template

WUST

So Like Coding? You Baldy

August 20, 2019

Contents

0 Header

0.1 pbds

0.1.1 head

```cpp
#include <bits/extc++.h>
#pragma comment(linker, "/STACK:102400000,102400000")
using namespace __gnu_pbds; // tree, gp_hash_table, trie
using namespace __gnu_cxx; // rope
tree<TYPE, null_type, less<>, rb_tree_tag, tree_order_statistics_node_update> tr;
// 可并堆
#include <ext/pb_ds/priority_queue.hpp>
using namespace __gnu_pbds;
__gnu_pbds::priority_queue<int,greater<int>,pairing_heap_tag> q[MAX];
//q[i].join(q[j]) 将j堆并入i

```

0.2 FastIO

0.2.1 FastScanner

```cpp
// 适用于正负整数
template <class T>
inline bool scan(T &ret){
    char c;
    int sgn;
    if (c = getchar(), c == EOF) return 0; //EOF
    while (c != '-' && (c < '0' || c > '9')) c = getchar();
    sgn = (c == '-') ? -1 : 1;
    ret = (c == '-') ? 0 : (c - '0');
    while (c = getchar(), c >= '0' && c <= '9') ret = ret * 10 + (c - '0');
    ret *= sgn;
    return 1;
}

template <class T>
inline void out(T x) {
    if (x > 9) out(x / 10);
    putchar(x % 10 + '0');
}

inline int read() {
    int x = 0;
    char ch = getchar();
    while (ch > '9' || ch < '0')ch = getchar();
    while (ch >= '0' && ch <= '9') {
        x = x * 10 + ch - '0';
        ch = getchar();
    }
    return x;
}

// 代替gets
scanf("%[^\n]%*c", ss)

// python一行读入
a,b = map(int, input().split())

```

```
a = []
for i in input().split():
    a.append(int(i))
```

0.2.2 FastPowAndAdd

```
// 精确快速乘
ll qpmul(ll a, ll b) {
    a %= mod; b %= mod;
    ll res = 0;
    while (b > 0) {
        if (b & 1) {
            res = (res + a);
            if (res >= mod) res -= mod;
        }
        a = (a + a);
        if (a >= mod) a -= mod;
        b >>= 1;
    }
    return res;
}

// O(1)快速乘
ll mul2(ll x,ll y,ll p) {
    ll res=(x*y-ll((long double)x/p*y+1.0e-8)*p);
    return res<0?res+p:res;
}

//int128
ll ans = ((__int128) a * b) % p;

// 10进制快速幂, 直接读入%s,c 预处理字符串len
char c[1000005], len;
ll qp(ll a) {
    len --;
    a %= mod;
    ll s = a;
    ll res = 1;
    while (len >= 0) {
        ll cur = s;
        for (int i = 1; i <= c[len] - '0'; ++i) {
            res = res * s % mod;
        }
        for (int i = 1; i < 10; ++i) {
            cur = cur * s % mod;
        }
        s = cur;
        len --;
    }
    return res;
}
```

0.3 header

```
#include <bits/stdc++.h>
#define ll long long
using namespace std;
```

```
4
5  template <class T>
6  inline bool scan(T &ret){
7      char c;
8      int sgn;
9      if (c = getchar(), c == EOF) return 0; //EOF
10     while (c != '-' && (c < '0' || c > '9')) c = getchar();
11     sgn = (c == '-') ? -1 : 1;
12     ret = (c == '-') ? 0 : (c - '0');
13     while (c = getchar(), c >= '0' && c <= '9') ret = ret * 10 + (c - '0');
14     ret *= sgn;
15     return 1;
16 }
17
18 const ll mod = 998244353;
19 const int maxn = 1e5+5;
20
21 ll qp(ll x, ll n) {
22     ll res = 1; x %= mod;
23     while (n > 0) {
24         if (n & 1) res = res * x % mod;
25         x = x * x % mod;
26         n >>= 1;
27     }
28     return res;
29 }
30
31 int main() {
32     return 0;
33 }
```

1 Math

1.1 素数

1.1.1 Eratosthenes 筛法

```
const int maxn = 1e6 + 10;
bool vis[maxn];
vector<int> prime;

void init()
{
    for(int i = 2; i < maxn; i ++) if(!vis[i])
        for(int j = i * i; j < maxn; j += i) vis[i] = true;
    for(int i = 2; i < maxn; i ++) if(!vis[i]) prime.push_back(i);
}
```

1.1.2 Euler 筛

```
const int maxn = 1e6 + 10;
int prime[maxn], v[maxn], n, cnt;   //每个合数只会被它的最小质因子p筛一次

void Euler_Sieve()
{
    for(int i = 2; i <= n; i ++)
    {
        if(!v[i]) v[i] = i, prime[++cnt] = i;
        for(int j = 1; j <= cnt && i * prime[j] <= n; j ++)
        {
            v[i * prime[j]] = prime[j];
            if(i % prime[j] == 0) break;
        }
    }
}
```

1.1.3 MillerRabin 素性测试

```
typedef long long ll;

bool check(ll a, ll n)
{
    if(n == 2 || a >= n) return true;
    if(n == 1 || !(n & 1)) return false;
    ll d = n - 1;
    while(!(d & 1)) d >>= 1;
    ll t = qp(a, d, n);
    while(d != n - 1 && t != 1 && t != n - 1)
    {
        t = mul(t, t, n);
        d <<= 1;
    }
    return t == n - 1 || d & 1;
}

bool Miller_Rabin(ll n)
{
    static vector<ll> t = {2, 325, 9375, 28178, 450775, 9780504, 1795265022};
    if (n <= 1) return false;
```

```
    for (ll k: t) if (!check(k, n)) return false;
    return true;
}
```

1.1.4 PollardRho 快速因数分解

```
mt19937 mt(time(0));
LL pollard_rho(LL n, LL c) {
    LL x = uniform_int_distribution<LL>(1, n - 1)(mt), y = x;
    auto f = [&](LL v) { LL t = mul(v, v, n) + c; return t < n ? t : t - n; };
    while (1) {
        x = f(x); y = f(f(y));
        if (x == y) return n;
        LL d = gcd(abs(x - y), n);
        if (d != 1) return d;
    }
}

LL fac[100], fcnt;
void get_fac(LL n, LL cc = 19260817) {
    if (n == 4) { fac[fcnt++] = 2; fac[fcnt++] = 2; return; }
    if (Miller_Rabin(n)) { fac[fcnt++] = n; return; }
    LL p = n;
    while (p == n) p = pollard_rho(n, --cc);
    get_fac(p); get_fac(n / p);
}
```

