ICPC 代码模板-V2.0

SDU(QD)　sdu1801

2019 年 11 月 14 日

目录

1 STL 模板库

1.1 pair

```
#include <utility>
pair<double, double> p;
cin >> p.first >> p.second;
p = make_pair(a, b);
```

1.2 set

```
#include <set>
set<int> s;
set<double> ss;
s.begin() // 返回指向第一个元素的迭代器
s.clear() // 清除所有元素
s.count() // 返回某个值元素的个数
s.empty() // 如果集合为空，返回true(真)
s.end() // 返回指向最后一个元素之后的迭代器，不是最后一个元素
s.equal_range() // 返回集合中与给定值相等的上下限的两个迭代器
s.erase() // 删除集合中的元素
s.find() // 返回一个指向被查找到元素的迭代器
s.get_allocator() // 返回集合的分配器
s.insert() // 在集合中插入元素
s.lower_bound() // 返回指向大于（或等于）某值的第一个元素的迭代器
s.key_comp() // 返回一个用于元素间值比较的函数
s.max_size() // 返回集合能容纳的元素的最大限值
s.rbegin() // 返回指向集合中最后一个元素的反向迭代器
s.rend() // 返回指向集合中第一个元素的反向迭代器
s.size() // 集合中元素的数目
s.swap() // 交换两个集合变量
s.upper_bound() // 返回大于某个值元素的迭代器
s.value_comp() // 返回一个用于比较元素间的值的函数
```

在 <set> 头文件中，还定义了另一个非常实用的模版类 multiset（多重集合）。多重集合与集合的区别在于集合中不能存在相同元素，而多重集合中可以存在。
multiset 和 set 的基本操作相似,需要注意的是,集合的 count() 能返回 0(无)或者 1(有),而多重集合是有多少个返回多少个。

```
multiset<int> s;
multiset<double> ss;
```

1.3 vector

```
vector<int> s; //定义一个空的vector对象，存储的是int类型的元素
vector<int> s(n); //定义一个含有n个int元素的vector对象
vector<int> s(first, last); //定义一个vector对象，并从由迭代器 first和last定义的序列[
    first, last)中复制初值
s[i] //直接以下标方式访问容器中的元素
s.front() //返回首元素
s.back() //返回尾元素
s.push_back(x) //向表尾插入元素x
s.size() //返回表长
s.empty() //表为空时，返回真，否则返回假
```

```
s.pop_back() //删除表尾元素
s.begin() //返回指向首元素的随机存取迭代器
s.end() //返回指向尾元素的下一个位置的随机存取迭代器
s.insert(it, val) //向迭代器 it指向的元素前插入新元素val
s.insert(it, n, val) // 向迭代器 it指向的元素前插入n个新元素val s.insert(it, first, last)
//将由迭代器 first和last所指定的序列[first, last)插入到迭代器 it指向的元素前面
s.erase(it) //删除由迭代器 it所指向的元素
s.erase(first, last) //删除由迭代器 first和last所指定的序列[first, last)
s.reserve(n) //预分配缓冲空间，使存储空间至少可容纳n个元素
s.resize(n) //改变序列 长度，超出的元素将会全部被删除，如果序列需要扩展（原空间小于n），元素默认值将填满扩展出的空间
s.resize(n, val) //改变序列 长度，超出的元素将会全部被删除，如果序列需要扩展（原空间小于n），val将填满扩展出的空间
s.clear() //删除容器中的所有元素
s.swap(v) //将s与另一个vector对象进行交换
s.assign(first, last)//将序列替换成由迭代器 first和last所指定的序列[first, last)，[first, last)不能是原序列中的一部分
```

要注意的是，resize 操作和 clear 操作都是对表的有效元素进行的操作，但并不一定会改变缓冲空间的大小
另外，vector 还有其他的一些操作，如反转、取反等，不再一一列举
vector 上还定义了序列之间的比较操作运算符（>、<、>=、<=、==、!=），可以按照字典序比较两个序列。

1.4 bitset

```
const int MAXN = 32;
bitset<MAXN> bt; //bt 包括 MAXN 位，下标 0 ~ MAXN - 1，默认初始化为 0
bitset<MAXN> bt1(0xf); //0xf 表示十六进制数 f，将 bt1 低 4 位初始化为 1
bitset<MAXN> bt2(012); //012 表示八进制数 12，即将 bt2 低 4 位初始化为 1010
bitset<MAXN> bt3("1010"); //将 bt3 低 4 位初始化为 1010
bitset<MAXN> bt4(s, pos, n); //将 01 字符串? s 的 pos 位开始的 n 位初始化 bt4
bt.any() //bt 中是否存在置为 1 的二进制位?
bt.none() //bt 中不存在置为 1 的二进制位吗?
bt.count() //bt 中置为 1的二进制位的个数
bt.size() //bt 中二进制位的个数
bt[pos] //访问 bt 中在 pos 处的二进制位
bt.test(pos) //bt 中在 pos 处的二进制位是否为 1
bt.set() //把 bt 中所有二进制位都置为 1
bt.set(pos) //把 bt 中在 pos 处的二进制位置为 1
bt.reset() //把 bt 中所有二进制位都置为 0
bt.reset(pos) //把 bt 中在pos处的二进制位置为0
bt.flip() //把 bt 中所有二进制位逐位取反
bt.flip(pos) //把 bt 中在 pos 处的二进制位取反
bt[pos].flip() //同上
bt.to_ulong() // 用 bt 中同样的二进制位返回一个 unsigned long 值
os << bt //把 bt 中的位集输出到 os 流
```

1.5 algorithm

库函数列表

表 1: algorithm 函数列表 1

功能	函数名
对序列中的每个元素执行某操作	for_each()
在序列中找出某个值的第一次出现的位置	find()
在序列中找出符合某谓词的第一个元素	find_if()
在序列中找出一子序列的最后一次出现的位置	find_end()
在序列中找出第一次出现指定值集中之值的位置	find_first_of()
在序列中找出相邻的一对值	adjacent_find()
在序列中统计某个值出现的次数	count()
在序列中统计与某谓词匹配的次数	count_if()
找出两个序列相异的第一个元素	mismatch()
两个序列中的对应元素都相同时为真	equal()
在序列中找出一子序列的第一次出现的位置	search()
在序列中找出一值的连续 n 次出现的位置	search_n()
从序列的第一个元素起进行复制	copy()
从序列的最后一个元素起进行复制	copy_backward()
交换两个元素	swap()
交换指定范围的元素	swap_ranges()
交换由迭代器所指的两个元素	iter_swap()
将某操作应用于指定范围的每个元素	transform()
用一个给定值替换一些值	replace()
替换满足谓词的一些元素	replace_if()
复制序列时用一给定值替换元素	replace_copy()
复制序列时替换满足谓词的元素	replace_copy_if()
用一给定值取代所有元素	fill()
用一给定值取代前 n 个元素	fill_n()
用一操作的结果取代所有元素	generate()
用一操作的结果取代前 n 个元素	generate_n()
删除具有给定值的元素	remove()
删除满足谓词的元素	remove_if()
复制序列时删除具有给定值的元素	remove_copy()
复制序列时删除满足谓词的元素	remove_copy_if()
删除相邻的重复元素	unique()
复制序列时删除相邻的重复元素	unique_copy()
反转元素的次序	reverse()
复制序列时反转元素的次序	reverse_copy()
循环移动元素	rotate()
复制序列时循环移动元素	rotate_copy()

表 2: algorithm 函数列表 2

功能	函数名
采用均匀分布来随机移动元素	random_shuffle()
将满足某谓词的元素都放到前面	partition()
将满足某谓词的元素都放到前面并维持原顺序	stable_partition()
以很好的平均效率排序	sort()
排序，并维持相同元素的原有顺序	stable_sort()
将序列的前一部分排好序	partial_sort()
复制的同时将序列的前一部分排好序	partial_sort_copy()
将第 n 各元素放到它的正确位置	nth_element()
找到大于等于某值的第一次出现	lower_bound()
找到大于某值的第一次出现	upper_bound()
找到（在不破坏顺序的前提下）可插入给定值的最大范围	equal_range()
在有序序列中确定给定元素是否存在	binary_search()
归并两个有序序列	merge()
归并两个接续的有序序列	inplace_merge()
一序列为另一序列的子序列时为真	includes()
构造两个集合的有序并集	set_union()
构造两个集合的有序交集	set_intersection()
构造两个集合的有序差集	set_difference()
构造两个集合的有序对称差集（并-交）	set_symmetric_difference()
向堆中加入元素	push_heap()
从堆中弹出元素	pop_heap()
从序列构造堆	make_heap()
给堆排序	sort_heap()
两个值中较小的	min()
两个值中较大的	max()
序列中的最小元素	min_element()
序列中的最大元素	max_element()
两个序列按字典序的第一个在前	lexicographical_compare()
按字典序的下一个排列	next_permutation()
按字典序的前一个排列	prev_permutation()

2 数学

2.1 线性筛法

2.1.1 欧拉函数

```
typedef long long ll;
const ll N = 10000000;
bool is[N+10];
int prime[N+10], cnt;
int phi[N+10];
ll n, ans;
void init(){
    phi[1] = 1;
    is[1] = is[0] = 1;
    for(int i = 2; i <= N; i ++){
        if(!is[i]) prime[++cnt] = i, phi[i] = prime[cnt]-1;
        for(int j = 1; j <= cnt; j ++){
            if(i*prime[j] >= N) break;
            is[i*prime[j]] = 1;
            if(i % prime[j] == 0){
                phi[i*prime[j]] = phi[i] * prime[j];
                break;
            }else phi[i*prime[j]] = phi[i] * (prime[j] - 1);
        }
    }
}
```

2.1.2 莫比乌斯函数

```
typedef long long ll;
const ll N = 10000000;
bool is[N+10];
int prime[N+10], cnt;
int miu[N+10];
ll n, ans;
void init(){
    miu[1] = 1;
    is[1] = is[0] = 1;
    for(int i = 2; i <= N; i ++){
        if(!is[i]) prime[++cnt] = i, miu[i] = -1;
        for(int j = 1; j <= cnt; j ++){
            if(i*prime[j] >= N) break;
            is[i*prime[j]] = 1;
            if(i % prime[j] == 0){
                miu[i*prime[j]] = 0;
                break;
            }else miu[i*prime[j]] = - miu[i];
        }
    }
}
```

2.1.3 约数个数

```cpp
//d 约数个数      num 最小素因子的个数
typedef long long ll;
const ll N = 10000000;
bool is[N+10];
int prime[N+10], cnt;
int d[N+10], num[N+10];
ll n, ans;
void init(){
    d[1] = 1,num[1] = 0;
    is[1] = is[0] = 1;
    for(int i = 2; i <= N; i ++){
        if(!is[i]) prime[++cnt] = i, d[i] = 2, num[i] = 1;
        for(int j = 1; j <= cnt; j ++){
            if(i*prime[j] >= N) break;
            is[i*prime[j]] = 1;
            if(i % prime[j] == 0){
                num[i*prime[j]] = num[i] + 1;
                d[i*prime[j]] = d[i] / (num[i] + 1) * (num[i*prime[j]]+1);
                break;
            }else d[i*prime[j]] = d[i] * d[prime[j]], num[i*prime[j]] = 1;
        }
    }
}
```

2.1.4 约数和

```cpp
//sd 约数和  sp      表示从1到最小素因子的最大幂次的和
typedef long long ll;
const ll N = 10000000;
bool is[N+10];
int prime[N+10], cnt;
int sd[N+10], sp[N+10];
ll n, ans;
void init(){
    sp[1] = 0, sd[1] = 1;
    is[1] = is[0] = 1;
    for(int i = 2; i <= N; i ++){
        if(!is[i]) prime[++cnt] = i, sd[i] = i + 1, sp[i] = i + 1;
        for(int j = 1; j <= cnt; j ++){
            if(i*prime[j] >= N) break;
            is[i*prime[j]] = 1;
            if(i % prime[j] == 0){
                sp[i*prim[j]] = sp[i] * prim[j] + 1;
                sd[i*prim[j]] = sd[i] / sp[i] * sp[i*prim[j]];
                break;
            }else sd[i*prim[j]] = sd[i] * sd[prim[j]], sp[i*prim[j]] = 1 + prim[j];
        }
    }
}
```

2.2 求单个欧拉函数

```cpp
typedef long long ll;
```

```
bool is[100010];
ll prime[100010], cnt;
ll n;
ll phi(ll x){
    ll ans = x, i;
    for(i = 1; prime[i] <= x && i <= cnt; i ++){
        if(x % prime[i] == 0) ans = ans/prime[i]*(prime[i]-1);
        while(x % prime[i] == 0) x /= prime[i];
    }
    if(x != 1) ans = ans/x*(x-1);
    return ans;
}
void init(){
    is[1] = is[0] = 1;
    for(int i = 2; i <= 100000; i ++){
        if(!is[i]) prime[++cnt] = i;
        for(int j = 1; j <= cnt; j ++){
            if(i*prime[j] >= 100000) break;
            is[i*prime[j]] = 1;
            if(i % prime[j] == 0) break;
        }
    }
}
```

2.3 拓展欧几里得（同余方程/逆元）

```
int exgcd(int a, int b, int &x, int &y){
    if(b == 0){
        x = 1, y = 0;
        return a;
    }
    int GCD = exgcd(b, a%b, x, y);
    int xx = x;
    x = y, y = xx - a/b * y;
    return GCD;
}
//时间复杂度：O(log n)
```

2.3.1 求乘法逆元

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

当 $gcd(a, b) = 1$ 时 (a,b 互质)，在 $\pmod b$ 的同余系下，有：

$$ax + by \equiv 1 \pmod b$$

$$ax \equiv 1 \pmod b$$

所以当 $gcd(a, b) = 1$ 时，a 存在 $\pmod b$ 意义下的乘法逆元，$a^{-1} = x$。

2.3.2 欧拉定理求逆元

若 n, a 为正整数，且 n, a 互质，则:

$$a^{\phi(n)} \equiv 1 (mod\ n)$$

逆元即为：

$$a^{\phi(n)-1}$$

2.3.3 求解线性同余方程

线性同余方程是最基本的同余方程，“线性”表示方程的未知数次数是一次，即形如：

$$ax \equiv b \pmod p$$

此方程有解当且仅当 b 能够被 a 与 p 的最大公约数整除。这时，如果 x_0 是方程的一个解，那么所有的解可以表示为：

$$\{x_0 + k\frac{p}{d} | (k\ z)\}$$

其中 d 是 a 与 p 的最大公约数。在模 p 的完全剩余系 $0, 1, ..., p-1$ 中，恰有 d 个解。注意：算完记得判断答案是否合法，不合法用上式加到合法。假如 $gcd(a, p) = d$，我们可以把 a, b, p 同除以 d，使得 $gcd(a, p) = 1$。

$$ax \equiv b \pmod p$$

$$x \equiv ba^{-1} \pmod p$$

2.3.4 拓展欧几里得算法推导

不妨设 $a > b, A > B$ 因为 $ax + by = gcd(a, b)$，$gcd(a, b) = gcd(b, a\%b)$ 考虑递归的过程，令 $a = b\ b = a\%b$ 所以

$$bx' + (a\%b)y' = gcd(b, a\%b) = gcd(a, b) = ax + by$$

即：

$$bx' + (a\%b)y' = ax + by$$
$$bx' + (a - \lfloor\frac{a}{b}\rfloor b)y' = ax + by$$
$$bx' + ay' - \lfloor\frac{a}{b}\rfloor by' = ax + by$$
$$ay' + b(x' - \lfloor\frac{a}{b}\rfloor y') = ax + by$$

解得：

$$x = y' \quad y = x' - \lfloor\frac{a}{b}\rfloor y'$$

2.4 高斯消元

```cpp
inline int dcmp(double x){
    if(fabs(x) <= eps) return 0;
    return x < 0 ? -1 : 1;
}

struct mr{
    const static int sz = 501;
    double d[sz][sz];
    int sx, sy;
    mr(int x, int y){memset(d, 0, sizeof(d)), sx = x, sy = y;}
    void init(int *a, int x, int y){
        if(x > sz || y > sz) exit(-1);
        else sx = x, sy = y;
        for(int i = 0; i < x*y; i ++) d[i/x][i%y] = a[i];
    }
    bool Gaussian(){
        for(int i = 0; i < sx; i ++){
            if(dcmp(d[i][i]) == 0){
                int j = i+1;
                while(j < sx && dcmp(d[j][i]) == 0) j ++;
                if(j >= sx) return false;
                else swap(d[i], d[j]);
            }
            for(int j = i+1; j < sx; j ++){
                double v = d[j][i]/d[i][i];
                for(int k = i; k < sy; k ++) d[j][k] -= v * d[i][k];
            }
        }
        for(int i = sx-1; i >= 0; i --){
            if(d[i][i] == 0) return false;
            d[i][sy-1] /= d[i][i], d[i][i] = 1;
            for(int j = i-1; j >= 0; j --) d[j][sy-1] -= d[j][i] * d[i][sy-1], d[j][i] = 0;
        }
        return true;
    }
};
```

2.5 中国剩余定理（拓展）

2.5.1 中国剩余定理

中国剩余定理可以求出以下的一元线性同余方程组中 x 的一个解：

$$(S):\begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \\ \quad\vdots \\ x \equiv a_n \pmod{m_n} \end{cases}$$

假设整数 $m_1, m_2, ..., m_n$ 两两互质，则对任意的整数：$a_1, a_2, ..., a_n$，方程组 (S) 有解，并且通解可以用如下方式构造得到：设 $M = \prod_{i=1}^{n} m_i$，并设 $M_i = \frac{M}{m_i}$，$t_i = M_i^{-1}\ (\mathrm{mod}\ m)_i$ 即：

$$x \equiv \sum_{i=1}^{n} a_i t_i M_i \pmod{M}$$

2.5.2 拓展中国剩余定理

当 $gcd(m_1, m_2) \neq 1$ 时，考虑合并两个方程：

$$x \equiv c_1 \pmod{m_1}$$

$$x \equiv c_2 \pmod{m_2}$$

显然有 $x = c_1 + k_1 m_1, x = c_2 + k_2 m_2$, 即 $c_1 + k_1 m_1 = c_2 + k_2 m_2$ 移项得 $k_1 m_1 = k_2 m_2 + c_2 - c_1$ 我们令 $t = gcd(m_1, m_2)$, 则有 $\frac{m_1}{t} k_1 = \frac{m_2}{t} k_2 + \frac{c_2 - c_1}{t}$ 当且仅当 $t | c_2 - c_1$ 时可以合并此时有

$$\frac{m_1}{t} k_1 \equiv \frac{c_2 - c_1}{t} (mod\ \frac{m_2}{t})$$

使用扩展欧几里得求得 k_1 的通解：

$$k_1 = ans + k \times \frac{m_2}{t}\ (k \in \mathbb{Z})$$

将 k_1 代回第一个方程得:

$$x \equiv ans \times m_1 + k \times \frac{m_1 \times m_2}{t} + c_1$$

最终再变换成同余方程的形式：

$$x \equiv ans \times m_1 + c_1 (mod\ \frac{m_1 \times m_2}{t})$$

这是一个同余方程的形式，如此不断合并所有方程即可。

```
#include<iostream>
#include<cstdio>
#define LL long long
using namespace std;
const LL MAXN = 1e6 + 10;
LL K, C[MAXN], M[MAXN], x, y;
```

```cpp
inline LL ksc(unsigned LL x,unsigned LL y, LL p){return ((LL)(x*y-(unsigned LL)((long double)x/p*y)*p)%p+p)%p;}
LL gcd(LL a, LL b) {
    return b == 0 ? a : gcd(b, a % b);
}
LL exgcd(LL a, LL b, LL &x, LL &y) {
    if (b == 0) {x = 1, y = 0; return a;}
    LL r = exgcd(b, a % b, x, y), tmp;
    tmp = x; x = y; y = tmp - (a / b) * y;
    return r;
}
LL inv(LL a, LL b) {
    LL r = exgcd(a, b, x, y);
    while (x < 0) x += b;
    return x;
}
//x%m=c
//最后X%M[n]=C[n]
int main() {
    cin>>K;
    for (LL i = 1; i <= K; i++) scanf("%lld%lld", &M[i], &C[i]);
    bool flag = 1;
    for (LL i = 2; i <= K; i++) {
        LL M1 = M[i - 1], M2 = M[i], C2 = C[i], C1 = C[i - 1], T = gcd(M1, M2);
        if ((C2 - C1) % T != 0) {flag = 0; break;}
        M[i] = (M1 /T * M2);
        C[i] = ksc(inv( M1 / T , M2 / T ),((C2 - C1) / T%(M2 / T)+(M2 / T))%(M2 / T), (M2 / T))  * M1 + C1;
        C[i] = (C[i] % M[i] + M[i]) % M[i];
    }
    printf("%lld\n", flag ? C[K] : -1);
    return 0;
}
```

