

Standard Code Library

Magic Vegetable(!!snb!)

Shanghai University

October 30, 2019

Contents

数据范围表	3
宏定义	3
VIM 配置	3
数学	6
欧拉筛	6
筛莫比乌斯函数	6
莫比乌斯反演, 整除分块	7
公式	8
连分数	10
常见狄利克雷卷积	10
杜教筛	11
一些积性函数的表	11
快速幂	12
快速乘	12
扩展欧几里得算法	12
类欧几里得算法	13
逆元	14
组合数	14
拉格朗日插值	15
线性递推	15
Fast Transfroming	17
离散对数	23
二次剩余	24
中国剩余定理	25
线性基	26
高斯消元	27
素数测试	28
Pollard Rho	29
博弈	29
字符串	30
KMP	30
Manacher	30
后缀数组	31
后缀自动机	33
广义后缀自动机	34
AC 自动机	34
最小表示法	36
回文自动机	37
EXKMP	38
数据结构	40
无旋 Treap	40
ST 表	40
树状数组	41
笛卡尔树	42
主席树优化建图	42
Link-Cut Tree	43

Splay	46
时间分治线段树	50
柯朵莉树	52
K-D Tree	52
多边形	57
圆和线段	58
其他	62
图论	65
树链剖分	65
边双联通分量	65
虚树	66
网络流	68
最小树形图	71
树的最大匹配	73
支配树	73
斯坦纳树	75
2-SAT	77
网络流基本建图技巧	78
一些公式	78
杂项	79
简单随机数	79
输入输出挂	79
HDU 专用	79
日期	80
子集枚举	81
大数	81

数据范围表

unsigned int : $0 \sim 4294967295$

int : $-2147483648 \sim 2147483647 \quad (1 \ll 30) + ((1 \ll 30) - 1) \quad 2.1 \times 10^9$

ull : $0 \sim 18446744073709551615 \quad 1.8 \times 10^{19}$

ll_{max} : $9223372036854775807 \quad (1LL \ll 62) + ((1LL \ll 62) - 1) \quad 9.2 \times 10^{18}$

0x3f3f3f3f : $1061109567 \quad 1 \times 10^9$

0x3f3f3f3f3f3f3f3f : $4557430888798830399 \quad 4.5 \times 10^{18}$

宏定义

- debug 宏

```
1  #define compute
2  #ifdef compute
3  #define dbg(x...) do{cout << "\033[32;1m" << #x << "->" ; err(x);} while(0)
4  void err(){cout << "\033[39;0m" << endl;}
5  template<template<typename...> class T,typename t,typename... A>
6  void err(T<t> a,A... x){for (auto v:a) cout << v << ' '; err(x...);}
7  template<typename T,typename... A>
8  void err(T a,A... x){cout << a << ' '; err(x...);}
9  #else
10 #define dbg(...)
11 #endif
```

- 更多配色: 33-黄色, 34-蓝色, 31-橙色

VIM 配置

- 比赛用

```
1  set nu
2  set hlsearch
3  set tabstop=4
4  syntax on
5  set shiftwidth=4
6  set cindent
7  set mouse=a
8  set cursorline
9  set cursorcolumn
```

- compute 用

```
1  set nocompatible " be iMproved, required
2  filetype off " required
3  " set the runtime path to include Vundle and initialize
4  set rtp+=~/.vim/bundle/Vundle.vim
5  call vundle#begin()
6  " alternatively, pass a path where Vundle should install plugins
7  "call vundle#begin('~/.vim/bundle/Vundle.vim')
8  " let Vundle manage Vundle, required
```

```

9  Plugin 'VundleVim/Vundle.vim'
10 Plugin 'luochen1990/rainbow'
11 call vundle#end() " required
12 filetype plugin indent on " required
13 " To ignore plugin indent changes, instead use:
14 "filetype plugin on
15 "
16 " Brief help
17 " :PluginList - lists configured plugins
18 " :PluginInstall - installs plugins; append `!` to update or just :PluginUpdate
19 " :PluginSearch foo - searches for foo; append `!` to refresh local cache
20 " :PluginClean - confirms removal of unused plugins; append `!` to auto-approve
   ↪ removal
21 "
22 " see :h vundle for more details or wiki for FAQ
23 " Put your non-Plugin stuff after this line
24 set nu
25 set tabstop=4
26 syntax on
27 set shiftwidth=4
28 set cin
29 set mouse=a
30 set ruler
31 set cursorline
32 set cursorcolumn
33 set cindent
34 set autoindent
35 let g:rainbow_active=1
36 " 主题 solarized
37 Bundle 'altercation/vim-colors-solarized'
38 "let g:solarized_termcolors=256
39 let g:solarized_termtrans=1
40 let g:solarized_contrast="normal"
41 let g:solarized_visibility="normal"
42 " 主题 molokai
43 Bundle 'tomasr/molokai'
44 let g:molokai_original = 1
45 set background=dark
46 set t_Co=256
47 "colorscheme solarized
48 colorscheme molokai
49 "colorscheme phd
50 "kakko comp
51 inoremap ( (<Esc>i
52 inoremap [ [<Esc>i
53 inoremap { {<Esc>i
54 inoremap ' '<Esc>i
55 inoremap " "<Esc>i
56 inoremap ) <c-r>=ClosePair('')<CR>
57 inoremap } <c-r>=ClosePair('}')<CR>
58 inoremap ] <c-r>=ClosePair(']')<CR>
59 function ClosePair(char)

```

```
60     if getline('.')[col('.')-1]==a:char
61         return "\<Right>"
62     else
63         return a:char
64     endif
65 endfunction
66 set completeopt=longest,menu
```

数学

欧拉筛

```
1  const int maxn = 1e7 + 10;
2  int prime[maxn] = {0}, phi[maxn] = {0}, tot;
3
4  void euler()
5  {
6      phi[1] = 1;
7      for (int i = 2; i < maxn; i++)
8      {
9          if (!phi[i])
10         {
11             prime[tot++] = i;
12             phi[i] = i - 1;
13         }
14         for (int j = 0; j < tot && i * prime[j] < maxn; j++)
15         {
16             if (i % prime[j] == 0)
17             {
18                 phi[i * prime[j]] = phi[i] * prime[j];
19                 break;
20             }
21             phi[i * prime[j]] = phi[i] * phi[prime[j]];
22         }
23     }
24 }
```

筛莫比乌斯函数

```
1  const int maxn = 1e7 + 10;
2  int prime[maxn], tot = 0, mu[maxn];
3  bool check[maxn];
4
5  void mobius()
6  {
7      mu[1] = 1;
8      for (int i = 2; i < maxn; i++)
9      {
10         if (!check[i])
11         {
12             prime[tot++] = i;
13             mu[i] = -1;
14         }
15         for (int j = 0; j < tot && i * prime[j] < maxn; j++)
16         {
17             check[i * prime[j]] = true;
18             if (i % prime[j] == 0)
19             {
20                 mu[i * prime[j]] = 0;
21                 break;
22             }
23         }
24     }
25 }
```

```

23         mu[i * prime[j]] = -mu[i];
24     }
25 }
26 }

```

莫比乌斯反演，整除分块

$$F(n) = \sum_{d|n} f(d) \Rightarrow f(n) = \sum_{d|n} \mu(d) F\left(\frac{n}{d}\right)$$

$$F(n) = \sum_{n|d} f(d) \Rightarrow f(n) = \sum_{n|d} \mu\left(\frac{d}{n}\right) F(d)$$

```

1  ll prime[maxn], tot = 0, mu[maxn], sum[maxn];
2  bool check[maxn];
3
4  void getmu()
5  {
6      mu[1] = 1;
7      for (int i = 2; i < maxn; i++)
8      {
9          if (!check[i])
10             {
11                 prime[tot++] = i;
12                 mu[i] = -1;
13             }
14             for (int j = 0; j < tot && i * prime[j] < maxn; j++)
15             {
16                 check[i * prime[j]] = true;
17                 if (i % prime[j] == 0)
18                 {
19                     mu[i * prime[j]] = 0;
20                     break;
21                 }
22                 mu[i * prime[j]] = -mu[i];
23             }
24     }
25     for (int i = 1; i < maxn; i++) //前缀和
26         sum[i] = sum[i - 1] + mu[i];
27 }
28
29 ll cal(int a, int b) //整除分块 i:1->a j:1->b gcd(i,j)=1 对数
30 {
31     if (a > b)
32         swap(a, b);
33     ll l = 1, r, ans = 0;
34     while (l <= a)
35     {
36         r = min(a / (a / l), b / (b / l));
37         ans += (sum[r] - sum[l - 1]) * (a / l) * (b / l);
38         l = r + 1;
39     }
40     return ans;
41 }

```


公式

一些数论公式

- 当 $x \geq \phi(p)$ 时有 $a^x \equiv a^{x \bmod \phi(p) + \phi(p)} \pmod{p}$
- $\mu^2(n) = \sum_{d^2|n} \mu(d)$
- $\sum_{d|n} \varphi(d) = n$
- $\sum_{d|n} 2^{\omega(d)} = \sigma_0(n^2)$, 其中 ω 是不同素因子个数
- $\sum_{d|n} \mu^2(d) = 2^{\omega(n)}$

一些数论函数求和的例子

- $\sum_{i=1}^n i[\gcd(i, n) = 1] = \frac{n\varphi(n) + [n=1]}{2}$
- $\sum_{i=1}^n \sum_{j=1}^m [\gcd(i, j) = x] = \sum_d \mu(d) \lfloor \frac{n}{dx} \rfloor \lfloor \frac{m}{dx} \rfloor$
- $\sum_{i=1}^n \sum_{j=1}^m \gcd(i, j) = \sum_{i=1}^n \sum_{j=1}^m \sum_{d|\gcd(i, j)} \varphi(d) = \sum_d \varphi(d) \lfloor \frac{n}{d} \rfloor \lfloor \frac{m}{d} \rfloor$
- $S(n) = \sum_{i=1}^n \mu(i) = 1 - \sum_{i=1}^n \sum_{d|i, d < i} \mu(d) \stackrel{t=\frac{i}{d}}{=} 1 - \sum_{t=2}^n S(\lfloor \frac{n}{t} \rfloor)$
– 利用 $[n=1] = \sum_{d|n} \mu(d)$
- $S(n) = \sum_{i=1}^n \varphi(i) = \sum_{i=1}^n i - \sum_{i=1}^n \sum_{d|i, d < i} \varphi(i) \stackrel{t=\frac{i}{d}}{=} \frac{i(i+1)}{2} - \sum_{t=2}^n S(\frac{n}{t})$
– 利用 $n = \sum_{d|n} \varphi(d)$
- $\sum_{i=1}^n \mu^2(i) = \sum_{i=1}^n \sum_{d^2|n} \mu(d) = \sum_{d=1}^{\lfloor \sqrt{n} \rfloor} \mu(d) \lfloor \frac{n}{d^2} \rfloor$
- $\sum_{i=1}^n \sum_{j=1}^n \gcd^2(i, j) = \sum_d d^2 \sum_t \mu(t) \lfloor \frac{n}{dt} \rfloor^2$
 $\stackrel{x=dt}{=} \sum_x \lfloor \frac{n}{x} \rfloor^2 \sum_{d|x} d^2 \mu(\frac{x}{d})$
- $\sum_{i=1}^n \varphi(i) = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n [i \perp j] - 1 = \frac{1}{2} \sum_{i=1}^n \mu(i) \cdot \lfloor \frac{n}{i} \rfloor^2 - 1$

斐波那契数列性质

- $F_{a+b} = F_{a-1} \cdot F_b + F_a \cdot F_{b+1}$
- $F_1 + F_3 + \dots + F_{2n-1} = F_{2n}, F_2 + F_4 + \dots + F_{2n} = F_{2n+1} - 1$
- $\sum_{i=1}^n F_i = F_{n+2} - 1$
- $\sum_{i=1}^n F_i^2 = F_n \cdot F_{n+1}$
- $F_n^2 = (-1)^{n-1} + F_{n-1} \cdot F_{n+1}$
- $\gcd(F_a, F_b) = F_{\gcd(a, b)}$
- 模 n 周期 (皮萨诺周期)
 - $\pi(p^k) = p^{k-1} \pi(p)$
 - $\pi(nm) = \text{lcm}(\pi(n), \pi(m)), \forall n \perp m$
 - $\pi(2) = 3, \pi(5) = 20$
 - $\forall p \equiv \pm 1 \pmod{10}, \pi(p) | p - 1$
 - $\forall p \equiv \pm 2 \pmod{5}, \pi(p) | 2p + 2$

常见生成函数

- $(1 + ax)^n = \sum_{k=0}^n \binom{n}{k} a^k x^k$
- $\frac{1 - x^{r+1}}{1 - x} = \sum_{k=0}^r x^k$
- $\frac{1}{1 - ax} = \sum_{k=0}^{\infty} a^k x^k$
- $\frac{1}{(1 - x)^2} = \sum_{k=0}^{\infty} (k + 1) x^k$

- $\frac{1}{(1-x)^n} = \sum_{k=0}^{\infty} \binom{n+k-1}{k} x^k$
- $e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}$
- $\ln(1+x) = \sum_{k=0}^{\infty} \frac{(-1)^{k+1}}{k} x^k$

佩尔方程

若一个丢番图方程具有以下形式： $x^2 - ny^2 = 1$ 。且 n 为正整数，则称此二元二次不定方程为**佩尔方程**。

若 n 是完全平方数，则这个方程式只有平凡解 $(\pm 1, 0)$ （实际上对任意的 n ， $(\pm 1, 0)$ 都是解）。对于其余情况，拉格朗日证明了佩尔方程总有非平凡解。而这些解可由 \sqrt{n} 的连分数求出。

$$x = [a_0; a_1, a_2, a_3] = x = a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \ddots}}}$$

设 $\frac{p_i}{q_i}$ 是 \sqrt{n} 的连分数表示： $[a_0; a_1, a_2, a_3, \dots]$ 的渐近分数列，由连分数理论知存在 i 使得 (p_i, q_i) 为佩尔方程的解。取其中最小的 i ，将对应的 (p_i, q_i) 称为佩尔方程的基本解，或最小解，记作 (x_1, y_1) ，则所有的解 (x_i, y_i) 可表示成如下形式： $x_i + y_i \sqrt{n} = (x_1 + y_1 \sqrt{n})^i$ 。或者由以下的递回关系式得到：

$$x_{i+1} = x_1 x_i + n y_1 y_i, y_{i+1} = x_1 y_i + y_1 x_i$$

但是：佩尔方程千万不要去推（虽然推起来很有趣，但结果不一定好看，会是两个式子）。记住佩尔方程结果的形式通常是 $a_n = k a_{n-1} - a_{n-2}$ （ a_{n-2} 前的系数通常是 -1 ）。暴力 / 凑出两个基础解之后加上一个 0，容易解出 k 并验证。

Burnside & Polya

$$\bullet |X/G| = \frac{1}{|G|} \sum_{g \in G} |X^g|$$

注： X^g 是 g 下的不动点数量，也就是说有多少种东西用 g 作用之后可以保持不变。

$$\bullet |Y^X/G| = \frac{1}{|G|} \sum_{g \in G} m^{c(g)}$$

注：用 m 种颜色染色，然后对于某一种置换 g ，有 $c(g)$ 个置换环，为了保证置换后颜色仍然相同，每个置换环必须染成同色。

皮克定理

$$2S = 2a + b - 2$$

- S 多边形面积
- a 多边形内部点数
- b 多边形边上点数

莫比乌斯反演

- $g(n) = \sum_{d|n} f(d) \Leftrightarrow f(n) = \sum_{d|n} \mu(d) g(\frac{n}{d})$
- $f(n) = \sum_{n|d} g(d) \Leftrightarrow g(n) = \sum_{n|d} \mu(\frac{d}{n}) f(d)$

低阶等幂求和

- $\sum_{i=1}^n i^1 = \frac{n(n+1)}{2} = \frac{1}{2}n^2 + \frac{1}{2}n$
- $\sum_{i=1}^n i^2 = \frac{n(n+1)(2n+1)}{6} = \frac{1}{3}n^3 + \frac{1}{2}n^2 + \frac{1}{6}n$
- $\sum_{i=1}^n i^3 = \left[\frac{n(n+1)}{2}\right]^2 = \frac{1}{4}n^4 + \frac{1}{2}n^3 + \frac{1}{4}n^2$
- $\sum_{i=1}^n i^4 = \frac{n(n+1)(2n+1)(3n^2+3n-1)}{30} = \frac{1}{5}n^5 + \frac{1}{2}n^4 + \frac{1}{3}n^3 - \frac{1}{30}n$
- $\sum_{i=1}^n i^5 = \frac{n^2(n+1)^2(2n^2+2n-1)}{12} = \frac{1}{6}n^6 + \frac{1}{2}n^5 + \frac{5}{12}n^4 - \frac{1}{12}n^2$

一些组合公式

- 错排公式: $D_1 = 0, D_2 = 1, D_n = (n-1)(D_{n-1} + D_{n-2}) = n! \left(\frac{1}{2!} - \frac{1}{3!} + \dots + (-1)^n \frac{1}{n!} \right) = \lfloor \frac{n!}{e} + 0.5 \rfloor$
- 卡特兰数 (n 对括号合法方案数, n 个结点二叉树个数, $n \times n$ 方格中对角线下方的单调路径数, 凸 $n+2$ 边形的三角形划分数, n 个元素的合法出栈序列数): $C_n = \frac{1}{n+1} \binom{2n}{n} = \frac{(2n)!}{(n+1)!n!}$

连分数

- 可表示为 $\frac{p_k}{q_k} = a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \frac{1}{a_4 + \dots}}} = [a_1, a_2, a_3, a_4, \dots]$
- 可以根据线性递得到以下式子:

$$p_k = \begin{cases} a_1 & k=1 \\ a_1 a_2 + 1 & k=2 \\ a_k p_{k-1} + p_{k-2} & k \geq 3 \end{cases}$$

$$q_k = \begin{cases} 1 & k=1 \\ a_2 & k=2 \\ a_k q_{k-1} + q_{k-2} & k \geq 3 \end{cases}$$