1.2 约数

1.2.1 EulerPhi

```
//计算欧拉phi函数，phi(n)且与n互素的正整数个数

int oula(int n){
    int rea=n;
    for(int i=2; i*i<=n; i++)
        if(n%i==0)//第一次找到的必为素因子
        {
            rea=rea-rea/i;
            do
                n/=i;//把该素因子全部约掉
            while(n%i==0);
        }
    if(n>1)
        rea=rea-rea/n;
    return rea;
} //单点欧拉 O(sqrt(n))

bool boo[50000];
int p[20000];
void prim(){
    memset(boo,0,sizeof(boo));
    boo[0]=boo[1]=1;
    int k=0;
    for(int i=2; i<50000; i++)
    {
        if(!boo[i])
            p[k++]=i;
```

```
28          for(int j=0; j<k&&i*p[j]<50000; j++)
29          {
30              boo[i*p[j]=1;
31              if(!(i%p[j]))
32                  break;
33          }
34      }
35  }//筛选法打表
36  int phi(int n)
37  {
38      int rea=n;
39      for(int i=0; p[i]*p[i]<=n; i++)//对于一些不是素数的可不遍历
40          if(n%p[i]==0)
41          {
42              rea=rea-rea/n;
43              do
44                  n/=p[i];
45              while(n%p[i]==0);
46          }
47      if(n>1)
48          rea=rea-rea/n;
49      return rea;
50  } //素数+欧拉
51
52  int euler[maxn];
53  void init() {
54      int i, j;
55      for(i=1; i<maxn; i++)
56          euler[i]=i;
57      for(i=2; i<maxn; i+=2)
58          euler[i]/=2;
59      for(i=3; i<maxn; i+=2)
60          if(euler[i]==i) {
61              for(j=i; j<=maxn; j+=i)
62                  euler[j]=euler[j]/i*(i-1);
63          }
64  } //递推欧拉表
```

1.2.2 Sieve

```
1  //用类似筛法的方法计算phi(1),phi(2),...,phi(n)
2  int phi[maxn];
3
4  void phi_table(int n)
5  {
6      for (int i = 2; i <= n; i++) phi[i] = 0;
7      phi[1] = 1;
8      for (int i = 2; i <= n; i++) if (!phi[i])
9          for (int j = i; j <= n; j += i)
10         {
11             if (!phi[j]) phi[j] = j;
12             phi[j] = phi[j] / i * (i - 1);
13         }
14 }
```

1.2.3 gcd

```
ll gcd(ll a,ll b) {while(b^=a^=b^=a%=b);return a;}

void exgcd(ll a, ll b, ll& x, ll& y, ll& c) {
    if(!b) {y = 0; x = 1; c = a; return;}
    exgcd(b, a % b, y, x); y -= a / b * x;
}
```

1.2.4 解乘法逆元

```
void exgcd(ll a, ll b, ll c, ll d, ll &x, ll &y) {
    ll z = (a + b - 1) / b;
    if (z <= c / d) {
        x = z;
        y = 1;
        return;
    }
    a -= (z - 1) * b; c -= (z - 1) * d;
    exgcd(d, c, b, a, y, x);
    x += (z - 1) * y;
}

int main(int argc, char* argv[]) {
    int T;
    scanf("%d", &T);
    ll p, x;
    for (int kase = 1; kase <= T; ++kase) {
        scanf("%lld%lld", &p, &x);
        ll b, y;
        exgcd(p, x, p, x - 1, b, y);
        printf("%lld/%lld\n", b * x - p * y, b);
    }
    return 0;
}
```

1.3 同余

1.3.1 扩展欧几里得算法

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

1.3.2 中国剩余定理

```
typedef long long ll;

void exgcd(ll a, ll b, ll &x, ll &y)
{
    if(b == 0) { x = 1; y = 0; return; }
    exgcd(b, a % b, x, y);
    ll t = x; x = y, y = t - a / b * y;
}

ll crt(ll *a, ll *m, int n)
```

```
{
    ll M = 1, ans = 0;
    for(int i = 1; i <= n; i ++) M *= m[i];
    for(int i = 1; i <= n; i ++)
    {
        ll x = 0, y = 0;
        ll Mi = M / m[i];
        exgcd(Mi, m[i], x, y);
        ans = (ans + Mi % M * x % M * a[i] % M + M) % M;
    }
    if(ans < 0) ans += M;
    return ans;
}
```

1.3.3 扩展中国剩余定理

```
typedef long long ll;

const int N = 1e5 + 10;

int n;
ll a[N], r[N];

ll exgcd(ll a, ll b, ll& x, ll& y)
{
    if(b == 0) { x = 1, y = 0; return a; }
    ll ret = exgcd(b, a % b, y, x); y -= a / b * x;
    return ret;
}

ll excrt()
{
    ll M = a[1], R = r[1], x, y, d;
    for(int i = 2; i <= n; i ++)
    {
        d = exgcd(M, a[i], x, y);
        if((R - r[i]) % d) return -1;
        x = (R - r[i]) / d * x % a[i];
        R -= M * x;
        M = M / d * a[i];
        R %= M;
    }
    return (R % M + M) % M;
}
```

1.3.4 BSGS

```
int qp(int a, int n, int mod)
{
    long long ans = 1, base = a;
    while(n)
    {
        if(n & 1) (ans *= base) %= mod;
        (base *= base) %= mod;
        n >>= 1;
    }
    return ans;
}
```

```cpp

int BSGS(int a, int b, int p)
{
    map<int, int> hash;
    b %= p;
    int t = (int)sqrt(p) + 1;
    for(int j =0; j < t; j ++)
    {
        int val = 1ll * b * qp(a, j, p) % p;
        hash[val] = j;
    }
    a = qp(a, t, p);
    if(a == 0) return b == 0 ? 1 : -1;
    for(int i = 0; i <= t; i ++)
    {
        int val = qp(a, i, p);
        int j = hash.find(val) == hash.end() ? -1 : hash[val];
        if(j >= 0 && i * t - j >= 0) return i * t - j;
    }
    return -1;
}
```

1.3.5 逆元

```cpp
/*
1.费马小定理
条件:mod为素数
*/
ll inv(ll x){return qp(x,mod-2);}

/*
2.扩展欧几里得
条件:gcd(a,mod)==1
如果gcd(a,mod)!=1 返回-1
*/
ll inv(ll a,ll p)
{
    ll g,x,y;
    g=exgcd(a,p,x,y);
    return g==1?(x+p)%p:-1;
}

/*
3.公式
a/b%mod=c
->a%(b*mod)/b=c
*/

/*
4.逆元打表
p是模
p要求是奇素数
*/
ll inv[MAX];
void getinv(int n,ll p)
{
    ll i;
    inv[1]=1;
```

```
    for(i=2;i<=n;i++) inv[i]=(p-p/i)*inv[p%i]%p;
}

// log逆元
ll dlog(ll g, ll b, ll p) {
    ll m = sqrt(p - 1);
    map<ll, ll> powers;
    for (long j = 0; j < m; j++) powers[qp(g, j, p)] = j;
    long gm = qp(g, -m + 2 * (p - 1), p);
    for (int i = 0; i < m; i++) {
        if (powers[b]) return i * m + powers[b];
        b = b * gm % p;
    }
    return -1;
}
```

1.3.6 模素数二次同余方程

```
// 要求模为素数，输入n，mod，返回 x^2 % mod = n，可解任意一次二元方程

bool Legendre(ll a,ll p) {
    return qp(a,p-1>>1,p)==1;
}

ll modsqr(ll a,ll p) {
    ll x;
    ll i,k,b;
    if(p==2) x=a%p;
    else if(p%4==3) x=qp(a,p+1>>2,p);
    else {
        for(b=1;Legendre(b,p);++b);
        i=p-1>>1;
        k=0;
        do
        {
            i>>=1;
            k>>=1;
            if(!((1LL*qp(a,i,p)*qp(b,k,p)+1)%p)) k+=p-1>>1;
        }while(!(i&1));
        x=1ll*qp(a,i+1>>1,p)*qp(b,k>>1,p)%p;
    }
    return min(x, p - x);
//    if(p-x<x) x=p-x;
//    if(x==p-x) printf("%d\n",x);
//    else printf("%d %d\n",x,p-x);
}
```