2.6 FFT

```cpp
struct Com{
    double x, y;
    Com(double _x = 0, double _y = 0){x = _x, y = _y;}
    Com operator + (const Com &t)const{return Com(x+t.x, y+t.y);}
    Com operator - (const Com &t)const{return Com(x-t.x, y-t.y);}
    Com operator * (const Com &t)const{return Com(x*t.x-y*t.y, x*t.y+y*t.x);}
};
const int maxn = 300010;
const double pi = acos(-1);
int BIT, R[maxn];
void fft(Com *a, int n, int inv){
    if(R[n-1] != n-1) // 分治FFT把这个if去掉
        for(int i = 0; i < n; i ++) R[i] = (R[i>>1]>>1) | ((i&1)<<(BIT-1));
    for(int i = 0; i < n; i ++) if(i < R[i]) std::swap(a[i], a[R[i]]);
    for(int i = 1; i < n; i <<= 1){
        Com mi(cos(pi/i), sin(pi/i) * inv);
        for(int j = 0; j < n; j += (i<<1)){
            Com x(1, 0);
```

```
                for(int k = 0; k < i; k ++, x = x * mi){
                    Com t1 = a[j+k], t2 = x * a[j+k+i];
                    a[j+k] = t1 + t2, a[j+k+i] = t1 - t2;
                }
            }
        }
}
Com a[maxn], b[maxn];
char c1[maxn], c2[maxn];
int res[maxn];
int main(){
    int n, m;
    scanf("%s%s", c1, c2);
    n = strlen(c1)-1, m = strlen(c2)-1;
    for(int i = 0; i <= n; i ++) a[i].x = c1[i]-'0';
    for(int i = 0; i <= m; i ++) b[i].x = c2[i]-'0';
    m += n;
    for(n = 1; n <= m; n <<= 1, BIT ++);
    fft(a, n, 1), fft(b, n, 1);
    for(int i = 0; i < n; i ++) a[i] = a[i] * b[i];
    fft(a, n, -1);
    for(int i = 0; i <= m; i ++) res[m-i] = (long long)(a[i].x/n+0.5);
    for(int i = 0; i < m; i ++) res[i+1] += res[i] / 10, res[i] %= 10;
    while(res[m] >= 10) res[m+1] += res[m] / 10, res[m ++] %= 10;
    for(int i = m; i >= 0; i --) printf("%d", res[i]);
    puts("");
    return 0;
}
```

2.7 NTT

2.7.1 代码实现

```
// g 为原根, inv=1正向, inv=-1逆向
void ntt(ll *a, int n, int inv){
    if(R[n-1] != n-1) // 分治NTT把这个if去掉
        for(int i = 0; i < n; i ++) R[i] = (R[i>>1]>>1) | ((i&1) << (BIT-1));
    for(int i = 0; i < n; i ++) if(i < R[i]) swap(a[i], a[R[i]]);
    for(int i = 1; i < n; i <<= 1){
        ll mi = inv == 1 ? pow(g, (mod-1)/(2*i)) : ni(pow(g, (mod-1)/(2*i)));
        for(int j = 0; j < n; j += (i<<1)){
            ll x = 1;
            for(int k = 0; k < i; k ++, x = (x * mi) % mod){
                ll t1 = a[j+k], t2 = (x * a[j+k+i]) % mod;
                a[j+k] = (t1 + t2) % mod;
                a[j+k+i] = (t1 - t2) % mod;
            }
        }
    }
}
```

2.7.2 常用 NTT 模数

g 是 $(\bmod\ r * 2^k + 1)$ 的原根

表 3: 常用 NTT 模数

素数	r	k	g
3	1	1	2
5	1	2	2
17	1	4	3
97	3	5	5
193	3	6	5
257	1	8	3
7681	15	9	17
12289	3	12	11
40961	5	13	3
65537	1	16	3
786433	3	18	10
5767169	11	19	3
7340033	7	20	3
23068673	11	21	3
104857601	25	22	3
167772161	5	25	3
469762049	7	26	3
1004535809	479	21	3
2013265921	15	27	31
2281701377	17	27	3
3221225473	3	30	5
75161927681	35	31	3
77309411329	9	33	7
206158430209	3	36	22
2061584302081	15	37	7
2748779069441	5	39	3
6597069766657	3	41	5
39582418599937	9	42	5
79164837199873	9	43	5
263882790666241	15	44	7
1231453023109121	35	45	3
1337006139375617	19	46	3
3799912185593857	27	47	5
4222124650659841	15	48	19
7881299347898369	7	50	6
31525197391593473	7	52	3
180143985094819841	5	55	6
1945555039024054273	27	56	5
4179340454199820289	29	57	3

2.8 FWT

2.8.1 代码实现

```
#include<iostream>
#define LL long long
using namespace std;
const int MOD=998244353;
const int inv2=(MOD+1)/2;//逆元
void FWT_or(LL *a,int N,int opt){//数组(from 0) 数组长度 1 or -1
    for(int i=1;i<N;i<<=1)
        for(int p=i<<1,j=0;j<N;j+=p)
            for(int k=0;k<i;++k)
                if(opt==1)a[i+j+k]=(a[j+k]+a[i+j+k])%MOD;
                else a[i+j+k]=(a[i+j+k]+MOD-a[j+k])%MOD;
}
void FWT_and(LL *a,int N,int opt){
    for(int i=1;i<N;i<<=1)
        for(int p=i<<1,j=0;j<N;j+=p)
            for(int k=0;k<i;++k)
                if(opt==1)a[j+k]=(a[j+k]+a[i+j+k])%MOD;
                else a[j+k]=(a[j+k]+MOD-a[i+j+k])%MOD;
}
void FWT_xor(LL *a,int N,int opt){
    for(int i=1;i<N;i<<=1)
        for(int p=i<<1,j=0;j<N;j+=p)
            for(int k=0;k<i;++k)
            {
                LL X=a[j+k],Y=a[i+j+k];
                a[j+k]=(X+Y)%MOD;a[i+j+k]=(X+MOD-Y)%MOD;
                if(opt==-1)a[j+k]=1ll*a[j+k]*inv2%MOD,a[i+j+k]=1ll*a[i+j+k]*inv2%MOD;
            }
}
LL A[(1<<17)|1],B[(1<<17)|1];
LL a[(1<<17)|1],b[(1<<17)|1];
int main(){
    int n;cin>>n;
    int N=(1<<n);
    for(int i=0;i<N;i++)scanf("%lld",&A[i]);
    for(int i=0;i<N;i++)scanf("%lld",&B[i]);

    for(int i=0;i<N;i++)a[i]=A[i],b[i]=B[i];
    FWT_or(a,N,1),FWT_or(b,N,1);
    for(int i=0;i<N;i++)a[i]=a[i]*b[i]%MOD;
    FWT_or(a,N,-1);
    for(int i=0;i<N;i++)printf("%lld%c",a[i]," \n"[i+1==N]);

    for(int i=0;i<N;i++)a[i]=A[i],b[i]=B[i];
    FWT_and(a,N,1),FWT_and(b,N,1);
    for(int i=0;i<N;i++)a[i]=a[i]*b[i]%MOD;
    FWT_and(a,N,-1);
    for(int i=0;i<N;i++)printf("%lld%c",a[i]," \n"[i+1==N]);

    for(int i=0;i<N;i++)a[i]=A[i],b[i]=B[i];
    FWT_xor(a,N,1),FWT_xor(b,N,1);
    for(int i=0;i<N;i++)a[i]=a[i]*b[i]%MOD;
    FWT_xor(a,N,-1);
    for(int i=0;i<N;i++)printf("%lld%c",a[i]," \n"[i+1==N]);
    return 0;
}
```

2.9 博弈游戏

2.9.1 博弈问题综述

所讨论的博弈问题满足以下条件：

1. 玩家只有两个人，轮流做出决策，且操作公平。
2. 游戏的状态集有限，保证游戏在有限步后结束.
3. 博弈双方都使用最优决策。

一个游戏有两个状态，我们称为必胜态和必败态，他们的关系：

- 必胜态表示，从当前状态可以转移到一个必败态。
- 必败态表示，从当前状态无法转移到一个必败态。

也就是说，一个状态不是必胜态，就是必败态。

2.9.2 Bash 博弈（同余理论）

问题：一堆 n 个物品，两个人从轮流中取出 1 到 m 个，无法取者失败。

解法：当 $(m+1)|n$ 时，先手必败，其余情况，先手必胜。当剩余的物品数量为 $(m+1)$ 的整数倍时，为必败态，此时无论先手取多少个，后手只需要使两次取的物品数的总和是 $(m+1)$，下一个状态依然处于必败态。

2.9.3 Nim 博弈（异或理论）

问题：n 堆物品，每堆有 a_i 个，每次在某一堆中操作，取出至少 1 个物品，无法操作者失败。

解法：定义 $S=\oplus_{i=0}^{n}a_i$ 为 n 堆石子的异或和，则当 $S=0$ 时，先手必败，其余情况先手必胜。原理为对于一个异或和为 0 的状态，只要对其中一个数的任意一个二进制位进行改动，其异或和一定会发生变化，变为非 0 值。此时对于异或和非 0 的局面，后手一定可以找到现在异或和中非 0 的最高位，既然次位的异或和为 1，那么在所有的堆中, 至少有一堆该位二进制值为 1。我们选定此数进行减值，将该位改为 0（这时，后面的位无论是什么，都一定比原来小)，低于该位的位数按照异或和进行构造，使新的异或和为 0，此时的下一个状态依然是必败态。

2.9.4 Wythoff 博弈

问题：有两堆石子，个数为 x,y（不妨设 $x\le y$)，每次每个人可以从任意一堆石子中取任意多的石子或者从两堆石子中取同样多的石子，不能取得人输。

解法：当 $\lfloor\frac{(y-x)(\sqrt{5}+1)}{2}\rfloor=x$ 时先手必败，否则先手必胜。该题可以转化为：在一个仅有第一象限的棋盘上的 (x,y) 处有一个棋子，该棋子只能向两个坐标减小的方向移动。具体地，该棋子可以向平行于 x,y 轴的方向走任意多步，也可以向平行于 $y=x$ 的直线方向走任意多步，不能移动棋子的一方失败。通过在棋盘划出必胜位置线，剩余的位置即为必败位置。

2.9.5 Fibonacci 博弈

问题：一堆石子有 n 个，两人轮流取，先取者第 1 次可以取任意多个，但不能全部取完，以后每次取的石子数不能超过上次取子数的 2 倍，不能取者输。

解法：当 n 是斐波那契数的时候，先手必败，否则先手必胜。

2.9.6 SG 函数（异或理论扩展）

$$SG[x]=\text{mex}\{SG[y]\}$$

其中 y 是 x 所有的后继状态（通过操作改变当前状态为后继态)，mex 是最小的没有出现的连续自然数。

游戏的和 游戏 $x_1, x_2, ..., x_n$ 的和为 x，且 $x = \{x_1, x_2, ..., x_n\}$。理解为 x 是由 $x_1, x_2, ..., x_n$ 构成的新游戏。

SG 定理 如果一个游戏的所有子游戏相互独立（即一个子游戏的操作不会影响其他的子游戏），那么

$$SG[x] = SG[x_1] \text{ xor } SG[x_2] \text{ xor } ... \text{ xor } SG[x_n]$$

文字表述为，一些游戏的和的 SG 函数等于各个游戏 SG 函数的异或和。

原理 **SG** 函数本质上是把博弈问题转化为 **Nim** 博弈进行求解，一个子游戏相当于一堆石子，SG 函数的值即为该堆石子的个数，整个游戏的 SG 值为所有子游戏的异或和，即 Nim 博弈中，将每堆石子个数异或起来。若当前局面为必败态（SG 值为 0），改变整个游戏中任意一个子游戏到其后继状态时，所有子游戏 SG 函数值的异或和必然发生变化，取次值二进制的最高位，所有子游戏中的 SG 函数值二进制中，至少有一个数的二进制该位为 1，找到次数所表示的子游戏。该子游戏的 SG 值，是对该子游戏所有后继状态的 SG 值取 mex，这就意味着，我们可以通过执行当前子游戏的某个后继状态，把当前子游戏的 SG 值变为小于该值的任意一个自然数。这相当于 Nim 博弈中取走一些石子做减法。通过这样的变化，后手可以把当前游戏的 SG 函数值再次变为 0，此时下一个状态依旧必败。

2.10 线性预处理逆元

设 P 为题目中的模数。

$$P = \lfloor \frac{P}{i} \rfloor \times i + P\%i$$

一下运算在 $\%P$ 的同余系下进行。

$$-P\%i = \lfloor \frac{P}{i} \rfloor \times i$$

$$-P\%i \times (i)^{-1} \times (P\%i)^{-1} = \lfloor \frac{P}{i} \rfloor \times i \times (i)^{-1} \times (P\%i)^{-1}$$

$$-(i)^{-1} = \lfloor \frac{P}{i} \rfloor \times (P\%i)^{-1}$$

$$(i)^{-1} = -\lfloor \frac{P}{i} \rfloor \times (P\%i)^{-1}$$

因为 $P\%i < i$ 这个数的逆元已经求得，所以借此公式就可以线性处理逆元了。

```
ni[i]=-(p/i)*ni[p%i];
```

2.11 Miller-Rabin 素数判定

```

#define ll long long
#define ull unsigned long long
#define ld long double
#define inl inline
using namespace std;
inl ll ksc(ull x, ull y, ll p) { return (x * y - (ull) ((ld) x / p * y) * p + p) % p;
    }
inl ll ksm(ll x, ll y, ll p) {
    ll res = 1;
    for (; y; y >>= 1, x = ksc(x, x, p))if (y & 1)res = ksc(res, x, p);
    return res;
}
inl bool mr(ll x, ll p) {
    if (ksm(x, p - 1, p) != 1)return 0;
    ll y = p - 1, z;
    while (!(y & 1)) {
        y >>= 1;
        z = ksm(x, y, p);
        if (z != 1 && z != p - 1)return 0;
        if (z == p - 1)return 1;
    }
    return 1;
}
int te_pri[20] = {0, 2, 3, 5, 7, 43, 61, 24251}, te_num = 7;
inl bool isprime(ll x) {
    if (x < 2)return 0;
    for (int i = 1; i <= te_num; i++)
        if (x == te_pri[i])return 1;
    for (int i = 1; i <= te_num; i++)
        if (!(x % te_pri[i]) || !mr(te_pri[i], x))return 0;
    return 1;
}

int main() {
    return 0;
}
```

2.12 Pollard-Rho 算法 (大数质因数分解)

```

#define ll long long
#define ull unsigned long long
#define ld long double
#define inl inline
#define re register
using namespace std;

inl ll ksc(ull x, ull y, ll p) { return (x * y - (ull) ((ld) x / p * y) * p + p) % p;
    }
inl ll ksm(ll x, ll y, ll p) {
    ll res = 1;
    for (; y; y >>= 1, x = ksc(x, x, p))if (y & 1)res = ksc(res, x, p);
    return res;
}
inl ll Abs(ll x) { return x < 0 ? -x : x; }
inl ll gcd(ll x, ll y) {
    ll z;
    while (y)  z = x, x = y, y = z % y;
    return x;
}
inl bool mr(ll x, ll p) {
    if (ksm(x, p - 1, p) != 1)return 0;
    ll y = p - 1, z;
    while (!(y & 1)) {
        y >>= 1;
        z = ksm(x, y, p);
        if (z != 1 && z != p - 1)return 0;
        if (z == p - 1)return 1;
    }
    return 1;
}
ll te_pri[20] = {0, 2, 3, 5, 7, 43, 61, 24251};
int te_num = 7;
inl bool isprime(ll x) {
    if (x < 2)return 0;
    for (int i = 1; i <= te_num; i++)
        if (x == te_pri[i])return 1;
    for (int i = 1; i <= te_num; i++)
        if (!(x % te_pri[i]) || !mr(te_pri[i], x))return 0;
    return 1;
}
inl ll rho(ll p) {
    ll x, y, z, c, g;
    re int i, j;
    while (1) {
        y = x = rand() % p;
        z = 1, c = rand() % p;
        i = 0, j = 1;
        while (++i) {
            x = (ksc(x, x, p) + c) % p;
            z = ksc(z, Abs(y - x), p);
            if (x == y || !z)break;
            if (!(i % 127) || i == j) {
                g = gcd(z, p);
                if (g > 1)return g;
                if (i == j)y = x, j <<= 1;
            }
        }
    }
}
ll ys[1010];//最后的质数分解在ys中
int ind;//每次重新使用ind记得清零
inl void prho(ll p) {
    if (p == 1)return;
    if (isprime(p)) {
        ys[++ind] = p;
        return;
    }
    ll pi = rho(p);
    while (p % pi == 0)p /= pi;
    prho(pi);
    prho(p);
}
ll x, pos;
int main() {
    srand(time(0));
    cin >> x;
    prho(x);
    sort(ys + 1, ys + ind + 1);
    int m = unique(ys + 1, ys + ind + 1) - ys - 1;
    return 0;
}
```

2.13 组合数学相关

2.13.1 相关定理

定理 1 $\{1, 2, ...n\}$ 的 r 组合 $a_1, a_2, ..., a_r$ 出现在所有 r 组合中的字典序位置编号：

$$index = C(n, r) - C(n - a_1, r) - C(n - a_2, r - 1) - ... - C(n - a_r, 1)$$

定理 2

$$k*C(n,k)=n*C(n-1,k-1)$$

$$C(n,0)+C(n,2)+...=C(n,1)+C(n,3)+...$$

$$1*C(n,1)+2*C(n,2)+...+n*C(n,n)=n*2^{n-1}$$

2.13.2 错位排列

$$D_n=n!\sum_{k=0}^{n}\frac{(-1)^k}{k!}$$

2.13.3 Catalan 数

凸多边形三角划分,n 个节点组成二叉搜索树，n 对括号正确匹配数目，1-n 的出栈序列

$$C_n=\frac{C(2*n,n)}{n+1}$$

$$C_n=\frac{(4*n-2)}{n+1}*C_n-1$$
$$C_1=1$$

2.13.4 Stirling 数

第一类 Stirling 数 s(p, k) 将 p 个不同元素构成 k 个圆排列的数目。

$$s(p,k)=(p-1)*s(p-1,k)+s(p-1,k-1)$$

第二类 Stirling 数 n 个元素拆分 k 个集合的方案数记为 S(n, k)。

$$S(p,k)=k*S(p-1,k)+S(p-1,k-1)$$

$$S(p,0)=0,(p>=1)$$

$$S(p,p)=1,(p>=0)$$

$$S(p,1)=1,(p>=1)$$

$$S(p,2)=2^{p-1}-1,(p>=2)$$

$$S(p,p-1)=C(p,2)$$

2.13.5 Bell 数

元素个数为 n 的集合的划分数目。

$$B_p=\sum_{i=0}^{p}S(p,i)$$

其中 $S(p,i)$ 表示第二类 Stirling 数。

$$B_p=C(p-1,0)*B_0+C(p-1,1)*B_1+...+C(p-1,p-1)*B_{p-1}$$

2.13.6 卢卡斯定理

若 p 是质数：

$$C(n,m)\%p=C(n/p,m/p)*C(n\%p,m\%p)\%p$$

```
#include<iostream>
#define LL long long
using namespace std;
LL n,m;
LL fac[100100],mod;
LL inv(LL i){
    LL res=1,j=mod-2;
    for(;j;j>>=1,i=i*i%mod) if(j&1)res=res*i%mod;
    return res;
}
LL C(LL a,LL b){return a>b?0:fac[b]*inv(fac[a]*fac[b-a]%mod)%mod;}
//b个里面选a个
LL lucas(LL a,LL b){return a?C(a%mod,b%mod)*lucas(a/mod,b/mod)%mod:1;}
int main(){
    int T;cin>>T;
    fac[0]=1;
    while(T--){
        cin>>n>>m>>mod;
        for(int i=1;i<mod;i++)fac[i]=fac[i-1]*i%mod;
        cout<<lucas(m,n+m)<<endl;
    }
    return 0;
}
```

若 p 不是质数，令 $p = p_1 * p_2 * ... * p_n$，这里 p_i 是质数

$$\begin{cases} C_n^m \equiv a_1 \pmod{p_1} \\ C_n^m \equiv a_2 \pmod{p_2} \\ \quad\vdots \\ C_n^m \equiv a_n \pmod{p_n} \end{cases}$$

则分别求出 $a_1, a_2, ..., a_n$ 用中国剩余定理合并即可。

2.14 蔡勒公式（日期转星期）

0-星期日，1-星期一，2-星期二，3-星期三，4-星期四，5-星期五，6-星期六

```
int Change(int year, int month, int day){
    if(month == 1 || month == 2) month += 12, year --;
    int c = year/100, y = year%100, m = month, d = day;
    int W = c/4 - 2*c + y + y/4 + 26*(m+1)/10 + d - 1;
    if(W < 0) return (W + (-W/7 + 1)*7)%7;
    return W % 7;
}
```