- 写成矩阵形式即为:

$$\begin{bmatrix} p_n & p_{n-1} \\ q_n & q_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} a_1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} a_2 & 1 \\ 1 & 0 \end{bmatrix} \dots \begin{bmatrix} a_n & 1 \\ 1 & 0 \end{bmatrix}$$

常见狄利克雷卷积

积性函数

$$\epsilon(n) = [n=1]$$

$$1(n) = 1$$

$$Id(n) = n$$

$$d(n) = \sum_{d|n} 1$$

$$\sigma(n) = \sum_{d|n} d$$

卷积

$$\mu * 1 = \epsilon$$

$$\phi * 1 = Id$$

$$1 * 1 = d$$

杜教筛

$O(n^{2/3})$ 求积性函数前缀和。

假设要求的积性函数为 f ，前缀和为 S ，能找到较为简单（前缀和好求）的积性函数 g ，使得 $f * g = h$ 前缀和容易求出。那么有：

$$g(1)S(n) = \sum_{i=1}^n h(i) - \sum_{d=2}^n g(d) \cdot S(\lfloor n/d \rfloor)$$

其中 $g(1)$ 一般来说都是 1， h 和 g 的前缀和需要比较好算，然后就能记忆化搜索了。

```
1 //杜教筛欧拉函数 原理:  $\phi * I = Id$ 
2 const int maxn=1e6+10; //通常预处理到  $n^{2/3}$  为最优
3 ll phi[maxn];
4 unordered_map<int,ll> phi2; //hash
5
6 ll djsphi(int n)
7 {
8     if (n < maxn)
9         return phi[n];
10    if (phi2[n])
11        return phi2[n];
12    ll ans = 1ll * n * (n + 1) / 2;
13    for (int l = 2, r; l <= n; l = r + 1)
14    {
15        r = n / (n / l);
16        ans -= djsphi(n / l) * (r - l + 1);
17    }
18    return phi2[n] = ans; //记忆化
19 }
```

一些积性函数的表

id	$\mu(i)$	$\phi(i)$	id	$\mu(i)$	$\phi(i)$	id	$\mu(i)$	$\phi(i)$
1	1	1	16	0	8	31*	-1	30
2*	-1	1	17*	-1	16	32	0	16
3*	-1	2	18	0	6	33	1	20
4	0	2	19*	-1	18	34	1	16
5*	-1	4	20	0	8	35	1	24
6	1	2	21	1	12	36	0	12
7*	-1	6	22	1	10	37*	-1	36
8	0	4	23*	-1	22	38	1	18
9	0	6	24	0	8	39	1	24
10	1	4	25	0	20	40	0	16
11*	-1	10	26	1	12	41*	-1	40
12	0	4	27	0	18	42	-1	12
13*	-1	12	28	0	12	43*	-1	42
14	1	6	29*	-1	28	44	0	20
15	1	8	30	-1	8	45	0	24

快速幂

```
1 ll quick(ll a, ll b)
2 {
3     ll ret = 1;
4     while (b)
5     {
6         if (b & 1)
7             ret = ret * a % mod;
8         a = a * a % mod;
9         b >>= 1;
10    }
11    return ret;
12 }
```

快速乘

```
1 ll mul(ll a, ll b, ll m)
2 {
3     ll ret = 0;
4     while (b)
5     {
6         if (b & 1)
7         {
8             ret += a;
9             if (ret >= m)
10                ret -= m;
11        }
12        a += a;
13        if (a >= m)
14            a -= m;
15        b >>= 1;
16    }
17    return ret;
18 }

1 ll mul(ll a, ll b, ll m)
2 {
3     return (a * b - ll((long double)a * b / m) * m + m) % m;
4 }

1 ll mul(ll a, ll b, ll m)
2 {
3     ll t = a * b - ll((long double)a * b / m) * m;
4     return t < 0 ? t + m : t;
5 }
```

扩展欧几里得算法

求解 $a \cdot x + b \cdot y = \gcd(a, b)$ 的一组特解。

```
1 ll exgcd(ll a, ll b, ll &x, ll &y) //返回 gcd(a,b)
2 {
3     if (b == 0)
4     {
```

```

5         x = 1, y = 0;
6         return a;
7     }
8     ll d = exgcd(b, a % b, y, x);
9     y -= x * (a / b);
10    return d;
11 }

```

类欧几里得算法

$$F(a, b, c, n) = \sum_{i=0}^n \lfloor \frac{a * i + b}{c} \rfloor$$

$$G(a, b, c, n) = \sum_{i=0}^n \lfloor i * \frac{a * i + b}{c} \rfloor$$

$$H(a, b, c, n) = \sum_{i=0}^n (\lfloor \frac{a * i + b}{c} \rfloor)^2$$

通用方法: 1. 当 $a \geq c$ 或者 $b \geq c$ 时, 通过 $\lfloor \frac{a}{c} \rfloor = \lfloor \frac{a \% c}{c} \rfloor + \lfloor \frac{a}{c} \rfloor$ 展开化简。2. 当 $a < c$ 且 $b < c$ 时, 通过枚举直线下的点, 交换枚举顺序展开化简后递归求解。3. 注意边界条件: $a = 0, n = 0$ 。4. 需要自己写多项取模相加/减/乘。

```

1  const ll mod = 1e9 + 7, inv2 = (mod + 1) / 2, inv6 = (mod + 1) / 6;
2
3  struct node
4  {
5      ll f, g, h;
6  };
7
8  node solve(ll a, ll b, ll c, ll n)
9  {
10     node ans, tmp;
11     if (a == 0)
12     {
13         ans.f = (n + 1) * (b / c) % mod;
14         ans.g = (b / c) * n % mod * (n + 1) % mod * inv2 % mod;
15         ans.h = (n + 1) * (b / c) % mod * (b / c) % mod;
16         return ans;
17     }
18     if (a >= c || b >= c)
19     {
20         tmp = solve(a % c, b % c, c, n);
21         ans.f = (tmp.f + (a / c) * n % mod * (n + 1) % mod * inv2 % mod + (b / c) *
22             ↪ (n + 1) % mod) % mod;
23         ans.g = (tmp.g + (a / c) * n % mod * (n + 1) % mod * (2 * n + 1) % mod * inv6
24             ↪ % mod + (b / c) * n % mod * (n + 1) % mod * inv2 % mod) % mod;
25         ans.h = ((a / c) * (a / c) % mod * n % mod * (n + 1) % mod * (2 * n + 1) %
26             ↪ mod * inv6 % mod +
27             (b / c) * (b / c) % mod * (n + 1) % mod + (a / c) * (b / c) % mod *
28             ↪ n % mod * (n + 1) % mod +
29             tmp.h + 2 * (a / c) % mod * tmp.g % mod + 2 * (b / c) % mod * tmp.f
30             ↪ % mod) %
31         mod;
32     }
33     return ans;

```

```

28     }
29     ll m = (a * n + b) / c;
30     tmp = solve(c, c - b - 1, a, m - 1);
31     ans.f = ((n * (m % mod) % mod - tmp.f) % mod + mod) % mod;
32     ans.g = ((n * (n + 1) % mod * (m % mod) % mod - tmp.f - tmp.h) % mod + mod) *
        ⇨ inv2 % mod;
33     ans.h = ((n * (m % mod) % mod * ((m + 1) % mod) % mod - 2 * tmp.g - 2 * tmp.f -
        ⇨ ans.f) % mod + mod) % mod;
34     return ans;
35 }

```

逆元

1. 使用费马小定理，要求模数 p 为素数。
2. 使用扩展欧几里得定理，不要求模数 p 为素数。

```

1 ll inv(ll a, ll p) //求 a 关于 p 的逆元
2 {
3     ll x, y;
4     ll d = exgcd(a, p, x, y);
5     if (d != 1)
6         return -1;
7     return (x % p + p) % p;
8 }

```

3. 基于 $inv(a) = (p - \lfloor p/a \rfloor) * inv(p\%a) \% p$ ，在 $O(n)$ 时间复杂度下求出逆元表。

```

1 void init()
2 {
3     inv[1] = 1;
4     for (int i = 2; i < maxn; i++)
5         inv[i] = (mod - mod / i) * 1LL * inv[mod % i] % mod;
6 }

```

组合数

$O(n)$ 时间复杂度内预处理出 n 以内的组合数。

```

1 void init() //inv, f, finv 都开 ll
2 {
3     inv[1] = 1;
4     for (int i = 2; i < maxn; i++)
5         inv[i] = (mod - mod / i) * inv[mod % i] % mod; //inv: 逆元
6     f[0] = finv[0] = 1; //f: 阶乘 finv: 阶乘逆元 (1/f)
7     for (int i = 1; i < maxn; i++)
8     {
9         f[i] = f[i - 1] * i % mod;
10        finv[i] = finv[i - 1] * inv[i] % mod;
11    }
12 }
13
14 ll C(int n, int m) //C(n, m)
15 {
16     if (m < 0 || m > n)

```

```

17         return 0;
18     return f[n] * finv[n - m] % mod * finv[m] % mod;
19 }

```

拉格朗日插值

$$L(x) = \sum_{i=0}^n y_i l_i(x)$$

其中

$$l_i(x) = \prod_{j=0 \& j \neq i}^n \frac{x - x_j}{x_i - x_j}$$

1. 需要组合数中的 init。
2. arr: 插值数组 n: 项数 (从 0 开始一共 n+1 项) x: 需要要求的值

```

1 ll lagrange(ll *arr, ll n, ll x)
2 {
3     if (x <= n)
4         return arr[x];
5     ll ans = 0, sgn = n & 1 ? -1 : 1;
6     for (int j = 0; j <= n; j++, sgn *= -1)
7         ans = (ans + mod + sgn * f[x] % mod * finv[x - n - 1] % mod * inv[x - j] %
            ↪ mod * finv[j] % mod * finv[n - j] % mod * arr[j] % mod) % mod;
8     return ans;
9 }

```

线性递推

BM 模板

要求: 1. 线性递推 2. 所有数都有逆元 3. k 阶线性递推需要 2k 项

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  #define rep(i,a,n) for (int i=a;i<n;i++)
4  #define per(i,a,n) for (int i=n-1;i>=a;i--)
5  #define pb push_back
6  #define mp make_pair
7  #define all(x) (x).begin(),(x).end()
8  #define fi first
9  #define se second
10 #define SZ(x) ((int)(x).size())
11 typedef vector<int> VI;
12 typedef long long ll;
13 typedef pair<int,int> PII;
14 const ll mod=1000000007; //修改成题目要求的模数
15 ll powmod(ll a,ll b) {ll res=1;a%=mod; assert(b>=0);
    ↪ for(;b;b>>=1){if(b&1)res=res*a%mod;a=a*a%mod;}return res;}
16 // head
17 int _,n;
18 namespace linear_seq {
19     const int N=10010;

```



```

20 ll res[N],base[N],_c[N],_md[N];
21
22 vector<int> Md;
23 void mul(ll *a,ll *b,int k) {
24     rep(i,0,k+k) _c[i]=0;
25     rep(i,0,k) if (a[i]) rep(j,0,k) _c[i+j]=(_c[i+j]+a[i]*b[j])%mod;
26     for (int i=k+k-1;i>=k;i--) if (_c[i])
27         rep(j,0,SZ(Md)) _c[i-k+Md[j]]=(_c[i-k+Md[j]]-_c[i]*_md[Md[j]])%mod;
28     rep(i,0,k) a[i]=_c[i];
29 }
30 int solve(ll n,VI a,VI b) { // a 系数 b 初值 b[n+1]=a[0]*b[n]+...
31     // printf("%d\n",SZ(b));
32     ll ans=0,pnt=0;
33     int k=SZ(a);
34     assert(SZ(a)==SZ(b));
35     rep(i,0,k) _md[k-1-i]=-a[i];_md[k]=1;
36     Md.clear();
37     rep(i,0,k) if (_md[i]!=0) Md.push_back(i);
38     rep(i,0,k) res[i]=base[i]=0;
39     res[0]=1;
40     while ((1ll<<pnt)<=n) pnt++;
41     for (int p=pnt;p>=0;p--) {
42         mul(res,res,k);
43         if ((n>p)&1) {
44             for (int i=k-1;i>=0;i--) res[i+1]=res[i];res[0]=0;
45             rep(j,0,SZ(Md)) res[Md[j]]=(res[Md[j]]-res[k]*_md[Md[j]])%mod;
46         }
47     }
48     rep(i,0,k) ans=(ans+res[i]*b[i])%mod;
49     if (ans<0) ans+=mod;
50     return ans;
51 }
52 VI BM(VI s) {
53     VI C(1,1),B(1,1);
54     int L=0,m=1,b=1;
55     rep(n,0,SZ(s)) {
56         ll d=0;
57         rep(i,0,L+1) d=(d+(1ll)C[i]*s[n-i])%mod;
58         if (d==0) ++m;
59         else if (2*L<=n) {
60             VI T=C;
61             ll c=mod-d*powmod(b,mod-2)%mod;
62             while (SZ(C)<SZ(B)+m) C.pb(0);
63             rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
64             L=n+1-L; B=T; b=d; m=1;
65         } else {
66             ll c=mod-d*powmod(b,mod-2)%mod;
67             while (SZ(C)<SZ(B)+m) C.pb(0);
68             rep(i,0,SZ(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
69             ++m;
70         }
71     }

```

```

72     return C;
73 }
74 int gao(VI a,ll n) { //c.size()= 阶数
75     VI c=BM(a);
76     c.erase(c.begin());
77     rep(i,0,SZ(c)) c[i]=(mod-c[i])%mod;
78     return solve(n,c,VI(a.begin(),a.begin()+SZ(c)));
79 }
80 };
81 int main()
82 {
83     while (~scanf("%d",&n))
84     {
85         vector<int>v;
86         v.push_back(1);
87         v.push_back(1);
88         v.push_back(2);
89         v.push_back(3);
90         v.push_back(5);
91         v.push_back(8);
92         //VI{1,1,2,3,5,8} 解出斐波那契数列
93         printf("i:%d arr:%d\n",n,linear_seq::gao(v,n-1));
94     }
95 }

```

Fast Transfroming

FFT

- 复数类实现，n 为 2 的幂次

```

1  typedef double LD;
2  const LD PI = 3.14159265358979;
3  struct C
4  {
5      LD r, i;
6      C(LD r = 0, LD i = 0) : r(r), i(i) {}
7      C operator+(const C &a) const
8      {
9          return C(r + a.r, i + a.i);
10     }
11     C operator-(const C &a) const
12     {
13         return C(r - a.r, i - a.i);
14     }
15     C operator*(const C &a) const
16     {
17         return C(r * a.r - i * a.i, r * a.i + i * a.r);
18     }
19 };
20 void FFT(C x[], int n, int p)
21 {
22     for (int i = 0, t = 0; i < n; ++i)

```

```

23     {
24         if (i > t)
25             swap(x[i], x[t]);
26         for (int j = n >> 1; (t ^= j) < j; j >>= 1)
27             ;
28     }
29     for (int h = 2; h <= n; h <<= 1)
30     {
31         C wn(cos(p * 2 * PI / h), sin(p * 2 * PI / h));
32         for (int i = 0; i < n; i += h)
33         {
34             C w(1, 0), u;
35             for (int j = i, k = h >> 1; j < i + k; ++j)
36             {
37                 u = x[j + k] * w;
38                 x[j + k] = x[j] - u;
39                 x[j] = x[j] + u;
40                 w = w * wn;
41             }
42         }
43     }
44     if (p == -1)
45         for (int i = 0; i < n; ++i)
46             x[i].r /= n;
47 }
48 void conv(C a[], C b[], int n)
49 {
50     FFT(a, n, 1);
51     FFT(b, n, 1);
52     for (int i = 0; i < n; ++i)
53         a[i] = a[i] * b[i];
54     FFT(a, n, -1);
55 }

```

NTT

```

1  const int maxn=1e6+7;
2  ll wn[maxn << 2], rev[maxn << 2];
3  int G=3;//998244353
4  int NTT_init(int n_) {
5      int step = 0; int n = 1;
6      for ( ; n < n_; n <<= 1) ++step;
7      for(int i=1;i<n;i++)
8          rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (step - 1));
9      int g = quick(G, (mod - 1) / n);
10     wn[0] = 1;
11     for (int i = 1; i <= n; ++i)
12         wn[i] = wn[i - 1] * g % mod;
13     return n;
14 }
15
16 void NTT(ll a[], int n, int f) {

```

```

17     for(int i=0;i<n;i++) if (i < rev[i])
18         std::swap(a[i], a[rev[i]]);
19     for (int k = 1; k < n; k <= 1) {
20         for (int i = 0; i < n; i += (k < 1)) {
21             int t = n / (k < 1);
22             for(int j=0;j<k;j++){
23                 ll w = f == 1 ? wn[t * j] : wn[n - t * j];
24                 ll x = a[i + j];
25                 ll y = a[i + j + k] * w % mod;
26                 a[i + j] = (x + y) % mod;
27                 a[i + j + k] = (x - y + mod) % mod;
28             }
29         }
30     }
31     if (f == -1) {
32         ll ninv = inv(n);
33         for(int i=0;i<n;i++)
34             a[i] = a[i] * ninv % mod;
35     }
36 }

```

FWT

- $C_k = \sum_{i \oplus j = k} A_i B_j$