1.4 矩阵与线性方程组

1.4.1 矩阵快速幂

```
const int mod = 1e9 + 7;
typedef long long ll;

int cur;
struct Matrix {ll a[105][105]; };

Matrix mul(Matrix a, Matrix b)
```

```
{
    Matrix res;
    memset(res.a, 0, sizeof res.a);
    for(int i = 0; i < cur; i++)
        for(int j = 0; j < cur; j++)
            for(int k = 0; k < cur; k++)
                (res.a[i][j] +=a.a[i][k] * b.a[k][j] % mod) %= mod;
    return res;
}

Matrix pow(Matrix a, ll n)
{
    Matrix ans, base = a;
    for(int i = 0; i < cur; i++) ans.a[i][i] = 1;
    while(n)
    {
        if(n & 1) ans = mul(ans, base);
        base = mul(base, base);
        n >>= 1;
    }
    return ans;
}
```

1.4.2 高斯消元

```
const int N = 20 + 10;

int n;
double b[N], c[N][N];
//c: 系数矩阵, b: 常数;  二者一起构成增广矩阵

void Gaussian_Elimination()
{
    for(int i = 1; i <= n; i ++)
    {
        //找到x[i]的系数不为0的一个方程
        for(int j = i; j <= n; j ++) if(fabs(c[j][i]) > 1e-8)
        {
            for(int k = 1; k <= n; k ++) swap(c[i][k], c[j][k]);
            swap(b[i], b[j]);
        }
        //消去其他方程的x[i]的系数
        for(int j = 1; j <= n; j ++)
        {
            if(i == j) continue;
            double rate = c[j][i] / c[i][i];
            for(int k = i; k <= n; k ++) c[j][k] -= c[i][k] * rate;
            b[j] -= b[i] * rate;
        }
    }
}
```

1.4.3 线性基

```
typedef long long ll;

struct LinearBasis
{
```

```
    ll d[64], tot;

    void ins(ll x)  //插入线性基
    {
        for(int i = 63; i >= 0; i --)
        {
            if((x >> i) & 1)
            {
                if(!d[i]) return void(d[i] = x);
                x ^= d[i];
            }
        }
    }

    ll max_xor()    //在一个序列中取若干个数，使其异或和最大
    {
        ll ans = 0;
        for(int i = 63; i >= 0; i --)
            if((ans ^ d[i]) > ans) ans ^= d[i];
        return ans;
    }

    void init()
    {
        for(int i = 0; i < 64; i ++) if(d[i])
            for(int j = 0; j < i; j ++)
                if(d[i] & (1ll << j)) d[i] ^= d[j];
        for(int i = 0; i < 64; i ++) if(d[i]) d[tot ++] = d[i];
    }

    ll k_th(ll k)   //取任意个元素进行异或的第k小个数
    {
        //考虑能异或出0的情况，tot表示线性基中的元素个数
        k -= (n != tot);
        if(k > (1ll << tot)) return -1;
        ll ans = 0;
        for(int i = 0; i < tot; i ++) if(k & (1ll << i)) ans ^= d[i];
        return ans;
    }
};
```

1.5 组合数学

1.5.1 Lucas

```
const int maxn = 1e6 + 10;

ll fac[maxn], inv[maxn], facinv[maxn];

void init()
{
    fac[0] = inv[0] = facinv[0] = 1;
    fac[1] = inv[1] = facinv[1] = 1;
    for(int i = 2; i < maxn; i++)
    {
        fac[i] = fac[i - 1] * i % mod;
        inv[i] = mod - mod / i * inv[mod % i] % mod;
        facinv[i] = facinv[i - 1] * inv[i] % mod;
    }
```

```
}

ll C(int n, int k)
{
    if(k > n || k < 0) return 0;
    return fac[n] * facinv[k] % mod * facinv[n - k] % mod;
}

ll lucas(ll n, ll m)
{
    ll res = 1;
    while(n && m)
    {
        res = res * C(n % mod, m % mod) % mod;
        n /= mod;
        m /= mod;
    }
    return res;
}
```

1.5.2 小模数组合数

p 小 n,m 大

```

const int NICO = 100000+10;
const int MOD  = 99991;
ll f[NICO];

ll Lucas(ll a,ll k)
{
    ll res = 1;
    while(a && k)
    {
        ll a1 = a % MOD;
        ll b1 = k % MOD;
        if(a1 < b1) return 0;
        res = res*f[a1]*qp(f[b1]*f[a1-b1]%MOD,MOD-2)%MOD;
        a /= MOD;
        k /= MOD;
    }
    return res;
}

void init()
{
    f[0] = 1;
    for(int i=1;i<=MOD;i++)
    {
        f[i] = f[i-1]*i%MOD;
    }
}

int main()
{
    init();
    cout << Lucas(5,2) << endl;
}
```

1.5.3 递推组合数

$0 \le m \le n \le 1000$

```
const int maxn = 1010;
ll C[maxn][maxn];
void init() {
    C[0][0] = 1;
    for (int i = 1; i < maxn; i++)
    {
        C[i][0] = 1;
        for (int j = 1; j <= i; j++) C[i][j] = (C[i - 1][j - 1] + C[i - 1][j]) % mod;
    }
}
```

$0 \le m \le n \le 10^5$, 模 p 为素数

```
const int maxn = 100010;
ll f[maxn];
ll inv[maxn]; // 阶乘的逆元
void CalFact() {
    f[0] = 1;
    for (int i = 1; i < maxn; i++) f[i] = (f[i - 1] * i) % p;
    inv[maxn - 1] = qp(f[maxn - 1], p - 2);
    for (int i = maxn - 2; ~i; i--) inv[i] = inv[i + 1] * (i + 1) % p;
}
ll C(int n, int m) { return f[n] * inv[m] % p * inv[n - m] % p; }
```

1.5.4 大模数组合数

n,m 小 p 大

```
map<int, ll> m;

const int MOD = 1e9+7;
void fun(int n, int k) {
    for (int i = 2; i <= sqrt(n * 1.0); i++) {
        while (n % i == 0) {
            n /= i;
            m[i] += k;
        }
    }
    if (n > 1) {
        m[n] += k;
    }
}

ll C(ll a, ll b) {
    if (a < b || a < 0 || b < 0)
        return 0;
    m.clear();
    ll ret = 1;
    b = min(a - b, b);
    for (int i = 0; i < b; i++) {
        fun(a - i, 1);
    }
    for (int i = b; i >= 1; i--) {
        fun(i, -1);
    }
```

```
    for (__typeof(m.begin()) it = m.begin(); it != m.end(); it++) {
        if ((*it).second != 0) {
            ret *= qp((*it).first, (*it).second);
            ret %= MOD;
        }
    }
    return ret;
}

int main(int argc,char *argv[])
{
    ll a, b;
    while (scanf("%lld%lld", &a, &b) != EOF) {
        printf("%lld\n", C(a, b));
    }
    return 0;
}
```