2.15 二次剩余

```
#define LL long long
using namespace std;
LL mod = 1e9 + 7, I_mul;

struct com {
    LL a, b;
    com(LL a = 0, LL b = 0) : a(a), b(b) {}
    bool operator==(com y) { return a == y.a && b == y.b; }
    com operator*(com y) { return com((a * y.a + I_mul * b % mod * y.b) % mod, (b * y.
        a + a * y.b) % mod); }
    com operator^(LL k) {
        com res(1, 0);
        com x(a, b);
        for (; k; x = x * x, k >>= 1) if (k & 1)res = res * x;
        return res;
    }
};
//判断一个数x是否有二次剩余
bool check_if_residue(LL x, LL M) { return !x || (com(x, 0) ^ ((M - 1) >> 1)) == com
    (1, 0); }
//二次剩余的两个解放在x0和x1中
void solve(LL n, LL p, LL &x0, LL &x1) {
    if (!n) {
        x0 = x1 = 0;
        return;
    }
    mod = p;
    LL g = rand() % mod;
    while (!g || check_if_residue((g * g + mod - n) % mod, mod))g = rand() % mod;
    I_mul = (g * g + mod - n) % mod;
    x0 = (LL) (com(g, 1) ^ ((mod + 1) >> 1)).a;
    x1 = mod - x0;
}
LL ans1, ans2;
LL p;
int main() {
    int T;
    cin >> T;
    while (T--) {
        scanf("%lld%lld", &p, &mod);
        if (!check_if_residue(p, mod)) {
            puts("Hola!");
            continue;
        }
        solve(p, mod, ans1, ans2);
        if (ans1 == ans2)printf("%lld\n", ans1);
        else {
            if (ans1 > ans2)swap(ans1, ans2);
            printf("%lld␣%lld\n", ans1, ans2);
        }

    }
}
```

2.16 BSGS/EX-BSGS

处理最小化 X 使 $Y^X \equiv Z \pmod{P}$

```

//A^X=B(mod C) 求X C为整数
#define MOD 76543
int hs[MOD], head[MOD], next[MOD], id[MOD], top;

void insert(int x, int y) {
    int k = x % MOD;
    hs[top] = x, id[top] = y, next[top] = head[k], head[k] = top++;
}

int find(int x) {
    int k = x % MOD;
    for (int i = head[k]; i != -1; i = next[i])
```

```
        if (hs[i] == x)
            return id[i];
    return -1;
}

int BSGS(int a, int b, int c) {
    memset(head, -1, sizeof(head));
    top = 1;
    if (b == 1)
        return 0;
    int m = sqrt(c * 1.0), j;
    long long x = 1, p = 1;
    for (int i = 0; i < m; ++i, p = p * a % c)
        insert(p * b % c, i);//存的是(a^j*b, j)
    for (long long i = m;; i += m) {
        if ((j = find(x = x * p % c)) != -1)
            return i - j;  //a^(ms-j)=b(mod c)
        if (i > c)
            break;
    }
    return -1;
}

//EXBSGS
LL xiao(LL &a, LL &b, LL &c) {
    LL r = 0, d, x, y, a1 = 1;
    while ((d = extend_Euclid(a, c, x, y)) != 1) {
        if (b % d)
            return -1;
        c /= d;
        b /= d;
        a1 = a1 * (a / d) % c;
        r++;
    }
    if (r == 0)
        return 0;
    extend_Euclid(a1, c, x, y);
    b = (b * x % c + c) % c;
    return r;
}

LL extend_BSGS(LL a, LL b, LL c) {
    a %= c;    //简化运算
    LL r = 1;
    for (int i = 0; i < 50; i++) {
        if ((r - b) % c == 0)
            return i;
        r *= a;
        r %= c;
    }
    memset(head, -1, sizeof(head));
    LL cnt = xiao(a, b, c);
    if (cnt == -1)
        return -1;
    top = 1;
    if (b == 1)
        return cnt;
    LL m = ceil(sqrt(c * 1.0)), j;
    LL x = 1, p = 1;
    for (LL i = 0; i < m; ++i, p = p * a % c)
        insert(p * b % c, i);
    for (LL i = m;; i += m) {
        if ((j = find(x = x * p % c)) != -1)
            return i - j + cnt;
        if (i > c)
            break;
    }
    return -1;
}
```

2.17 杜教筛

```
#define LL long long
using namespace std;
typedef long long ll;
const ll N = 5000000;
bool is[N + 10];
int prime[N + 10], cnt;
LL miu[N + 10], phi[N + 10];

void init() {
    phi[1] = miu[1] = 1;
    is[1] = is[0] = 1;
    for (int i = 2; i <= N; i++) {
        if (!is[i]) prime[++cnt] = i, miu[i] = -1, phi[i] = prime[cnt] - 1;
        for (int j = 1; j <= cnt; j++) {
            if (i * prime[j] >= N) break;
            is[i * prime[j]] = 1;
            if (i % prime[j] == 0) {
                miu[i * prime[j]] = 0;
                phi[i * prime[j]] = phi[i] * prime[j];
                break;
            } else miu[i * prime[j]] = -miu[i], phi[i * prime[j]] = phi[i] * (prime[j] - 1);
        }
    }
    for (int i = 1; i <= N; i++)miu[i] += miu[i - 1], phi[i] += phi[i - 1];
}
unordered_map<int, LL> ansmiu, ansphi;
LL S_phi(int n) {
    if (n <= N)return phi[n];
    if (ansphi[n])return ansphi[n];
    LL ans = 0;
    for (int l = 2, r; l <= n; l = r + 1)
        r = n / (n / l), ans += (r - l + 1) * S_phi(n / l);
    return ansphi[n] = 1ll * (LL) n * (n + 1) / 2 - ans;//杜教筛公式
}
LL S_miu(int n) {
    if (n <= N)return miu[n];
    if (ansmiu[n])return ansmiu[n];
    LL ans = 0;
    for (int l = 2, r; l <= n; l = r + 1)
        r = n / (n / l), ans += (r - l + 1) * S_miu(n / l);
    return ansmiu[n] = 1ll - ans;//杜教筛公式
}
int n;
int main() {
    init();
    int T;
    cin >> T;
    while (T--) {
        scanf("%d", &n);
        printf("%lld␣%lld\n", S_phi(n), S_miu(n));
    }
    return 0;
}
```

2.18 min-25 筛

2.18.1 求区间质数和

```
#include<iostream>
#include<cstdio>
#include<cstdlib>
#include<cstring>
#include<algorithm>
#include<cmath>
using namespace std;

#define ll long long
#define ld long double
#define mem(Arr,x) memset(Arr,x,sizeof(Arr))

const int maxN=1010000;
const int inf=2147483647;

ll nn,srt;
bool notprime[maxN];
int pcnt,P[maxN];
ll num,Num[maxN+maxN],Id[maxN+maxN];
ld G[maxN+maxN],Sum[maxN];

ld Calc(ll n);
void Init();
int GetId(ll x);

int main(){
    Init();
    ll L,R;scanf("%lld%lld",&L,&R);
    printf("%.0LF\n",Calc(R)-Calc(L-1));return 0;
}

```

```
ld Calc(ll n){
    if (n==0) return 0;
    srt=sqrt(n);nn=n;num=0;mem(Id,0);mem(G,0);mem(Num,0);
    for (ll i=1,j;i<=n;i=j+1){
        j=n/i;Num[++num]=j;G[num]=(j<=1)?(0):((ld)(j-1)*(ld)(j+2)/2.0);
        if (j<=srt) Id[j]=num;
        else Id[i+maxN]=num;
        j=n/j;
    }
    for (int j=1;j<=pcnt;j++){
        for (int i=1;(i<=num)&&(1ll*P[j]*P[j]<=Num[i]);i++){
            G[i]=G[i]-1ll*P[j]*(G[GetId(Num[i]/P[j])]-Sum[j-1]);
        }
    }
    return G[1];
}

void Init(){
    notprime[1]=1;
    for (int i=1;i<maxN;i++){
        if (notprime[i]==0) P[++pcnt]=i,Sum[pcnt]=Sum[pcnt-1]+i;
        for (int j=1;(j<=pcnt)&&(1ll*i*P[j]<maxN);j++){
            notprime[i*P[j]]=1;if (i%P[j]==0) break;
        }
    }
    return;
}

int GetId(ll x){
    if (x<=srt) return Id[x];
    else return Id[nn/x+maxN];
}
```

2.18.2 求区间质数个数

```
#include<iostream>
#include<cstdio>
#include<cstdlib>
#include<cstring>
#include<algorithm>
#include<cmath>
using namespace std;

#define ll long long
#define mem(Arr,x) memset(Arr,x,sizeof(Arr))

const int maxN=1001000;
const int inf=2147483647;

ll n,srt;
ll G[maxN];
ll num,Num[maxN+maxN],Id[maxN+maxN];
bool notprime[maxN];
int pcnt,P[maxN];
```

```

void Init(){
    notprime[1]=1;
    for (int i=2;i<maxN;i++){
        if (notprime[i]==0) P[++pcnt]=i;
        for (int j=1;(j<=pcnt)&&(1ll*i*P[j]<maxN);j++){
            notprime[i*P[j]]=1;
            if (i%P[j]==0) break;
        }
    }
    return;
}
int GetId(ll x){
    if (x<=srt) return Id[x];
    else return Id[n/x+maxN];
}
ll S(ll n,int j);

int main(){
    Init();
    scanf("%lld",&n);srt=sqrt(n);
    for (ll i=1,j;i<=n;i=j+1){
        j=n/(n/i);Num[++num]=n/i;G[num]=n/i-1;
        if (n/i<=srt) Id[n/i]=num;
        else Id[i+maxN]=num;
    }
    for (int j=1;j<=pcnt;j++)
        for (int i=1;(i<=num)&&(1ll*P[j]*P[j]<=Num[i]);i++)
            G[i]=G[i]-G[GetId(Num[i]/P[j])]+(j-1);
    printf("%lld\n",G[1]);
}
```

2.19 线性基

```
#define size 60
struct LinearBase
{
    LL v[size+7],p[size+7];
    int cnt=0;
    bool now=0,flag;
    void clear(){
        memset(v,0,sizeof(v));
        memset(p,0,sizeof(p));
        cnt=0;flag=now=0;
    }
    void ins(LL a){
        now=0;
        for(int i=size;i>=0;i--)
            if(a&(1ll<<i)){
                if(!v[i]){v[i]=a;break;}
                a^=v[i];
            }
        if(!a)flag=1;
    }
    bool find(LL a){
        if(!a&&flag)return 1;
        for(int i=size;i>=0;i--)
            if(a&(1ll<<i)){
                if(!v[i])return 0;
                a^=v[i];
            }
        return 1;
    }
    LL getmax(){
        LL anss=0;
        for(int i=size;i>=0;i--)if((anss^v[i])>anss)anss^=v[i];
        return anss;
    }
    LL getmin(){
        if(flag)return 0;
        for(int i=0;i<=size;i++)if(!v[i])return v[i];
    }
    void rebuild(){
        now=1,cnt=0;
        memset(p,0,sizeof(p));
        for(int i=size;i>=0;i--)
            for(int j=i-1;j>=0;j--)
                if(v[i]&(1ll<<j))v[i]^=v[j];
        for(int i=0;i<=size;i++)if(v[i])p[cnt++]=v[i];
    }
    LL kth_min(LL k){
        if(!now)rebuild();
        LL res=0;
        if(flag)k--;
        if(k>=(1LL<<cnt))return -1;
        for(int i=size;i>=0;i--)if(k&(1LL<<i))res^=p[i];
        return res;
    }
    LL kth_max(LL k){
        if(!now)rebuild();
        LL t=(1LL<<cnt);
        if(flag)t--;
        return kth_min(t-k+1);
    }
    LinearBase operator+(const LinearBase &a){
        LinearBase c=a;
        for(int i=0;i<=size;i++)if(v[i])c.ins(v[i]);
        return c;
    }
    friend LinearBase operator*(const LinearBase &a,const LinearBase &b){
        LinearBase all,c,d;
        all.clear(),c.clear(),d.clear();
        for(int i=size;i>=0;i--)all.v[i]=a.v[i];
        for(int i=size;i>=0;i--){
            if(b.v[i]){
                LL v=b.v[i],k=1LL>>i;
                bool can=1;
                for(int j=size;j>=0;j--)
```

```
                    if(v&(1ll<<j))
                        if(all.v[j])v^=all.v[i],k^=d.v[j];
                        else{can=0,all.v[j]=v,d.v[j]=k;break;}
                if(can){
                    LL v=0;
                    for(int j=size;j>=0;j--)if(k&(1ll<<j))v^=b.v[j];
                    c.ins(v);
                }
            }
        }
        return all;
    }
};
int main()
{
    return 0;
}
```

2.20 类欧几里得

```
const LL p=998244353;
LL i2=499122177,i6=166374059;//2^(-1),6^(-1)
struct data{
    data(){f=g=h=0;}
    LL f,g,h;
};
data calc(LL n,LL a,LL b,LL c){
    LL ac=a/c,bc=b/c,m=(a*n+b)/c,n1=n+1,n21=n*2+1;
    data d;
    if(!a){
        d.f=bc*n1%p;
        d.g=bc*n%p*n1%p*i2%p;
        d.h=bc*bc%p*n1%p;
        return d;
    }
    if(a>=c||b>=c){
        d.f=n*n1%p*i2%p*ac%p+n1*bc%p;
        d.g=ac*n%p*n1%p*n21%p*i6%p+bc*n%p*n1%p*i2%p;
        d.h=ac*ac%p*n%p*n1%p*n21%p*i6%p+bc*bc%p*n1%p+ac*bc%p*n%p*n1%p;
        d.f%=p,d.g%=p,d.h%=p;
        data e=calc(n,a%c,b%c,c);
        d.h+=e.h+2*bc%p*e.f%p+2*ac%p*e.g%p;
        d.g+=e.g,d.f+=e.f;
        d.f%=p,d.g%=p,d.h%=p;
        return d;
    }
    data e=calc(m-1,c,c-b-1,a);
    d.f=n*m%p-e.f,d.f=(d.f%p+p)%p;
    d.g=m*n%p*n1%p-e.h-e.f,d.g=(d.g*i2%p+p)%p;
    d.h=n*m%p*(m+1)%p-2*e.g-2*e.f-d.f,d.h=(d.h%p+p)%p;
    return d;
}
```

3 字符串

3.1 后缀数组

```
#include <cstdio>
#include <iostream>
#include <cstring>
#include <string>
#include <algorithm>

using namespace std;
int n;
const int N = 1000010;
int sa[N], x[N], c[N], y[N];
char s[N];
inline void SA()
{
    for (int i = 1; i <= n; ++i) x[i] = s[i];
    int m = 128;
    for (int i = 0; i <= m; ++i) c[i] = 0;
    for (int i = 1; i <= n; ++i) c[x[i]]++;
    for (int i = 1; i <= m; ++i) c[i] += c[i-1];
    for (int i = n; i; i--) sa[c[x[i]]--] = i;

    for (int k = 1, p; k <= n; k <<= 1){
        p = 0;
        for (int i = n; i > n - k; i--) y[++p] = i;
        for (int i = 1; i <= n; ++i)
            if(sa[i] > k) y[++p] = sa[i] - k;

        for (int i = 0; i <= m; ++i) c[i] = 0;
        for (int i = 1; i <= n; ++i) c[x[i]]++;
        for (int i = 1; i <= m; ++i) c[i] += c[i-1];
        for (int i = n; i; i--) sa[c[x[y[i]]]--] = y[i];

        p = y[sa[1]] = 1;
        for (int i = 2, a, b; i <= n; ++i){
            a = sa[i] + k > n? -1 : x[sa[i] + k];
            b = sa[i-1] + k > n?-1 : x[sa[i-1] + k];
            y[sa[i]] = (x[sa[i]] == x[sa[i-1]] && (a == b)? p : ++p);
        }
        swap(x, y);
        m = p;
        if (p == n)break;
    }
}
int height[N], rk[N], st[N][20];
inline void build_H(){
    int k = 0;
    for (int i = 1; i <= n; ++i) rk[sa[i]] = i;
    for (int i = 1; i <= n; ++i){
        if (rk[i] == 1) continue;
        if (k) --k;
        int j = sa[rk[i]-1];
```

```
            while(j + k <= n && i + k <= n && s[i+k]==s[j+k]) ++k;
            height[rk[i]] = k;
        }
}
inline void build_lcp(){
        for (int i = 1; i <= n; ++i)st[i][0] = height[i];
        for (int i = 1; i <= 17; ++i){
            for (int j = 1; j+(1<<i)-1 <= n; ++j)
                st[j][i] = max(st[j][i-1], st[j+(1<<(i-1))][i-1]);
        }
}
inline int query(int l, int r){
        int k = log2(r - l + 1);
        return max(st[l][k], st[r-(1<<k)+1][k]);
}
int main()
{
        scanf("%s", s + 1);
        n = strlen(s + 1);
        SA();
        build_H();
        for (int i = 1; i <= n; ++i)printf("%d␣", sa[i]);
}
```

3.2 KMP

3.2.1 KMP

```
#include<cstdio>
#include<iostream>
#include<cstring>
#include<string>

using namespace std;
const int N = 10010;
char s[N];
int nxt[N];
//求 t 在 s 中是否出现，需要t的nxt数组
inline int check(char *s, int sl, char *t, int tl, int *nxt){
        int l = 0;
        for (int i = 1; i <= sl; ++i){
            while(l && s[i] != t[l + 1]) l = nxt[l];
            if (s[i] == t[l + 1]) l++;
            if (l == tl) return 1;
        }
        return 0;
}
int main(){
        scanf("%s", s + 1);
        int p = 0, n = strlen(s + 1);
        for (int i = 2; i <= n; ++i){
            while(p && s[p + 1] != s[i]) p = nxt[p];
            if (s[p + 1] == s[i]) p++;
            nxt[i] = p;
```

```
    }
    for (int i = 1; i <= n; ++i)
        printf("%d␣", nxt[i]);
}
```

3.2.2 exkmp

```
//有两个字符串a,b,要求输出b与a的每一个后缀的最长公共前缀
//nxt里为b的next数组, extend里是b与a的每一个后缀的最长公共前缀
#include<cstdio>
#include<iostream>
#include<cstring>
#include<string>
#include<algorithm>


#define N 1000010

using namespace std;

int q,nxt[N],extend[N];
string s,t;

void getnxt()
{
    nxt[0]=t.size();//nxt[0]一定是T的长度
    int now=0;
    while(t[now]==t[1+now]&&now+1<(int)t.size())now++;//这就是从1开始暴力
    nxt[1]=now;
    int p0=1;
    for(int i=2;i<(int)t.size();i++)
    {
        if(i+nxt[i-p0]<nxt[p0]+p0)nxt[i]=nxt[i-p0];//第一种情况
        else
        {//第二种情况
            int now=nxt[p0]+p0-i;
            now=max(now,0);//这里是为了防止i>p的情况
            while(t[now]==t[i+now]&&i+now<(int)t.size())now++;//暴力
            nxt[i]=now;
            p0=i;//更新p0
        }
    }
}

void exkmp()
{
    getnxt();
    int now=0;
    while(s[now]==t[now]&&now<min((int)s.size(),(int)t.size()))now++;//暴力
    extend[0]=now;
    int p0=0;
    for(int i=1;i<(int)s.size();i++)
    {
        if(i+nxt[i-p0]<extend[p0]+p0)extend[i]=nxt[i-p0];//第一种情况
        else
        {//第二种情况
            int now=extend[p0]+p0-i;
            now=max(now,0);//这里是为了防止i>p的情况
            while(t[now]==s[i+now]&&now<(int)t.size()&&now+i<(int)s.size())now++;//暴
                力
            extend[i]=now;
            p0=i;//更新p0
        }
    }
}

int main()
{
    cin>>s>>t;
    exkmp();
    int len=t.size();
    for(int i=0;i<len;i++)printf("%d␣",nxt[i]);//输出nxt
    puts("");
    len=s.size();
    for(int i=0;i<len;i++)printf("%d␣",extend[i]);//输出extend
    return 0;
}
```

3.3 manacher

```
void Manacher(char s[],int len) {//原字符串和串长
    int l = 0;
    String[l++] = '$'; // 0下标存储为其他字符,防止越界
    String[l++] = '#';
    for (int i = 0; i < len; i++) {
        String[l++] = s[i];
        String[l++] = '#';
    }
    String[l] = 0; // 空字符
    int MaxR = 0;
    int flag = 0;
    for (int i = 0; i < l; i++) {
        cnt[i] = MaxR > i ? min(cnt[2 * flag - i], MaxR - i) : 1;//2*flag-i是i点关于
            flag的对称点
        while (String[i + cnt[i]] == String[i - cnt[i]])
            cnt[i]++;
        if (i + cnt[i] > MaxR) {
            MaxR = i + cnt[i];
            flag = i;
        }
    }
}
/*
* String: $ # a # b # a # a # b # a # 0
* cnt:      1 1 2 1 4 1 2 7 2 1 4 1 2 1
*/
```