```

1  template<typename T>
2  void fwt(ll a[], int n, T f) {
3      for (int d = 1; d < n; d *= 2)
4          for (int i = 0, t = d * 2; i < n; i += t)
5              for(int j = 0; j < d; j++)
6                  f(a[i + j], a[i + j + d]);
7  }
8
9  void AND(ll& a, ll& b) { a += b; }
10 void OR(ll& a, ll& b) { b += a; }
11 void XOR (ll& a, ll& b) {
12     ll x = a, y = b;
13     a = (x + y) % mod;
14     b = (x - y + mod) % mod;
15 }
16 void rAND(ll& a, ll& b) { a -= b; }
17 void rOR(ll& a, ll& b) { b -= a; }
18 void rXOR(ll& a, ll& b) {
19     static ll inv2 = (mod + 1) / 2;
20     ll x = a, y = b;
21     a = (x + y) * inv2 % mod;
22     b = (x - y + mod) * inv2 % mod;
23 }

```

万能 FFT

```

1  namespace fft
2  {

```

```

3 struct num
4 {
5     double x,y;
6     num() {x=y=0;}
7     num(double x,double y):x(x),y(y){}
8 };
9 inline num operator+(num a,num b) {return num(a.x+b.x,a.y+b.y);}
10 inline num operator-(num a,num b) {return num(a.x-b.x,a.y-b.y);}
11 inline num operator*(num a,num b) {return num(a.x*b.x-a.y*b.y,a.x*b.y+a.y*b.x);}
12 inline num conj(num a) {return num(a.x,-a.y);}
13
14 int base=1;
15 vector<num> roots={{0,0},{1,0}};
16 vector<int> rev={0,1};
17 const double PI=acosl(-1.0);
18
19 void ensure_base(int nbase)
20 {
21     if(nbase<=base) return;
22     rev.resize(1<<nbase);
23     for(int i=0;i<(1<<nbase);i++)
24         rev[i]=(rev[i]>>1)>>1+((i&1)<<(nbase-1));
25     roots.resize(1<<nbase);
26     while(base<nbase)
27     {
28         double angle=2*PI/(1<<(base+1));
29         for(int i=1<<(base-1);i<(1<<base);i++)
30         {
31             roots[i<<1]=roots[i];
32             double angle_i=angle*(2*i+1-(1<<base));
33             roots[(i<<1)+1]=num(cos(angle_i),sin(angle_i));
34         }
35         base++;
36     }
37 }
38
39 void fft(vector<num> &a,int n=-1)
40 {
41     if(n==-1) n=a.size();
42     assert((n&(n-1))==0);
43     int zeros=__builtin_ctz(n);
44     ensure_base(zeros);
45     int shift=base-zeros;
46     for(int i=0;i<n;i++)
47         if(i<(rev[i]>>shift))
48             swap(a[i],a[rev[i]>>shift]);
49     for(int k=1;k<n;k<=<=1)
50     {
51         for(int i=0;i<n;i+=2*k)
52         {
53             for(int j=0;j<k;j++)
54 
```

```

55         num z=a[i+j+k]*roots[j+k];
56         a[i+j+k]=a[i+j]-z;
57         a[i+j]=a[i+j]+z;
58     }
59 }
60 }
61 }
62
63 vector<num> fa,fb;
64
65 vector<int> multiply(vector<int> &a, vector<int> &b)
66 {
67     int need=a.size()+b.size()-1;
68     int nbase=0;
69     while((1<<nbase)<need) nbase++;
70     ensure_base(nbase);
71     int sz=1<<nbase;
72     if(sz>(int)fa.size()) fa.resize(sz);
73     for(int i=0;i<sz;i++)
74     {
75         int x=(i<(int)a.size()?a[i]:0);
76         int y=(i<(int)b.size()?b[i]:0);
77         fa[i]=num(x,y);
78     }
79     fft(fa,sz);
80     num r(0,-0.25/sz);
81     for(int i=0;i<=(sz>>1);i++)
82     {
83         int j=(sz-i)&(sz-1);
84         num z=(fa[j]*fa[j]-conj(fa[i]*fa[i]))*r;
85         if(i!=j) fa[j]=(fa[i]*fa[i]-conj(fa[j]*fa[j]))*r;
86         fa[i]=z;
87     }
88     fft(fa,sz);
89     vector<int> res(need);
90     for(int i=0;i<need;i++) res[i]=fa[i].x+0.5;
91     return res;
92 }
93
94 vector<int> multiply_mod(vector<int> &a,vector<int> &b,int m,int eq=0)
95 {
96     int need=a.size()+b.size()-1;
97     int nbase=0;
98     while((1<<nbase)<need) nbase++;
99     ensure_base(nbase);
100    int sz=1<<nbase;
101    if(sz>(int)fa.size()) fa.resize(sz);
102    for(int i=0;i<(int)a.size();i++)
103    {
104        int x=(a[i]%m+m)%m;
105        fa[i]=num(x&((1<<15)-1),x>>15);
106    }

```

```

107     fill(fa.begin()+a.size(),fa.begin()+sz,num{0,0});
108     fft(fa,sz);
109     if(sz>(int)fb.size()) fb.resize(sz);
110     if(eq) copy(fa.begin(),fa.begin()+sz,fb.begin());
111     else
112     {
113         for(int i=0;i<(int)b.size();i++)
114         {
115             int x=(b[i]%m+m)%m;
116             fb[i]=num(x&((1<<15)-1),x>>15);
117         }
118         fill(fb.begin()+b.size(),fb.begin()+sz,num{0,0});
119         fft(fb,sz);
120     }
121     double ratio=0.25/sz;
122     num r2(0,-1),r3(ratio,0),r4(0,-ratio),r5(0,1);
123     for(int i=0;i<=(sz>>1);i++)
124     {
125         int j=(sz-i)&(sz-1);
126         num a1=(fa[i]+conj(fa[j]));
127         num a2=(fa[i]-conj(fa[j]))*r2;
128         num b1=(fb[i]+conj(fb[j]))*r3;
129         num b2=(fb[i]-conj(fb[j]))*r4;
130         if(i!=j)
131         {
132             num c1=(fa[j]+conj(fa[i]));
133             num c2=(fa[j]-conj(fa[i]))*r2;
134             num d1=(fb[j]+conj(fb[i]))*r3;
135             num d2=(fb[j]-conj(fb[i]))*r4;
136             fa[i]=c1*d1+c2*d2*r5;
137             fb[i]=c1*d2+c2*d1;
138         }
139         fa[j]=a1*b1+a2*b2*r5;
140         fb[j]=a1*b2+a2*b1;
141     }
142     fft(fa,sz);fft(fb,sz);
143     vector<int> res(need);
144     for(int i=0;i<need;i++)
145     {
146         ll aa=fa[i].x+0.5;
147         ll bb=fb[i].x+0.5;
148         ll cc=fa[i].y+0.5;
149         res[i]=(aa+((bb%m)<<15)+((cc%m)<<30))%m;
150     }
151     return res;
152 }
153 vector<int> square_mod(vector<int> &a,int m)
154 {
155     return multiply_mod(a,a,m,1);
156 }
157 };

```

离散对数

BSGS

- 北上广深, 拔山盖世应用于模数为质数

```
1 ll BSGS(ll a, ll b, ll p) //  $a^x = b \pmod p$ 
2 {
3     a %= p;
4     if (!a && !b)
5         return 1;
6     if (!a)
7         return -1;
8     static map<ll, ll> mp;
9     mp.clear();
10    ll m = sqrt(p + 1.5);
11    ll v = 1;
12    for (int i = 1; i < m + 1; ++i)
13    {
14        v = v * a % p;
15        mp[v * b % p] = i;
16    }
17    ll vv = v;
18    for (int i = 1; i < m + 1; ++i)
19    {
20        auto it = mp.find(vv);
21        if (it != mp.end())
22            return i * m - it->second;
23        vv = vv * v % p;
24    }
25    return -1;
26 }
```

exBSGS

```
1 ll exBSGS(ll a, ll b, ll p) //  $a^x = b \pmod p$ 
2 {
3     a %= p; b %= p;
4     if (a == 0) return b > 1 ? -1 : b == 0 && p != 1;
5     LL c = 0, q = 1;
6     while (1) {
7         ll g = __gcd(a, p);
8         if (g == 1) break;
9         if (b == 1) return c;
10        if (b % g) return -1;
11        ++c; b /= g; p /= g; q = a / g * q % p;
12    }
13    static map<ll, ll> mp; mp.clear();
14    ll m = sqrt(p + 1.5);
15    ll v = 1;
16    for(int i = 1; i < m + 1 ; i++)
17    {
18        v = v * a % p;
19        mp[v * b % p] = i;
```



```

20     }
21     for(int i = 1; i < m + 1; i++)
22     {
23         q = q * v % p;
24         auto it = mp.find(q);
25         if (it != mp.end()) return i * m - it->second + c;
26     }
27     return -1;
28 }

```

二次剩余

- $x \equiv -1$ 时无根
- 否则有两个解: $x, p - x$

```

1  ll a, p, w;
2  struct T
3  {
4      ll x, y;
5  };
6
7  T mul_two(T a, T b, ll p)
8  {
9      T ans;
10     ans.x = (a.x * b.x % p + a.y * b.y % p * w % p) % p;
11     ans.y = (a.x * b.y % p + a.y * b.x % p) % p;
12     return ans;
13 }
14
15 T qpow_two(T a, ll n, ll p)
16 {
17     T ans;
18     ans.x = 1;
19     ans.y = 0;
20     while (n)
21     {
22         if (n & 1)
23             ans = mul_two(ans, a, p);
24         n >>= 1;
25         a = mul_two(a, a, p);
26     }
27     return ans;
28 }
29
30 ll qpow(ll a, ll n, ll p)
31 {
32     ll ans = 1;
33     a %= p;
34     while (n)
35     {
36         if (n & 1)
37             ans = ans * a % p;
38         n >>= 1;

```

```

39     a = a * a % p;
40 }
41 return ans % p;
42 }
43
44 ll Legendre(ll a, ll p)
45 {
46     return qpow(a, (p - 1) >> 1, p);
47 }
48
49 int solve(ll n, ll p)
50 {
51     if (n == 0)
52         return 0;
53     if (p == 2)
54         return 1;
55     if (Legendre(n, p) + 1 == p)
56         return -1;
57     ll a, t;
58     while (1)
59     {
60         a = rand() % p;
61         t = a * a - n;
62         w = (t % p + p) % p;
63         if (Legendre(w, p) + 1 == p)
64             break;
65     }
66     T tmp;
67     tmp.x = a;
68     tmp.y = 1;
69     T ans = qpow_two(tmp, (p + 1) >> 1, p);
70     return ans.x;
71 }

```

中国剩余定理

- 逐项合并，支持不互质，无解返回-1
- 前置 exgcd

```

1 ll CRT(ll *m, ll *r, ll n)
2 {
3     if (!n)
4         return 0;
5     ll M = m[0], R = r[0], x, y, d;
6     for (int i = 1; i < n; i++)
7     {
8         d = exgcd(M, m[i], x, y);
9         if ((r[i] - R) % d)
10             return -1;
11         x = (r[i] - R) / d * x % (m[i] / d);
12         R += x * M;
13         M = M / d * m[i];
14         R %= M;

```

```

15     }
16     return R >= 0 ? R : R + M;
17 }

```

线性基

```

1  template<typename T,int D>
2  struct Base{
3      T a[D];
4      int m;
5      Base(){m = 0, memset(a, 0, sizeof(a));}
6      void clear(){m = 0, memset(a, 0, sizeof(a));}
7      bool ins(T x)
8      {
9          for(int i = D - 1; ~i; --i)
10             if(x >> i & 1)
11             {
12                 if(a[i]) x ^= a[i];
13                 else{
14                     m++;
15                     a[i] = x;return 1;
16                 }
17             }
18             return 0;
19     }
20 };
21 //求交
22 template<typename T,int D>
23 Base<T,D> Merge(Base<T,D> A,Base<T,D> B)
24 {
25     if(A.m==D) return B;
26     if(B.m==D) return A;
27     Base<T,D> All,C,D;
28     All=A;
29     D.ful();
30     for(int i=D-1;i>=0;i--)
31     {
32         if(B.a[i]){
33             T v=B.a[i],k=0;
34             bool can=1;
35             for(int j=D-1;j>=0;j--)
36             {
37                 if(v>>j&1)
38                 {
39                     if(All.a[j])
40                     {
41                         v^=All.a[j];
42                         k^=D.a[j];
43                     }
44                     else{
45                         can=0;
46                         All.a[j]=v;

```

```

47         D.a[j]=k;
48         break;
49     }
50 }
51 }
52 if(can)
53 {
54     T v=0;
55     for(int j=D-1;j>=0;j--)
56     {
57         if(k>>j&1)
58             v^=A.a[j];
59     }
60     C.ins(v);
61 }
62 }
63 }
64 return C;
65 }

```

高斯消元

```

1  const double eps=1e-8;
2  typedef vector<double> vec;
3  typedef vector<vec> mat;
4  int sz;
5  vec gauss_jordan(const mat& A, const vec& b)
6  {
7      int n=A.size();
8      mat B(n,vec(n+1));
9      for(int i=0;i<n;i++)
10         for(int j=0;j<n;j++)
11             B[i][j]=A[i][j];
12
13     for(int i=0;i<n;i++) B[i][n]=b[i];
14     for(int i=0;i<n;i++)
15     {
16         int pivot=i;
17         for(int j=i;j<n;j++)
18             if(abs(B[j][i])>abs(B[pivot][i])) pivot=j;
19         swap(B[i],B[pivot]);
20         if(abs(B[i][i])<eps) return vec();
21         for(int j=i+1;j<=n;j++) B[i][j]/=B[i][i];
22         for(int j=0;j<n;j++)
23         {
24             if(i!=j)
25             {
26                 for(int k=i+1;k<=n;k++)
27                     B[j][k]-=B[j][i]*B[i][k];
28             }
29         }
30     }

```

```

31     vec x(n);
32     for(int i=0;i<n;i++)
33         x[i]=B[i][n];
34     return x;
35 }

```

素数测试

miller_rabin

```

1  ll power(ll v, ll p, ll m)
2  {
3      ll r = 1;
4      while (p)
5      {
6          if (p & 1)
7              r = r * v % m;
8          v = v * v % m;
9          p >>= 1;
10     }
11
12     return r;
13 }
14
15 bool witness(ll a, ll p)
16 {
17     int k = 0;
18     ll q = p - 1;
19     while ((q & 1) == 0)
20         ++k, q >>= 1;
21     ll v = power(a, q, p);
22     if (v == 1 || v == p - 1)
23         return false; // probably prime number
24     while (k-- != 0)
25     {
26         v = v * v % p;
27         if (v == p - 1)
28             return false;
29     }
30
31     return true; // composite number
32 }
33
34 bool miller_rabin(ll p)
35 {
36     if (p == 1)
37         return false;
38     if (p == 2)
39         return true;
40     if (p % 2 == 0)
41         return false;
42
43     for (int i = 0; i != 50; ++i)

```

```

44     {
45         ll a = std::rand() % (p - 1) + 1;
46         if (witness(a, p))
47             return false;
48     }
49
50     return true;
51 }

```

Pollard Rho

```

1  ll pollard_rho(ll n, int a)
2  {
3      ll x = 2, y = 2, d = 1, k = 0, i = 1;
4      while(d == 1)
5      {
6          ++k;
7          x = mul_mod(x, x, n) + a;
8          d = __gcd(x >= y ? x - y : y - x, n);
9          if(k == i)
10         {
11             y = x;
12             i <= 1;
13         }
14     }
15     if(d == n) return pollard_rho(n, a + 1);
16     return d;
17 }

```

博弈

- 巴什博弈：P 态为 $n \equiv 0(\text{mod } m + 1)$
- 阶梯博弈：阶梯博弈等效为奇数号阶梯的 nim 博弈
- 威佐夫博弈：有两堆各若干个物品，两个人轮流从某一堆取至少一个或同时从两堆中取同样多的物品，规定每次至少取一个，多者不限，最后取光者得胜。P 态为 $(y - x) \times \frac{\sqrt{5}+1}{2} = x$
- NP 图、SG 函数找规律，博弈 dp
- 考虑模仿操作

字符串

KMP

```
1 char s[maxn], t[maxn];
2 int fail[maxn];
3 void getfail()
4 {
5     memset(fail, 0, sizeof(fail));
6     int len = strlen(t);
7     int j = 0, k = fail[0] = -1;
8     while (j < len)
9     {
10         while (k != -1 && t[j] != t[k])
11             k = fail[k];
12         fail[++j] = ++k;
13     }
14 }
15
16 int kmp()
17 {
18     int n = strlen(s), m = strlen(t);
19     int i = 0, j = 0;
20     int ret = 0;
21     while (i < n)
22     {
23         while (j != -1 && s[i] != t[j])
24             j = fail[j];
25         i++, j++;
26         if (j == m)
27             ret++, j = fail[j];
28     }
29     return ret;
30 }
```

Manacher

```
1 const int maxn = 2e5;
2
3 string Mnc(string &s)
4 {
5     string t = "$#";
6     for (int i = 0; i < s.length(); ++i) //构造辅助串
7     {
8         t += s[i];
9         t += '#';
10    }
11
12    int ml = 0, p = 0, R = 0, M = 0;
13    //最大长度, 最长回文中心, 当前最大回文串右端, 当前最长回文中心
14
15    int len = t.length();
16    vector<int> P(len, 0); //回长度数组
```

```

17     for (int i = 0; i < len; ++i)
18     {
19         P[i] = R > i ? min(P[2 * M - i], R - i) : 1; //转移方程
20
21         while (t[i + P[i]] == t[i - P[i]]) //长度扩张
22             ++P[i];
23
24         if (i + P[i] > R) //更新右端和中心
25         {
26             R = i + P[i];
27             M = i;
28         }
29         if (ml < P[i]) //记录极大
30         {
31             ml = P[i];
32             p = i;
33         }
34     }
35
36     return s.substr((p - ml) / 2, ml - 1); //返回回文串
37 }

```

后缀数组