1.6 卷积

1.6.1 FFT

```
const int maxn = 1e7 + 10;
const double Pi = acos(-1.0);

struct complex
{
    double x, y;
    complex (double xx = 0, double yy = 0) { x = xx, y = yy; }
}a[maxn], b[maxn];

complex operator + (complex a, complex b) { return complex(a.x + b.x, a.y + b.y); }
complex operator - (complex a, complex b) { return complex(a.x - b.x, a.y - b.y); }
complex operator * (complex a, complex b) { return complex(a.x * b.x - a.y * b.y, a.x * b.y + a.y *
     b.x); }

int n, m;
int l, r[maxn];
int limit;

void FFT(complex *A, int type)
{
    for(int i = 0; i < limit; i ++)
        if(i < r[i]) swap(A[i], A[r[i]]);
    for(int mid = 1; mid < limit; mid <<= 1)
    {
        complex Wn(cos(Pi / mid), type * sin(Pi / mid));
        for(int R = mid << 1, j = 0; j < limit; j += R)
        {
            complex w(1, 0);
            for(int k = 0; k < mid; k ++, w = w * Wn)
            {
                complex x = A[j + k], y = w * A[j + mid + k];
                A[j + k] = x + y;
                A[j + mid + k] = x - y;
            }
        }
    }
}
```

```

void mul()
{
    l = 0, limit = 1;
    while(limit <= n + m) limit <<= 1, l ++;
    for(int i = 0; i < limit; i ++)
        r[i] = (r[i >> 1] >> 1) | ( (i & 1) << (l - 1));
    FFT(a, 1);
    FFT(b, 1);
    for(int i = 0; i <= limit; i ++) a[i] = a[i] * b[i];
    FFT(a, -1);
    for(int i = 0; i <= n + m; i ++)
        printf("%d ", (int)(a[i].x / limit + 0.5));
}
```

1.6.2 NTT

```
const int maxn = 2097152;
const int mod = 998244353;
const int root = 3;
// 998244353 -> 3, 1e9+7 -> 5,

template<long long mod, long long root>
struct NTT {
    vector<long long> omega;

    NTT() {
        omega.resize(maxn + 1);
        long long x = fpow(root, (mod - 1) / maxn);
        omega[0] = 1ll;
        for (int i = 1; i <= maxn; ++i)
            omega[i] = omega[i - 1] * x % mod;
    }

    long long fpow(long long a, long long n) {
        (n += mod - 1) %= mod - 1;
        long long r = 1;
        for (; n; n >>= 1) {
            if (n & 1) (r *= a) %= mod;
            (a *= a) %= mod;
        }
        return r;
    }

    void bitrev(vector<long long> &v, int n) {
        int z = __builtin_ctz(n) - 1;
        for (int i = 0; i < n; ++i) {
            int x = 0;
            for (int j = 0; j <= z; ++j) x ^= (i >> j & 1) << (z - j);
            if (x > i) swap(v[x], v[i]);
        }
    }

    void ntt(vector<long long> &v, int n) {
        bitrev(v, n);
        for (int s = 2; s <= n; s <<= 1) {
            int z = s >> 1;
            for (int i = 0; i < n; i += s) {
```

```
                for (int k = 0; k < z; ++k) {
                    long long x = v[i + k + z] * omega[maxn / s * k] % mod;
                    v[i + k + z] = (v[i + k] + mod - x) % mod;
                    (v[i + k] += x) %= mod;
                }
            }
        }
    }

    void intt(vector<long long> &v, int n) {
        ntt(v, n);
        for (int i = 1; i < n / 2; ++i) swap(v[i], v[n - i]);
        long long inv = fpow(n, -1);
        for (int i = 0; i < n; ++i) (v[i] *= inv) %= mod;
    }

    vector<long long> operator()(vector<long long> a, vector<long long> b) {
        int sz = 1;
        while (sz < a.size() + b.size() - 1) sz <<= 1;
        while (a.size() < sz) a.push_back(0);
        while (b.size() < sz) b.push_back(0);
        ntt(a, sz), ntt(b, sz);
        vector<long long> c(sz);
        for (int i = 0; i < sz; ++i) c[i] = a[i] * b[i] % mod;
        intt(c, sz);
        while (c.size() && c.back() == 0) c.pop_back();
        return c;
    }

    vector<long long> operator()(vector<long long> a, int n) {
        int sz = 1;
        while (sz < n * a.size()) sz <<= 1;
        while (a.size() < sz) a.push_back(0);
        ntt(a, sz);
        for (int i = 0; i < sz; ++i) a[i] = fpow(a[i], n);
        intt(a, sz);
        while (a.size() && a.back() == 0) a.pop_back();
        return a;
    }
};

NTT<mod, root> conv;
```

1.6.3 原根

```
#include<bits/stdc++.h>
#define ll long long
#define IL inline
#define RG register
using namespace std;

ll prm[1000],tot,N,root;

ll Power(ll bs,ll js,ll MOD){
    ll S = 1,T = bs;
    while(js){
        if(js&1)S = S*T%MOD;
        T = T*T%MOD;
```

```
        js >>= 1;
    } return S;
}

IL ll GetRoot(RG ll n){
    RG ll tmp = n - 1 , tot = 0;
    for(RG ll i = 2; i <= sqrt(tmp); i ++){
        if(tmp%i==0){
            prm[++tot] = i;
            while(tmp%i==0)tmp /= i;
        }
    }
    if(tmp != 1)prm[++tot] = tmp;            //质因数分解
    for(RG ll g = 2; g <= n-1; g ++){
        bool flag = 1;
        for(RG int i = 1; i <= tot; i ++){     //检测是否符合条件
            if(Power(g,(n-1)/prm[i],n) == 1)
                { flag = 0; break; }
        }
        if(flag)return g;
    }return 0;                          //无解
}

int main(){
    cin >> N;
    root = GetRoot(N);
    cout<<root<<endl;
    return 0;
}
```

1.7 多项式

1.7.1 拉格朗日插值

```
typedef long long ll;

const int mod = 998244353;
const int maxn = 1e5 + 10;

int x[maxn], y[maxn];

int qp(int a, int n)
{
    ll ans = 1, base = a;
    for(; n; (base *= base) %= mod, n >>= 1) if(n & 1) (ans *= base) %= mod;
    return ans;
}

int lagrange(int n, int *x, int *y, int xi)
{
    int ans = 0;
    for(int i = 0; i <= n; i ++)
    {
        int s1 = 1, s2 = 1;
        for(int j = 0; j <= n; j ++) if(i != j)
        {
            s1 = 1ll * s1 * (xi - x[j]) % mod;
            s2 = 1ll * s2 * (x[i] - x[j]) % mod;
        }
```

```
        ans = (1ll * ans + 1ll * y[i] * s1 % mod * qp(s2, mod - 2) % mod) % mod;
    }
    return (ans + mod) % mod;
}
```