3.4 PAM

```
//本质不同回文串数量 cnt[i] - 2;

#include <cstdio>
#include <iostream>
#include <cstring>
#include <string>
#include <algorithm>

using namespace std;
const int N = 300010;
struct Pam{
    int len;
    int nxt[30];
    int fail;
}st[N];
int n, tot, last;
int S[N], cnt[N], num[N];
inline int newnode(int x){
    memset(st[tot].nxt, 0, sizeof(st[tot].nxt));
    cnt[tot] = 0, st[tot].len = x, num[tot] = 0;
    return tot++;
}
inline void pam_init(){
    tot = n = last = 0, newnode(0), newnode(-1);
    S[0] = - 1, st[0].fail = 1;
}
inline int get_fail(int x){
    while(S[n-st[x].len-1] != S[n]) x = st[x].fail;
    return x;
}
inline int insert(int c){
    S[++n] = c;
    int cur = get_fail(last);
    if (!st[cur].nxt[c]){
        int now = newnode(st[cur].len + 2);
        st[now].fail = st[get_fail(st[cur].fail)].nxt[c];
        st[cur].nxt[c] = now;
        num[now] = num[st[now].fail] + 1;
    }
    last = st[cur].nxt[c];
    cnt[last]++;
    return num[last];
}
inline void count(){
    for (int i = tot - 1; i >= 0; --i) cnt[st[i].fail] += cnt[i];
}
char s[N];
int main()
{
    pam_init();
    scanf("%s", s+1);
    int len = strlen(s + 1);
    long long ans = 0;
    for (int i = 1; i <= len; ++i) insert(s[i] - 'a' + 1);
    count();
    for (int i = 1; i < tot; ++i) ans = max(ans, 1ll * st[i].len * cnt[i]);
    printf("%lld\n", ans);
}
```

3.5 AC 自动机

3.5.1 求最大出现次数

```
//AC自动机求最大出现次数
//注意AC自动机上是可能出现自环的，所以在判环的时候要注意
//记得build
#include<cstdio>
#include<iostream>
#include<cstring>
#include<string>
#include<algorithm>
#include<queue>

using namespace std;
const int N = 156, L = 1e6 + 6;
namespace AC{
    const int SZ = N * 80;
    int tot, tr[SZ][26];
    int fail[SZ], idx[SZ], val[SZ];
    int cnt[N];
    void init(){
        memset(fail, 0, sizeof(fail));
        memset(tr, 0, sizeof(tr));
        memset(val, 0, sizeof(val));
        memset(cnt, 0, sizeof(cnt));
        memset(idx, 0, sizeof(idx));
        tot = 0;
    }
    void insert(char *s, int id){
        int u = 0;
        for (int i = 1; s[i]; i++){
            if (!tr[u][s[i] - 'a']) tr[u][s[i] - 'a'] = ++tot;
            u = tr[u][s[i] - 'a'];
        }
        idx[u] = id;
    }
    queue<int> q;
    void build(){
        for (int i = 0; i < 26; ++i)
            if (tr[0][i]) q.push(tr[0][i]);
        while(!q.empty()){
            int u = q.front();
            q.pop();
            for (int i = 0; i < 26; ++i){
                if (tr[u][i]) fail[tr[u][i]] = tr[fail[u]][i], q.push(tr[u][i]);
                else tr[u][i] = tr[fail[u]][i];
            }
        }
    }
    inline int query(char *t){
        int u = 0, res = 0;
        for (int i = 1; t[i]; i++){
            u = tr[u][t[i] - 'a'];
            for (int j = u; j; j = fail[j]) val[j]++;
        }
        for (int i = 0; i <= tot; ++i)
            if (idx[i]) res = max(res, val[i]), cnt[idx[i]] = val[i];
        return res;
    }
}
```

3.5.2 AC 自动机 (map) 版本

```
#include<cstdio>
#include<iostream>
#include<cstring>
#include<string>
#include<algorithm>
#include<queue>
#include<map>

using namespace std;
const int N = 50010, L = 1e6 + 6;
int pos[N];
namespace AC{
    const int SZ = 100010;
    int tot;
    int vis[SZ], cnt[SZ], val[SZ];
    int fail[SZ];
    map<int, int> tr[SZ];
    void insert(int *s, int l, int id){
        int u = 0;
        for (int i = 1; i <= l; i++){
            if (!tr[u][s[i]]) tr[u][s[i]] = ++tot;
            u = tr[u][s[i]];
            // cout<<u<<' ';
        }
        pos[id] = u;
        val[u]++;
        // cout<<endl;
    }
    queue<int>q;
    inline void build(){
        for (auto i:tr[0]) q.push(i.second);
        while(!q.empty()){
            int u = q.front();
            q.pop();
            // cout<<u<<endl;
            for (auto i:tr[u]){
                fail[i.second] = tr[fail[u]][i.first], q.push(i.second);
            }
        }
    }
    inline int query(int id, vector<int>t1){
        int u = 0, res = 0;
        int l = t1.size();
        for (int i = 0; i < l; i++){
            while(u && !tr[u][t1[i]]) u = fail[u];
            u = tr[u][t1[i]];
            for (int j = u; j; j = fail[j]){
                if (vis[j] != id){
                    cnt[j]++, vis[j] = id;
                    if (val[j]) res += val[j];
                }
                else break;
            }
        }
        return res;
    }
}
vector<int> q1[N], q2[N];
int t[N * 2], ans[N];
```

3.5.3 AC 自动机 + 矩阵快速幂

```
int n;
char s[N][100], t[L];
int main()
{
    while(~scanf("%d", &n)){
        if (n == 0) break;
        AC::init();
        for (int i = 1; i <= n; ++i) scanf("%s", s[i] + 1), AC::insert(s[i], i);
        AC::build();
        scanf("%s", t + 1);
        int x = AC::query(t);
        printf("%d\n", x);
        for (int i = 1; i <= n; ++i)
            if (AC::cnt[i] == x) printf("%s\n", s[i] + 1);
    }
    return 0;
}
//有很多题需要判断是否是终点，在求fail结点的时候可以顺便求一下
if (tr[u][i]) fail[tr[u][i]] = tr[fail[u]][i], idx[tr[u][i]] |= idx[fail[tr[u][i]]], q
    .push(tr[u][i]);
//矩阵快速幂求出现过给定字符串的的所有长度的串
typedef unsigned long long ull;
struct rec{
    int sz;
    ull a[N][N];
    rec(){}
    void init(int _sz){
        sz = _sz;
        memset(a, 0, sizeof(a));
    }
    rec operator* (const rec &b) const {
        rec res;
        res.init(sz);
        for (int i = 0; i <= sz; ++i){
            for (int j = 0; j <= sz; ++j){
                for (int k = 0; k <= sz; ++k)
                    res.a[i][j] += a[i][k] * b.a[k][j];
            }
        }
        return res;
    }
}e, ans, res;
inline rec qpow(rec a, long long q){
    rec res = e;
    for (;q; q >>= 1, a = a * a){
        if (q & 1) res = res * a;
    }
    return res;
}
//省略了init和insert
namespace AC{
    queue<int>q;
    void build(){
        for (int i = 0; i < 26; ++i)
            if (tr[0][i]) q.push(tr[0][i]);
        while(!q.empty()){
            int u = q.front();
            q.pop();
            for (int i = 0; i < 26; ++i){
                if (tr[u][i]) fail[tr[u][i]] = tr[fail[u]][i], idx[tr[u][i]] |= idx[
                    fail[tr[u][i]]], q.push(tr[u][i]);
                else tr[u][i] = tr[fail[u]][i];
            }
        }
    }
    inline rec chengzi(){
        res.init(tot + 1);
        for (int i = 0; i <= tot; ++i){
            if (idx[i]) res.a[i][tot + 1] = 1, res.a[i][i] = 26;
            else for (int j = 0; j < 26; ++j){
                    res.a[i][tr[i][j]]++;
                }
        }
        res.a[tot + 1][tot + 1] = 1;
        return res;
    }
};
```

3.6 SAM

3.6.1 SAM

```
const int N = 40010;
```

```
struct state{
    int len, link;
    int next[180];
}st[N];
int sz, last;
int s[N], a[N], n;
int l[N], r[N], c[N], q[N];
void sam_init(){
    memset(st, 0, sizeof(st));
    sz = last = 1;
}
void sam_extend(int c){
    int cur = ++sz;
    st[cur].len = st[last].len + 1;
    int p = last;
    while(p && !st[p].next[c]){
        st[p].next[c] = cur;
        p = st[p].link;
    }
    if (!p){
        st[cur].link = 1;
    } else {
        int q = st[p].next[c];
        if (st[p].len + 1 == st[q].len) {
            st[cur].link = q;
        } else {
            int clone = ++sz;
            st[clone].len = st[p].len + 1;
            memcpy(st[clone].next, st[q].next, sizeof(st[q].next));
            st[clone].link = st[q].link;
            while(p && st[p].next[c] == q) {
                st[p].next[c] = clone;
                p = st[p].link;
            }
            st[q].link = st[cur].link = clone;
        }
    }
    last = cur;
}
```

3.6.2 求子串出现次数 (树上 dp)

```
// 求字典序第k大子串，T为0则表示不同位置的相同子串算作一个。T=1则表示不同位置的相同子串算作多个
// T=0时，跑一个后缀自动机上的dp，求不同子串个数，相当于求路径数目，然后然后在树上一个一个点的找就可以了。
// T=1时，现求一个子串出现次数，做法是按照后缀链接跑树形dp，字符串上的点赋值为1，克隆出的点初值为0，跑完dp后在按照第一种方法做一下就可以
#include <cstdio>
#include <iostream>
#include <cstring>
#include <string>
#include <algorithm>

```

```
using namespace std;
const int N = 1000010;
struct state{
    int len, link;
    int nxt[27];
}st[N];
int n, sz, T, last;
long long k;
char s[N];
int c[N], q[N];
long long dp[N], cnt[N];
inline void sam_init(){
    sz = last = 1;
}
inline void sam_extend(int c){
    int cur = ++sz;
    st[cur].len = st[last].len + 1;
    int p = last;
    while(p && !st[p].nxt[c]){
        st[p].nxt[c] = cur;
        p = st[p].link;
    }
    if (!p) st[cur].link = 1;
    else {
        int q = st[p].nxt[c];
        if (st[p].len + 1 == st[q].len){
            st[cur].link = q;
        } else {
            int clone = ++sz;
            st[clone].len = st[p].len + 1;
            memcpy(st[clone].nxt, st[q].nxt, sizeof(st[q].nxt));
            st[clone].link = st[q].link;
            while(p && st[p].nxt[c] == q){
                st[p].nxt[c] = clone;
                p = st[p].link;
            }
            st[q].link = st[cur].link = clone;
        }
    }
    last = cur;
}
inline void sam(){
    long long now = 0;
    int p = 1;
    while(1){
        for (int i = 1; i <= 26; ++i){
            if (now + dp[st[p].nxt[i]] < k)
                now += dp[st[p].nxt[i]];
            else {
                putchar('a' + i - 1);
                now += cnt[st[p].nxt[i]];
                if (now >= k) return;
                p = st[p].nxt[i];
                break;
                }
            }
        }
}
int main()
{
    scanf("%s", s + 1);
    n = strlen(s + 1);
    sam_init();
    for (int i = 1; i <= n; ++i)
        sam_extend(s[i] - 'a' + 1);
    for (int i = 1; i <= sz; ++i) c[st[i].len]++;
    for (int i = 1; i <= n; ++i) c[i] += c[i-1];
    for (int i = sz; i; i--) q[c[st[i].len]--] = i;
    scanf("%d%lld", &T, &k);
    if (!T){
        for (int i = 1; i <= sz; ++i) cnt[i] = 1;
    } else {
        int p = 1;
        for (int i = 1; i <= n; ++i){
            int x = s[i] - 'a' + 1;
            p = st[p].nxt[x];
            cnt[p] = 1;
        }
        for (int i = sz; i >= 1; --i){
            p = q[i];
            cnt[st[p].link] += cnt[p];
        }
    }
    for (int i = sz; i >= 1; --i){
        int p = q[i];
        if (p != 1) dp[p] = cnt[p];
        for (int j = 1; j <= 26; ++j)
            dp[p] += dp[st[p].nxt[j]];
    }
    if (k > dp[1])puts("-1");
    else sam();
}
```

3.6.3 求最长不重叠子串，使用 lastpos 的性质

```
//求最长不重叠子串
const int N = 40010;
struct state{
    int len, link;
    int next[180];
}st[N];
int sz, last;
int s[N], a[N], n;
int l[N], r[N], c[N], q[N];
void sam_init(){
    memset(l, 63, sizeof(l));
    memset(r, 0, sizeof(r));
    memset(c, 0, sizeof(c));
    memset(q, 0, sizeof(q));
    memset(st, 0, sizeof(st));
    sz = last = 1;
}
void sam_extend(int c){
    int cur = ++sz;
    st[cur].len = st[last].len + 1;
    l[cur] = r[cur] = st[cur].len;
    int p = last;
    while(p && !st[p].next[c]){
        st[p].next[c] = cur;
        p = st[p].link;
    }
    if (!p){
        st[cur].link = 1;
    } else {
        int q = st[p].next[c];
        if (st[p].len + 1 == st[q].len) {
            st[cur].link = q;
        } else {
            int clone = ++sz;
            st[clone].len = st[p].len + 1;
            memcpy(st[clone].next, st[q].next, sizeof(st[q].next));
            st[clone].link = st[q].link;
            while(p && st[p].next[c] == q) {
                st[p].next[c] = clone;
                p = st[p].link;
            }
            st[q].link = st[cur].link = clone;
        }
    }
    last = cur;
}
int main()
{
    while(scanf("%d", &n)){
        if (!n)break;
        int ans = 0;
        for (int i = 1; i <= n; ++i){
            scanf("%d", &s[i]);
            a[i] = s[i] - s[i - 1];
        }
        sam_init();
        for (int i = 2; i <= n; ++i){
            sam_extend(a[i] + 88);
        }
        for (int i = 1; i <= sz; ++i)c[st[i].len]++;
        for (int i = 1; i < n; ++i) c[i] += c[i-1];
        for (int i = sz; i; i--)q[c[st[i].len]--] = i;
        for (int i = sz; i; i--){
            int p = q[i];
            l[st[p].link] = min(l[st[p].link], l[p]);
            r[st[p].link] = max(r[st[p].link], r[p]);
        }
        for (int i = 1; i <= sz; ++i)
            ans = max(ans, min(st[i].len, r[i] - l[i]));
        if (ans < 4)puts("0");
        else printf("%d\n", ans + 1);
    }
}
```

4 图论

4.1 2-SAT

对于 a_i，用第 $2i-1$ 和 $2i$ 表示取 0 和 1，一条边 $x->y$ 表示选了 x 必选 y i' 表示 i 的反面：

1. i,j 不能同时选：$i \to j', j \to i'$，一般为 $a_i xor a_j = 1$
2. i,j 必须同时选：$i \to j, j \to i$，一般为 $a_i xor a_j = 0$
3. i,j 只能选其一：$i \to j', j \to i', i' \to j, j' \to i$ 一般为 $a_i or a_j = 1$
4. 必须选 i：$i' \to i$ 一般为 $a_i = 1$ 或 $a_i and a_j = 1$

4.2 生成树计数

G 的度数矩阵 $D[G]$,G 的邻接矩阵 $A[G]$ G 的 $Kirchhoff$ 矩阵 $C[G] = D[G]-A[G]$ $Matrix-Tree$ 定理 $C[G]$ 的任何一个 $n-1$ 阶主子式的行列式的绝对值，为所有不同生成树的个数求行列式用高斯消元

```

LL a[MAXN][MAXN];
const LL mod=1e9;
void add(int u,int v){//加边 双向
    a[u][u]++,a[v][v]++;
    a[u][v]--,a[v][u]--;
}
int Gauss(int n){//高斯消元
    int ans=1;
    for(int i=1;i<=n;i++){
        for(int k=i+1;k<=n;k++){
            while(a[k][i]){
                int d=a[i][i]/a[k][i];
                for(int j=i;j<=n;j++)a[i][j]=(a[i][j]-1ll*d*a[k][j]%mod+mod)%mod;
                swap(a[i],a[k]),ans=-ans;
            }
        }
        ans=1ll*ans*a[i][i]%mod,ans=(ans%mod+mod)%mod;
    }
    return ans;
}
int n,m,id[MAXN][MAXN];
char s[MAXN][MAXN];
int main(){
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n;i++)scanf("%s",s[i]+1);
    int idx=0;
    for(int i=1;i<=n;i++)
        for(int j=1;j<=m;j++)
            if(s[i][j]=='.')id[i][j]=++idx;
    for(int i=1;i<=n;i++)
```

```
        for(int j=1;j<=m;j++)
            if(id[i][j]){
                if(id[i-1][j])add(id[i][j],id[i-1][j]);
                if(id[i][j-1])add(id[i][j],id[i][j-1]);
            }
    cout<<Gauss(idx-1);
    return 0;
}
```

4.3 Floyd 求最小环

```
#include<algorithm>
#include<iostream>
#include<cstdio>
#define INF 0x3f3f
#define MAXN 511
using namespace std;
int dis[MAXN][MAXN], mp[MAXN][MAXN], n, m;
int mincost_route(){
    int mincost = INF;
    for(int k = 1; k <= n; k ++){
        for(int i = 1; i < k; i ++)
            for(int j = i+1; j < k; j ++)
                mincost = min(mincost, dis[i][j] + mp[i][k] + mp[k][j]);
        for(int i = 1; i <= n; i ++)
            for(int j = 1; j <= n; j ++)
                dis[i][j] = min(dis[i][j], dis[i][k] + dis[k][j]);
    }
    return mincost;
}
int main(){
    scanf("%d%d", &n, &m);
    for(int i = 1; i <= n; i ++)
        for(int j = 1; j <= n; j ++)
            mp[i][j] = dis[i][j] = INF;
    for(int u, v, w, i = 1; i <= m; i ++){
        scanf("%d%d%d",&u, &v, &w);
        mp[u][v] = mp[v][u] = dis[u][v] = dis[v][u] = min(w, mp[u][v]);
    }
    cout << mincost_route() << endl;
}
```

4.4 BFS 判负环

```

const int maxn=200010;
struct gra {
    bool vis[maxn];
    int head[maxn], to[maxn << 1], nxt[maxn << 1], cnt;
    int dis[maxn],c[maxn];
    int f[maxn << 1],N;
    void clear(int n) {N=n,fill(head, head+5+n, 0),cnt = 0;}
    void add(int a, int b, int c) {nxt[++cnt] = head[a], to[head[a] = cnt] = b, f[cnt] = c;}
    bool spfa(int i){
        queue<int>q;
        q.push(i),c[i]++;
        dis[i]=0;
        while(!q.empty()){
            i=q.front(),q.pop();
            vis[i]=0;
            for(int j=head[i];j;j=nxt[j]){
                if(dis[to[j]]>dis[i]+f[j]){
                    dis[to[j]]=dis[i]+f[j];
                    if(!vis[to[j]]){
                        c[to[j]]++;
                        if(c[to[j]]>N)return 1;
                        vis[to[j]]=1,q.push(to[j]);
                    }
                }
            }
        }
        return 0;
    }
    bool ne_c(int n,int i){
        fill(dis,dis+5+n,1000000000);
        fill(vis,vis+5+n,0);
        fill(c,c+5+n,0);
        return spfa(i);
    }
}G;
int n,m1,m2;
int edge1[maxn][4],edge2[maxn][4];
bool check(int p){
    G.clear(n+1);
    for(int i=1;i<=m1;i++)
        G.add(edge1[i][2],edge1[i][1]-1,-edge1[i][3]);
    for(int i=1;i<=m2;i++)
        G.add(edge2[i][1]-1,edge2[i][2],p-edge2[i][3]);
    for(int i=1;i<=n;i++)G.add(i,i-1,0),G.add(i-1,i,1);
    G.add(0,n,p),G.add(n,0,-p);
    return G.ne_c(n,0);
}
int main(){
    int T;cin>>T;
    while(T--){
        scanf("%d%d%d",&n,&m1,&m2);
        for(int i=1;i<=m1;i++)
            scanf("%d%d%d",&edge1[i][1],&edge1[i][2],&edge1[i][3]);
        for(int i=1;i<=m2;i++)
            scanf("%d%d%d",&edge2[i][1],&edge2[i][2],&edge2[i][3]);
        int l=0,r=n,mid;
        while(l<r){
            mid=(l+r)>>1;
            if(check(mid)) l=mid+1;
            else r=mid;
        }
        printf("%d\n",l);
    }
    return 0;
}
```