```

1  char s[maxn];
2  int sa[maxn], t[maxn], t2[maxn], c[maxn], rk[maxn], height[maxn];
3  //sa[],height[] 下标从 1 开始, rk[] 下标从 0 开始
4  void getsa(int m, int n)
5  { //n 为字符串的长度, 字符集的值 0~m-1
6      n++;
7      int *x = t, *y = t2;
8      //基数排序
9      for (int i = 0; i < m; i++)
10         c[i] = 0;
11      for (int i = 0; i < n; i++)
12         c[x[i] = s[i]]++;
13      for (int i = 1; i < m; i++)
14         c[i] += c[i - 1];
15      for (int i = n - 1; ~i; i--)
16         sa[--c[x[i]]] = i;
17      for (int k = 1; k <= n; k <= 1)
18      { //直接利用 sa 数组排序第二关键字
19         int p = 0;
20         for (int i = n - k; i < n; i++)
21             y[p++] = i;
22         for (int i = 0; i < n; i++)
23             if (sa[i] >= k)
24                 y[p++] = sa[i] - k;
25         //基数排序第一关键字
26         for (int i = 0; i < m; i++)
27             c[i] = 0;
28         for (int i = 0; i < n; i++)

```



```

29         c[x[y[i]]]++;
30     for (int i = 1; i < m; i++)
31         c[i] += c[i - 1];
32     for (int i = n - 1; ~i; i--)
33         sa[--c[x[y[i]]]] = y[i];
34     //根据 sa 和 y 数组计算新的 x 数组
35     swap(x, y);
36     p = 1;
37     x[sa[0]] = 0;
38     for (int i = 1; i < n; i++)
39         x[sa[i]] = y[sa[i - 1]] == y[sa[i]] && y[sa[i - 1] + k] == y[sa[i] + k] ?
40             ↪ p - 1 : p++;
41     if (p >= n)
42         break; //以后即使继续倍增, sa 也不会改变, 推出
43         //下次基数排序的最大值
44     m = p;
45     n--;
46     int k = 0;
47     for (int i = 0; i <= n; i++)
48         rk[sa[i]] = i;
49     for (int i = 0; i < n; i++)
50     {
51         if (k) k--;
52         int j = sa[rk[i] - 1];
53         while (s[i + k] == s[j + k])
54             k++;
55         height[rk[i]] = k;
56     }
57 }
58 int dp[maxn][30];
59 void initrmq(int n)
60 {
61     for (int i = 1; i <= n; i++)
62         dp[i][0] = height[i];
63     for (int j = 1; (1 << j) <= n; j++)
64         for (int i = 1; i + (1 << j) - 1 <= n; i++)
65             dp[i][j] = min(dp[i][j - 1], dp[i + (1 << (j - 1))][j - 1]);
66 }
67 int rmq(int l, int r)
68 {
69     int k = 31 - __builtin_clz(r - l + 1);
70     return min(dp[l][k], dp[r - (1 << k) + 1][k]);
71 }
72 int lcp(int a, int b)
73 { // 求两个后缀的最长公共前缀
74     a = rk[a], b = rk[b];
75     if (a > b)
76         swap(a, b);
77     return rmq(a + 1, b);
78 }

```

后缀自动机

```
1 char s[maxn];
2 int ch[maxn][26], step[maxn], pre[maxn];
3 int to[maxn], topo[maxn], cntr[maxn], sum[maxn];
4 int sz, last; // init(){sz=last=1;}
5 void ins(int x)
6 {
7     int np = ++sz, p = last;
8     last = np;
9     step[np] = step[p] + 1;
10    cntr[np] = 1;
11    while (!ch[p][x] && p)
12        ch[p][x] = np, p = pre[p];
13    if (!p)
14        pre[np] = 1;
15    else
16    {
17        int q = ch[p][x];
18        if (step[q] == step[p] + 1)
19            pre[np] = q;
20        else
21        {
22            int nq = ++sz;
23            step[nq] = step[p] + 1;
24            for (int i = 0; i < 26; ++i)
25                ch[nq][i] = ch[q][i];
26            pre[nq] = pre[q];
27            pre[q] = pre[np] = nq;
28            while (ch[p][x] == q && p)
29                ch[p][x] = nq, p = pre[p];
30        }
31    }
32 }
33
34 void getr()
35 {
36     for (int i = 1; i <= sz; ++i)
37         ++to[step[i]]; //利用后缀自动机性质拓扑排序
38     for (int i = 1; i <= sz; ++i)
39         to[i] += to[i - 1];
40     for (int i = 1; i <= sz; ++i)
41         topo[to[step[i]]--] = i;
42     for (int i = sz; i >= 1; --i)
43         if (ty)
44             cntr[pre[topo[i]]] += cntr[topo[i]];
45         else
46             cntr[i] = 1;
47     cntr[1] = 0;
48     for (int i = sz; i >= 1; --i)
49     {
50         int x = topo[i];
```

```

51         sum[x] = cntr[x]; //sum: 停下或继续, 你还能走出多少个子串
52         for (int j = 0; j < 26; ++j)
53             if (ch[x][j])
54                 sum[x] += sum[ch[x][j]];
55     }
56 }

```

广义后缀自动机

```

1  //每个串 last 置 1
2  void ins(int x)
3  {
4      x--='a';
5      int p=last,np=0,nq=0,q=-1;
6      if(!ch[p][x])
7      {
8          np = ++sz;
9          step[np] = step[p] + 1;
10         while (!ch[p][x] && p)
11             ch[p][x] = np, p = pre[p];
12     }
13     if (!p)
14         pre[np] = 1;
15     else
16     {
17         q = ch[p][x];
18         if (step[q] == step[p] + 1)
19             pre[np] = q;
20         else
21         {
22             int nq = ++sz;
23             step[nq] = step[p] + 1;
24             for (int i = 0; i < 26; ++i)
25                 ch[nq][i] = ch[q][i];
26             pre[nq] = pre[q];
27             pre[q] = pre[np] = nq;
28             while (ch[p][x] == q && p)
29                 ch[p][x] = nq, p = pre[p];
30         }
31     }
32     last=np ? np : nq ? nq : q;
33 }

```

AC 自动机

```

1  struct AC_auto
2  {
3      int ch[maxn][26];
4      int num[maxn], fail[maxn];
5      // f 即为 fail 指针.
6      int tot;
7      void init()

```

```

8      {
9          tot = 0;
10         for (int i = 0; i < 26; ++i)
11             ch[0][i] = 0;
12     }
13     void insert(char s[], int len)
14     {
15         int u = 0;
16         for (int i = 0; i < len; i++)
17         {
18             if (!ch[u][s[i] - 'a'])
19             {
20                 ch[u][s[i] - 'a'] = ++tot;
21                 for (int j = 0; j < 26; ++j)
22                     ch[tot][j] = 0;
23                 num[tot] = fail[tot] = 0;
24             }
25             u = ch[u][s[i] - 'a'];
26         }
27         num[u]++;
28     } //往 Trie 树里插入元素.
29     void build()
30     {
31         queue<int> q;
32         for (int i = 0; i < 26; i++)
33         {
34             if (ch[0][i])
35                 fail[ch[0][i]] = 0,
36                 //第一层与其他单词不可能有公共前后缀,fail 直接为根.
37                 q.push(ch[0][i]);
38         }
39         while (!q.empty())
40         {
41             int u = q.front();
42             q.pop();
43             for (int i = 0; i < 26; i++)
44                 if (ch[u][i])
45                 {
46                     fail[ch[u][i]] = ch[fail[u]][i];
47                     q.push(ch[u][i]);
48                 }
49                 else
50                     ch[u][i] = ch[fail[u]][i];
51             //这一步直接省略了查询时的比较.
52         }
53     } //构建 Fail 指针.
54     int query(char s[], int len)
55     {
56         int u = 0, ans = 0;
57         for (int i = 0; i < len; i++)
58         {
59             u = ch[u][s[i] - 'a'];

```

```

60         for (int j = u; j && num[j] != -1; j = fail[j])
61             ans += num[j], num[j] = -1;
62         //因为直接已经在每个单词的最后面打了标记，所以直接加上即可。
63     }
64     return ans;
65 }
66 } AC;

```

最小表示法

```

1  int getMin(char *s)
2  {
3      int i = 0, j = 1, l;
4      int len = strlen(s);
5      while (i < len && j < len)
6      {
7          for (l = 0; l < len; l++)
8              if (s[(i + l) % len] != s[(j + l) % len])
9                  break;
10         if (l >= len)
11             break;
12         if (s[(i + l) % len] > s[(j + l) % len])
13         {
14             if (i + l + 1 > j)
15                 i = i + l + 1;
16             else
17                 i = j + 1;
18         }
19         else if (j + l + 1 > i)
20             j = j + l + 1;
21         else
22             j = i + 1;
23     }
24     return i < j ? i : j;
25 }
26
27 int getMax(char *s)
28 {
29     int len = strlen(s);
30     int i = 0, j = 1, k = 0;
31     while (i < len && j < len && k < len)
32     {
33         int t = s[(i + k) % len] - s[(j + k) % len];
34         if (!t)
35             k++;
36         else
37         {
38             if (t > 0)
39             {
40                 if (j + k + 1 > i)
41                     j = j + k + 1;
42                 else

```

```

43         j = i + 1;
44     }
45     else if (i + k + 1 > j)
46         i = i + k + 1;
47     else
48         i = j + 1;
49     k = 0;
50 }
51 }
52 return i < j ? i : j;
53 }

```

回文自动机

```

1  const int maxn = 5e5 + 7;
2
3  struct PAM
4  {
5      int next[maxn][26]; //next 指针, 和字典树类似, 指向的串为当前串两端加上同一个字符构成。
6      int fail[maxn], cnt[maxn], num[maxn], len[maxn], s[maxn];
7      int last, n, p;
8      int newnode(int rt)
9      {
10         for (int i = 0; i < 26; i++)
11             next[p][i] = 0;
12         cnt[p] = 0;
13         num[p] = 0;
14         len[p] = rt;
15         return p++;
16     }
17
18     void init()
19     {
20         p = last = n = 0;
21         newnode(0);
22         newnode(-1);
23         s[n] = -1;
24         fail[0] = 1;
25     }
26
27     int getFail(int x) //fail 指针的构建
28     {
29         while (s[n - len[x] - 1] != s[n])
30             x = fail[x];
31         return x;
32     }
33
34     int ins(int c) //插入字符
35     {
36         c -= 'a';
37         s[++n] = c;
38         int cur = getFail(last);

```

```

39     if (!next[cur][c]) //如果不存此字符节点
40     {
41         int now = newnode(len[cur] + 2); //+2: 回文所以两段同时加 1
42         fail[now] = next[getFail(fail[cur])][c]; //构建此处的 fail
43         next[cur][c] = now; //构建此处的 next
44         num[now] = num[fail[now]] + 1; //以此末尾字母结尾的回文串个数
45     }
46     last = next[cur][c]; //last 指针
47     cnt[last]++;
48     return last;
49 }
50
51 void count()
52 {
53     for (int i = p - 1; i >= 0; i--)
54         cnt[fail[i]] += cnt[i]; //父节点累加子节点的 cnt (若 fail[v]=u, 则 u 一定是 v
55         ↳ 的子回文串)
56 }
57 } pam;

```

EXKMP

```

1  char s[maxn];
2  int nxt[maxn], ex[maxn]; //ex 数组即为 extend 数组
3  //预处理计算 nxt 数组
4  void GETNEXT(char *str)
5  {
6      int i=0, j, po, len=strlen(str);
7      nxt[0]=len; //初始化 nxt[0]
8      while(str[i]==str[i+1]&&i+1<len) //计算 nxt[i]
9          i++;
10     nxt[1]=i;
11     po=1; //初始化 po 的位置
12     for(i=2; i<len; i++)
13     {
14         if(nxt[i-po]+i<nxt[po]+po) //第一种情况, 可以直接得到 nxt[i] 的值
15             nxt[i]=nxt[i-po];
16         else //第二种情况, 要继续匹配才能得到 nxt[i] 的值
17         {
18             j=nxt[po]+po-i;
19             if(j<0) j=0; //如果 i>po+nxt[po], 则要从头开始匹配
20             while(i+j<len&&str[j]==str[j+i]) //计算 nxt[i]
21                 j++;
22             nxt[i]=j;
23             po=i; //更新 po 的位置
24         }
25     }
26 }
27
28 //计算 extend 数组
29 void EXKMP(char *s1, char *s2)

```

```

30 {
31     int i=0,j,po,len=strlen(s1),l2=strlen(s2);
32     GETNEXT(s2); //计算子串的 nxt 数组
33     while(s1[i]==s2[i]&&i<l2&&i<len) //计算 ex[0]
34         i++;
35     ex[0]=i;
36     po=0; //初始化 po 的位置
37     for(i=1;i<len;i++)
38     {
39         if(nxt[i-po]+i<ex[po]+po) //第一种情况, 直接可以得到 ex[i] 的值
40             ex[i]=nxt[i-po];
41         else //第二种情况, 要继续匹配才能得到 ex[i] 的值
42         {
43             j=ex[po]+po-i;
44             if(j<0)j=0; //如果 i>ex[po]+po 则要从头开始匹配
45             while(i+j<len&&j<l2&&s1[j+i]==s2[j]) //计算 ex[i]
46                 j++;
47             ex[i]=j;
48             po=i; //更新 po 的位置
49         }
50     }
51 }

```


数据结构

无旋 Treap

- 大根堆

维护集合

- 小于 k 分为左子树，大于等于 k 分为右子树

```
1 struct Treap
2 {
3     Treap *l, *r;
4     int val, prior;
5     Treap(int _val) : val(_val), l(NULL), r(NULL), prior(rnd()) {}
6 };
7 typedef Treap *pt;
8 void split(pt o, int k, pt &l, pt &r)
9 {
10     if (!o)
11         l = r = NULL;
12     else if (o->val >= k)
13         split(o->l, k, l, o->l), r = o;
14     else
15         split(o->r, k, o->r, r), l = o;
16 }
17 void merge(pt &o, pt l, pt r)
18 {
19     if (!l || !r)
20         o = l ? l : r;
21     else if (l->prior > r->prior)
22         merge(l->r, l->r, r), o = l;
23     else
24         merge(r->l, l, r->l), o = r;
25 }
26 pt root;
```

ST 表

- 一维

```
1 void ST(int n) //处理出 [1,n] 的 RMQ
2 {
3     for (int i = 1; i <= n; i++)
4         dp[i][0] = arr[i];
5
6     for (int j = 1; (1 << j) <= n; j++)
7     {
8         for (int i = 1; i + (1 << j) - 1 <= n; i++)
9             dp[i][j] = min(dp[i][j - 1], dp[i + (1 << (j - 1))][j - 1]);
10    }
11 }
12
13 int query(int l, int r)
```

```

14 {
15     int k = 31 - __builtin_clz(r - l + 1);
16     return min(dp[l][k], dp[r - (1 << k) + 1][k]);
17 }

```

树状数组

- 区间最值
- $O(\log^2(n))$

```

1  int a[maxn], tree[maxn]; //a[] 存原始数据, tree[] 存树状数组
2  int n;
3  //先改 a[x], 然后 update(x)
4  void update(int x)
5  {
6      int lx, i;
7      while (x < n)
8      {
9          tree[x] = a[x];
10         lx = -x & x;
11         for (i = 1; i < lx; i <= 1)
12             tree[x] = max(tree[x], tree[x - i]);
13         x += -x & x;
14     }
15 }
16 int query(int x, int y) //[x,y] 区间最值
17 {
18     int ret = 0;
19     while (y >= x)
20     {
21         ret = max(a[y], ret);
22         y--;
23         for (; y - (-y & y) >= x; y -= -y & y)
24             ret = max(tree[y], ret);
25     }
26     return ret;
27 }

```

- 区间修改、区间查询（查询前缀和的前缀和）

```

1  int tr[maxn], trr[maxn];
2  void add(int x, int val)
3  {
4      for (int i = x; i < maxn; i += i & -i)
5      {
6          tr[i] += val;
7          trr[i] += x * val;
8      }
9  }
10 void add(int l, int r, int val)
11 {
12     add(l, val);
13     add(r + 1, -val);
14 }

```

```

15 int sum(int x)
16 {
17     int ret = 0;
18     for (int i = x; i > 0; i -= i & -i)
19         ret += (x + 1) * tr[i] - trr[i];
20     return ret;
21 }
22 int sum(int l, int r) { return sum(r) - sum(l - 1); }

```

笛卡尔树

- $O(n)$ 建树，大根堆

```

1 stack<int> st;
2 for (int i = 0; i < n; i++)
3 {
4     int last = -1;
5     while (!st.empty() && arr[i] > arr[st.top()])
6         last = st.top(), st.pop();
7     if (!st.empty())
8         rc[st.top()] = i, fa[i] = st.top();
9     lc[i] = last;
10    if (~last)
11        fa[last] = i;
12    st.push(i);
13 }
14 int root = -1;
15 for (int i = 0; i < n; i++)
16     if (!~fa[i])
17         root = i;

```

主席树优化建图

- 只写了点向区间连边

```

1 struct Node
2 {
3     Node *l, *r;
4     int id;
5     Node(int _id) : id(_id), l(NULL), r(NULL) {}
6 };
7 Node *rt[maxn];
8 int tot; //编号
9 int ins[maxn];
10 #define Lson L, mid, o->l
11 #define Rson mid + 1, R, o->r
12 void build(int L, int R, Node *&o)
13 {
14     o = new Node(tot++);
15     if (L == R)
16     {
17         addedge(o->id, L);
18         return;
19     }
20 }

```

