3     if (x < 0)
4     {
5         putchar('-');
6         x = -x;
7     }
8     if (x > 9)
9     {
10         write(x / 10);
11     }
12     putchar(x % 10 + '0');
13 }
```

更快的读写：

- 短板 (仅快读)

```

1 #define gc() (is == it ? it = (is = in) + fread(in, 1, Q, stdin), (is == it ? 
2 EOF : *is++) : *is++)
3 const int Q = (1 << 24) + 1;
4 char in[Q], *is = in, *it = in, c;
5 void read(long long &n) {
6     for (n = 0; (c = gc()) < '0' || c > '9';)
7         for (; c <= '9' && c >= '0'; c = gc()) n = n * 10 + c - 48;
7 }

```

- 长版

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #define sc(x) scanf("%lld", &x)
5 const ll MAXSIZE = 1 << 20;
6 char buf[MAXSIZE], *p1, *p2;
7 #define gc() (p1 == p2 && (p2 = (p1 = buf) + fread(buf, 1, MAXSIZE, stdin), 
8 p1 == p2) ? EOF : *p1++)
9 inline ll rd()
10 {
11     ll x = 0, f = 1;
12     char c = gc();
13     while (!isdigit(c))
14     {
15         if (c == '-')
16             f = -1;
17         c = gc();
18     }
19     while (isdigit(c))
20         x = x * 10 + (c ^ 48), c = gc();
21     return x * f;
22 }
23 char pbuf[1 << 20], *pp = pbuf;
24 inline void push(const char &c)
25 {
26     if (pp - pbuf == 1 << 20)
27         fwrite(pbuf, 1, 1 << 20, stdout), pp = pbuf;
28     *pp++ = c;
29 }
30 inline void write(ll x)
31 {
32     if (x < 0)
33         x = -x, push('-');
34     static ll sta[35];
35     ll top = 0;
36     do
37     {
38         sta[top++] = x % 10, x /= 10;
39     } while (x);
40     while (top)
41         push(sta[--top] + '0');
42 }
43 signed main()
44 {

```

```
44     ll a = rd(), b = rd();
45     write(a + b);
46     fwrite(pbuf, 1, pp - pbuf, stdout);
47     return 0;
48 }
```

Python:

```
1 import sys
2 input = sys.stdin.readline # 实测约快10倍
3 print = sys.stdout.write # 仅字符串输出(注意不会换行)
```

Java

```
1 import java.io.*;
2
3 public class Main {
4     static StreamTokenizer scanner = new StreamTokenizer(new
5 BufferedReader(new InputStreamReader(System.in)));
6     static PrintWriter out = new PrintWriter(new BufferedWriter(new
7 OutputStreamWriter(System.out)));
8
9     public static int nextInt() throws IOException {// 快约一倍
10        scanner.nextToken();
11        return (int) scanner.nval;
12    }
13
14    public static long nextLong() throws IOException {
15        scanner.nextToken();
16        return (long) scanner.nval;
17    }
18
19    static String next() throws IOException {
20        scanner.nextToken();
21        return scanner.sval;
22    }
23
24    public static void main(String[] args) throws IOException {
25        int n = nextInt() / 10;
26        for (int i = 0; i < n; ++i) {
27            int s = 0;
28            for (int j = 0; j < 10; ++j) {
29                s += nextInt();
30            }
31            out.println(s); // 实测快约4倍
32        }
33        out.close(); // 没有的话什么也不输出
34    }
35 }
```

## 其他

```
1 | #pragma GCC optimize(2)
```

快速乘:  $(a \times b \bmod p)$

```
1 | return ((ull)a*b-(ull)((ull)a/k*b)*k)%k;
```

防炸 || 不开 i128 的模乘:

```
1 | LL mul(LL a, LL b, LL P){  
2 |     LL L = a * (b >> 25LL) % P * (1LL << 25) % P;  
3 |     LL R = a * (b & ((1LL << 25) - 1)) % P;  
4 |     return (L + R) % P;  
5 | }
```

|