4.5 DFS 判负环

```


const int maxn = 200010;

struct gra {
    bool ins[maxn];
    int head[maxn], to[maxn << 1], nxt[maxn << 1], cnt, flag;
    int dis[maxn], f[maxn << 1];
    void clear(int n) { fill(head, head + 5 + n, 0), flag = cnt = 0; }
    void add(int a, int b, int c) { nxt[++cnt] = head[a], to[head[a] = cnt] = b, f[cnt
        ] = c; }
    void spfa(int h) {
        ins[h] = 1;
        for (int i = head[h]; i; i = nxt[i])
            if (dis[to[i]] > dis[h] + f[i]) {
                if (ins[to[i]] || flag) {
                    flag = 1;
                    break;
                }
                dis[to[i]] = dis[h] + f[i];
                spfa(to[i]);
            }
        ins[h] = 0;
    }

    bool ne_c(int n) {
        fill(dis, dis + 5 + n, 0);
        fill(ins, ins + 5 + n, 0);
        flag = 0;
        for (int i = 0; i <= n && !flag; i++)spfa(i);
        return flag;
    }
} G;

int n, m1, m2;
int edge1[maxn][4], edge2[maxn][4];

bool check(int p) {
    G.clear(n + 1);
    for (int i = 1; i <= m1; i++)
        G.add(edge1[i][2], edge1[i][1] - 1, -edge1[i][3]);
    for (int i = 1; i <= m2; i++)
        G.add(edge2[i][1] - 1, edge2[i][2], p - edge2[i][3]);
    for (int i = 1; i <= n; i++)G.add(i, i - 1, 0), G.add(i - 1, i, 1);
    G.add(0, n, p), G.add(n, 0, -p);
    return G.ne_c(n);
}

int main() {
    int T;
    cin >> T;
    while (T--) {
        scanf("%d%d%d", &n, &m1, &m2);
        for (int i = 1; i <= m1; i++)
            scanf("%d%d%d", &edge1[i][1], &edge1[i][2], &edge1[i][3]);
        for (int i = 1; i <= m2; i++)
            scanf("%d%d%d", &edge2[i][1], &edge2[i][2], &edge2[i][3]);
        int l = 0, r = n, mid;
        while (l < r) {
            mid = (l + r) >> 1;
            if (check(mid)) l = mid + 1;
            else r = mid;
        }
        printf("%d\n", l);
    }
    return 0;
}
```

4.6 匈牙利算法

```
//匈牙利求最小边覆盖（点数 - 最大匹配）
//最小点覆盖 = 最大匹配
#include<iostream>
#include<cstdio>
#include<cstring>

using namespace std;
const int N = 100010;
int matching[N], te, check[N], ans, head[N], n, m;
struct edge {
    int v, next;
} e[200010];

void add(int u, int v) {
    e[++te].v = v;
    e[te].next = head[u];
    head[u] = te;
}

int dfs(int u) {
    for (int i = head[u]; i; i = e[i].next) {
        int v = e[i].v;
        if (!check[v]) {
            check[v] = 1;
            if (matching[v] == -1 || dfs(matching[v])) {
                matching[u] = v;
                matching[v] = u;
                return true;
            }
        }
    }
    return false;
}

int hungarian() {
    memset(matching, -1, sizeof(matching));
    for (int i = 1; i <= n; i++) {
        if (matching[i] == -1) {
            memset(check, 0, sizeof(check));
            if (dfs(i))++ans;
        }
    }
}

int main() {
    ans = 0;
    memset(head, 0, sizeof(head));
    cin >> n >> m;
    for (int i = 1; i <= m; ++i) {
        int u, v;
        scanf("%d%d", &u, &v);
        add(u, v + n);
    }
    hungarian();
    cout << n - ans;
}
```

(1) 二分图的最大匹配

匈牙利算法

(2) 二分图的最小点覆盖

二分图的最小点覆盖 = 二分图的最大匹配

求最小点覆盖：从右边所有没有匹配过的点出发，按照增广路的“交替出现”的要求 DFS。最终右边没有访问过的点和左边访问过的点组成最小点覆盖。

(3) 二分图的最少边覆盖

二分图的最少边覆盖 = 点数-二分图的最大匹配

证明：

先贪心选一组最大匹配的边放进集合，对于剩下的没有匹配的点，随便选一条与之关联的边放进集合，那么得到的集合就是最小边覆盖。所以有：最小边覆盖 = 最大匹配 + 点数-2* 最大匹配 = 点数-最大匹配

(4) 二分图的最大独立集

二分图的最大独立集 = 点数-二分图的最大匹配

证明：

我们可以这样想，先把所有的点放进集合，然后删去最少的点和与之相关联的边，使得全部边都被删完，这就是最小点覆盖。所以有：最大独立集 = 点数-最小点覆盖

(5) 有向无环图的最少不相交路径覆盖

我们把原图中的点 V 拆成两个点 Vx 和 Vy，对于原图中的边 A>B，我们在新图中连 Ax>By。那么最少不相交路径覆盖 = 原图的点数-新图的最大匹配

证明：

一开始每个点都独立为一条路径，在二分图中连边就是将路径合并，每连一条边路径数就减一。因为路径不能相交，所以不能有公共点，这恰好就是匹配的定义。所以有：最少不相交路径覆盖 = 原图的点数-新图的最大匹配

(6) 有向无环图的最少可相交路径覆盖

先用 floyd 求出原图的传递闭包，如果 a 到 b 有路，那么就加边 a->b。然后就转化成了最少不相交路径覆盖问题。

(7) 有向无环图中最少不相交路径覆盖和最大独立集的相互转化

用偏序集，一般可以抽象为有向无环图。

Dilworth 定理：有向无环图的最大独立集 = 有向无环图最少不相交路径覆盖

4.7 费用流-EK(Dij-单路增广)

```
typedef int T;

struct gra{
    int head[maxn], to[maxn<<1], nxt[maxn<<1], cnt;
    T f[maxn<<1], v[maxn << 1];
    void clear(int n) {fill(head, head+1+n, 0), cnt = 1;}
    void add(int a, int b, T c, T d){
        nxt[++ cnt] = head[a], head[a] = cnt, to[cnt] = b, f[cnt] = c, v[cnt] = d;
        nxt[++ cnt] = head[b], head[b] = cnt, to[cnt] = a, f[cnt] = 0, v[cnt] = -d;
    }
};

struct cost_flow : public gra{
    int pr[maxn], q[maxn], s, t, mx;
    bool vis[maxn];
    T dis[maxn], mnf[maxn];
    void init(int ss, int tt, int mxx){s = ss, t = tt, mx = mxx, clear(mx);}
    bool spfa(){
        for(int i = 0; i <= mx; i ++) dis[i] = inf, vis[i] = 0;
        int l = 1, r = 0;
        mnf[s] = inf, vis[s] = 1, dis[s] = 0, q[++ r] = s;
        while(l <= r){
            int x = q[l ++];
            for(int i = head[x]; i; i = nxt[i]){
                int u = to[i];
                if(f[i] > 0 && dis[u] > dis[x] + v[i]){
                    dis[u] = dis[x] + v[i];
                    pr[u] = i, mnf[u] = min(mnf[x], f[i]);
                    if(vis[u] == 0) vis[u] = 1, q[++ r] = u;
                }
            }
            vis[x] = 0;
        }
        return dis[t] != inf;
    }
    pair<T, T> ek(){
        T flow = 0, cost = 0;
        while(spfa()){
            cost += mnf[t]*dis[t], flow += mnf[t];
            for(int x = t; x != s; x = to[pr[x] ^ 1]){
                f[pr[x]] -= mnf[t], f[pr[x]^1] += mnf[t];
            }
        }
        return make_pair(flow, cost);
    }
};
```

4.8 最大流-Dinic

```
typedef int T;

struct gra{
    int head[maxn], to[maxn<<1], nxt[maxn<<1], cnt;
    T f[maxn<<1];
    void clear(int n) {fill(head, head+1+n, 0), cnt = 1;}
    void add(int a, int b, T c){
        nxt[++ cnt] = head[a], head[a] = cnt, to[cnt] = b, f[cnt] = c;
        nxt[++ cnt] = head[b], head[b] = cnt, to[cnt] = a, f[cnt] = 0;
    }
};

struct flow : public gra{
    int dep[maxn], q[maxn], cur[maxn], s, t, mx;
    void init(int ss, int tt, int mxx){s = ss, t = tt, mx = mxx, clear(mx);}
    bool bfs(){
        for(int i = 0; i <= mx; i ++) dep[i] = -1;
        int l = 1, r = 0;
        q[++ r] = s, dep[s] = 1;
        while(l <= r){
            int x = q[l ++];
            for(int i = head[x]; i; i = nxt[i]){
                int u = to[i];
                if(dep[u] == -1 && f[i] > 0)
                    q[++ r] = u, dep[u] = dep[x] + 1;
            }
        }
        return dep[t] != -1;
    }
    T dfs(int x, T mr){
        if(x == t || mr == 0) return mr;
        T c = 0, res = 0;
        for(int &i = cur[x]; i; i = nxt[i]){
            int u = to[i];
            if(dep[u] == dep[x] + 1 && (c = dfs(u, min(mr, f[i]))))
                f[i] -= c, f[i^1] += c, mr -= c, res += c;
            if(mr == 0) break;
        }
        return res;
    }
    T dinic(){
        T res = 0;
        while(bfs()){
            for(int i = 0; i <= mx; i ++) cur[i] = head[i];
            res += dfs(s, inf);
        }
        return res;
    }
};
```

4.9 最大流-Dinic

```
typedef int T;

```

```
struct gra{
    int head[maxn], to[maxn<<1], nxt[maxn<<1], cnt;
    T f[maxn<<1];
    void clear(int n) {fill(head, head+1+n, 0), cnt = 1;}
    void add(int a, int b, T c){
        nxt[++ cnt] = head[a], head[a] = cnt, to[cnt] = b, f[cnt] = c;
        nxt[++ cnt] = head[b], head[b] = cnt, to[cnt] = a, f[cnt] = 0;
    }
};

struct flow : public gra{
    int dep[maxn], q[maxn], cur[maxn], s, t, mx;
    void init(int ss, int tt, int mxx){s = ss, t = tt, mx = mxx, clear(mx);}
    bool bfs(){
        for(int i = 0; i <= mx; i ++) dep[i] = -1;
        int l = 1, r = 0;
        q[++ r] = s, dep[s] = 1;
        while(l <= r){
            int x = q[l ++];
            for(int i = head[x]; i; i = nxt[i]){
                int u = to[i];
                if(dep[u] == -1 && f[i] > 0)
                    q[++ r] = u, dep[u] = dep[x] + 1;
            }
        }
        return dep[t] != -1;
    }
    T dfs(int x, T mr){
        if(x == t || mr == 0) return mr;
        T c = 0, res = 0;
        for(int &i = cur[x]; i; i = nxt[i]){
            int u = to[i];
            if(dep[u] == dep[x] + 1 && (c = dfs(u, min(mr, f[i]))))
                f[i] -= c, f[i^1] += c, mr -= c, res += c;
            if(mr == 0) break;
        }
        return res;
    }
    T dinic(){
        T res = 0;
        while(bfs()){
            for(int i = 0; i <= mx; i ++) cur[i] = head[i];
            res += dfs(s, inf);
        }
        return res;
    }
};
```

4.10 最大流-ISAP

```
typedef int T;

struct gra{
    int head[maxn], to[maxn<<1], nxt[maxn<<1], cnt;
    T f[maxn<<1];
    void clear(int n) {fill(head, head+1+n, 0), cnt = 1;}
    void add(int a, int b, T c){
        nxt[++ cnt] = head[a], head[a] = cnt, to[cnt] = b, f[cnt] = c;
        nxt[++ cnt] = head[b], head[b] = cnt, to[cnt] = a, f[cnt] = 0;
    }
};

struct flow : public gra{
    int dep[maxn], vm[maxn], q[maxn], cur[maxn], s, t, mx;
    void init(int ss, int tt, int mxx){s = ss, t = tt, mx = mxx, clear(mx);}
    void bfs(){
        for(int i = 0; i <= mx; i ++) dep[i] = -1, vm[i] = 0;
        int l = 1, r = 0;
        q[++ r] = t, dep[t] = 1, vm[1] = 1;
        while(l <= r){
            int x = q[l ++];
            for(int i = head[x]; i; i = nxt[i]){
                int u = to[i];
                if(dep[u] == -1)
                    dep[u] = dep[x] + 1, vm[dep[u]] ++, q[++ r] = u;
            }
        }
    }
    T dfs(int x, T mr){
        if(x == t || mr == 0) return mr;
        T c = 0, res = 0;
        for(int &i = cur[x]; i; i = nxt[i]){
            int u = to[i];
            if(f[i] && dep[u] + 1 == dep[x] && (c = dfs(u, min(f[i], mr))))
                f[i] -= c, f[i^1] += c, mr -= c, res += c;
            if(mr == 0) return res;
        }
        if(-- vm[dep[x]] == 0) dep[s] = mx + 1;
        dep[x] ++, vm[dep[x]] ++;
        return res;
    }
    T isap(){
        T res = 0; bfs();
        while(dep[s] <= mx) {
            for(int i = 0; i <= mx; i ++) cur[i] = head[i];
            res += dfs(s, inf);
        }
        return res;
    }
};
```

4.11 最大流-ISAP(非封装版)

```
//注意：te初值为1， mx为总点数， head记得清空
#include<cstdio>
#include<iostream>
#include<cstring>
#include<algorithm>
```

```
#include<string>

using namespace std;

const int N = 510, M = 100010;
const int inf = 0x3f3f3f3f;
int te;
int head[N];
struct edge{
    int u, v, f, nxt;
}e[M];
inline void add(int u, int v, int cap){
    e[++te] = (edge){u, v, cap, head[u]};
    head[u] = te;
}
int dep[N], vm[N], q[N], cur[N], s, t, mx;
void bfs(){
    for (int i = 0; i <= mx; ++i) dep[i] = -1, vm[i] = 0;
    int l = 1, r = 0;
    q[++r] = t, dep[t] = 1, vm[1] = 1;
    while(l <= r){
        int x = q[l++];
        for (int i = head[x]; i; i = e[i].nxt){
            int v = e[i].v;
            if (dep[v] == -1)
                dep[v] = dep[x] + 1, vm[dep[v]]++, q[++r] = v;
        }
    }
}
inline int dfs(int x, int mr){
    if (x == t || mr == 0) return mr;
    int c = 0, res = 0;
    for (int &i = cur[x]; i; i = e[i].nxt){
        int v = e[i].v;
        if (e[i].f && dep[v] + 1 == dep[x] && (c = dfs(v, min(e[i].f, mr))))
            e[i].f -= c, e[i ^ 1].f += c, mr -= c, res += c;
        if (mr == 0) return res;
    }
    if (-- vm[dep[x]] == 0) dep[s] = mx + 1;
    dep[x]++, vm[dep[x]]++;
    return res;
}
int res;
void isap(){
    bfs();
    while(dep[s] <= mx){
        for (int i = 0; i <= mx; ++i) cur[i] = head[i];
        res += dfs(s, inf);
        // cout<<11111<<' '; cout<<res<<endl;
    }
}
int n;
```

4.12 tarjan

4.12.1 求割点割边

```
// 数函数调用init与work
struct Tarjan : public gra{
    int dfn[maxn], low[maxn], st[maxn], scc[maxn], sz[maxn];
    int tp, tim, num, n;
    gra e;
    void init(int N){clear(n = N), tim = 0, num = 0, tp = 0, mem(dfn, 0), mem(scc, 0)
        ;}
    int dfs(int x){
        dfn[x] = low[x] = ++ tim, st[++tp] = x;
        int stx = tp;
        for(int i = head[x]; i; i = nxt[i]){
            int u = to[i];
            if(!dfn[u]) dfs(u), low[x] = min(low[x], low[u]);
            else if(!scc[u]) low[x] = min(low[x], dfn[u]);
        }
        if(dfn[x] == low[x]){
            sz[++ num] = tp - stx + 1;
            while(tp >= stx) scc[st[tp --]] = num;
        }
    }
    void build(){
        unordered_set <int> has[n+1];
        for(int x = 1; x <= n; x ++){
            for(int i = head[x]; i; i = nxt[i]){
                int u = to[i], sx = scc[x], su = scc[u];
                if(has[sx].find(su) == has[sx].end() && sx != su)
                    e.add(sx, su), has[sx].insert(su);
            }
        }
    }
    void work(){
        for(int i = 1; i <= n; i ++) if(!dfn[i]) dfs(i);
        build();
        // 这里写操作
    }
};
```

4.12.2 边双联通分量

```
const int maxn = 1e4 + 10;
int n, m;
struct data {
    int to, next;
} e[maxn * 10];
int head[maxn], cnt = 0;
int cur = 0, top = 0;
int low[maxn], dfn[maxn], vis[maxn], ans[maxn * 10];

void ins(int u, int v) {
    e[++cnt].to = v;
    e[cnt].next = head[u];
    head[u] = cnt;
}

void tarjan(int u, int fa) {
    vis[u] = 1;
    dfn[u] = low[u] = ++cur;
    for (int i = head[u]; i; i = e[i].next) {
        int x = e[i].to;
        if (vis[x] == 1 && x != fa) {
            if (low[u] > dfn[x])low[u] = dfn[x];
            ans[++top] = u;
            ans[++top] = x;
        }
        if (vis[x] == 0) {
            tarjan(x, u);
            if (low[x] < low[u])low[u] = low[x];
            if (low[x] > dfn[u]) {
                ans[++top] = u;
                ans[++top] = x;
                ans[++top] = x;
                ans[++top] = u;
            } else {
                ans[++top] = u;
                ans[++top] = x;
            }
        }
    }
    vis[u] = 2;
}
//tarjan(i ,-1);
```

4.12.3 点双联通分量

```
const int maxn = 5010;
int n, m, q, head[maxn];
struct data {
    int to, next;
} e[maxn * 4];
int cnt = 0, top = 0, tot = 0;
int dfn[maxn], low[maxn], s[maxn], vis[maxn];
vector<int> color[maxn];
int block[maxn];
int block_color = 0;

void ins(int u, int v) {
    e[++cnt].to = v;
    e[cnt].next = head[u];
    head[u] = cnt;
}

void bic(int v, int fa, int dep) {
    vis[v] = 1;
    s[++top] = v;
    block[v] = block_color;
    dfn[v] = low[v] = dep;
    for (int i = head[v]; i; i = e[i].next) {
        int x = e[i].to;
        if (vis[x] == 0) {
            bic(x, v, dep + 1);
            if (low[x] >= dfn[v]) {
                tot++;
                do {
                    color[s[top]].push_back(tot);
                    top--;
                } while (s[top + 1] != x);
                if (low[x] == dfn[v]) color[v].push_back(tot);
            }
            if (low[x] <= low[v]) low[v] = low[x];
        } else if (x != fa && vis[x] == 1) low[v] = min(low[v], dfn[x]);
    }
    vis[v] = 2;
}

//bic(x, -1, 0);
```

4.12.4 极大强连通分量

```
const int maxn = 1e4 + 10;

int n, m;
int cur = 0;
vector<int> v[maxn];
bool vis[maxn];
int low[maxn], dfn[maxn];
int s[maxn];
int top = 0, tot = 0;

void tarjan(int x) {
    s[++top] = x;
    vis[x] = 1;
    dfn[x] = low[x] = ++cur;
    for (int i = 0; i < v[x].size(); ++i) {
        int u = v[x][i];
        if (dfn[u] == 0) tarjan(u);
        if (vis[u] && low[x] > low[u]) low[x] = low[u];
    }
    if (low[x] == dfn[x]) {
        tot = 0;
        while (s[top + 1] != x) {
            top--;
            tot++;
        }
    }
}
```

5 数据结构

5.1 倍增 LCA

```
struct gra {
    int head[maxn], to[maxn << 1], nxt[maxn << 1], cnt;
    void clear(int n) {fill(head, head+1+n, 0), cnt = 0;}
    void add(int a, int b) {nxt[++cnt] = head[a], to[head[a] = cnt] = b;}
};

const int LOGSZ = 19;

struct StLca : public gra{
    T d[maxn][LOGSZ+1], up[maxn][LOGSZ+1];
    int dep[maxn], n, R;
    void init(int N, int root){ n = N, R = root, mem(up[R], 0), dep[R] = 1, clear(n);}
    void CalcLca(){ dfs(R);}
    void dfs(int x){
        for(int i = 1; i <= LOGSZ; i ++) up[x][i] = up[up[x][i-1]][i-1];
        for(int i = head[x]; i; i = nxt[i]){
            int u = to[i]; if(u == up[x][0]) continue;
            dep[u] = dep[x] + 1, up[u][0] = x, dfs(u);
        }
    }
    int lca(int x, int y){
        if(x == y) return x;
        if(dep[x] < dep[y]) swap(x, y);
        for(int i = LOGSZ; i >= 0; i --)
            if(dep[up[x][i]] >= dep[y]) x = up[x][i];
        if(x == y) return x;
        for(int i = LOGSZ; i >= 0; i --)
            if(up[x][i] != up[y][i]) x = up[x][i], y = up[y][i];
        return up[x][0];
    }
};
```