```

19     }
20     int mid = L + R >> 1;
21     build(Lson);
22     build(Rson);
23     addedge(o->id, o->l->id);
24     addedge(o->id, o->r->id);
25 }
26 void update(int p, int l, int r, int L, int R, Node *o)
27 {
28     if (l <= L && r >= R)
29     {
30         addedge(ins[p], o->id);
31         return;
32     }
33     int mid = L + R >> 1;
34     if (l <= mid)
35         update(p, l, r, Lson);
36     if (r > mid)
37         update(p, l, r, Rson);
38 }
39 void add(int pos, int L, int R, Node *&o, Node *pre)
40 {
41     o = new Node(tot++);
42     if (L == R)
43     {
44         addedge(o->id, ins[pos]);
45         return;
46     }
47     int mid = L + R >> 1;
48     if (pos <= mid)
49     {
50         add(pos, Lson, pre->l);
51         o->r = pre->r;
52     }
53     else
54     {
55         add(pos, Rson, pre->r);
56         o->l = pre->l;
57     }
58     addedge(o->id, o->l->id);
59     addedge(o->id, o->r->id);
60 }

```

Link-Cut Tree

- 修改点值先 makeroot

```

1  #define lc ch[x][0]
2  #define rc ch[x][1]
3  namespace LCT
4  {
5      int fa[maxn], ch[maxn][2], val[maxn], pre[maxn], lz[maxn];
6      inline bool nroot(int x)

```

```

7  {
8      return ch[fa[x]][0] == x || ch[fa[x]][1] == x;
9  }
10 inline void pushup(int x) //维护链信息
11 {
12     pre[x] = pre[lc] ^ pre[rc] ^ val[x];
13 }
14 inline void pushr(int x)
15 {
16     swap(lc, rc);
17     lz[x] ^= 1;
18 } //反转
19 inline void pushdown(int x)
20 {
21     if (lz[x])
22     {
23         if (lc)
24             pushr(lc);
25         if (rc)
26             pushr(rc);
27         lz[x] = 0;
28     }
29 }
30 void rotate(int x) //单次旋转
31 {
32     int y = fa[x], z = fa[y], k = ch[y][1] == x, w = ch[x][!k];
33     if (nroot(y))
34         ch[z][ch[z][1] == y] = x;
35     ch[x][!k] = y;
36     ch[y][k] = w;
37     if (w)
38         fa[w] = y;
39     fa[y] = x;
40     fa[x] = z;
41     pushup(y);
42 }
43 void pushall(int x) //递归下放标记
44 {
45     if (nroot(x))
46         pushall(fa[x]);
47     pushdown(x);
48 }
49 void splay(int x)
50 {
51     pushall(x);
52     while (nroot(x))
53     {
54         int y = fa[x];
55         int z = fa[y];
56         if (nroot(y))
57             rotate((ch[y][0] == x) ^ (ch[z][0] == y) ? x : y);
58         rotate(x);

```

```

59     }
60     pushup(x);
61 }
62 void access(int x)
63 {
64     for (int y = 0; x; x = fa[y = x])
65     {
66         splay(x);
67         rc = y;
68         pushup(x);
69     }
70 }
71 void makeroot(int x)
72 {
73     access(x);
74     splay(x);
75     pushr(x);
76 }
77 int findroot(int x)
78 {
79     access(x);
80     splay(x);
81     while (lc)
82         pushdown(x), x = lc;
83     splay(x);
84     return x;
85 }
86 void split(int x, int y)
87 {
88     makeroot(x);
89     access(y);
90     splay(y);
91 }
92 void link(int x, int y)
93 {
94     makeroot(x);
95     if (findroot(y) != x)
96         fa[x] = y;
97 }
98 void cut(int x, int y)
99 {
100     makeroot(x);
101     if (findroot(y) == x && fa[y] == x && !ch[y][0])
102     {
103         fa[y] = ch[x][1] = 0;
104         pushup(x);
105     }
106 }
107 }; // namespace LCT

```

Splay

```
1  struct Splay
2  {
3
4      struct Node
5      {
6          int father, childs[2], key, cnt, sz;
7          void init() {father = childs[0] = childs[1] = key = cnt = sz = 0;}
8          void init(int fa, int lc, int rc, int k, int c, int s)
9          {
10              father = fa;
11              childs[0] = lc;
12              childs[1] = rc;
13              key = k;
14              cnt = c;
15              sz = s;
16          }
17      } tre[maxn];
18
19      int tot, root;
20      void init() {tot = root = 0;}
21
22      inline bool judge(int x) {return tre[ tre[x].father ].childs[1] == x;}
23
24      inline void update(int x)
25      {
26          if(x)
27          {
28              tre[x].sz = tre[x].cnt;
29              if(tre[x].childs[0])
30                  tre[x].sz += tre[ tre[x].childs[0] ].sz;
31              if(tre[x].childs[1])
32                  tre[x].sz += tre[ tre[x].childs[1] ].sz;
33          }
34      }
35
36      inline void rotate(int x)
37      {
38          int y = tre[x].father, z = tre[y].father, k = judge(x);
39          tre[y].childs[k] = tre[x].childs[!k];
40          tre[ tre[x].childs[!k] ].father = y;
41          tre[x].childs[!k] = y;
42          tre[y].father = x;
43          tre[z].childs[ tre[z].childs[1] == y ] = x;
44          tre[x].father = z;
45          update(y);
46      }
47
48      void splay(int x,int goal)
49      {
50          for(int father; (father = tre[x].father) != goal; rotate(x) )
```

```

51         if(tre[father].father != goal)
52             rotate(judge(x) == judge(father) ? father : x);
53     if(goal == 0)
54         root = x;
55 }
56
57 void insert(int x)
58 {
59     if(root == 0)
60     {
61         tre[++tot].init(0, 0, 0, x, 1, 1);
62         root = tot;
63         return ;
64     }
65     int now = root, father = 0;
66     while(1)
67     {
68         if(tre[now].key == x)
69         {
70             tre[now].cnt ++;
71             update(now), update(father);
72             splay(now, 0);
73             break;
74         }
75         father = now;
76         if(x > tre[now].key)
77             now = tre[now].childs[1];
78         else
79             now = tre[now].childs[0];
80
81         if(now == 0)
82         {
83             tre[++tot].init(father, 0, 0, x, 1, 1);
84             if(x > tre[father].key)
85                 tre[father].childs[1] = tot;
86             else
87                 tre[father].childs[0] = tot;
88             update(father);
89             splay(tot, 0);
90             break;
91         }
92     }
93 }
94
95 int pre()
96 {
97     int now = tre[root].childs[0];
98     while(tre[now].childs[1])
99         now = tre[now].childs[1];
100     return now;
101 }
102

```



```

103     int next()
104     {
105         int now = tre[root].childs[1];
106         while(tre[now].childs[0])
107             now = tre[now].childs[0];
108         return now;
109     }
110
111     int rnk(int x)
112     { /// 找 x 的排名
113         int now = root, ans = 0;
114         while(1)
115         {
116             if(x < tre[now].key)
117                 now = tre[now].childs[0];
118             else
119             {
120                 if(tre[now].childs[0])
121                     ans += tre[ tre[now].childs[0] ].sz;
122                 if(x == tre[now].key)
123                 {
124                     splay(now, 0);
125                     return ans + 1;
126                 }
127                 ans += tre[now].cnt;
128                 now = tre[now].childs[1];
129             }
130         }
131     }
132
133     int kth(int x)
134     { /// 找排名为 x 的数字
135         int now = root;
136         while(1)
137         {
138             if(tre[now].childs[0] && x <= tre[ tre[now].childs[0] ].sz )
139                 now = tre[now].childs[0];
140             else
141             {
142                 int lchild = tre[now].childs[0], sum = tre[now].cnt;
143                 if(lchild)
144                     sum += tre[lchild].sz;
145                 if(x <= sum)
146                     return tre[now].key;
147                 x -= sum;
148                 now = tre[now].childs[1];
149             }
150         }
151     }
152
153     void del(int x)
154     {

```

```

155     find(x);
156     if(tre[root].cnt > 1)
157     {
158         tre[root].cnt --;
159         update(root);
160         return ;
161     }
162     if(!tre[root].childs[0] && !tre[root].childs[1])
163     {
164         tre[root].init();
165         root = 0;
166         return ;
167     }
168     if(!tre[root].childs[0])
169     {
170         int old_root = root;
171         root = tre[root].childs[1];
172         tre[root].father = 0;
173         tre[old_root].init();
174         return ;
175     }
176     if(!tre[root].childs[1])
177     {
178         int old_root = root;
179         root = tre[root].childs[0];
180         tre[root].father = 0;
181         tre[old_root].init();
182         return ;
183     }
184     int pre_node = pre(), old_root = root;
185     splay(pre_node, 0);
186     tre[root].childs[1] = tre[old_root].childs[1];
187     tre[ tre[old_root].childs[1] ].father = root;
188     tre[old_root].init();
189     update(root);
190 }
191
192 bool find(int x)
193 {
194     int now = root;
195     while(1)
196     {
197         if(now == 0)
198             return 0;
199         if(x == tre[now].key)
200         {
201             splay(now, 0);
202             return 1;
203         }
204         if(x > tre[now].key)
205             now = tre[now].childs[1];
206         else

```

```

207         now = tre[now].childs[0];
208     }
209 }
210 } S;

```

时间分治线段树

- 内含可回滚并查集和并查集判二分图

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  typedef pair<int,int> PII;
5  const int maxn=1e5+7;
6  struct Edge{
7      int u,v;
8  };
9  vector<Edge> seg[maxn<<2];
10 #define lson o<<1
11 #define rson o<<1|1
12 #define Lson L,mid,lson
13 #define Rson mid+1,R,rson
14 int n,m,T;
15 int fa[maxn],col[maxn],sz[maxn];
16 void init()
17 {
18     for(int i=0;i<=n;i++) fa[i]=i,sz[i]=1,col[i]=0;
19 }
20 int Find(int x)
21 {
22     return fa[x]==x?x:Find(fa[x]);
23 }
24 int getval(int x)
25 {
26     int ret=0;
27     while(fa[x]!=x) ret^=col[x],x=fa[x];
28     return ret;
29 }
30 void update(int l,int r,Edge e,int L=1,int R=T,int o=1)
31 {
32     if(l<=L&&r>=R)
33     {
34         seg[o].push_back(e);
35         return;
36     }
37     int mid=L+R>>1;
38     if(l<=mid)
39         update(l,r,e,Lson);
40     if(r>mid) update(l,r,e,Rson);
41 }
42 void solve(int L=1,int R=T,int o=1,bool ok=0)
43 {
44     int mid=L+R>>1;

```

```

45     if(ok){
46         if(L==R) puts(ok?"No":"Yes");
47         else
48         {
49             solve(Lson,1);
50             solve(Rson,1);
51         }
52     }
53     else{
54         vector<int> cur;
55         //insert
56         for(int i=0;i<seg[o].size();i++)
57         {
58             Edge e=seg[o][i];
59             int u=e.u,v=e.v;
60             int colu=getval(u),colv=getval(v);
61             u=Find(u),v=Find(v);
62             if(u==v&&colu==colv)
63                 ok=1;
64             else{
65                 if(sz[u]>sz[v]) swap(u,v);
66                 sz[v]+=sz[u];
67                 fa[u]=v;
68                 col[u]=colu^colv^1;
69                 cur.push_back(u);
70             }
71         }
72         if(L==R)
73             puts(ok?"No":"Yes");
74         else solve(Lson,ok),solve(Rson,ok);
75         //deleta
76         for(int i=cur.size()-1;i>=0;i--)
77         {
78             int u=cur[i];
79             sz[fa[u]]-=sz[u],fa[u]=u,col[u]=0;
80         }
81     }
82 }
83 int main()
84 {
85     scanf("%d%d%d",&n,&m,&T);
86     init();
87     for(int i=0,u,v,s,e;i<m;i++)
88     {
89         scanf("%d%d%d",&u,&v,&s,&e);
90         s++;
91         if(s<=e)
92             update(s,e,Edge{u,v});
93     }
94     solve();
95 }

```

柯朵莉树

```
1 struct node{
2     int l,r;
3     mutable ll val;
4     bool operator<(const node &a)const{
5         return r<a.r;
6     }
7 };
8 set<node> st;
9 void split(int p)
10 {
11     auto it = st.lower_bound({p, p, 0});
12     if(it -> l == p) return;
13     int l = it -> l, r = it -> r; ll val = it -> val;
14     st.erase(it);
15     st.insert({l, p-1, val});
16     st.insert({p, r, val});
17 }
18 void update(ll l, ll r, int v)
19 {
20     split(l), split(r + 1);
21     auto cur=st.lower_bound({l, l, 0});
22     while(cur->r <= r)
23     {
24         auto tmp = cur;
25         cur++;
26         st.erase(tmp);
27     }
28     st.insert({l, r, v});
29 }
```

K-D Tree

- 最近/远点对

```
1 const int maxn=5e5+7;
2 const int inf=0x3f3f3f3f;
3 int cur,ans,root;
4
5 struct P
6 {
7     int mn[2],mx[2],d[2],lch,rch;
8     int& operator[](int x) {return d[x];}
9     friend bool operator<(P x,P y) {return x[cur]<y[cur];}
10    friend int dis(P x,P y) {return abs(x[0]-y[0])+abs(x[1]-y[1]);}
11 }p[maxn];
12
13 struct kdtree
14 {
15     P t[maxn],T;
16     int ans;
17     void update(int k)
```

```

18 {
19     int l=t[k].lch,r=t[k].rch;
20     for (int i=0;i<2;i++)
21     {
22         t[k].mn[i]=t[k].mx[i]=t[k][i];
23         if (l) t[k].mn[i]=min(t[k].mn[i],t[l].mn[i]);
24         if (r) t[k].mn[i]=min(t[k].mn[i],t[r].mn[i]);
25         if (l) t[k].mx[i]=max(t[k].mx[i],t[l].mx[i]);
26         if (r) t[k].mx[i]=max(t[k].mx[i],t[r].mx[i]);
27     }
28 }
29 int build(int l,int r,int now)
30 {
31     cur=now;
32     int mid=(l+r)/2;
33     nth_element(p+l,p+mid,p+r+1);
34     t[mid]=p[mid];
35     for (int i=0;i<2;i++) t[mid].mx[i]=t[mid].mn[i]=t[mid][i];
36     if (l<mid) t[mid].lch=build(l,mid-1,now^1);
37     if (r>mid) t[mid].rch=build(mid+1,r,now^1);
38     update(mid);
39     return mid;
40 }
41 int getmn(P x)
42 {
43     int ans=0;
44     for (int i=0;i<2;i++)
45     {
46         ans+=max(T[i]-x.mx[i],0);
47         ans+=max(x.mn[i]-T[i],0);
48     }
49     return ans;
50 }
51 int getmx(P x)
52 {
53     int ans=0;
54     for (int i=0;i<2;i++) ans+=max(abs(T[i]-x.mn[i]),abs(T[i]-x.mx[i]));
55     return ans;
56 }
57 void querymx(int k)
58 {
59     ans=max(ans,dis(t[k],T));
60     int l=t[k].lch,r=t[k].rch,dl=-inf,dr=-inf;
61     if (l) dl=getmx(t[l]);
62     if (r) dr=getmx(t[r]);
63     if (dl>dr)
64     {
65         if (dl>ans) querymx(l);
66         if (dr>ans) querymx(r);
67     }
68     else
69     {

```

```

70         if (dr>ans) querymx(r);
71         if (dl>ans) querymx(l);
72     }
73 }
74 void querymn(int k)
75 {
76     if (dis(t[k],T)) ans=min(ans,dis(t[k],T));
77     int l=t[k].lch,r=t[k].rch,dl=inf,dr=inf;
78     if (l) dl=getmn(t[l]);
79     if (r) dr=getmn(t[r]);
80     if (dl<dr)
81     {
82         if (dl<ans) querymn(l);
83         if (dr<ans) querymn(r);
84     }
85     else
86     {
87         if (dr<ans) querymn(r);
88         if (dl<ans) querymn(l);
89     }
90 }
91 int query(int f,int x,int y)
92 {
93     T[0]=x;T[1]=y;
94     if (f==0) ans=-inf,querymx(root);
95     else ans=inf,querymn(root);
96     return ans;
97 }
98 }kd;
99 ```\newpage
100
101 ## 二维几何 点和向量
102
103 ```c++
104 #include <bits/stdc++.h>
105 using namespace std;
106 #define mp make_pair
107 #define fi first
108 #define se second
109 #define pb push_back
110 typedef double db;
111 const db eps = 1e-6;
112 const db pi = acos(-1.0);
113 int sign(db k)
114 {
115     if (k > eps)
116         return 1;
117     else if (k < -eps)
118         return -1;
119     return 0;
120 }
121 int cmp(db k1, db k2) { return sign(k1 - k2); }

```

```

122 int inmid(db k1, db k2, db k3) { return sign(k1 - k3) * sign(k2 - k3) <= 0; } // k3 在
    ↪ [k1,k2] 内
123 struct point
124 {
125     db x, y;
126     point operator+(const point &k1) const { return (point){k1.x + x, k1.y + y}; }
127     point operator-(const point &k1) const { return (point){x - k1.x, y - k1.y}; }
128     point operator*(db k1) const { return (point){x * k1, y * k1}; }
129     point operator/(db k1) const { return (point){x / k1, y / k1}; }
130     int operator==(const point &k1) const { return cmp(x, k1.x) == 0 && cmp(y, k1.y)
    ↪ == 0; }
131     // 逆时针旋转
132     point turn(db k1) { return (point){x * cos(k1) - y * sin(k1), x * sin(k1) + y *
    ↪ cos(k1)}; }
133     point turn90() { return (point){-y, x}; }
134     bool operator<(const point k1) const //x 为第一关键词 y 为第二关键词
135     {
136         int a = cmp(x, k1.x);
137         if (a == -1)
138             return 1;
139         else if (a == 1)
140             return 0;
141         else
142             return cmp(y, k1.y) == -1;
143     }
144     db abs() { return sqrt(x * x + y * y); }
145     db abs2() { return x * x + y * y; }
146     db dis(point k1) { return ((*this) - k1).abs(); }
147     point unit()
148     {
149         db w = abs();
150         return (point){x / w, y / w};
151     }
152     void scan()
153     {
154         double k1, k2;
155         scanf("%lf%lf", &k1, &k2);
156         x = k1;
157         y = k2;
158     }
159     void print() { printf("%.11lf %.11lf\n", x, y); }
160     db getw() { return atan2(y, x); }
161     point getdel()
162     {
163         if (sign(x) == -1 || (sign(x) == 0 && sign(y) == -1))
164             return (*this) * (-1);
165         else
166             return (*this);
167     }
168     int getP() const { return sign(y) == 1 || (sign(y) == 0 && sign(x) == -1); }
169 };