1.7.2 拉格朗日插值（连续取值）

```
const int mod = 'edit';
const int maxn = 'edit';

int x[maxn], y[maxn];
int s1[maxn], s2[maxn], ifac[maxn];

//如果x的取值是连续一段，可以做到O(n)求解
int lagrange(int n, int *x, int *y, int xi)
{
    int ans = 0;
    s1[0] = (xi - x[0]) % mod, s2[n + 1] = 1;
    for(int i = 1; i <= n; i ++) s1[i] = 1ll * s1[i - 1] * (xi - x[i]) % mod;
    for(int i = n; i >= 0; i --) s2[i] = 1ll * s2[i + 1] * (xi - x[i]) % mod;
    ifac[0] = ifac[1] = 1;
    for(int i = 2; i <= n; i ++) ifac[i] = -1ll * mod / i * ifac[mod % i] % mod;
    for(int i = 2; i <= n; i ++) ifac[i] = 1ll * ifac[i] * ifac[i - 1] % mod;
    for(int i = 0; i <= n; i ++)
        (ans += 1ll * y[i] * (i == 0 ? 1 : s1[i - 1]) % mod * s2[i + 1] % mod * ifac[i] % mod * (((
    n - i) & 1) ? -1 : 1) * ifac[n - i] % mod) %= mod;
    return (ans + mod) % mod;
}
```

1.8 Others

1. 约数定理：若 $n = \prod_{i=1}^{k} p_i^{a_i}$，则
   (a) 约数个数 $f(n) = \prod_{i=1}^{k}(a_i + 1)$
   (b) 约数和 $g(n) = \prod_{i=1}^{k}(\sum_{j=0}^{a_i} p_i^j)$
2. 小于 n 且互素的数之和为 $n\varphi(n)/2$
3. 若 $\gcd(n, i) = 1$，则 $\gcd(n, n - i) = 1 (1 \le i \le n)$
4. 错排公式：$D(n) = (n-1)(D(n-2) + D(n-1)) = \sum_{i=2}^{n} \frac{(-1)^k n!}{k!} = [\frac{n!}{e} + 0.5]$
5. 威尔逊定理：$p\ is\ prime\ \Rightarrow (p-1)! \equiv -1 \pmod{p}$
6. 欧拉定理：$\gcd(a, n) = 1 \Rightarrow a^{\varphi(n)} \equiv 1 \pmod{n}$
7. 欧拉定理推广：$\gcd(n, p) = 1 \Rightarrow a^n \equiv a^{n\%\varphi(p)} \pmod{p}$
8. 模的幂公式：$a^n \pmod{m} = \begin{cases} a^n \mod m & n < \varphi(m) \\ a^{n\%\varphi(m)+\varphi(m)} \mod m & n \ge \varphi(m) \end{cases}$
9. 素数定理：对于不大于 n 的素数个数 $\pi(n)$，$\lim_{n\to\infty} \pi(n) = \frac{n}{\ln n}$
10. 位数公式：正整数 x 的位数 $N = \log_{10}(n) + 1$
11. 斯特灵公式 $n! \approx \sqrt{2\pi n}(\frac{n}{e})^n$
12. 设 $a > 1, m, n > 0$，则 $\gcd(a^m - 1, a^n - 1) = a^{\gcd(m,n)} - 1$
13. 设 $a > b, \gcd(a, b) = 1$，则 $\gcd(a^m - b^m, a^n - b^n) = a^{\gcd(m,n)} - b^{\gcd(m,n)}$

$$G = \gcd(C_n^1, C_n^2, ..., C_n^{n-1}) = \begin{cases} n, & n \text{ is prime} \\ 1, & n \text{ has multy prime factors} \\ p, & n \text{ has single prime factor } p \end{cases}$$

$\gcd(Fib(m), Fib(n)) = Fib(\gcd(m, n))$

14. 若 $\gcd(m,n)=1$, 则:

    (a) 最大不能组合的数为 $m*n-m-n$

    (b) 不能组合数个数 $N=\frac{(m-1)(n-1)}{2}$

15. $(n+1)lcm(C_n^0,C_n^1,...,C_n^{n-1},C_n^n)=lcm(1,2,...,n+1)$

16. 若 p 为素数，则 $(x+y+...+w)^p\equiv x^p+y^p+...+w^p \pmod p$

17. 卡特兰数：1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012

    $h(0)=h(1)=1, h(n)=\frac{(4n-2)h(n-1)}{n+1}=\frac{C_{2n}^n}{n+1}=C_{2n}^n-C_{2n}^{n-1}$

18. 伯努利数：$B_n=-\frac{1}{n+1}\sum_{i=0}^{n-1}C_{n+1}^iB_i$

$$\sum_{i=1}^{n}i^k=\frac{1}{k+1}\sum_{i=1}^{k+1}C_{k+1}^iB_{k+1-i}(n+1)^i$$

19. 二项式反演：

$$f_n=\sum_{i=0}^{n}(-1)^i\binom{n}{i}g_i\Leftrightarrow g_n=\sum_{i=0}^{n}(-1)^i\binom{n}{i}f_i$$

$$f_n=\sum_{i=0}^{n}\binom{n}{i}g_i\Leftrightarrow g_n=\sum_{i=0}^{n}(-1)^{n-i}\binom{n}{i}f_i$$

20. 2 的 n 次方，在 pow 时可以精确输出最大 2^1023，pow(2,1023)

21. FFT 常用素数

$r\ 2^k+1$	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
998244353	119	23	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

1.8.1 BM

```
using VI = vector<int>;
const int mod = 1e9+7;
const int maxn = 1 << 14;
ll res[maxn], base[maxn], _c[maxn], _md[maxn];
vector<int> Md;
void mul(ll* a, ll* b, int k)
{
    for (int i = 0; i < k + k; i++) _c[i] = 0;
    for (int i = 0; i < k; i++)
        if (a[i])
            for (int j = 0; j < k; j++) _c[i + j] = (_c[i + j] + a[i] * b[j]) % mod;
    for (int i = k + k - 1; i >= k; i--)
        if (_c[i])
            for (int j = 0; j < Md.size(); j++) _c[i - k + Md[j]] = (_c[i - k + Md[j]] - _c[i] *
    _md[Md[j]]) % mod;
```

```
15      for (int i = 0; i < k; i++) a[i] = _c[i];
16  }
17  int solve(ll n, VI a, VI b)
18  {
19      ll ans = 0, pnt = 0;
20      int k = a.size();
21      assert(a.size() == b.size());
22      for (int i = 0; i < k; i++) _md[k - 1 - i] = -a[i];
23      _md[k] = 1;
24      Md.clear();
25      for (int i = 0; i < k; i++)
26          if (_md[i] != 0) Md.push_back(i);
27      for (int i = 0; i < k; i++) res[i] = base[i] = 0;
28      res[0] = 1;
29      while ((1LL << pnt) <= n) pnt++;
30      for (int p = pnt; p >= 0; p--)
31      {
32          mul(res, res, k);
33          if ((n >> p) & 1)
34          {
35              for (int i = k - 1; i >= 0; i--) res[i + 1] = res[i];
36              res[0] = 0;
37              for (int j = 0; j < Md.size(); j++) res[Md[j]] = (res[Md[j]] - res[k] * _md[Md[j]]) %
        mod;
38          }
39      }
40      for (int i = 0; i < k; i++) ans = (ans + res[i] * b[i]) % mod;
41      if (ans < 0) ans += mod;
42      return ans;
43  }
44  VI BM(VI s)
45  {
46      VI C(1, 1), B(1, 1);
47      int L = 0, m = 1, b = 1;
48      for (int n = 0; n < s.size(); n++)
49      {
50          ll d = 0;
51          for (int i = 0; i <= L; i++) d = (d + (ll)C[i] * s[n - i]) % mod;
52          if (d == 0)
53              ++m;
54          else if (2 * L <= n)
55          {
56              VI T = C;
57              ll c = mod - d * Pow(b, mod - 2) % mod;
58              while (C.size() < B.size() + m) C.push_back(0);
59              for (int i = 0; i < B.size(); i++) C[i + m] = (C[i + m] + c * B[i]) % mod;
60              L = n + 1 - L, B = T, b = d, m = 1;
61          }
62          else
63          {
64              ll c = mod - d * Pow(b, mod - 2) % mod;
65              while (C.size() < B.size() + m) C.push_back(0);
66              for (int i = 0; i < B.size(); i++) C[i + m] = (C[i + m] + c * B[i]) % mod;
67              ++m;
68          }
69      }
70      return C;
71  }
72  int gao(VI a, ll n)
```

```
{
    VI c = BM(a);
    c.erase(c.begin());
    for (int i = 0; i < c.size(); i++) c[i] = (mod - c[i]) % mod;
    return solve(n, c, VI(a.begin(), a.begin() + c.size()));
}
```