5.2 倍增 LCA(求链值)

```
struct gra {
    int head[maxn], to[maxn << 1], nxt[maxn << 1], cnt;
    T f[maxn << 1];
    void clear(int n) {fill(head, head+1+n, 0), cnt = 0;}
    void add(int a, int b, T c) {nxt[++cnt] = head[a], to[head[a] = cnt] = b, f[cnt] =
        c;}
};

const int LOGSZ = 19;

struct StLca : public gra{
    T d[maxn][LOGSZ+1], up[maxn][LOGSZ+1];
    int dep[maxn], n, R;
    void init(int N, int root, const T *v = nullptr){
        n = N, R = root, mem(up[R], 0), dep[R] = 1, clear(n);
        if(v != nullptr) for(int i = 1; i <= n; i ++) d[i][0] = v[i];
    }
    void CalcLca(){ dfs(R);}
    void dfs(int x){
        for(int i = 1; i <= LOGSZ; i ++){
            up[x][i] = up[up[x][i-1]][i-1];
            d[x][i] = min(d[x][i-1], d[up[x][i-1]][i-1]);
        }
        for(int i = head[x]; i; i = nxt[i]){
            int u = to[i]; if(u == up[x][0]) continue;
            dep[u] = dep[x] + 1, up[u][0] = x;
            d[u][0] = f[i], dfs(u);
        }
    }
    int lca(int x, int y){
        if(x == y) return x;
        if(dep[x] < dep[y]) swap(x, y);
        for(int i = LOGSZ; i >= 0; i --)
            if(dep[up[x][i]] >= dep[y]) x = up[x][i];
        if(x == y) return x;
        for(int i = LOGSZ; i >= 0; i --)
            if(up[x][i] != up[y][i]) x = up[x][i], y = up[y][i];
        return up[x][0];
    }
    T get(int x, int y){
        T res = inf;
        if(dep[x] < dep[y]) swap(x, y);
        for(int i = LOGSZ; i >= 0; i --)
            if(dep[up[x][i]] >= dep[y])
                res = min(res, d[x][i]), x = up[x][i];
        return res;
    }
    T que(int x, int y){
        int LCA = lca(x, y);
        return min(get(x, LCA), get(y, LCA));
    }
};
```

5.3 点分治

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

#define mem(x, v) memset(x, v, sizeof(x))

using namespace std;

typedef long long ll;

const int maxn = 10010;
const int inf = ~0u >> 1u;
//const ll inf = ~0llu >> 1u;


struct gra {
    int head[maxn], to[maxn << 1], nxt[maxn << 1], f[maxn << 1], cnt;
    void clear(int n) { fill(head, head + 1 + n, 0), cnt = 0; }
    void add(int a, int b, int c) { nxt[++cnt] = head[a], to[head[a] = cnt] = b, f[cnt
        ] = c; }
};

struct PointDivide : public gra{
    int sz[maxn], TS, SZmx, root;
    bool vis[maxn], res[maxn];

    void GetRoot(int x, int fa) {
        int SZ = 0;
        sz[x] = 1;
        for (int i = head[x]; i; i = nxt[i]) {
            int u = to[i];
            if (vis[u] || u == fa) continue;
            GetRoot(u, x);
            sz[x] += sz[u];
            if (sz[u] > SZ) SZ = sz[u];
        }
        if (TS - sz[x] > SZ) SZ = TS - sz[x];
        if (SZmx > SZ) SZmx = SZ, root = x;
    }

    int GetSize(int x, int fa) {
        int siz = 1;
        for (int i = head[x]; i; i = nxt[i]) {
            int u = to[i];
            if (vis[u] || u == fa) continue;
            siz += GetSize(u, x);
        }
        return siz;
    }

    void calc(int x, int fa) {

        for (int i = head[x]; i; i = nxt[i]) {
            int u = to[i];
            if (vis[u] || u == fa) continue;
            calc(u, x);
        }
    }

    void sol(int x) {
        vis[x] = true;
        for (int i = head[x]; i; i = nxt[i]) {
            int u = to[i];
            if (vis[u]) continue;

            calc(u, x);
        }

        for (int i = head[x]; i; i = nxt[i]) {
            int u = to[i];
            if (!vis[u]) getAns(u, x);
        }
    }

    void getAns(int x, int fa){ SZmx = inf, TS = GetSize(x, fa), GetRoot(x, fa), sol(
        root); }
}S;



int main() {
    int n, m;
    scanf("%d%d", &n, &m);
    scanf("%d", &n), S.clear(n);
    for (int i = 1, a, b, c; i < n; i++) {
        scanf("%d%d%d", &a, &b, &c);
        S.add(a, b, c), S.add(b, a, c);
    }
    S.getAns(1, 0);

    return 0;
}
```

5.4 树链剖分

```
template <class T>
struct TreeHeavy{
    gra e;
    SegmentTree <T> SeTree;
    int sz[maxn], dep[maxn], fa[maxn], son[maxn], top[maxn];
    int pos[maxn], q[maxn], st[maxn], n, R;
    T *val;
    // 传入“点数”与“根节点编号”
    void init(int N, int root){e.clear(n=N), R = root;}
    // 传入节点初值
    void work(T *v) {
        val = v, fill(son+1, son+1+n, 0);
        int l = 1, r = 0, tp = 0, tim = 0;
        q[++r] = R, dep[R] = 1;
        while (l <= r) {
            int x = q[l++]; sz[x] = 1;
            for (int i = e.head[x]; i; i = e.nxt[i]) {
                int u = e.to[i]; if (u == fa[x]) continue;
                dep[u] = dep[x] + 1, fa[u] = x, q[++r] = u;
            }
        }
        for (int i = n; i >= 1; i--) {
            int x = q[i]; if (!fa[x]) continue;
            sz[fa[x]] += sz[x];
            if (sz[son[fa[x]]] < sz[x]) son[fa[x]] = x;
        }
        st[++tp] = R, top[R] = R;
        while (tp > 0) {
            int x = st[tp--];
            pos[x] = ++tim, q[tim] = val[x];
            for (int i = e.head[x]; i; i = e.nxt[i]) {
                int u = e.to[i]; if (u == fa[x] || u == son[x]) continue;
                st[++tp] = u, top[u] = u;
            }
            if (son[x]) st[++tp] = son[x], top[son[x]] = top[x];
        }
        for(int i = 1; i <= n; i ++) val[i] = q[i];
        SeTree.init(n, val);
    }
    // 链操作 $O(n \log^2 n)$
    T ask(int x, int y) {
        T res = 0;
        while (top[x] != top[y]) {
            if (dep[top[x]] < dep[top[y]]) swap(x, y);
            res = SeTree.merge(res, SeTree.que(pos[top[x]], pos[x]));
            x = fa[top[x]];
        }
        if (dep[x] < dep[y]) swap(x, y);
        res = SeTree.merge(res, SeTree.que(pos[y], pos[x]));
        return res;
    }
    void mdy(int x, int y, T v) {
        while (top[x] != top[y]) {
            if (dep[top[x]] < dep[top[y]]) swap(x, y);
            SeTree.upd(pos[top[x]], pos[x], v);
            x = fa[top[x]];
        }
        if (dep[x] < dep[y]) swap(x, y);
        SeTree.upd(pos[y], pos[x], v);
    }
    // 子树操作 $O(n \log n)$
    T ask(int x) { return SeTree.que(pos[x], pos[x] + sz[x] - 1); }
    void mdy(int x, T v) { SeTree.upd(pos[x], pos[x] + sz[x] - 1, v); }
};
```

5.5 虚树

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

#define mem(x, v) memset(x, v, sizeof(x))

using namespace std;

typedef long long ll;
typedef long long T;
const int maxn = 250010;
const int inf = ~0u >> 1u;
//const ll inf = ~0llu >> 1u;

struct gra {
    int head[maxn], to[maxn << 1], nxt[maxn << 1], cnt;
    T f[maxn << 1];
    void clear(int n) {fill(head, head+1+n, 0), cnt = 0;}
    void add(int a, int b, T c) {nxt[++cnt] = head[a], to[head[a] = cnt] = b, f[cnt] =
        c;}
};

const int LOGSZ = 18;

struct StLca : public gra{
    T d[maxn][LOGSZ+1], up[maxn][LOGSZ+1];
    int dep[maxn], n, R;
    static int dfn[maxn], dct; // 维护dfs序
    void init(int N, int root, const T *v = nullptr){
        n = N, dct = 0, R = root, mem(up[R], 0), dep[R] = 1, clear(n);
        if(v != nullptr) for(int i = 1; i <= n; i ++) d[i][0] = v[i];
    }
    void CalcLca(){ dfs(R);}
    void dfs(int x){
        dfn[x] = ++ dct;
        for(int i = 1; i <= LOGSZ; i ++){
            up[x][i] = up[up[x][i-1]][i-1];
            d[x][i] = min(d[x][i-1], d[up[x][i-1]][i-1]);
        }
        for(int i = head[x]; i; i = nxt[i]){
            int u = to[i]; if(u == up[x][0]) continue;
            dep[u] = dep[x] + 1, up[u][0] = x;
            d[u][0] = f[i], dfs(u);
        }
    }
    int lca(int x, int y){
        if(x == y) return x;
        if(dep[x] < dep[y]) swap(x, y);
        for(int i = LOGSZ; i >= 0; i --)
            if(dep[up[x][i]] >= dep[y]) x = up[x][i];
        if(x == y) return x;
        for(int i = LOGSZ; i >= 0; i --)
            if(up[x][i] != up[y][i]) x = up[x][i], y = up[y][i];
        return up[x][0];
    }
    T get(int x, int y){
        T res = inf;
        if(dep[x] < dep[y]) swap(x, y);
        for(int i = LOGSZ; i >= 0; i --)
            if(dep[up[x][i]] >= dep[y])
                res = min(res, d[x][i]), x = up[x][i];
        return res;
    }
};

int StLca::dfn[maxn] = {0}, StLca::dct = 0;

struct VirTree : public StLca{
    gra e;
    bool vis[maxn], use[maxn];
    int p[maxn], st[maxn], id[maxn], iid[maxn], sz, tp, idt;
    static bool cmp(int x, int y){ return dfn[x] < dfn[y];}
    void reset(){
        while(sz) use[p[sz--]] = false;
        e.clear(idt), idt = 0;
    }
    void push(int x){ p[++sz] = x, use[x] = true; }
    void st_pop(){
        e.add(id[st[tp-1]], id[st[tp]], get(st[tp - 1], st[tp]));
        vis[st[tp --]] = false;
    }
    void st_push(int x){st[++tp] = x, id[x] = ++idt, iid[idt] = x, vis[x] = true;}
    void build(){
        sort(p+1, p+1+sz, cmp),
                st_push(R);
        for(int i = 1; i <= sz; i ++){
            int LCA = lca(st[tp], p[i]);
            while(dep[st[tp-1]] >= dep[LCA]) st_pop();
            if(!vis[LCA]) st_push(LCA), swap(st[tp], st[tp-1]), st_pop();
            st_push(p[i]);
        }
        while(tp > 1) st_pop();
        vis[st[tp --]] = false;
    }
    T dp(int x){
        T res = 0;
        for(int i = e.head[x]; i; i = e.nxt[i]){
            int u = e.to[i];
            if(use[iid[u]]) res += e.f[i];
            else res += min(e.f[i], dp(u));
        }
        return res;
    }
    T dp(){ return dp(R);}
};


int n, m;
VirTree vt;

int main(){
    scanf("%d", &n), vt.init(n, 1);
    for(int i = 1, a, b, c; i < n; i ++){
        scanf("%d%d%d", &a, &b, &c);
        vt.add(a, b, c), vt.add(b, a, c);
    }
    // 预处理LCA
    vt.CalcLca(), scanf("%d", &m);
    for(int i = 1, nn; i <= m; i ++){
        // 每次执行完进行reset
        scanf("%d", &nn), vt.reset();
        for(int j = 1, a; j <= nn; j ++)
            scanf("%d", &a), vt.push(a); // 使用push压入节点信息
        vt.build(), printf("%lld\n", vt.dp()); // build构建，dp求解
    }
    return 0;
```

```
}
```

5.6 Splay（普通平衡树）

```
#include <cstdio>

const int inf = 0x7fffffff;

struct sn{
    int val, cnt, size;
    sn *ch[2], *pre;
    sn(int v = 0);
    void set_ch(int wh, sn *child);
    int wh(){return pre->ch[0] == this ? 0 : 1;}
}*null;

sn::sn(int v){ val = v, size = cnt = 1, pre = ch[0] = ch[1] = null; }
void sn::set_ch(int wh, sn *child) {
    ch[wh] = child;
    if(child != null) child->pre = this;
    size = ch[0]->size + ch[1]->size + cnt;
}
struct Splay{
    sn *root;
    Splay(){
        null = new sn(0);
        null->pre = null->ch[0] = null->ch[1] = null;
        null->size = null->cnt = 0;
        root = null;
    }
    void rotate(sn *now){
        int wh = now->wh();
        sn *fa = now->pre, *gra = now->pre->pre;
        fa->set_ch(wh, now->ch[wh^1]);
        now->set_ch(wh^1, fa), now->pre = gra;
        if(gra != null) gra->ch[gra->ch[0] == fa ? 0 : 1] = now;
    }
    void splay(sn *now, sn *tar){
        for( ; now->pre != tar; rotate(now))
            if(now->pre->pre != tar)
                now->wh() == now->pre->wh() ? rotate(now->pre) : rotate(now);
        if(tar == null) root = now;
    }
    void insert(int x){
        sn *now = root, *ins = new sn(x);
        while(now != null){
            if(now->val == ins->val) {
                now->size ++, now->cnt ++;
                splay(now, null);
                return;
            } else{
                x = ins->val < now->val ? 0 : 1;
                if(now->ch[x] == null) now->set_ch(x, ins), now = null;
                else now = now->ch[x];
            }
        }
        if(root == null) root = ins;
        else splay(ins, null);
    }
    sn *find(int x){
        sn *now = root;
        while(now != null){
            if(now->val == x) break;
            now = now->val < x ? now->ch[1] : now->ch[0];
        }
        if(now != null) splay(now, null);
        return now;
    }
    void del(int x){
        sn *now = find(x);
        if(now == null) return;
        if(now->cnt > 1){now->cnt --, now->size --;return;}
        if(now->ch[0] == null && now->ch[1] == null) {root = null;}
        else if(now->ch[0] == null) root = now->ch[1], now->ch[1]->pre = null;
        else if(now->ch[1] == null) root = now->ch[0], now->ch[0]->pre = null;
        else{
            sn *t = now->ch[0];
            while(t->ch[1] != null) t = t->ch[1];
            splay(t, now);
            t->set_ch(1, now->ch[1]);
            t->pre = null, root = t;
        }
        delete now;
    }
    int get_rank(int x){
        sn *now = find(x);
        if(now == null) return -1;
        return now->ch[0]->size + 1;
    }
    sn* get_kth(int k) {
        sn *now = root;
        int left = k;
        while(now != null){
            if(left <= now->ch[0]->size+now->cnt && left >= now->ch[0]->size+1){
                splay(now, null);
                return now;
            }
            if(left <= now->ch[0]->size) now = now->ch[0];
            else left -= now->ch[0]->size + now->cnt, now = now->ch[1];
        }
        return null;
    }
    sn* pre(int val)const {
        sn *now = root, *ans = null;
        while(now != null){
            if(val <= now->val) now = now->ch[0];
            else {
                if(ans == null || ans->val < now->val) ans = now;
```

```
                    now = now->ch[1];
                }
            }
            return ans;
        }
        sn* nxt(int val)const {
            sn *now = root, *ans = null;
            while(now != null){
                if(val >= now->val) now = now->ch[1];
                else {
                    if(ans == null || ans->val > now->val) ans = now;
                    now = now->ch[0];
                }
            }
            return ans;
        }
}s;
int main(){

    int q; scanf("%d", &q);
    while(q--){
        int order, val;
        scanf("%d%d", &order, &val);
        switch(order){
            case 1: s.insert(val); break;
            case 2: s.del(val); break;
            case 3: printf("%d\n", s.get_rank(val)); break;
            case 4: printf("%d\n", s.get_kth(val)->val); break;
            case 5: printf("%d\n", s.pre(val)->val); break;
            case 6: printf("%d\n", s.nxt(val)->val); break;
        }
    }
    return 0;
}
```

5.7 Splay（文艺平衡树）

```
#include <cstdio>
const int maxn = 100010;
struct node{
    int val, num, size, tag;
    node *pre, *ch[2];
    void update(){size = ch[0]->size + ch[1]->size + 1;}
    void set_ch(int wh, node *child);
    int wh(){return pre->ch[0] == this ? 0 : 1;}
} Pool[maxn], *root, *null;
void node::set_ch(int wh, node *child){
    ch[wh] = child;
    if(child != null) child->pre = this;
    update();
}
void rotate(node *now){
    node *fa = now->pre, *gra = now->pre->pre;
    int wh = now->wh();
    fa->set_ch(wh, now->ch[wh^1]);
    now->set_ch(wh^1, fa);

    now->pre = gra;
    if(gra != null) gra->ch[gra->ch[0] == fa ? 0:1] = now;

}
void splay(node *now, node *tar){
    for( ; now->pre != tar; rotate(now))
        if(now->pre->pre != tar)
            now->wh() == now->pre->wh() ? rotate(now->pre) : rotate(now);
    if(tar == null) root = now;
}
int cnt, ct, n, m;
node *one(int val, int num){
    node *one = &Pool[++cnt];
    one->ch[0] = one->ch[1] = one->pre = null;
    one->val = val, one->num = num;
    one->tag = 0, one->size = 1;
    return one;
}
void change(node *now){
    node *t = now->ch[0];
    now->ch[0] = now->ch[1];
    now->ch[1] = t;
}
void down(node *now){
    if(now->tag == 0) return;
    now->tag ^= 1;
    change(now->ch[0]);
    change(now->ch[1]);
    now->ch[0]->tag ^= 1;
    now->ch[1]->tag ^= 1;

}
void insert(int val){
    node *now = root, *last = null;
    while(now != null){
        last = now, down(now);
        if(val < now->val) now = now->ch[0];
        else now = now->ch[1];
    }
    now = one(val, ++ct);
    if(last == null) root = now;
    else{
        if(val < last->val) last->set_ch(0, now);
        else last->set_ch(1, now);
        splay(now, null);
    }
    return;
}
node *kth(int k){
    int left = k;
    node *now = root;
    while(now != null){
        down(now);
        if(now->ch[0]->size + 1 == left) return now;
        if(now->ch[0]->size + 1 > left) now = now->ch[0];
        else{
            left -= (now->ch[0]->size + 1);
            now = now->ch[1];
        }
    }
    return null;
}

void turn(int l, int r){
    node *ll, *rr, *now;
    *now->ch[0];
    *now->ch[1];
    if(l == 1 && r == n){now->tag ^= 1; return;}
    else if(l == 1){
        rr = kth(r+1);
        splay(rr, null);
        now = rr->ch[0];
        change(now), now->tag ^= 1;
    }else if(r == n){
        ll = kth(l-1);
        splay(ll, null);
        now = ll->ch[1];
        change(now);
        now->tag ^= 1;
    }else{
        ll = kth(l-1);
        rr = kth(r+1);
        splay(ll, null);
        splay(rr, ll);
        now = rr->ch[0];
        change(now), now->tag ^= 1;
    }

}
void print(node *now){
    down(now);
    if(now->ch[0] == null) printf("%d␣", now->num);
    else print(now->ch[0]), printf("%d␣", now->num);
    if(now->ch[1] != null) print(now->ch[1]);
}
int main(){
    null = Pool, null->num = null->size = null->tag = null->val = 0;
    null->pre = null->ch[0] = null->ch[1] = null, root = null;
    scanf("%d%d", &n, &m);
    for(int i = 1; i <= n; i ++)
        insert(i);
    for(int i = 1; i <= m; i ++){
        int l, r;
        scanf("%d%d", &l, &r);
        turn(l, r);
    }
    print(root);
}
```

5.8 Link-Cut-Tree 动态树

```
#include <iostream>
#include <cstdio>
#include <algorithm>
#include <cstring>
using namespace std;

const int maxn = 10010;
int n, m;

struct node{
    bool rev;
    int id;
    node *pre, *ch[2];
    int wh(){return pre->ch[0] == this ? 0 : 1;}
    void set_ch(int wh, node *child);
    void down(){
        if(rev){
            rev ^= 1;
            ch[0]->rev ^= 1;
            ch[1]->rev ^= 1;
            swap(ch[0], ch[1]);
        }
    }
    bool is_root(){return pre->ch[0] != this && pre->ch[1] != this;}
}po[maxn], *null, *st[maxn];

void node::set_ch(int wh, node *child){
    ch[wh] = child;
    if(child != null) child->pre = this;
}

void rotate(node *now){
    node *fa = now->pre, *gra = fa->pre;
    int wh = now->wh();
    if(!fa->is_root()) gra->ch[gra->ch[0] == fa ? 0 : 1] = now;
    fa->set_ch(wh, now->ch[wh^1]);
    now->set_ch(wh^1, fa), now->pre = gra;
}

void splay(node *now){
    int stt = 0; st[++stt] = now;
    for(node *i = now; !i->is_root(); i = i->pre) st[++stt] = i->pre;
    for(int i = stt; i >= 1; i--) st[i]->down();
    for( ; !now->is_root(); rotate(now))
        if(!now->pre->is_root())
            now->wh() == now->pre->wh() ? rotate(now->pre) : rotate(now);
}