```



```

170 int inmid(point k1, point k2, point k3) { return inmid(k1.x, k2.x, k3.x) &&
    ↪ inmid(k1.y, k2.y, k3.y); }
171 db cross(point k1, point k2) { return k1.x * k2.y - k1.y * k2.x; }
172 db dot(point k1, point k2) { return k1.x * k2.x + k1.y * k2.y; }
173 db rad(point k1, point k2) { return atan2(cross(k1, k2), dot(k1, k2)); }
174 // -pi -> pi
175 int compareangle(point k1, point k2)
176 {
177     return k1.getP() < k2.getP() || (k1.getP() == k2.getP() && sign(cross(k1, k2)) >
    ↪ 0);
178 }
179 point proj(point k1, point k2, point q)
180 { // q 到直线 k1,k2 的投影
181     point k = k2 - k1;
182     return k1 + k * (dot(q - k1, k) / k.abs2());
183 }
184 point reflect(point k1, point k2, point q) { return proj(k1, k2, q) * 2 - q; } q
    ↪ 对于直线 k1,k2 的对称点
185 int clockwise(point k1, point k2, point k3)
186 { // k1 k2 k3 逆时针 1 顺时针 -1 否则 0
187     return sign(cross(k2 - k1, k3 - k1));
188 }
189 int checkLL(point k1, point k2, point k3, point k4)
190 { // 求直线 (L) 线段 (S)k1,k2 和 k3,k4 的交点
191     return cmp(cross(k3 - k1, k4 - k1), cross(k3 - k2, k4 - k2)) != 0;
192 }
193 point getLL(point k1, point k2, point k3, point k4)
194 {
195     db w1 = cross(k1 - k3, k4 - k3), w2 = cross(k4 - k3, k2 - k3);
196     return (k1 * w2 + k2 * w1) / (w1 + w2);
197 }
198 int intersect(db l1, db r1, db l2, db r2)
199 {
200     if (l1 > r1)
201         swap(l1, r1);
202     if (l2 > r2)
203         swap(l2, r2);
204     return cmp(r1, l2) != -1 && cmp(r2, l1) != -1;
205 }
206 int checkSS(point k1, point k2, point k3, point k4)
207 { // 非规范相交 <= 0 ; 规范相交 < 0
208     return intersect(k1.x, k2.x, k3.x, k4.x) && intersect(k1.y, k2.y, k3.y, k4.y) &&
209         sign(cross(k3 - k1, k4 - k1)) * sign(cross(k3 - k2, k4 - k2)) <= 0 &&
210         sign(cross(k1 - k3, k2 - k3)) * sign(cross(k1 - k4, k2 - k4)) <= 0;
211 }
212 db disSP(point k1, point k2, point q)
213 { // 点 (q) 到线段 (k1, k2) 距离
214     point k3 = proj(k1, k2, q);
215     if (inmid(k1, k2, k3))
216         return q.dis(k3);
217     else
218         return min(q.dis(k1), q.dis(k2));

```

```

219 }
220 db disSS(point k1, point k2, point k3, point k4)
221 {
222     if (checkSS(k1, k2, k3, k4))
223         return 0;
224     else
225         return min(min(disSP(k1, k2, k3), disSP(k1, k2, k4)), min(disSP(k3, k4, k1),
            ↪ disSP(k3, k4, k2)));
226 }
227 int onS(point k1, point k2, point q) //点 q 在点 k1,k2 之间
228 {
229     return inmid(k1, k2, q) && sign(cross(k1 - q, k2 - k1)) == 0;
230 }

```

多边形

```

1 db area(vector<point> A)
2 { // 多边形用 vector<point> 表示 , 逆时针
3     db ans = 0;
4     for (int i = 0; i < A.size(); i++)
5         ans += cross(A[i], A[(i + 1) % A.size()]);
6     return ans / 2;
7 }
8 int checkconvex(vector<point> A)
9 { // 逆时针
10     int n = A.size();
11     A.push_back(A[0]);
12     A.push_back(A[1]);
13     for (int i = 0; i < n; i++)
14         if (sign(cross(A[i + 1] - A[i], A[i + 2] - A[i])) == -1)
15             return 0;
16     return 1;
17 }
18 int contain(vector<point> A, point q)
19 { // 2 内部 1 边界 0 外部
20     int pd = 0;
21     A.push_back(A[0]);
22     for (int i = 1; i < A.size(); i++)
23     {
24         point u = A[i - 1], v = A[i];
25         if (onS(u, v, q))
26             return 1;
27         if (cmp(u.y, v.y) > 0)
28             swap(u, v);
29         if (cmp(u.y, q.y) >= 0 || cmp(v.y, q.y) < 0)
30             continue;
31         if (sign(cross(u - v, q - v)) < 0)
32             pd ^= 1;
33     }
34     return pd << 1;
35 }
36 vector<point> ConvexHull(vector<point> A, int flag = 1) //凸包

```

```

37 { // flag=0 不严格 flag=1 严格
38     int n = A.size();
39     vector<point> ans(n * 2);
40     sort(A.begin(), A.end());
41     int now = -1;
42     for (int i = 0; i < A.size(); i++)
43     {
44         while (now > 0 && sign(cross(ans[now] - ans[now - 1], A[i] - ans[now - 1])) <
45             ↪ flag)
46             now--;
47         ans[++now] = A[i];
48     }
49     int pre = now;
50     for (int i = n - 2; i >= 0; i--)
51     {
52         while (now > pre && sign(cross(ans[now] - ans[now - 1], A[i] - ans[now - 1]))
53             ↪ < flag)
54             now--;
55         ans[++now] = A[i];
56     }
57     ans.resize(now);
58     return ans;
59 }
60 db convexDiameter(vector<point> A)
61 { // 凸包直径
62     int now = 0, n = A.size();
63     db ans = 0;
64     for (int i = 0; i < A.size(); i++)
65     {
66         now = max(now, i);
67         while (1)
68         {
69             db k1 = A[i].dis(A[now % n]), k2 = A[i].dis(A[(now + 1) % n]);
70             ans = max(ans, max(k1, k2));
71             if (k2 > k1)
72                 now++;
73             else
74                 break;
75         }
76     }
77     return ans;
78 }

```

圆和线段

```

1 struct circle
2 {
3     point o;
4     db r;
5     void scan()
6     {
7         o.scan();

```

```

8         scanf("%lf", &r);
9     }
10    int inside(point k) { return cmp(r, o.dis(k)); }
11};
12struct line
13{
14    // p[0]->p[1]
15    point p[2];
16    line(point k1, point k2)
17    {
18        p[0] = k1;
19        p[1] = k2;
20    }
21    point &operator[](int k) { return p[k]; }
22    int include(point k) { return sign(cross(p[1] - p[0], k - p[0])) > 0; }
23    point dir() { return p[1] - p[0]; }
24    line push()
25    { // 向外 ( 左手边 ) 平移 eps
26        const db eps = 1e-6;
27        point delta = (p[1] - p[0]).turn90().unit() * eps;
28        return {p[0] - delta, p[1] - delta};
29    }
30};
31point getLL(line k1, line k2) { return getLL(k1[0], k1[1], k2[0], k2[1]); }
32int parallel(line k1, line k2) { return sign(cross(k1.dir(), k2.dir())) == 0; }
33int sameDir(line k1, line k2) { return parallel(k1, k2) && sign(dot(k1.dir(),
    ↪ k2.dir())) == 1; }
34int operator<(line k1, line k2)
35{
36    if (sameDir(k1, k2))
37        return k2.include(k1[0]);
38    return compareangle(k1.dir(), k2.dir());
39}
40int checkpos(line k1, line k2, line k3) { return k3.include(getLL(k1, k2)); }
41
42int checkposCC(circle k1, circle k2)
43{ // 返回两个圆的公切线数量
44    if (cmp(k1.r, k2.r) == -1)
45        swap(k1, k2);
46    db dis = k1.o.dis(k2.o);
47    int w1 = cmp(dis, k1.r + k2.r), w2 = cmp(dis, k1.r - k2.r);
48    if (w1 > 0)
49        return 4;
50    else if (w1 == 0)
51        return 3;
52    else if (w2 > 0)
53        return 2;
54    else if (w2 == 0)
55        return 1;
56    else
57        return 0;
58}

```

```

59 vector<point> getCL(circle k1, point k2, point k3)
60 { // 沿着 k2->k3 方向给出 , 相切给出两个
61     point k = proj(k2, k3, k1.o);
62     db d = k1.r * k1.r - (k - k1.o).abs2();
63     if (sign(d) == -1)
64         return {};
65     point del = (k3 - k2).unit() * sqrt(max((db)0.0, d));
66     return {k - del, k + del};
67 }
68 vector<point> getCC(circle k1, circle k2)
69 { // 沿圆 k1 逆时针给出 , 相切给出两个
70     int pd = checkposCC(k1, k2);
71     if (pd == 0 || pd == 4)
72         return {};
73     db a = (k2.o - k1.o).abs2(), cosA = (k1.r * k1.r + a - k2.r * k2.r) / (2 * k1.r *
74         ↪ sqrt(max(a, (db)0.0)));
75     db b = k1.r * cosA, c = sqrt(max((db)0.0, k1.r * k1.r - b * b));
76     point k = (k2.o - k1.o).unit(), m = k1.o + k * b, del = k.turn90() * c;
77     return {m - del, m + del};
78 }
79 vector<point> TangentCP(circle k1, point k2)
80 { // 沿圆 k1 逆时针给出
81     db a = (k2 - k1.o).abs(), b = k1.r * k1.r / a, c = sqrt(max((db)0.0, k1.r * k1.r
82         ↪ - b * b));
83     point k = (k2 - k1.o).unit(), m = k1.o + k * b, del = k.turn90() * c;
84     return {m - del, m + del};
85 }
86 vector<line> TangentoutCC(circle k1, circle k2)
87 {
88     int pd = checkposCC(k1, k2);
89     if (pd == 0)
90         return {};
91     if (pd == 1)
92     {
93         point k = getCC(k1, k2)[0];
94         return {(line){k, k}};
95     }
96     if (cmp(k1.r, k2.r) == 0)
97     {
98         point del = (k2.o - k1.o).unit().turn90().getdel();
99         return {(line){k1.o - del * k1.r, k2.o - del * k2.r}, (line){k1.o + del *
100             ↪ k1.r, k2.o + del * k2.r}};
101     }
102     else
103     {
104         point p = (k2.o * k1.r - k1.o * k2.r) / (k1.r - k2.r);
105         vector<point> A = TangentCP(k1, p), B = TangentCP(k2, p);
106         vector<line> ans;
107         for (int i = 0; i < A.size(); i++)
108             ans.push_back((line){A[i], B[i]});
109         return ans;
110     }
111 }

```

```

108 }
109 vector<line> TangentinCC(circle k1, circle k2)
110 {
111     int pd = checkposCC(k1, k2);
112     if (pd <= 2)
113         return {};
114     if (pd == 3)
115     {
116         point k = getCC(k1, k2)[0];
117         return {(line){k, k}};
118     }
119     point p = (k2.o * k1.r + k1.o * k2.r) / (k1.r + k2.r);
120     vector<point> A = TangentCP(k1, p), B = TangentCP(k2, p);
121     vector<line> ans;
122     for (int i = 0; i < A.size(); i++)
123         ans.push_back((line){A[i], B[i]});
124     return ans;
125 }
126 vector<line> TangentCC(circle k1, circle k2)
127 {
128     int flag = 0;
129     if (k1.r < k2.r)
130         swap(k1, k2), flag = 1;
131     vector<line> A = TangentoutCC(k1, k2), B = TangentinCC(k1, k2);
132     for (line k : B)
133         A.push_back(k);
134     if (flag)
135         for (line &k : A)
136             swap(k[0], k[1]);
137     return A;
138 }
139 db getarea(circle k1, point k2, point k3)
140 {
141     // 圆 k1 与三角形 k2 k3 k1.o 的有向面积交
142     point k = k1.o;
143     k1.o = k1.o - k;
144     k2 = k2 - k;
145     k3 = k3 - k;
146     int pd1 = k1.inside(k2), pd2 = k1.inside(k3);
147     vector<point> A = getCL(k1, k2, k3);
148     if (pd1 >= 0)
149     {
150         if (pd2 >= 0)
151             return cross(k2, k3) / 2;
152         return k1.r * k1.r * rad(A[1], k3) / 2 + cross(k2, A[1]) / 2;
153     }
154     else if (pd2 >= 0)
155     {
156         return k1.r * k1.r * rad(k2, A[0]) / 2 + cross(A[0], k3) / 2;
157     }
158     else
159     {

```

```

160         int pd = cmp(k1.r, disSP(k2, k3, k1.o));
161         if (pd <= 0)
162             return k1.r * k1.r * rad(k2, k3) / 2;
163         return cross(A[0], A[1]) / 2 + k1.r * k1.r * (rad(k2, A[0]) + rad(A[1], k3))
           ↪ / 2;
164     }
165 }
166 circle getcircle(point k1, point k2, point k3)
167 {
168     // 三点求圆
169     db a1 = k2.x - k1.x, b1 = k2.y - k1.y, c1 = (a1 * a1 + b1 * b1) / 2;
170     db a2 = k3.x - k1.x, b2 = k3.y - k1.y, c2 = (a2 * a2 + b2 * b2) / 2;
171     db d = a1 * b2 - a2 * b1;
172     point o = (point){k1.x + (c1 * b2 - c2 * b1) / d, k1.y + (a1 * c2 - a2 * c1) /
           ↪ d};
173     return (circle){o, k1.dis(o)};
174 }
175 circle getScircle(vector<point> A)
176 {
177     //随机增量法 最小圆覆盖
178     random_shuffle(A.begin(), A.end());
179     circle ans = (circle){A[0], 0};
180     for (int i = 1; i < A.size(); i++)
181         if (ans.inside(A[i]) == -1)
182             {
183                 ans = (circle){A[i], 0};
184                 for (int j = 0; j < i; j++)
185                     if (ans.inside(A[j]) == -1)
186                         {
187                             ans.o = (A[i] + A[j]) / 2;
188                             ans.r = ans.o.dis(A[i]);
189                             for (int k = 0; k < j; k++)
190                                 if (ans.inside(A[k]) == -1)
191                                     ans = getcircle(A[i], A[j], A[k]);
192                         }
193             }
194     return ans;
195 }

```

其他

```

1 vector<line> getHL(vector<line> &L)
2 { // 求半平面交 , 半平面是逆时针方向 , 输出按照逆时针
3     sort(L.begin(), L.end());
4     deque<line> q;
5     for (int i = 0; i < (int)L.size(); i++)
6     {
7         if (i && sameDir(L[i], L[i - 1]))
8             continue;
9         while (q.size() > 1 && !checkpos(q[q.size() - 2], q[q.size() - 1], L[i]))
10             q.pop_back();
11         while (q.size() > 1 && !checkpos(q[1], q[0], L[i]))

```

```

12         q.pop_front();
13         q.push_back(L[i]);
14     }
15     while (q.size() > 2 && !checkpos(q[q.size() - 2], q[q.size() - 1], q[0]))
16         q.pop_back();
17     while (q.size() > 2 && !checkpos(q[1], q[0], q[q.size() - 1]))
18         q.pop_front();
19     vector<line> ans;
20     for (int i = 0; i < q.size(); i++)
21         ans.push_back(q[i]);
22     return ans;
23 }
24 db closepoint(vector<point> &A, int l, int r)
25 { // 最近点对, 先要按照 x 坐标排序
26     if (r - l <= 5)
27     {
28         db ans = 1e20;
29         for (int i = l; i <= r; i++)
30             for (int j = i + 1; j <= r; j++)
31                 ans = min(ans, A[i].dis(A[j]));
32         return ans;
33     }
34     int mid = ((l + r) >> 1);
35     db ans = min(closepoint(A, l, mid), closepoint(A, mid + 1, r));
36     vector<point> B;
37     for (int i = l; i <= r; i++)
38         if (abs(A[i].x - A[mid].x) <= ans)
39             B.push_back(A[i]);
40     sort(B.begin(), B.end(), [](point k1, point k2) { return k1.y < k2.y; });
41     for (int i = 0; i < B.size(); i++)
42         for (int j = i + 1; j < B.size() && B[j].y - B[i].y < ans; j++)
43             ans = min(ans, B[i].dis(B[j]));
44     return ans;
45 }
46 vector<point> convexcut(vector<point> A, point k1, point k2)
47 {
48     // 保留 k1, k2, p 逆时针的所有点
49     int n = A.size();
50     A.push_back(A[0]);
51     vector<point> ans;
52     for (int i = 0; i < n; i++)
53     {
54         int w1 = clockwise(k1, k2, A[i]), w2 = clockwise(k1, k2, A[i + 1]);
55         if (w1 >= 0)
56             ans.push_back(A[i]);
57         if (w1 * w2 < 0)
58             ans.push_back(getLL(k1, k2, A[i], A[i + 1]));
59     }
60     return ans;
61 }
62 int checkPoS(vector<point> A, point k1, point k2)
63 {