1.8.2 exBM

```
// given first m items init[0..m-1] and coefficents trans[0..m-1] or
// given first 2 *m items init[0..2m-1], it will compute trans[0..m-1]
// for you. trans[0..m] should be given as that
//      init[m] = sum_{i=0}^{m-1} init[i] * trans[i]
struct LinearRecurrence
{
    using int64 = long long;
    using vec = std::vector<int64>;

    static void extand(vec& a, size_t d, int64 value = 0)
    {
        if (d <= a.size()) return;
        a.resize(d, value);
    }
    static vec BerlekampMassey(const vec& s, int64 mod)
    {
        std::function<int64(int64)> inverse = [&](int64 a) {
            return a == 1 ? 1 : (int64)(mod - mod / a) * inverse(mod % a) % mod;
        };
        vec A = {1}, B = {1};
        int64 b = s[0];
        for (size_t i = 1, m = 1; i < s.size(); ++i, m++)
        {
            int64 d = 0;
            for (size_t j = 0; j < A.size(); ++j)
            {
                d += A[j] * s[i - j] % mod;
            }
            if (!(d %= mod)) continue;
            if (2 * (A.size() - 1) <= i)
            {
                auto temp = A;
                extand(A, B.size() + m);
                int64 coef = d * inverse(b) % mod;
                for (size_t j = 0; j < B.size(); ++j)
                {
                    A[j + m] -= coef * B[j] % mod;
                    if (A[j + m] < 0) A[j + m] += mod;
                }
                B = temp, b = d, m = 0;
            }
            else
            {
                extand(A, B.size() + m);
                int64 coef = d * inverse(b) % mod;
                for (size_t j = 0; j < B.size(); ++j)
                {
                    A[j + m] -= coef * B[j] % mod;
                    if (A[j + m] < 0) A[j + m] += mod;
```

```
50                  }
51              }
52          }
53          return A;
54      }
55      static void exgcd(int64 a, int64 b, int64& g, int64& x, int64& y)
56      {
57          if (!b)
58              x = 1, y = 0, g = a;
59          else
60          {
61              exgcd(b, a % b, g, y, x);
62              y -= x * (a / b);
63          }
64      }
65      static int64 crt(const vec& c, const vec& m)
66      {
67          int n = c.size();
68          int64 M = 1, ans = 0;
69          for (int i = 0; i < n; ++i) M *= m[i];
70          for (int i = 0; i < n; ++i)
71          {
72              int64 x, y, g, tm = M / m[i];
73              exgcd(tm, m[i], g, x, y);
74              ans = (ans + tm * x * c[i] % M) % M;
75          }
76          return (ans + M) % M;
77      }
78      static vec ReedsSloane(const vec& s, int64 mod)
79      {
80          auto inverse = [](int64 a, int64 m) {
81              int64 d, x, y;
82              exgcd(a, m, d, x, y);
83              return d == 1 ? (x % m + m) % m : -1;
84          };
85          auto L = [](const vec& a, const vec& b) {
86              int da = (a.size() > 1 || (a.size() == 1 && a[0])) ? a.size() - 1 : -1000;
87              int db = (b.size() > 1 || (b.size() == 1 && b[0])) ? b.size() - 1 : -1000;
88              return std::max(da, db + 1);
89          };
90          auto prime_power = [&](const vec& s, int64 mod, int64 p, int64 e) {
91              // linear feedback shift register mod p^e, p is prime
92              std::vector<vec> a(e), b(e), an(e), bn(e), ao(e), bo(e);
93              vec t(e), u(e), r(e), to(e, 1), uo(e), pw(e + 1);
94              ;
95              pw[0] = 1;
96              for (int i = pw[0] = 1; i <= e; ++i) pw[i] = pw[i - 1] * p;
97              for (int64 i = 0; i < e; ++i)
98              {
99                  a[i] = {pw[i]}, an[i] = {pw[i]};
100                 b[i] = {0}, bn[i] = {s[0] * pw[i] % mod};
101                 t[i] = s[0] * pw[i] % mod;
102                 if (t[i] == 0)
103                 {
104                     t[i] = 1, u[i] = e;
105                 }
106                 else
107                 {
108                     for (u[i] = 0; t[i] % p == 0; t[i] /= p, ++u[i])
```

```
109                     ;
110             }
111         }
112         for (size_t k = 1; k < s.size(); ++k)
113         {
114             for (int g = 0; g < e; ++g)
115             {
116                 if (L(an[g], bn[g]) > L(a[g], b[g]))
117                 {
118                     ao[g] = a[e - 1 - u[g]];
119                     bo[g] = b[e - 1 - u[g]];
120                     to[g] = t[e - 1 - u[g]];
121                     uo[g] = u[e - 1 - u[g]];
122                     r[g] = k - 1;
123                 }
124             }
125             a = an, b = bn;
126             for (int o = 0; o < e; ++o)
127             {
128                 int64 d = 0;
129                 for (size_t i = 0; i < a[o].size() && i <= k; ++i)
130                 {
131                     d = (d + a[o][i] * s[k - i]) % mod;
132                 }
133                 if (d == 0)
134                 {
135                     t[o] = 1, u[o] = e;
136                 }
137                 else
138                 {
139                     for (u[o] = 0, t[o] = d; t[o] % p == 0; t[o] /= p, ++u[o])
140                         ;
141                     int g = e - 1 - u[o];
142                     if (L(a[g], b[g]) == 0)
143                     {
144                         extand(bn[o], k + 1);
145                         bn[o][k] = (bn[o][k] + d) % mod;
146                     }
147                     else
148                     {
149                         int64 coef = t[o] * inverse(to[g], mod) % mod * pw[u[o] - uo[g]] % mod;
150                         int m = k - r[g];
151                         extand(an[o], ao[g].size() + m);
152                         extand(bn[o], bo[g].size() + m);
153                         for (size_t i = 0; i < ao[g].size(); ++i)
154                         {
155                             an[o][i + m] -= coef * ao[g][i] % mod;
156                             if (an[o][i + m] < 0) an[o][i + m] += mod;
157                         }
158                         while (an[o].size() && an[o].back() == 0) an[o].pop_back();
159                         for (size_t i = 0; i < bo[g].size(); ++i)
160                         {
161                             bn[o][i + m] -= coef * bo[g][i] % mod;
162                             if (bn[o][i + m] < 0) bn[o][i + m] -= mod;
163                         }
164                         while (bn[o].size() && bn[o].back() == 0) bn[o].pop_back();
165                     }
166                 }
167             }
```

```
168             }
169             return std::make_pair(an[0], bn[0]);
170         };
171 
172         std::vector<std::tuple<int64, int64, int>> fac;
173         for (int64 i = 2; i * i <= mod; ++i)
174         {
175             if (mod % i == 0)
176             {
177                 int64 cnt = 0, pw = 1;
178                 while (mod % i == 0) mod /= i, ++cnt, pw *= i;
179                 fac.emplace_back(pw, i, cnt);
180             }
181         }
182         if (mod > 1) fac.emplace_back(mod, mod, 1);
183         std::vector<vec> as;
184         size_t n = 0;
185         for (auto&& x : fac)
186         {
187             int64 mod, p, e;
188             vec a, b;
189             std::tie(mod, p, e) = x;
190             auto ss = s;
191             for (auto&& x : ss) x %= mod;
192             std::tie(a, b) = prime_power(ss, mod, p, e);
193             as.emplace_back(a);
194             n = std::max(n, a.size());
195         }
196         vec a(n), c(as.size()), m(as.size());
197         for (size_t i = 0; i < n; ++i)
198         {
199             for (size_t j = 0; j < as.size(); ++j)
200             {
201                 m[j] = std::get<0>(fac[j]);
202                 c[j] = i < as[j].size() ? as[j][i] : 0;
203             }
204             a[i] = crt(c, m);
205         }
206         return a;
207     }
208 
209     LinearRecurrence(const vec& s, const vec& c, int64 mod) : init(s), trans(c), mod(mod), m(s.size
    ()) {}
210     LinearRecurrence(const vec& s, int64 mod, bool is_prime = true) : mod(mod)
211     {
212         vec A;
213         if (is_prime)
214             A = BerlekampMassey(s, mod);
215         else
216             A = ReedsSloane(s, mod);
217         if (A.empty()) A = {0};
218         m = A.size() - 1;
219         trans.resize(m);
220         for (int i = 0; i < m; ++i)
221         {
222             trans[i] = (mod - A[i + 1]) % mod;
223         }
224         std::reverse(trans.begin(), trans.end());
225         init = {s.begin(), s.begin() + m};
```

```
226     }
227     int64 calc(int64 n)
228     {
229         if (mod == 1) return 0;
230         if (n < m) return init[n];
231         vec v(m), u(m << 1);
232         int msk = !!n;
233         for (int64 m = n; m > 1; m >>= 1) msk <<= 1;
234         v[0] = 1 % mod;
235         for (int x = 0; msk; msk >>= 1, x <<= 1)
236         {
237             std::fill_n(u.begin(), m * 2, 0);
238             x |= !!(n & msk);
239             if (x < m)
240                 u[x] = 1 % mod;
241             else
242             { // can be optimized by fft/ntt
243                 for (int i = 0; i < m; ++i)
244                 {
245                     for (int j = 0, t = i + (x & 1); j < m; ++j, ++t)
246                     {
247                         u[t] = (u[t] + v[i] * v[j]) % mod;
248                     }
249                 }
250                 for (int i = m * 2 - 1; i >= m; --i)
251                 {
252                     for (int j = 0, t = i - m; j < m; ++j, ++t)
253                     {
254                         u[t] = (u[t] + trans[j] * u[i]) % mod;
255                     }
256                 }
257             }
258             v = {u.begin(), u.begin() + m};
259         }
260         int64 ret = 0;
261         for (int i = 0; i < m; ++i)
262         {
263             ret = (ret + v[i] * init[i]) % mod;
264         }
265         return ret;
266     }
267
268     vec init, trans;
269     int64 mod;
270     int m;
271 };
```