```

```
void access(node *x){
    for(node *i = null; x != null; i = x, x = x->pre){
        splay(x), x->set_ch(1, i);
    }
}

void makeroot(node *x){
    access(x), splay(x), x->rev ^= 1;
}

void link(node *x, node *y){
    makeroot(y), y->pre = x;
}

void cut(node *x, node *y){
    makeroot(x), access(y);
    splay(y); y->set_ch(0, null);
    x->pre = null;
}

int find(node *x){
    access(x), splay(x);
    node *now = x;
    while(now->ch[0] != null) now = now->ch[0];
    if(now != null) splay(now);
    return now->id;
}

char ch[10];

int main(){
    null = po; null->pre = null->ch[0] = null->ch[1] = null;
    scanf("%d%d", &n, &m);
    for(int i = 1; i <= n; i ++){
        po[i].id = i, po[i].ch[0] = po[i].ch[1] = po[i].pre = null;
    }
    for(int i = 1; i <= m; i ++){
        int u, v;
        scanf("%s%d%d", ch+1, &u, &v);
        if(ch[1] == 'Q'){
            if(find(&po[u]) == find(&po[v])) printf("Yes\n");
            else printf("No\n");
        }else if(ch[1] == 'C'){
            link(&po[u], &po[v]);
        }else if(ch[1] == 'D'){
            cut(&po[u], &po[v]);
        }
    }
    return 0;
}
```

5.9 主席树

```
const int maxn = 100010;
```

```

struct node{
    node *ch[2];
    int val;
    node *up(){return val = ch[0]->val + ch[0]->val, this;}
}d[100010*32], *rt[maxn];

node *null = d;

int n, m, k;
ull mi[25];
int v[maxn], cnt, w[25], ct;

node *get(int v = 0){
    node *now = &d[++ cnt];
    now->ch[0] = now->ch[1] = null;
    now->val = v;
    return now;
}
#define mid ((l+r)>>1)

void add(node *r1, node *r2, int l, int r, int pos){
    if(l == r) return;
    int wh = pos <= mid ? 0 : 1;
    r2->ch[wh] = get(1+r1->ch[wh]->val);
    r2->ch[wh^1] = r1->ch[wh^1], r2->up();
    add(r1->ch[wh], r2->ch[wh], wh==0?l:mid+1, wh==0?mid:r, pos);
}

bool que(node *r1, node *r2, int l, int r, int pos){
    if(l == r) return r2->val - r1->val;
    int wh = pos <= mid ? 0 : 1;
    return que(r1->ch[wh], r2->ch[wh], wh==0?l:mid+1, wh==0?mid:r, pos);
}
ull tt[maxn];
int main(){
    null->ch[0] = null->ch[1] = null, null->val = 0;
    rt[0] = get();
    // add(rt[i-1], rt[i], 1, ct, val);
    // que(rt[l-1], rt[r-k+1], 1, ct, tp);
    return 0;
}
```

5.10 Splay 套线段树

```
#include <cstdio>
#include <algorithm>

const int inf = 0x7fffffff;
const int maxn = 50010;

using namespace std;

struct sn{
    int val, cnt, size;
    sn *ch[2], *pre;
    sn(int v = 0);
    void set_ch(int wh, sn *child);
    int wh(){return pre->ch[0] == this ? 0 : 1;}
}*null;

sn::sn(int v){ val = v, size = cnt = 1, pre = ch[0] = ch[1] = null; }
void sn::set_ch(int wh, sn *child) {
    ch[wh] = child;
    if(child != null) child->pre = this;
    size = ch[0]->size + ch[1]->size + cnt;
}
bool is_init;
struct Splay{
    sn *root;
    Splay(){
        if(!is_init){
            is_init = true;
            null = new sn(0);
            null->pre = null->ch[0] = null->ch[1] = null;
            null->size = null->cnt = 0;
        }
        root = null;
    }
    void rotate(sn *now){
        int wh = now->wh();
        sn *fa = now->pre, *gra = now->pre->pre;
        fa->set_ch(wh, now->ch[wh^1]);
        now->set_ch(wh^1, fa), now->pre = gra;
        if(gra != null) gra->ch[gra->ch[0] == fa ? 0 : 1] = now;
    }
    void splay(sn *now, sn *tar){
        for( ; now->pre != tar; rotate(now))
            if(now->pre->pre != tar)
                now->wh() == now->pre->wh() ? rotate(now->pre) : rotate(now);
        if(tar == null) root = now;
    }
    void insert(int x){
        sn *now = root, *ins = new sn(x);
        while(now != null){
            if(now->val == ins->val) {
                now->size ++, now->cnt ++;
                splay(now, null);
                return;
            } else{
                x = ins->val < now->val ? 0 : 1;
                if(now->ch[x] == null) now->set_ch(x, ins), now = null;
                else now = now->ch[x];
            }
        }
        if(root == null) root = ins;
        else splay(ins, null);
    }
    sn *find(int x){
        sn *now = root;
        while(now != null){
            if(now->val == x) break;
            now = now->val < x ? now->ch[1] : now->ch[0];
        }
        if(now != null) splay(now, null);
        return now;
    }
    void del(int x){
        sn *now = find(x);
        if(now == null) return;
        if(now->cnt > 1){now->cnt --, now->size --;return;}
        if(now->ch[0] == null && now->ch[1] == null) {root = null;}
        else if(now->ch[0] == null) root = now->ch[1], now->ch[1]->pre = null;
        else if(now->ch[1] == null) root = now->ch[0], now->ch[0]->pre = null;
        else{
            sn *t = now->ch[0];
            while(t->ch[1] != null) t = t->ch[1];
            splay(t, now);
            t->set_ch(1, now->ch[1]);
            t->pre = null, root = t;
        }
        delete now;
    }
    int get_rank(int x){
        sn *now = find(x);
        if(now == null) return -1;
        return now->ch[0]->size + 1;
    }
    sn* pre(int val)const {
        sn *now = root, *ans = null;
        while(now != null){
            if(val <= now->val) now = now->ch[0];
            else {
                if(ans == null || ans->val < now->val) ans = now;
                now = now->ch[1];
            }
        }
        return ans;
    }
    sn* nxt(int val)const {
        sn *now = root, *ans = null;
        while(now != null){
            if(val >= now->val) now = now->ch[1];
            else {
                if(ans == null || ans->val > now->val) ans = now;
                now = now->ch[0];
            }
        }
        return ans;
    }
}s[maxn*6];

```

```
#define mid ((l+r)>>1)
#define lch (now<<1)
#define rch ((now<<1)+1)

int n, m;
int val[maxn];

void build(int now, int l, int r){
    for(int i = l; i <= r; i ++) s[now].insert(val[i]);
    if(l == r) return;
    build(lch, l, mid);
    build(rch, mid+1, r);
}
int que_rank(int now, int l, int r, int pos1, int pos2, int k){
    if(l == pos1 && r == pos2){return s[now].get_rank(k) - 1;}
    if(pos2 <= mid) return que_rank(lch, l, mid, pos1, pos2, k);
    else if(pos1 >= mid+1) return que_rank(rch, mid+1, r, pos1, pos2, k);
    else return que_rank(lch, l, mid, pos1, mid, k) + que_rank(rch, mid+1, r, mid+1,
        pos2, k);
}
int que_pre(int now, int l, int r, int pos1, int pos2, int k){
    if(l == pos1 && r == pos2){return s[now].pre(k)->val;}
    if(pos2 <= mid) return que_pre(lch, l, mid, pos1, pos2, k);
    else if(pos1 >= mid+1) return que_pre(rch, mid+1, r, pos1, pos2, k);
    else return max(que_pre(lch, l, mid, pos1, mid, k), que_pre(rch, mid+1, r, mid+1,
        pos2, k));
}
int que_nxt(int now, int l, int r, int pos1, int pos2, int k){
    if(l == pos1 && r == pos2){return s[now].nxt(k)->val;}
    if(pos2 <= mid) return que_nxt(lch, l, mid, pos1, pos2, k);
    else if(pos1 >= mid+1) return que_nxt(rch, mid+1, r, pos1, pos2, k);
    else return std::min(que_nxt(lch, l, mid, pos1, mid, k), que_nxt(rch, mid+1, r,
        mid+1, pos2, k));
}
int que_kth(int pos1, int pos2, int k){
    int l = 0, r = 1e8, res = -1;
    while(l <= r){
        int v = que_rank(1, 1, n, pos1, pos2, mid)+1;
        if(v > k) res = mid, r = mid - 1;
        else l = mid + 1;
    }
    return que_pre(1, 1, n, pos1, pos2, res);
}
void modify(int now, int l, int r, int pos, int k){
    s[now].insert(k);
    s[now].del(val[pos]);
    if(l == r) return;
    if(pos <= mid) modify(lch, l, mid, pos, k);
    else modify(rch, mid+1, r, pos, k);
}
int main(){
    scanf("%d%d", &n, &m);
    for(int i = 1; i <= n; i ++) scanf("%d", &val[i]);
    build(1, 1, n);
    for(int i = 1; i <= m; i ++){
        int a, b, c, d;
        scanf("%d", &a);
        switch(a){
            case 1:
                scanf("%d%d%d", &b, &c, &d);
                printf("%d\n", que_rank(1, 1, n, b, c, d)+1);
                break;
            case 2:
                scanf("%d%d%d", &b, &c, &d);
                printf("%d\n", que_kth(b, c, d));
                break;
            case 3:
                scanf("%d%d", &b, &c);
                modify(1, 1, n, b, c);
                val[b] = c;
                break;
            case 4:
                scanf("%d%d%d", &b, &c, &d);
                printf("%d\n", que_pre(1, 1, n, b, c, d));
                break;
            case 5:
                scanf("%d%d%d", &b, &c, &d);
                printf("%d\n", que_nxt(1, 1, n, b, c, d));
                break;
        }
    }
    return 0;
}
```

5.11 树状数组套主席树

```
#include <cstdio>
#include <iostream>
#include <algorithm>
using namespace std;
#define mid ((l+r)>>1)
const int maxn = 10010, MX = 1e9;
int n, m, cnt, sz1, sz2, val[maxn];
struct node{
    int val;
    node *ch[2];
}pool[maxn*900], *root[maxn], *null, *A[maxn], *B[maxn];
int lowbit(int x){return x&(-x);}
node *get(){
    node *now = &pool[++cnt];
    now->val = 0, now->ch[0] = now->ch[1] = null;
    return now;
}
void add(node *now, int l, int r, int pos, int v){
    if(l == r) return;
    int wh = 0; if(pos >= mid+1) wh = 1;
    if(now->ch[wh] == null) now->ch[wh] = get();
    now->ch[wh]->val += v;
    add(now->ch[wh], wh==0?l:mid+1, wh==0?mid:r, pos, v);
}
void update(int x, int val, int v){
    for( ; x <= n; x += lowbit(x)){
        if(root[x] == NULL) root[x] = get();
        add(root[x], 0, MX, val, v);
    }
    return;
}
void que(node *fi[], node *se[], int l, int r, int k){
    if(l == r){printf("%d\n", l); return;}
    int lv = 0;
    for(int i = 1; i <= sz1; i ++) lv -= fi[i]->ch[0]->val;
    for(int i = 1; i <= sz2; i ++) lv += se[i]->ch[0]->val;
    if(k <= lv){
        for(int i = 1; i <= sz1; i ++) fi[i] = fi[i]->ch[0];
        for(int i = 1; i <= sz2; i ++) se[i] = se[i]->ch[0];
        que(fi, se, l, mid, k);
    }else{
        for(int i = 1; i <= sz1; i ++) fi[i] = fi[i]->ch[1];
        for(int i = 1; i <= sz2; i ++) se[i] = se[i]->ch[1];
        que(fi, se, mid+1, r, k - lv);
    }
}
void GetRoot(int x, node *C[], int &sz){
    for( ; x >= 1; x -= lowbit(x)){
        C[++sz] = root[x];
    }
}
int main(){
    null = pool;
    null->val = 0, null->ch[0] = null->ch[1] = null;
    root[0] = get();
    scanf("%d%d", &n, &m);
    for(int i = 1; i <= n; i ++){
        scanf("%d", &val[i]);
        update(i, val[i], 1);
    }
    for(int i = 1, a, l, r, k; i <= m; i ++){
        char ch[5];
        scanf("%s", ch+1);
        if(ch[1] == 'Q'){
            sz1 = sz2 = 0;
            scanf("%d%d%d", &l, &r, &k);
            GetRoot(l-1, A, sz1);
            GetRoot(r, B, sz2);
            que(A, B, 0, MX, k);
        }else{
            scanf("%d%d", &a, &k);
            update(a, val[a], -1);
            update(a, k, 1);
            val[a] = k;
        }
    }
    return 0;
}
```

5.12 ST 表

```
int d[maxn], st[maxn][20];

void st_init(){
    for(int i = 1; i <= n; i ++) st[i][0] = d[i];
    for(int i = 1; i <= 17; i ++){
        for(int j = 1; j+(1<<i)-1 <= n; j ++){
            st[j][i] = max(st[j][i-1], st[j+(1<<(i-1))][i-1]);
        }
    }
}
int st_que(int l, int r){
    int k = log2(r - l + 1);
    return max(st[l][k], st[r-(1<<k)+1][k]);
}
```

5.13 ST 表（二维）

```
class st2{
public:
    static const int szX = 302, szY = 302;
    static const int lgX = (int)log2(szX)+1, lgY = (int)log2(szY)+1;
    bool MnOMx; // 0 --> min   1 --> max
    int d[szX][szY][lgX][lgY], n, m;
    int oper(int x, int y){
        if(MnOMx) return x > y ? x : y;
        return x < y ? x : y;
    }
    void init(int sx, int sy, bool MinOrMax){
        n = sx, m = sy, MnOMx = MinOrMax;
        for(int i = 0; i < lgX; i ++)
            for(int j = 0; j < lgY; j ++){
                if(i + j == 0) continue;
                for(int x = 1; x + (1<<i) - 1 <= n; x ++)
                    for(int y = 1; y + (1<<j) - 1 <= m; y ++){
                        if(i == 0) d[x][y][i][j] = oper(d[x][y+(1<<(j-1))][i][j-1], d[
                            x][y][i][j-1]);
                        else d[x][y][i][j] = oper(d[x+(1<<(i-1))][y][i-1][j], d[x][y][
                            i-1][j]);
                    }
            }
    }
    // x1 <= x2 && y1 <= y2
    int ask(int x1, int y1, int x2, int y2){
        int lx = log2(x2-x1+1), ly = log2(y2-y1+1), xx = x2-(1<<lx)+1, yy = y2-(1<<ly)
            +1;
        return oper(oper(d[x1][y1][lx][ly], d[xx][y1][lx][ly]),
                    oper(d[x1][yy][lx][ly], d[xx][yy][lx][ly]));
    }
    void clear(){memset(d, 0, sizeof(d));}
};
```

5.14 树状数组 (二维)

```
struct lb2{
    static const int szX = 1010, szY = 1010;
    int n, m;
    int d[szX][szY];
    inline int lb(int x){return x&(-x);}
    void set_sz(int sizeX=szX-10, int sizeY=szY-10){n = sizeX, m = sizeY;}
    void mdf(int x, int y, int v){
        for(int i = x; i <= n; i += lb(i))
            for(int j = y; j <= m; j += lb(j))
                d[i][j] += v;
    }
    int ask(int x, int y){
        int res = 0;
        for(int i = x; i; i -= lb(i))
            for(int j = y; j; j -= lb(j))
                res += d[i][j];
        return res;
    }
    void clear(){memset(d, 0, sizeof(d));}
};
```

5.15 cdq 分治

```
#include <cstdio>
#include <algorithm>
#include <iostream>
#include <cstring>
using namespace std;
const int maxn = 100010, maxm = 200010;
int n, k, ct, ans[maxn], res[maxn], cnt;
struct node{
    int a, b, c, id, sz;
    bool operator < (const node &t) const{
        if(a == t.a){
            if(b == t.b){
                return c < t.c;
            }else return b < t.b;
        }else return a < t.a;
    }
    bool operator == (const node &t)const{return a == t.a && b == t.b && c == t.c;}
}q[maxn], qq[maxn];
bool cmp(node a, node b){return a.b < b.b;}

int s[maxm];
int lowbit(int x){return x&(-x);}
void update(int x, int v){for(int i = x; i <= k; i += lowbit(i)) s[i] += v;}
int sum(int x){int res = 0; for(int i = x; i; i -= lowbit(i)) res += s[i]; return res
    ;}
```

```

void Solve(int l, int r){
    if(l == r) return;
    int mid = (l+r) >> 1;
    Solve(l, mid), Solve(mid+1, r);
    int i = l, j = mid+1, last = 0;
    while(j <= r){
        while(i <= mid && q[i].b <= q[j].b) update(q[i].c, q[i].sz), last = i ++;
        ans[q[j].id] += sum(q[j].c), j ++;
    }
    for(int i = l; i <= last; i ++) update(q[i].c, -q[i].sz);
    merge(q+l, q+mid+1, q+mid+1, q+r+1, qq+l, cmp);
    for(int i = l; i <= r; i ++) q[i] = qq[i];
}
int main(){
    scanf("%d%d", &n, &k);
    for(int i = 1; i <= n; i ++){
        scanf("%d%d%d", &q[i].a, &q[i].b, &q[i].c);
    }
    sort(q+1, q+1+n);
    node t = q[1]; qq[++ ct] = q[1], qq[ct].sz = 1;
    for(int i = 2; i <= n; i ++){
        if(q[i] == t) qq[ct].sz ++;
        else t = q[i], qq[++ ct] = q[i], qq[ct].sz = 1, qq[ct].id = ++ cnt;
    }
    for(int i = 1; i <= ct; i ++) q[i] = qq[i];
    Solve(1, ct);
    for(int i = 1; i <= ct; i ++) res[ans[q[i].id] + (q[i].sz-1)] += q[i].sz;
    for(int i = 0; i < n; i ++) printf("%d\n", res[i]);
    return 0;
}
```

5.16 KD-TREE 求最近 m 个点（欧几里得距离）

```
//非动态开点
#include<cstdio>
#include<iostream>
#include<cstring>
#include<string>
#include<algorithm>
#include<queue>
using namespace std;

#define sqr(x) (x) * (x)
const int N = 50010;
int n, k, idx;

struct Node{
    int f[5];
    bool operator < (const Node &u) const {
        return f[idx] < u.f[idx];
    }
}_data[N];

```

```
priority_queue<pair<double, Node> > Q;

struct KDT{
    Node val[4 * N];
    int flag[4 * N];
    void Build(int, int, int, int); //data[] 数组表示KDT的所有节点数据
    void Query(Node, int, int, int); // 用于标记某个节点是否存在, 1表示存在, -1表示
        不存在
}kd;

void KDT::Build(int l, int r, int x, int dept){// dept 表示深度
    if (l > r) return;
    flag[x] = 1;
    flag[x << 1] = flag[x << 1 | 1] = -1;
    idx = dept % k;
    int mid = (l + r) >> 1;
    nth_element(_data + l, _data + mid, _data + r + 1);

    val[x] = _data[mid];
    Build(l, mid - 1, x << 1, dept + 1);
    Build(mid + 1, r, x << 1 | 1, dept + 1);
}

void KDT::Query(Node p, int m, int x, int dept){// 寻找离p最近的m个特征属性
    if (flag[x] == -1) return;
    pair<double, Node> cur(0, val[x]);
    for (int i = 0; i < k; ++i)
        cur.first += sqr(cur.second.f[i] - p.f[i]);
    int dim = dept % k; // 保证相同节点dim值不变
    bool fg = 0;        //标记是否需要遍历右子树
    int lson = x << 1;
    int rson = x << 1 | 1;
    if (p.f[dim] >= val[x].f[dim]) swap(lson, rson); //p点dim大于当前数据, 则进入右子
        树
    if (~flag[lson]) Query(p, m, lson, dept + 1);       //节点lson存在, 则进入子树进行
        遍历

    if (Q.size() < m) Q.push(cur), fg = 1; //若队列未满, 放入
    else {
        if (cur.first < Q.top().first) Q.pop(), Q.push(cur);//若找到更小的距离, 替换最
            大距离点数据
        if (sqr(p.f[dim] - val[x].f[dim]) < Q.top().first) fg = 1;
    }
    if (~flag[rson] && fg) Query(p, m, rson, dept + 1);
}
int main()
{
    while(scanf("%d%d", &n, &k) != EOF){
        for (int i = 0; i < n; ++i)
            for (int j = 0; j < k; ++j)
                scanf("%d", &_data[i].f[j]);
        kd.Build(0, n - 1, 1, 0);
        int t, m;
        scanf("%d", &t);
        while(t--){
            Node p;
            for (int i = 0; i < k; ++i) scanf("%d", &p.f[i]);
            scanf("%d", &m);
            while(!Q.empty()) Q.pop();
            kd.Query(p, m, 1, 0);
            printf("the␣closest␣%d␣points␣are:\n", m);
            Node tmp[25];
            for (int i = 0; !Q.empty(); ++i){
                tmp[i] = Q.top().second;
                Q.pop();
            }
            for (int i = m - 1; i >= 0; --i){
                for (int j = 0; j < k; ++j)
                    printf("%d%c", tmp[i].f[j], j == k - 1 ? '\n' : '␣');
            }
        }
    }
    return 0;
}
```