```



```

64 // 多边形 A 和直线 ( 线段 )k1->k2 严格相交 , 注释部分为线段
65 struct ins
66 {
67     point m, u, v;
68     int operator<(const ins &k) const { return m < k.m; }
69 };
70 vector<ins> B;
71 //if (contain(A,k1)==2||contain(A,k2)==2) return 1;
72 vector<point> poly = A;
73 A.push_back(A[0]);
74 for (int i = 1; i < A.size(); i++)
75     if (checkLL(A[i - 1], A[i], k1, k2))
76     {
77         point m = getLL(A[i - 1], A[i], k1, k2);
78         if (inmid(A[i - 1], A[i], m) /*inmid(k1,k2,m)*/)
79             B.push_back((ins){m, A[i - 1], A[i]});
80     }
81 if (B.size() == 0)
82     return 0;
83 sort(B.begin(), B.end());
84 int now = 1;
85 while (now < B.size() && B[now].m == B[0].m)
86     now++;
87 if (now == B.size())
88     return 0;
89 int flag = contain(poly, (B[0].m + B[now].m) / 2);
90 if (flag == 2)
91     return 1;
92 point d = B[now].m - B[0].m;
93 for (int i = now; i < B.size(); i++)
94 {
95     if (!(B[i].m == B[i - 1].m) && flag == 2)
96         return 1;
97     int tag = sign(cross(B[i].v - B[i].u, B[i].m + d - B[i].u));
98     if (B[i].m == B[i].u || B[i].m == B[i].v)
99         flag += tag;
100     else
101         flag += tag * 2;
102 }
103 //return 0;
104 return flag == 2;
105 }

```

图论

树链剖分

- 维护点权，边权要下放

```
1  int fa[maxn], dep[maxn], maxson[maxn], son[maxn]; //dfs 数组
2  int top[maxn], dfn[maxn], tot; //link 数组
3  int dfs(int u)
4  {
5      int ret = 0;
6      for (int i = head[u]; i != -1; i = edge[i].nxt)
7      {
8          int v = edge[i].to;
9          if (v == fa[u])
10             continue;
11          fa[v] = u;
12          dep[v] = dep[u] + 1;
13          int sz = dfs(v);
14          ret += sz;
15          if (sz > maxson[u])
16          {
17              maxson[u] = sz;
18              son[u] = v;
19          }
20      }
21      return ret + 1;
22  }
23  void link(int u, int t)
24  {
25      dfn[u] = ++tot;
26      top[u] = t;
27      if (son[u])
28          link(son[u], t);
29      for (int i = head[u]; i != -1; i = edge[i].nxt)
30      {
31          int v = edge[i].to;
32          if (v == fa[u] || v == son[u])
33              continue;
34          link(v, v);
35      }
36  }
37  void hld()
38  {
39      dfs(1);
40      link(1, 1);
41  }
```

边双联通分量

- 将无向图缩成一棵树
- 有重边需要做一些改动
- 稍微改一下就是强联通缩点了

```

1  int dfn[maxn], low[maxn], bel[maxn];
2  int n, m;
3  int ti, scc; //时间戳与联通分量计数
4  stack<int> st;
5  void dfs(int u, int fa)
6  {
7      dfn[u] = low[u] = ++ti;
8      st.push(u);
9      for (int i = head[u]; i != -1; i = edge[i].nxt)
10     {
11         int v = edge[i].to;
12         if (v == fa)
13             continue;
14         if (!dfn[v])
15         {
16             dfs(v, u);
17             low[u] = min(low[u], low[v]);
18         }
19         else if (!bel[v])
20             low[u] = min(low[u], dfn[v]);
21     }
22     if (dfn[u] == low[u])
23     {
24         scc++;
25         while (1)
26         {
27             int t = st.top();
28             st.pop();
29             bel[t] = scc;
30             if (u == t)
31                 break;
32         }
33     }
34 }
35 void DCC()
36 {
37     for (int i = 1; i <= n; i++)
38         if (!dfn[i])
39             dfs(i, -1);
40     for (int i = 0; i < cur; i++) //遍历所有边建图
41     {
42         int u = edge[i].from, v = edge[i].to;
43         if (bel[u] != bel[v])
44             addedge2(bel[u], bel[v]);
45     }
46 }

```

虚树

- 注意每次清空虚树图
- 根节点必定是关键点

```

1  const int maxn = 5e5 + 7;

```

```

2  int ti; //时间戳
3  int ts[maxn];
4  int depth[maxn];
5  int far[maxn][22];
6  int dfn[maxn];
7  void dfs(int u, int fa = -1)
8  {
9      dfn[u] = ++ti; //dfs 序
10     ts[ti] = u;    //括号序列时间戳映射
11     for (int i = head[u]; i != -1; i = edge[i].nxt)
12     {
13         int v = edge[i].to;
14         if (v == fa)
15             continue;
16         depth[v] = depth[u] + 1;
17         far[v][0] = u; //可能会用到的倍增数组
18         dfs(v, u);
19         ts[++ti] = u;
20     }
21 }
22 int ST[maxn][22]; //LCA 转 RMQ 用 ST 表求
23 bool cmp(int x, int y) { return depth[x] < depth[y]; }
24 void RMQ()
25 {
26     for (int i = 1; i <= ti; i++)
27         ST[i][0] = ts[i];
28     for (int j = 1; (1 << j) <= ti; j++)
29     {
30         for (int i = 1; i + (1 << (j - 1)) - 1 <= ti; i++)
31         {
32             if (cmp(ST[i][j - 1], ST[i + (1 << (j - 1))][j - 1]))
33                 ST[i][j] = ST[i][j - 1];
34             else
35                 ST[i][j] = ST[i + (1 << (j - 1))][j - 1];
36         }
37     }
38 }
39 int LCA(int u, int v)
40 {
41     int l = dfn[u], r = dfn[v];
42     if (l > r)
43         swap(l, r);
44     int k = 31 - __builtin_clz(r - l + 1);
45     if (cmp(ST[l][k], ST[r - (1 << k) + 1][k]))
46         return ST[l][k];
47     return ST[r - (1 << k) + 1][k];
48 }
49 void Virtual(vector<int> &all) //关键点
50 {
51     int st[maxn]; //栈模拟访问
52     int top = 0;
53     for (auto &u : all)

```

```

54 {
55     if (top == 0)
56         st[++top] = u;
57     else
58     {
59         int lca = LCA(st[top], u);
60         while (top > 1 && dfn[st[top - 1]] >= dfn[lca])
61         { //栈中至少有两个元素，则开始向上连边
62             access(st[top], st[top-1]);
63             top--;
64         }
65         if (lca != st[top]) //最后将 lca 也放进去
66         {
67             access(st[top], lca);
68             st[top] = lca;
69         }
70         st[++top] = u;
71     }
72 }
73 while (top > 1) //所有元素出栈
74 {
75     access(st[top], st[top - 1]);
76     --top;
77 }
78 }

```

网络流

DINIC

```

1  const int maxn = 1e3 + 7;
2  const int INF = 0x3f3f3f3f;
3  struct Edge
4  {
5      int from, to, cap, flow;
6  };
7  struct Dinic
8  {
9      int n, m, s, t;
10     vector<Edge> edges;
11     vector<int> G[maxn];
12     bool vis[maxn];
13     int d[maxn];
14     int cur[maxn];
15
16     void AddEdge(int from, int to, int cap, int c = 0)
17     {
18         edges.push_back(Edge{from, to, cap, 0});
19         edges.push_back(Edge{to, from, c, 0});
20         m = edges.size();
21         G[from].push_back(m - 2);
22         G[to].push_back(m - 1);
23     }

```

```

24
25 bool BFS()
26 {
27     memset(vis, 0, sizeof(vis));
28     queue<int> q;
29     q.push(s);
30     d[s] = 0;
31     vis[s] = 1;
32     while (!q.empty())
33     {
34         int u = q.front();
35         q.pop();
36         for (int i = 0; i < G[u].size(); i++)
37         {
38             Edge &e = edges[G[u][i]];
39             if (!vis[e.to] && e.cap > e.flow)
40             {
41                 vis[e.to] = 1;
42                 d[e.to] = d[u] + 1;
43                 q.push(e.to);
44             }
45         }
46     }
47     return vis[t];
48 }
49 int DFS(int u, int dist)
50 {
51     if (u == t || dist == 0)
52         return dist;
53     int flow = 0, f;
54     for (int &i = cur[u]; i < G[u].size(); i++)
55     {
56         Edge &e = edges[G[u][i]];
57         if (d[u] + 1 == d[e.to] && (f = DFS(e.to, min(dist, e.cap - e.flow))) >
58             ↪ 0)
59         {
60             e.flow += f;
61             edges[G[u][i] ^ 1].flow -= f;
62             flow += f;
63             dist -= f;
64             if (!dist)
65                 break;
66         }
67     }
68     return flow;
69 }
70 int Maxflow(int s, int t)
71 {
72     this->s = s;
73     this->t = t;
74     int flow = 0;
75     while (BFS())

```

```

75     {
76         memset(cur, 0, sizeof(cur));
77         flow += DFS(s, INF);
78     }
79     return flow;
80 }
81 };

```

费用流

```

1  const int maxn = 1e3 + 7;
2  const int INF = 0x3f3f3f3f;
3  struct Edge
4  {
5      int from, to, cap, flow, cost;
6  };
7  struct MCMF
8  {
9      int n, m, s, t;
10     vector<Edge> edges;
11     vector<int> G[maxn];
12     int inq[maxn];
13     int d[maxn]; //最短路径数组
14     int p[maxn]; //记录路径
15     int a[maxn]; //记录流量
16     void init(int n)
17     {
18         this->n = n;
19         for (int i = 0; i < n; i++)
20             G[i].clear();
21         edges.clear();
22     }
23     void addedge(int from, int to, int cap, int cost)
24     {
25         edges.push_back(Edge{from, to, cap, 0, cost});
26         edges.push_back(Edge{to, from, 0, 0, -cost});
27         m = edges.size();
28         G[from].push_back(m - 2);
29         G[to].push_back(m - 1);
30     }
31     bool spfa(int s, int t, int &flow, int &cost)
32     {
33         for (int i = 0; i < n; i++)
34             d[i] = INF;
35         memset(inq, 0, sizeof(inq));
36         d[s] = 0;
37         inq[s] = 1;
38         p[s] = 0;
39         a[s] = INF;
40         queue<int> q;
41         q.push(s);
42         while (!q.empty())

```

```

43     {
44         int u = q.front();
45         q.pop();
46         inq[u] = 0;
47         for (int i = 0; i < G[u].size(); i++)
48         {
49             Edge &e = edges[G[u][i]];
50             if (e.cap > e.flow && d[e.to] > d[u] + e.cost)
51             {
52                 d[e.to] = d[u] + e.cost;           //松弛
53                 p[e.to] = G[u][i];                 //记录上一个点
54                 a[e.to] = min(a[u], e.cap - e.flow); //流量控制
55                 if (!inq[e.to])
56                 {
57                     q.push(e.to);
58                     inq[e.to] = 1;
59                 }
60             }
61         }
62     }
63     if (d[t] == INF)
64         return false; //不存在最短路
65     flow += a[t];
66     cost += d[t] * a[t];
67     int u = t;
68     while (u != s)
69     {
70         edges[p[u]].flow += a[t];
71         edges[p[u] ^ 1].flow -= a[t];
72         u = edges[p[u]].from;
73     }
74     return true;
75 }
76 int Mincost(int s, int t)
77 {
78     int flow = 0, cost = 0;
79     while (spfa(s, t, flow, cost))
80         ;
81     return cost;
82 }
83 };

```

最小树形图

```

1  struct Edge//边的权和顶点
2  {
3      int u, v;
4      ll w;
5  }edge[maxn * maxn];
6  int pre[maxn], id[maxn], vis[maxn], pos;
7  ll in[maxn]; //存最小入边权, pre[v] 为该边的起点
8

```



```

9  ll Directed_MST(int root, int V, int E)
10 {
11     ll ret = 0; //存最小树形图总权值
12     while(true)
13     {
14         int i;
15         //1. 找每个节点的最小入边
16         for( i = 0; i < V; i++)
17             in[i] = INF; //初始化为无穷大
18         for( i = 0; i < E; i++) //遍历每条边
19         {
20             int u = edge[i].u;
21             int v = edge[i].v;
22             if(edge[i].w < in[v] && u != v) //说明顶点 v 有条权值较小的入边 记录之
23             {
24                 pre[v] = u; //节点 u 指向 v
25                 in[v] = edge[i].w; //最小入边
26                 if(u == root) //这个点就是实际的起点
27                     pos = i;
28             }
29         }
30         for( i = 0; i < V; i++) //判断是否存在最小树形图
31         {
32             if(i == root)
33                 continue;
34             if(in[i] == INF)
35                 return -1; //除了根以外有点没有入边, 则根无法到达它 说明它是独立的点 一定不
36                                     //能构成树形图
37         }
38         //2. 找环
39         int cnt = 0; //记录环数
40         memset(id, -1, sizeof(id));
41         memset(vis, -1, sizeof(vis));
42         in[root] = 0;
43         for( i = 0; i < V; i++) //标记每个环
44         {
45             ret += in[i]; //记录权值
46             int v = i;
47             while(vis[v] != i && id[v] == -1 && v != root)
48             {
49                 vis[v] = i;
50                 v = pre[v];
51             }
52             if(v != root && id[v] == -1)
53             {
54                 for(int u = pre[v]; u != v; u = pre[u])
55                     id[u] = cnt; //标记节点 u 为第几个环
56                 id[v] = cnt++;
57             }
58         }
59         if(cnt == 0)
60             break; //无环 则 break

```

```

60     for( i = 0; i < V; i++)
61         if(id[i] == -1)
62             id[i] = cnt++;
63         //3. 建立新图    缩点，重新标记
64         for( i = 0; i < E; i++)
65             {
66                 int u = edge[i].u;
67                 int v = edge[i].v;
68                 edge[i].u = id[u];
69                 edge[i].v = id[v];
70                 if(id[u] != id[v])
71                     edge[i].w -= in[v];
72             }
73         V = cnt;
74         root = id[root];
75     }
76     return ret;
77 }

```

树的最大匹配

```

1  int f[N],g[N];
2  void dfs(int cur,int fa)
3  {
4      f[cur]=g[cur]=0;
5      for(int i=head[cur];~i;i=e[i].next)
6      {
7          if(e[i].v==fa)
8              continue;
9          int v=e[i].v;
10         dfs(v,cur);
11         g[cur]+=max(f[v],g[v]);
12     }
13     for(int i=head[cur];~i;i=e[i].next)
14     {
15         if(e[i].v==fa)
16             continue;
17         int v=e[i].v;
18         f[cur]=max(f[cur],g[cur]-max(f[v],g[v])+g[v]+1);
19     }
20 }
21
22 int maxmatch()
23 {
24     dfs(1,-1);
25     return max(f[1],g[1]);
26 }

```

支配树

```

1  vector<int> G[N], rG[N];
2  vector<int> dt[N];

```

```

3
4 namespace t1
5 {
6     int fa[N], idx[N], clk, ridx[N];
7     int c[N], best[N], semi[N], idom[N];
8
9     void init(int n)
10    {
11        clk = 0;
12        for(int i=0; i<=n; ++i)
13        {
14            idom[i]=0;
15            c[i]=-1;
16            idx[i]=0;
17            dt[i].clear();
18            semi[i] = best[i] = i;
19        }
20    }
21    void dfs(int u)
22    {
23        idx[u] = ++clk; ridx[clk] = u;
24        for (int& v: G[u])
25            if (!idx[v])
26            {
27                fa[v] = u;
28                dfs(v);
29            }
30    }
31    int fix(int x)
32    {
33        if (c[x] == -1)
34            return x;
35        int &f = c[x], rt = fix(f);
36        if (idx[semi[best[x]]] > idx[semi[best[f]]])
37            best[x] = best[f];
38        return f = rt;
39    }
40    void go(int rt)
41    {
42        dfs(rt);
43        for(int i=clk; i>=2; --i)
44        {
45            int x = ridx[i], mn = clk + 1;
46            for (int& u: rG[x])
47            {
48                if (!idx[u])
49                    continue; // 可能不能到达所有点
50                fix(u);
51                mn = min(mn, idx[semi[best[u]]]);
52            }
53            c[x] = fa[x];
54            dt[semi[x] = ridx[mn]].push_back(x);

```

```

55         x = ridx[i - 1];
56         for (int& u: dt[x])
57         {
58             fix(u);
59             if (semi[best[u]] != x)
60                 idom[u] = best[u];
61             else
62                 idom[u] = x;
63         }
64         dt[x].clear();
65     }
66     for(int i=2;i<=clk;++i)
67     {
68         int u = ridx[i];
69         if (idom[u] != semi[u])
70             idom[u] = idom[idom[u]];
71         dt[idom[u]].push_back(u);
72     }
73 }
74 }

```

斯坦纳树

```

1  const int maxn=35;
2  struct Edge{
3      int to,w;
4  };
5  vector<Edge> G[maxn];
6  int n,m;
7  PII tar[4];
8  int st[maxn];
9  int dp[maxn][1<<8];
10 int dis[1<<8];
11 bool inq[maxn];
12 queue<int> q;
13 int cur=0;
14 bool check(int S)
15 {
16     bool ok=1;
17     for(int i=0;i<4;i++)
18     {
19         if(S&st[tar[i].first])
20             ok&=((S&st[tar[i].second])!=0);
21     }
22     return ok;
23 }
24 void spfa(int S)
25 {
26     while(!q.empty())
27     {
28         int u=q.front();
29         q.pop();