2 Graph Theory

2.1 路径

2.1.1 Dijkstra

```
const int maxn = 1e5 + 10;
const int inf = 0x3f3f3f3f;

int head[maxn], dis[maxn], cnt, n;

struct Edge { int nex,to,w; }edge[20*maxn];

void add(int u,int v,int w)
{
    edge[++cnt].nex=head[u];
    edge[cnt].w=w;
    edge[cnt].to=v;
    head[u]=cnt;
}

void dijkstra(int s)
{
    priority_queue<pair<int, int>, vector<pair<int, int> >, greater<pair<int, int> > > que;
    memset(dis, 0x3f, sizeof dis);
    que.push({0, s}); dis[s] = 0;
    while(!que.empty())
    {
        auto f = que.top(); que.pop();
        int u = f.second, d = f.first;
        if(d != dis[u]) continue;
        for(int i = head[u]; ~i; i = edge[i].nex)
        {
            int v = edge[i].to, w = edge[i].w;
            if(dis[u] + w < dis[v])
            {
                dis[v] = dis[u] + w;
                que.push({dis[v], v});
            }
        }
    }
}
```

2.1.2 Euler Path

```
int S[N << 1], top;
Edge edges[N << 1];
set<int> G[N];

void DFS(int u) {
    S[top++] = u;
    for (int eid: G[u])
    {
        int v = edges[eid].get_other(u);
        G[u].erase(eid);
        G[v].erase(eid);
        DFS(v);
        return;
    }
```

```
}

void fleury(int start)
{
    int u = start;
    top = 0; path.clear();
    S[top++] = u;
    while (top)
    {
        u = S[--top];
        if (!G[u].empty())
            DFS(u);
        else path.push_back(u);
    }
}
```