5.17 扫描线

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

#define mem(x, v) memset(x, v, sizeof(x))

using namespace std;

typedef long long ll;

const int inf = ~0u >> 1u;
//const ll inf = ~0llu >> 1u;

const int N = 2097152;

ll n, rk[N], val[N];

struct SNode {
    int l, r;
    ll cnt, len;
};

struct SegmentTree {
#define ls (rt << 1)
#define rs (rt << 1 | 1)
    SNode t[N];

    void pushup(int rt) {
        if (t[rt].cnt) t[rt].len = val[t[rt].r + 1] - val[t[rt].l];
        else t[rt].len = t[ls].len + t[rs].len;
    }

    void build(int rt, int l, int r) {
        t[rt].l = l, t[rt].r = r;
        if (l == r) return;
        int mid = (t[rt].l + t[rt].r) >> 1;
        build(ls, l, mid);
        build(rs, mid + 1, r);
    }

    void add(int rt, int l, int r, int v) {
        if (l <= t[rt].l && t[rt].r <= r) {
            t[rt].cnt += v;
            pushup(rt);
            return;
        }
        int mid = (t[rt].l + t[rt].r) >> 1;
        if (l <= mid) add(ls, l, r, v);
        if (mid < r) add(rs, l, r, v);
        pushup(rt);
    }
} S;


struct node {
    int x, yh, yl, flag;

    bool operator<(const node &t) const {
        if (x != t.x) return x < t.x;
        return flag > t.flag;
    }
} e[N];


// x坐标是直接算的，y坐标是离散化的。

int main() {
    cin >> n;
    ll ans = 0;
    int n2 = n * 2, cnt = 0;
    for (int i = 1; i <= n; i++) {
        ll x1, y1, x2, y2, i2 = i * 2;
        scanf("%lld%lld%lld%lld", &x1, &y1, &x2, &y2);

        e[i2 - 1].x = x1, e[i2].x = x2;
        e[i2 - 1].yh = e[i2].yh = y2;
        e[i2 - 1].yl = e[i2].yl = y1;
        e[i2 - 1].flag = 1, e[i2].flag = -1;

        rk[++cnt] = y1;
        rk[++cnt] = y2;
    }

    sort(rk + 1, rk + n2 + 1);
    cnt = unique(rk + 1, rk + n2 + 1) - rk - 1;

    for (int i = 1; i <= n2; i++) {

        ll pos1 = lower_bound(rk + 1, rk + cnt + 1, e[i].yh) - rk;
        ll pos2 = lower_bound(rk + 1, rk + cnt + 1, e[i].yl) - rk;

        val[pos1] = e[i].yh;
        val[pos2] = e[i].yl;
        e[i].yh = pos1;
        e[i].yl = pos2;
    }

    sort(e + 1, e + n2 + 1);
    S.build(1, 1, n2);

    for (int i = 1; i <= n2; i++) {
        S.add(1, e[i].yl, e[i].yh - 1, e[i].flag);
        ans += S.t[1].len * (e[i + 1].x - e[i].x);
    }
    cout << ans << endl;
    return 0;
}
```

6 计算几何

6.1 计算几何基础

```
#include <cstdio>
#include <iostream>
#include <cstring>
#include <string>
#include <algorithm>
#include <cmath>

using namespace std;
const double inf = 1e18;
const double eps = 1e-9;
inline int dcmp(double x){
    if (fabs(x) <= eps) return 0;
    return x < 0 ? -1 : 1;
}
struct point{
    double x, y;
    point(double x = 0, double y = 0) : x(x), y(y) {}
    bool operator == (const point &t) const {return x == t.x && y == t.y;}
    point operator + (const point &t) const {return point(x + t.x, y + t.y);}
    point operator - (const point &t) const {return point(x - t.x, y - t.y);}
    double operator * (const point &t) const {return x * t.x + y * t.y;}
    point operator * (const double &k) {return point(k*x, k*y);}
    double operator ^ (const point &t) const {return x * t.y - y * t.x;}
    double dis() {return sqrt(x * x + y * y);}
    double dis2() {return x * x + y * y;}
};
typedef point Vector;
//两向量夹角
```

```
double angle(Vector a, Vector b){
    return acos(a * b / a.dis() / b.dis());
}
//向量旋转（逆时针 弧度）
Vector rotate(Vector A, double rad){
    return Vector(A.x * cos(rad) - A.y * sin(rad), A.x * sin(rad) + A.y * cos(rad));
}
//逆时针九十度的法向量
Vector Normal(Vector A){
    double L = A.dis();
    return Vector(-A.y/L, A.x/L);
}
struct Line{
    point v, p;
    Line(point v, point p):v(v), p(p) {}
    //返回点P = V + (P - V) * T;
    point Point(double t){
        return v + (p - v) * t;
    }
};
//两直线交点，需保证两直线不相交（叉积不为0）P to AB
// v ^ w != 0
point Inter(point p, Vector v, point q, Vector w){
    Vector u = p - q;
    double t = (w ^ u) / (v ^ w);
    return p + v * t;
}
//点到直线距离，绝对值为有向距离
double DisToLine(point P, point A, point B){
    Vector v1 = B - A, v2 = P - A;
    return fabs((v1 ^ v2) / v1.dis());
}
//点到线段距离 P to AB
double DisToSeg(point P, point A, point B){
    if (A == B) return (P - A).dis();
    Vector v1 = B - A, v2 = P - A, v3 = P - B;
    if (dcmp(v1 * v2) < 0)
        return v2.dis();
    if (dcmp(v1 * v3) > 0)
        return v3.dis();
    return DisToLine(P, A, B);
}
//P在AB上的投影点
point GetPro(point P, point A, point B){
    Vector v = B - A;
    return A + v * (v * (P - A) / (v * v));
}
//p是否在线段a1a2上
bool OnSeg(point p, point a1, point a2){
    return dcmp((a1 - p) ^ (a2 - p)) == 0 && dcmp((a1 - p) * (a2 - p)) < 0;
}
//判断两线段相交
bool SegInter(point a1, point a2, point b1, point b2){
    double c1 = (a2 - a1) ^ (b1 - a1), c2 = (a2 - a1) ^ (b2 - a1);
    double c3 = (b2 - b1) ^ (a1 - b1), c4 = (b2 - b1) ^ (a2 - b1);
    if (!dcmp(c1) || !dcmp(c2) || !dcmp(c3) || !dcmp(c4)){
        bool f1 = OnSeg(b1, a1, a2);
        bool f2 = OnSeg(b2, a1, a2);
        bool f3 = OnSeg(a1, b1, b2);
        bool f4 = OnSeg(a2, b1, b2);
        bool f = (f1 | f2 | f3 | f4);
        return f;
    }
    return (dcmp(c1) * dcmp(c2) < 0 && dcmp(c3) * dcmp(c4) < 0);
}

// pick定理 2S = 2A + B - 2, A为多边形内部的格点数, B为边上的格点数, 边上的格点数
    : GCD(X, Y).
```

6.2 求凸包

```
typedef vector<point> polygon;
inline int cmp(const point &a, const point &b){
    return a.x == b.x ? a.y < b.y : a.x < b.x;
}
int st[N], ct;
polygon ConvexHull(point *P, int sz){
    polygon res;
    sort(P, P + sz, cmp), ct = 0;
    st[++ct] = 0, st[++ct] = 1;
    for (int i = 2; i < sz; ++i){
        while(ct > 1 && dcmp((P[st[ct]] - P[st[ct-1]]) ^ (P[i] - P[st[ct]])) <= 0)
            ct--;
        st[++ct] = i;
    }
    for (int i = 1; i <= ct; ++i) res.push_back(P[st[i]]);
    ct = 0, st[++ct] = sz - 1, st[++ct] = sz - 2;
    for (int i = sz - 3; i >= 0; --i){
        while(ct > 1 && dcmp((P[st[ct]] - P[st[ct-1]]) ^ (P[i] - P[st[ct]])) <= 0)
            ct--;
        st[++ct] = i;
    }
    for (int i = 2; i < ct; ++i) res.push_back(P[st[i]]);
    return res;
}
```

6.3 旋转卡壳

```
//旋转卡壳, 平面最远点对
inline int caliper(polygon P) {
    int res = 0;
    if (n == 2){
        res = (p[0] - p[1]).dis2();
    } else if (n == 3) {
        res = max((p[0] - p[1]).dis2(), (p[0] - p[2]).dis2());
        res = max(res, (p[1] - p[2]).dis2());
    } else {
        P.push_back(point(0, 0));
        int m = P.size() - 1;
        int j = 2;
        for (int i = 0; i < m; ++i) {
            //万分注意这里一定是小于不能是小于等于，否则有可能被卡
            while (abs((P[i] - P[i + 1]) ^ (P[j] - P[i + 1])) < abs((P[i] - P[i + 1])
                ^ (P[j + 1] - P[i + 1])))
                j = (j + 1) % m;
            res = max(res, (P[i] - P[j]).dis2());
        }
    }
    return res;
}
```

6.4 最小圆覆盖

```
#include <cstdio>
#include <iostream>
#include <string>
#include <cstring>
#include <algorithm>
#include <cmath>
#include <vector>

using namespace std;
const double eps = 1e-9;
const double inf = 1e18;
const int N = 101000;
inline int dcmp(double x){
    if (fabs(x) <= eps) return 0;
    return x < 0 ? -1 : 1;
}
int n;
double r;

struct point{
    ...
}a[N], O;

//最小圆覆盖
bool in_circle(point a){return (a - O).dis() <= r? 1:0;}

inline int calc1(double a, double b, double c, double d, double e, double f){
    O.x = (b*f - d*e) / (b*c - a*d);
    O.y = (c*e - a*f) / (b*c - a*d);
}
inline void min_cover_circle(){
    random_shuffle(a+1, a+n+1);
    for (int i = 1; i <= n; ++i)
        if (!in_circle(a[i])){
            O = a[i], r = 0;
            for (int j = 1; j < i; ++j)
                if (!in_circle(a[j])){
                    O = (a[i] + a[j]) / 2.0;
                    r = (a[i] - O).dis();
                    for (int k = 1; k < j; ++k)
                        if (!in_circle(a[k])) {
                            calc1(a[i].x - a[j].x, a[i].y - a[j].y,a[i].x - a[k].x, a[i].y - a[k].y,
                                  (a[i].dis2() - a[j].dis2())/2.0,(a[i].dis2() - a[k].dis2())/2.0);
                            r = (a[i] - O).dis();
                        }
                }
        }
    printf("%.3f\n", r);
};
```

6.5 半平面交

```
#include<cmath>
#include<cstring>
#include<vector>
#include<cstdio>
#include<iostream>
#include<algorithm>
using namespace std;
const double eps=1e-6;
const int maxn=2e5+10;
const double Pi=acos(-1.00);
inline int dcmp(double x)
{
    if(x>eps)return 1;
    return x<-eps?-1:0;
}
struct Vector
{
    double x,y;
    Vector(double X=0,double Y=0)
    {
        x=X,y=Y;
    }
    bool operator == (const Vector &b)const
    {
        return dcmp(x-b.x)==0&&dcmp(y-b.y)==0;
    }
    double angle()
    {
        return atan2(y,x);//求出极角
    }
};
typedef Vector Point;
Vector operator + (Vector a,Vector b){return Vector(a.x+b.x,a.y+b.y);}
Vector operator - (Vector a,Vector b){return Vector(a.x-b.x,a.y-b.y);}
Vector operator * (Vector a,double b){return Vector(a.x*b,a.y*b);}
Vector operator / (Vector a,double b){return Vector(a.x/b,a.y/b);}
struct Line
{
    Point s,t;
    double ang;
    Line(Point X=Vector(),Point Y=Vector())
    {
        s=X,t=Y,ang=(Y-X).angle();
    }
};
typedef Line Segment;
double dot(Vector a,Vector b)
{
    return a.x*b.x+a.y*b.y;
}
double cross(Vector a,Vector b)
{
    return a.x*b.y-a.y*b.x;
}
bool is_parallel(Line a,Line b)//判断a,b直线是否平行
{
    return dcmp(cross(a.t-a.s,b.t-b.s))==0;
}
Point intersection(Line a,Line b)//求出a,b的交点
{
    return a.s+(a.t-a.s)*(cross(b.t-b.s,a.s-b.s)/cross(a.t-a.s,b.t-b.s));
}
double area(Point *p,int n)//求出多边形的面积
{
    double res=0;
    p[n+1]=p[1];
    for(int i=1;i<=n;i++)res+=cross(p[i],p[i+1]);
    return fabs(res/2);
}
bool operator < (const Line &a,const Line &b)//极角排序，如果极角相同则，选择最靠左的直线
{
    double r=a.ang-b.ang;
    if(dcmp(r)!=0)return dcmp(r)==-1;
    return dcmp(cross(a.t-a.s,b.t-a.s))==-1;
}
bool OnRight(Line a,Point b)//检查b是否在a直线的右边
{
    return dcmp(cross(a.t-a.s,b-a.s))<0;
}
bool SI(Line *l,int n,Point *s,int &m)//增量法求半平面交
{
    static Line que[maxn];
    static Point que2[maxn];//两个双端队列
    int head=0,tail=0;
    sort(l+1,l+1+n);
    que[0]=l[1];
    for(int i=2;i<=n;i++)
        if(dcmp(l[i].ang-l[i-1].ang)!=0)//极角相等的直线，取一个
        {
            if(head<tail&&(is_parallel(que[head],que[head+1])||is_parallel(que[tail],que[tail-1])))return false;//如果两个直线共线，但是极角不同，则没有半平
```

```
                面交
            while(head<tail&&OnRight(l[i],que2[tail-1]))tail--;//如果在直线右边，删除
                点
            while(head<tail&&OnRight(l[i],que2[head]))head++;
            que[++tail]=l[i];
            if(head<tail)que2[tail-1]=intersection(que[tail],que[tail-1]);//加入新点
        }
    while(head<tail&&OnRight(que[head],que2[tail-1]))tail--;//删去多余点
    while(head<tail&&OnRight(que[tail],que2[head]))head++;
    if(tail-head<=1)return false;//只有一个点或零个点，没有半平面交
    que2[tail]=intersection(que[head],que[tail]);//加入最后一条边，和第一条边的交点
    m=0;
    for(int i=head;i<=tail;i++)s[++m]=que2[i];
    return true;
}
const double lim=10000;
int n,m;
Point p[maxn];
Line l[maxn];
double solve()
{
    Point a=Point(0,0);//加入最大限制，防止半平面交无限大
    Point b=Point(lim,0);
    Point c=Point(lim,lim);
    Point d=Point(0,lim);
    l[++n]=Line(a,b);
    l[++n]=Line(b,c);
    l[++n]=Line(c,d);
    l[++n]=Line(d,a);
    if(!SI(l,n,p,m))return 0;
    return area(p,m);
}
int main()
{
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
    {
        Point a,b;
        scanf("%lf%lf%lf%lf",&a.x,&a.y,&b.x,&b.y);
        l[i]=Line(a,b);
    }
    printf("%.1f\n",solve());
}
```

7 其他

7.1 Java 高精度

```
import java.math.BigDecimal;
import java.math.BigInteger;

public class Main {

```

```java
public static void main(String[] args) {
        BigInteger a1 = new BigInteger("1"), a2 = new BigInteger("2"), ans;
        ans = a1.mod(a2);
        ans = a1.add(a2);
        ans = a1.subtract(a2);
        ans = a1.multiply(a2);
        ans = a1.divide(a2);
        System.out.println(ans);
        BigDecimal b1 = new BigDecimal(1), b2 = new BigDecimal(2), res;
        res = b1.add(b2);
        res = b1.subtract(b2);
        res = b1.multiply(b2);
        res = b1.divide(b2, 10, BigDecimal.ROUND_HALF_UP);/*保留10位，并四舍五入*/
        System.out.println(res);
    }
}
```

7.2 整数高精度 (加减乘)

```cpp
const ll mi = 1e9;
const ll mii[] = {1, 10, 100, 1000, 10000, 100000, 1000000, 10000000, 100000000,
    1000000000};

struct Bigint {
    ll d[501];
    bool op; // op == 0为正 1为负
    int sz;
    Bigint(ll x = 0) : sz(0) {
        mem(d, 0), op = x < 0, x = abs(x);
        if (x == 0) sz = 1, op = false;
        else while (x) d[sz++] = x % mi, x /= mi;
    }
    Bigint(const string &s) : sz(0) {
        int lw = 0;
        mem(d, 0), lw += op = (s[0] == '-');
        for (int i = s.length() - 1, j = 0; i >= lw; i--, j++) {
            d[sz] += mii[j] * (s[i] - '0');
            if (j == 8) sz++, j = -1;
        }
        if (sz == 0 || d[sz] != 0) sz++;
        if (sz == 1 && d[0] == 0) op = false;
    }

    inline void up(int p) { d[p + 1] += d[p] / mi, d[p] %= mi; }
    inline void refresh() {
        int i;
        for (i = 0; i < sz || d[i] != 0; i++) up(i);
        sz = i;
    }
    bool NumCmp(const Bigint &t) const {
        if (sz != t.sz) return sz < t.sz;
        for (int i = sz - 1; i >= 0; i--)
            if (d[i] != t.d[i]) return d[i] < t.d[i];
        return false;
    }
    Bigint NumSub(const Bigint &x) const {
        Bigint res = *this;
        for (int i = 0; i < x.sz; i++) res.d[i] -= x.d[i];
        for (int i = 0; i < res.sz || res.d[i] != 0; i++)
            if (res.d[i] < 0) res.d[i] += mi, res.d[i + 1]--;
        while (res.sz > 1 && res.d[res.sz - 1] == 0) res.sz--;
        return res;
    }
    Bigint NumAdd(const Bigint &x) const {
        Bigint res = *this;
        res.sz = max(sz, x.sz);
        int i;
        for (i = 0; i < x.sz; i++) res.d[i] += x.d[i];
        res.refresh();
        return res;
    }
    Bigint NumMul(const Bigint &x) const {
        Bigint res;
        res.sz = sz + x.sz - 1, res.op = op ^ x.op;
        for (int i = 0; i < sz; i++) {
            for (int j = 0; j < x.sz; j++) {
                res.d[i + j] += d[i] * x.d[j];
                res.up(i + j);
            }
        }
        res.refresh();
        return res;
    }
    Bigint flip() const {
        Bigint tmp = *this;
        tmp.op = true;
        return tmp;
    }
    Bigint operator+(const Bigint &x) const {
        if (!(op ^ x.op)) return NumAdd(x);
        if (op == 1 && x.op == 0) {
            if (NumCmp(x)) return x.NumSub(*this);
            else return NumSub(x);
        }
        return x + *this;
    }
    Bigint operator*(const Bigint &x) const { return NumMul(x); }
    Bigint operator-(const Bigint &x) const { return *this + x.flip(); }
    bool operator<(const Bigint &x) const {
        if (op != x.op) return op > x.op;
        return op == 0 == NumCmp(x);
    }
    bool operator>(const Bigint &x) const { return x < *this; }
    void print() {
        if (op) putchar('-');
        printf("%lld", d[sz - 1]);
        for (int i = sz - 2; i >= 0; i--) printf("%09lld", d[i]);
    }
```

```
};
```

7.3 整数读入输出优化

```
#include <iostream>
#include <cstdio>
#include <cctype>

#define SIZE (1 << 21)

#define Getchar() (pr1 == pr2 && (pr2 = (pr1 = fr) + fread(fr, 1, SIZE, stdin), pr1 == pr2) ? EOF : *pr1++)
#define Putchar(ch) (pw < SIZE ? fw[pw++] = (ch) : (fwrite(fw, 1, SIZE, stdout), fw[(pw = 0)++] = (ch)))

char fr[SIZE], * pr1 = fr, * pr2 = fr;
char fw[SIZE];
int pw;

int Read() {
    int res = 0, sign = 1;
    char ch = Getchar();
    while(!isdigit(ch)){if(ch == '-') sign = -1;ch = Getchar();}
    while(isdigit(ch)){res = res * 10 + ch - '0';ch = Getchar();}
    return res * sign;
}
void Write(int val) {
    char a[15];
    int len = 0;
    if(val < 0) {val = -val;Putchar('-');}
    do {a[++len] = val % 10 + '0';val /= 10;}
    while(val);

    while(len){Putchar(a[len--]);}
    return;
}
int main() {
    // program...
    return 0;
}
```

7.4 程序内开栈

```
#pragma comment(linker, "/STACK:102400000,102400000")
```