```

```

30     inq[u]=0;
31     for(auto e:G[u])
32     {
33         int v=e.to;
34         if(dp[v][S|st[v]]>dp[u][S]+e.w)
35         {
36             dp[v][S|st[v]]=dp[u][S]+e.w;
37             if(inq[v]||(st[v]|S)!=S) continue;
38             inq[v]=1;
39             q.push(v);
40         }
41     }
42 }
43 }
44 void solve()
45 {
46     memset(dp,0x3f,sizeof(dp)); //求解斯坦纳树
47     for(int i=0;i<n;i++)
48     {
49         if(st[i])
50             dp[i][st[i]]=0;
51     }
52     for(int S=0;S<(1<<cur);S++)
53     {
54         memset(inq,0,sizeof(inq));
55         for(int i=0;i<n;i++)
56         {
57             if(st[i]&&(st[i]&S)==0) continue;
58             for(int T=(S-1)&S;T;T=(T-1)&S)
59                 dp[i][S]=min(dp[i][S],dp[i][T|st[i]]+dp[i][(S-T)|st[i]]);
60             if(dp[i][S]!=INF)
61             {
62                 inq[i]=1;
63                 q.push(i);
64             }
65         }
66         spfa(S);
67     }
68     memset(dis,0x3f,sizeof(dis)); //计算答案
69     for(int S=0;S<(1<<cur);S++)
70     {
71         for(int i=0;i<n;i++)
72             dis[S]=min(dis[S],dp[i][S]);
73     }
74     for(int S=0;S<(1<<cur);S++)
75     {
76         if(!check(S)) continue; //满足要求的状态
77         for(int T=(S-1)&S;T;T=(T-1)&S)
78         {
79             if(!check(T)||!check(S-T)) continue;
80             dis[S]=min(dis[S],dis[T]+dis[S-T]);
81         }

```

```

82     }
83     cout<<dis[(1<<cur)-1]<<endl;
84 }

```

2-SAT

```

1  const int maxn=8e3+7; //HDU 826ms
2  struct Twosat    //O(n^2) 得到字典序最小的方案
3  {
4      //编号从 0~2n-1 顺序每两个为一对相斥的
5      int n;
6      vector<int> g[maxn*2];
7      bool mark[maxn*2];
8      int s[maxn*2],c;//一个栈
9      bool dfs(int x)
10     {
11         if(mark[x^1])
12             return false;
13         if(mark[x])
14             return true;
15         mark[x]=true;
16         s[c++]=x;
17         for(int to:g[x])
18             if(!dfs(to))
19                 return false;
20         return true;
21     }
22     void init(int n)
23     {
24         this->n=n;
25         for(int i=0;i<n*2;i++)
26         {
27             g[i].clear();
28             mark[i]=0;
29         }
30     }
31     void add(int x,int y)
32     { //这个函数随意变化
33         g[x].push_back(y^1); //选了 x 就必须选 y^1
34         g[y].push_back(x^1);
35     }
36     bool solve()
37     {
38         for(int i=0;i<n*2;i+=2)
39             if(!mark[i]&&!mark[i+1])
40             {
41                 c=0;
42                 if(!dfs(i))
43                 {
44                     while(c>0)
45                         mark[s[--c]]=false;
46                     if(!dfs(i+1))

```

```

47         return false;
48     }
49 }
50 return true;
51 }
52 }sat;

```

网络流基本建图技巧

- 最大流 = 最小割
- 拆点技巧
- 最小点覆盖 = 最大匹配
- 最大独立集 = 点数-最小覆盖集
- 最小路径覆盖最小路径覆盖 = 原图节点数-最大匹配拆点分出入度变二分图。一开始把每个点都视为一条路径，每次匹配相当于合并两条路径，匹配几次路径数就少了多少。
- 上下界网络流先让每条边有下界那么多的流量，然后再去增大某些弧的流量使流量平衡。具体操作即为平衡与补流。对于入流大于出流的点，由 S 向其建边补流，对于出流大于入流的点，向 T 连边平衡。具体操作时用一个数组统计入流与出流差即可。对于有源汇的情况，加一条由 t 到 s，容量为 INF 的边，改造成循环流即可。
- 最小流首先按照有源汇可行流的方法建模，但是不要建立 $\langle t, s \rangle$ 这条弧。然后在这个图上，跑从附加源 ss 到附加汇 tt 的最大流。这时候再添加弧 $\langle t, s \rangle$ 再跑从 ss 到 tt 的最大流，就是原图的最小流。
- 区间 k 覆盖最小费用最大流， $[i, j]$ 区间覆盖 $\text{add_edge}(i, j, 1, -w)$ 点之间连 $\text{add_edge}(i, i+1, k, 0)$
- 优化建图目前已知的有二进制优化和线段树优化。
- 最大权闭合子图建图：设置超级源汇 S 和所有的正权的点连接权值为点权的边，所有点权为负的点与 T 连接权值为点权绝对值的边。然后如果选择了某个 v 点才可以选 u 点，那么把 u 向 v 连接一条权值为 INF 的边。最大点权和 = 正点权和-最小割

一些公式

- Carley format : $(n+1)^{n-1}, T_{n,k} = kn^{n-k-1}$

杂项

简单随机数

```
1 inline int rnd()
2 {
3     static int seed=2333;
4     return seed=(int)seed*482711LL%2147483647;
5 }
```

输入输出挂

- 只支持整数
- 支持负数
- 不带空格

```
1 template <typename T>
2 bool scan_d(T &num)
3 {
4     char in;bool IsN=false;
5     in=getchar();
6     if(in==EOF) return false;
7     while(in!='-'&&(in<'0' || in>'9')) in=getchar();
8     if(in=='-'){ IsN=true;num=0;}
9     else num=in-'0';
10    while(in=getchar(),in>='0'&&in<='9'){
11        num*=10,num+=in-'0';
12    }
13    if(IsN) num=-num;
14    return true;
15 }
16 template <typename T>
17 void o(T p) {
18     static int stk[70], tp;
19     if (p == 0) { putchar('0'); return; }
20     if (p < 0) { p = -p; putchar('-'); }
21     while (p) stk[++tp] = p % 10, p /= 10;
22     while (tp) putchar(stk[tp--] + '0');
23 }
```

HDU 专用

```
1 #define reads(n) FastIO::read(n)
2 namespace FastIO {
3     const int SIZE = 1 << 16;
4     char buf[SIZE], obuf[SIZE], str[60];
5     int bi = SIZE, bn = SIZE, opt;
6     int read(char *s) {
7         while (bn) {
8             for (; bi < bn && buf[bi] <= ' '; bi++);
9             if (bi < bn) break;
10            bn = fread(buf, 1, SIZE, stdin);
11            bi = 0;
12        }
13    }
```



```

12     }
13     int sn = 0;
14     while (bn) {
15         for (; bi < bn && buf[bi] > ' '; bi++) s[sn++] = buf[bi];
16         if (bi < bn) break;
17         bn = fread(buf, 1, SIZE, stdin);
18         bi = 0;
19     }
20     s[sn] = 0;
21     return sn;
22 }
23 template<typename T>
24 bool read(T& x) {
25     int n = read(str), bf;
26     if (!n) return 0;
27     int i = 0; if (str[i] == '-') bf = -1, i++; else bf = 1;
28     for (x = 0; i < n; i++) x = x * 10 + str[i] - '0';
29     if (bf < 0) x = -x;
30     return 1;
31 }
32 };

```

日期

```

1  // Routines for performing computations on dates. In these routines,
2  // months are expressed as integers from 1 to 12, days are expressed
3  // as integers from 1 to 31, and years are expressed as 4-digit
4  // integers.
5
6  string dayOfWeek[] = {"Mo", "Tu", "We", "Th", "Fr", "Sa", "Su"};
7
8  // converts Gregorian date to integer (Julian day number)
9
10 int DateToInt (int m, int d, int y){
11     return
12         1461 * (y + 4800 + (m - 14) / 12) / 4 +
13         367 * (m - 2 - (m - 14) / 12 * 12) / 12 -
14         3 * ((y + 4900 + (m - 14) / 12) / 100) / 4 +
15         d - 32075;
16 }
17
18 // converts integer (Julian day number) to Gregorian date: month/day/year
19
20 void IntToDate (int jd, int &m, int &d, int &y){
21     int x, n, i, j;
22
23     x = jd + 68569;
24     n = 4 * x / 146097;
25     x -= (146097 * n + 3) / 4;
26     i = (4000 * (x + 1)) / 1461001;
27     x -= 1461 * i / 4 - 31;
28     j = 80 * x / 2447;

```

```

29     d = x - 2447 * j / 80;
30     x = j / 11;
31     m = j + 2 - 12 * x;
32     y = 100 * (n - 49) + i + x;
33 }
34
35 // converts integer (Julian day number) to day of week
36
37 string IntToDay (int jd){
38     return dayOfWeek[jd % 7];
39 }

```

子集枚举

```

1  for (int T = S; T; T = (T - 1) & S)
    • 枚举大小为 k 的子集

1  template<typename T>
2  void subset(int k, int n, T&& f) {
3      int t = (1 << k) - 1;
4      while (t < 1 << n)
5      {
6          f(t);
7          int x = t & -t, y = t + x;
8          t = ((t & ~y) / x >> 1) | y;
9      }
10 }

```

大数

```

1  const int DLEN=4,MAXSIZE=100,MAXN=9999;
2  class BigNum
3  {
4  private:
5      int a[MAXSIZE];    //可以控制大数的位数
6      int len;           //大数长度
7  public:
8      BigNum(){ len = 1;memset(a,0,sizeof(a)); }    //构造函数
9      BigNum(const int);    //将一个 int 类型的变量转化为大数
10     BigNum(const char*);    //将一个字符串类型的变量转化为大数
11     BigNum(const BigNum &);    //拷贝构造函数
12     BigNum &operator=(const BigNum &);    //重载赋值运算符，大数之间进行赋值运算
13
14     friend istream& operator>>(istream&, BigNum&);    //重载输入运算符
15     friend ostream& operator<<(ostream&, BigNum&);    //重载输出运算符
16
17     BigNum operator+(const BigNum &) const;    //重载加法运算符，两个大数之间的相加运算
18     BigNum operator-(const BigNum &) const;    //重载减法运算符，两个大数之间的相减运算
19     BigNum operator*(const BigNum &) const;    //重载乘法运算符，两个大数之间的相乘运算
20     BigNum operator/(const int &) const;    //重载除法运算符，大数对一个整数进行相除运算
21
22     BigNum operator^(const int &) const;    //大数的 n 次方运算

```

```

23     int    operator%(const int &) const;    //大数对一个 int 类型的变量进行取模运算
24     bool   operator>(const BigNum & T)const; //大数和另一个大数的大小比较
25     bool   operator>(const int & t)const;   //大数和一个 int 类型的变量的大小比较
26
27     void print();        //输出大数
28 };
29 BigNum::BigNum(const int b)    //将一个 int 类型的变量转化为大数
30 {
31     int c,d = b;
32     len = 0;
33     memset(a,0,sizeof(a));
34     while(d > MAXN)
35     {
36         c = d - (d / (MAXN + 1)) * (MAXN + 1);
37         d = d / (MAXN + 1);
38         a[len++] = c;
39     }
40     a[len++] = d;
41 }
42 BigNum::BigNum(const char*s)    //将一个字符串类型的变量转化为大数
43 {
44     int t,k,index,l,i;
45     memset(a,0,sizeof(a));
46     l=strlen(s);
47     len=l/DLEN;
48     if(l%DLEN)
49         len++;
50     index=0;
51     for(i=l-1;i>=0;i-=DLEN)
52     {
53         t=0;
54         k=i-DLEN+1;
55         if(k<0)
56             k=0;
57         for(int j=k;j<=i;j++)
58             t=t*10+s[j]-'0';
59         a[index++]=t;
60     }
61 }
62 BigNum::BigNum(const BigNum & T) : len(T.len)    //拷贝构造函数
63 {
64     int i;
65     memset(a,0,sizeof(a));
66     for(i = 0 ; i < len ; i++)
67         a[i] = T.a[i];
68 }
69 BigNum & BigNum::operator=(const BigNum & n)    //重载赋值运算符，大数之间进行赋值运算
70 {
71     int i;
72     len = n.len;
73     memset(a,0,sizeof(a));
74     for(i = 0 ; i < len ; i++)

```

```

75         a[i] = n.a[i];
76     return *this;
77 }
78 istream& operator>>(istream & in, BigNum & b)    //重载输入运算符
79 {
80     char ch[MAXSIZE*4];
81     int i = -1;
82     in>>ch;
83     int l=strlen(ch);
84     int count=0,sum=0;
85     for(i=l-1;i>=0;)
86     {
87         sum = 0;
88         int t=1;
89         for(int j=0;j<4&& i>=0;j++,i--,t*=10)
90         {
91             sum+=(ch[i]-'0')*t;
92         }
93         b.a[count]=sum;
94         count++;
95     }
96     b.len =count++;
97     return in;
98
99 }
100 ostream& operator<<(ostream& out, BigNum& b)    //重载输出运算符
101 {
102     int i;
103     cout << b.a[b.len - 1];
104     for(i = b.len - 2 ; i >= 0 ; i--)
105     {
106         cout.width(DLEN);
107         cout.fill('0');
108         cout << b.a[i];
109     }
110     return out;
111 }
112
113 BigNum BigNum::operator+(const BigNum & T) const    //两个大数之间的相加运算
114 {
115     BigNum t(*this);
116     int i,big;    //位数
117     big = T.len > len ? T.len : len;
118     for(i = 0 ; i < big ; i++)
119     {
120         t.a[i] +=T.a[i];
121         if(t.a[i] > MAXN)
122         {
123             t.a[i + 1]++;
124             t.a[i] -=MAXN+1;
125         }
126     }

```

```

127     if(t.a[big] != 0)
128         t.len = big + 1;
129     else
130         t.len = big;
131     return t;
132 }
133 BigNum BigNum::operator-(const BigNum & T) const    //两个大数之间的相减运算
134 {
135     int i,j,big;
136     bool flag;
137     BigNum t1,t2;
138     if(*this>T)
139     {
140         t1=*this;
141         t2=T;
142         flag=0;
143     }
144     else
145     {
146         t1=T;
147         t2=*this;
148         flag=1;
149     }
150     big=t1.len;
151     for(i = 0 ; i < big ; i++)
152     {
153         if(t1.a[i] < t2.a[i])
154         {
155             j = i + 1;
156             while(t1.a[j] == 0)
157                 j++;
158             t1.a[j--]--;
159             while(j > i)
160                 t1.a[j--] += MAXN;
161             t1.a[i] += MAXN + 1 - t2.a[i];
162         }
163         else
164             t1.a[i] -= t2.a[i];
165     }
166     t1.len = big;
167     while(t1.a[len - 1] == 0 && t1.len > 1)
168     {
169         t1.len--;
170         big--;
171     }
172     if(flag)
173         t1.a[big-1]=0-t1.a[big-1];
174     return t1;
175 }
176
177 BigNum BigNum::operator*(const BigNum & T) const    //两个大数之间的相乘运算
178 {

```

```

179     BigNum ret;
180     int i,j,up;
181     int temp,temp1;
182     for(i = 0 ; i < len ; i++)
183     {
184         up = 0;
185         for(j = 0 ; j < T.len ; j++)
186         {
187             temp = a[i] * T.a[j] + ret.a[i + j] + up;
188             if(temp > MAXN)
189             {
190                 temp1 = temp - temp / (MAXN + 1) * (MAXN + 1);
191                 up = temp / (MAXN + 1);
192                 ret.a[i + j] = temp1;
193             }
194             else
195             {
196                 up = 0;
197                 ret.a[i + j] = temp;
198             }
199         }
200         if(up != 0)
201             ret.a[i + j] = up;
202     }
203     ret.len = i + j;
204     while(ret.a[ret.len - 1] == 0 && ret.len > 1)
205         ret.len--;
206     return ret;
207 }
208 BigNum BigNum::operator/(const int & b) const    //大数对一个整数进行相除运算
209 {
210     BigNum ret;
211     int i,down = 0;
212     for(i = len - 1 ; i >= 0 ; i--)
213     {
214         ret.a[i] = (a[i] + down * (MAXN + 1)) / b;
215         down = a[i] + down * (MAXN + 1) - ret.a[i] * b;
216     }
217     ret.len = len;
218     while(ret.a[ret.len - 1] == 0 && ret.len > 1)
219         ret.len--;
220     return ret;
221 }
222 int BigNum::operator%(const int & b) const    //大数对一个 int 类型的变量进行取模运算
223 {
224     int i,d=0;
225     for (i = len-1; i>=0; i--)
226     {
227         d = ((d * (MAXN+1))% b + a[i])% b;
228     }
229     return d;
230 }

```

```

231 BigNum BigNum::operator^(const int & n) const    //大数的 n 次方运算
232 {
233     BigNum t,ret(1);
234     int i;
235     if(n<0)
236         exit(-1);
237     if(n==0)
238         return 1;
239     if(n==1)
240         return *this;
241     int m=n;
242     while(m>1)
243     {
244         t=*this;
245         for( i=1;i<=1<=m;i<=1)
246         {
247             t=t*t;
248         }
249         m-=i;
250         ret=ret*t;
251         if(m==1)
252             ret=ret*(*this);
253     }
254     return ret;
255 }
256 bool BigNum::operator>(const BigNum & T) const    //大数和另一个大数的大小比较
257 {
258     int ln;
259     if(len > T.len)
260         return true;
261     else if(len == T.len)
262     {
263         ln = len - 1;
264         while(a[ln] == T.a[ln] && ln >= 0)
265             ln--;
266         if(ln >= 0 && a[ln] > T.a[ln])
267             return true;
268         else
269             return false;
270     }
271     else
272         return false;
273 }
274 bool BigNum::operator >(const int & t) const    //大数和一个 int 类型的变量的大小比较
275 {
276     BigNum b(t);
277     return *this>b;
278 }
279
280 void BigNum::print()    //输出大数
281 {
282     int i;

```

```

283     //cout << a[len - 1];
284     printf("%d",a[len-1]);
285     for(i = len - 2 ; i >= 0 ; i--)
286     {
287         /*cout.width(DLEN);
288         cout.fill('0');
289         cout << a[i];*/
290         printf("%04d",a[i]);
291     }
292     //cout << endl;
293     printf("\n");
294 }

```