2.1.3 K-shortest Path

```
const int inf = 0x3f3f3f3f;
const int maxn = 1000 + 10;
const int maxm = 100000 + 10;

int n, k, cnt, head[maxn], revhead[maxn], dis[maxn];
bool vis[maxn];

struct node { int v, w, nex; } edge[maxm], revedge[maxm];

void init()
{
    cnt = 0;
    memset(head, 0xff, sizeof head);
    memset(revhead, 0xff, sizeof revhead);
}

void add(int u, int v, int w)
{
    edge[cnt].v = v, revedge[cnt].v = u;
    edge[cnt].w = revedge[cnt].w = w;
    edge[cnt].nex = head[u];
    revedge[cnt].nex = revhead[v];
    head[u] = revhead[v] = cnt;
    cnt++;
}

void spfa(int src)      //建立反向图，求图中所有点到终点的最短路径
{
    for (int i = 1; i <= n; i++) dis[i] = inf;
    memset(vis, false, sizeof vis);
    vis[src] = 0;
    queue<int> que;
    que.push(src);
    dis[src] = 0;
    while (!que.empty())
    {
        int u = que.front();
        que.pop();
        vis[u] = false;
        for (int i = revhead[u]; ~i; i = revedge[i].nex)
```

```
41          {
42              int v = revedge[i].v, w = revedge[i].w;
43              if (dis[v] > dis[u] + w)
44              {
45                  dis[v] = dis[u] + w;
46                  if (!vis[v])
47                  {
48                      que.push(v);
49                      vis[v] = true;
50                  }
51              }
52          }
53      }
54  }
55
56  struct A
57  {
58      int f, g, h;    //f(n),g(n),h(n)函数
59      int id;        //当前点的编号
60      bool operator<(const A a) const
61      {        //定义比较函数
62          if (a.f == f) return a.g < g;
63          return a.f < f;
64      }
65  };
66
67  int Astar(int src, int des)
68  {
69      int cnt = 0;
70      priority_queue <A> Q;
71      if (src == des) k++;   //如果起点即为终点
72      if (dis[src] == inf) return -1;      //如果起点不能到达终点
73      A st, now, tmp;
74      st.id = src, st.g = 0, st.f = st.g + dis[src];    //定义起始节点
75      Q.push(st);
76      while (!Q.empty())
77      {
78          now = Q.top();
79          Q.pop();
80          if (now.id == des)     //如果当前节点为终点
81          {
82              cnt++;
83              if (cnt == k) return now.g;    //找到第k短路
84          }
85          for (int i = head[now.id]; ~i; i = edge[i].nex)
86          {
87              tmp.id = edge[i].v;
88              tmp.g = now.g + edge[i].w;     //到该点的实际花费
89              tmp.f = tmp.g + dis[tmp.id];   //到最终状态的估计花费
90              Q.push(tmp);
91          }
92      }
93      return -1;  //路径总数小于k
94  }
95
96  int main()
97  {
98      int m, s, t, u, v, w;
99      while (scanf("%d%d", &n, &m) != EOF)
```

```
100     {
101         init();
102         while (m--)
103         {
104             scanf("%d%d%d", &u, &v, &w);
105             add(u, v, w);
106         }
107         scanf("%d%d%d", &s, &t, &k);
108         spfa(t);    //求所有点到终点的最短路
109         printf("%d\n", Astar(s, t));
110     }
111     return 0;
112 }
```

2.2 生成树

2.2.1 Kruskal

```
1  const int maxn = 1e5 + 10;
2
3  int n, m, pre[maxn];
4  struct edge {int u, v, w; } es[maxn];
5  int Find(int x) { return x == pre[x] ? x : pre[x] = Find(pre[x]); }
6  bool cmp(const edge &x, const edge &y) { return x.cost < y.cost; }
7
8  int kruskal()
9  {
10     sort(es, es + m, cmp);
11     int res = 0;
12     for(int i = 0; i < m; i ++)
13     {
14         int fx = Find(es[i].u), fy = Find(es[i].v);
15         if(fx != fy) pre[fx] = fy, res += es[i].cost;
16     }
17     return res;
18 }
```

2.2.2 Prim

```
1  const int maxn = 1000 + 10;
2  const int inf = 0x3f3f3f3f;
3
4  int n, mp[maxn][maxn], cost[maxn];
5  bool vis[maxn];
6
7  int prim()
8  {
9      for(int i = 0; i < n; i ++) cost[u] = inf, vis[u] = false;
10     int res = 0; cost[0] = 0;
11     for(;;)
12     {
13         int v = -1;
14         for(int u = 0; u < n; u ++)
15             if(!vis[u] && (v == -1 || cost[u] < cost[v])) v = u;
16         if(v == -1) break;
17         res += cost[v];
18         vis[v] = true;
19         for(int u = 0; u < n; u ++) cost[u] = min(cost[u], mp[v][u]);
```

```
    }
    return res;
}
```

2.2.3 最小树形图

```
const int INF = 0x3f3f3f3f;
const int maxn = 10000;
const int maxm = 10000;

struct Edge{int u,v,cost; } edge[maxm];

int pre[maxn], id[maxn], vis[maxn], in[maxn];

int zhuliu(int root, int n, int m)
{
    int res=0, u, v;
    for(;;)
    {
        for(int i=0; i<n; i++) in[i] = INF;
        for(int i=0; i<m; i++) if(edge[i].u != edge[i].v && edge[i].cost < in[edge[i].v])
        {
            pre[edge[i].v] = edge[i].u;
            in[edge[i].v] = edge[i].cost;
        }
        for(int i=0; i<n; i++) if(i != root && in[i] ==INF) return -1;
        int tn=0;
        memset(id, 0xff, sizeof id);
        memset(vis, 0xff, sizeof vis);
        in[root] = 0;
        for(int i=0; i<n;i++)
        {
            res += in[i];
            v = i;
            while( vis[v] != i && id[v] == -1 && v!= root) vis[v] = i, v = pre[v];
            if(v != root && id[v] == -1)
            {
                for(int u = pre[v]; u != v; u = pre[u]) id[u] = tn;
                id[v] = tn++;
            }
        }
        if(tn == 0) break;
        for(int i=0; i<n; i++) if(id[i] == -1) id[i] = tn++;
        for(int i=0; i<m; )
        {
            v = edge[i].v;
            edge[i].u = id[edge[i].u];
            edge[i].v = id[edge[i].v];
            if(edge[i].u != edge[i].v) edge[i++].cost -= in[v];
            else swap(edge[i], edge[--m]);
        }
        n = tn;
        root = id[root];
    }
    return res;
}
```

2.2.4 Matrix Tree

```
const int N = 305;
const int mod = 1e9 + 7;

int n, m, a[N][N];

int Gauss(int n) {
    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;
                }
                std::swap(a[i], a[k]);
                ans = - ans;
            }
        }
        ans = 1LL * ans * a[i][i] % mod;
    }
    return (ans % mod + mod) % mod;
}
int main() {
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= m; i++) {
        int u, v;
        scanf("%d%d", &u, &v);
        a[u][v]--, a[v][u]--;
        a[u][u]++, a[v][v]++;
    }
    printf("%d\n", Gauss(n - 1));
    return 0;
}
```

2.3 连通性

2.3.1 割点

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

vector<int> edge[maxn];
int n, dfn[maxn], low[maxn], cnt = 0;
bool vis[maxn], cut[maxn];

void Tarjan(int u, int fa)
{
    dfn[u] = low[u] = ++cnt;
    vis[u] = true;
    int children = 0;
    for (int i = 0; i < edge[u].size(); i++)
    {
        int v = edge[u][i];
        if (v != fa && vis[v])
            low[u] = min(low[u], dfn[v]);
        else if (!vis[v])
        {
            Tarjan(v, u);
```

```
            children++;
            low[u] = min(low[u], low[v]);
            if (fa == -1 && children > 1) //若u是根节点且子节点数大于1
                cut[u] = true;    //u是割点
            else if (fa != -1 && low[v] >= dfn[u])    //若u不是根节点且v不能访问到u的父节点
                cut[u] = true;    //u是割点
        }
    }
}
```

2.3.2 桥

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

vector<int> edge[maxn];
int n, dfn[maxn], low[maxn], father[maxn], cnt = 0;
bool bridge[maxn][maxn];

void Tarjan(int u, int fa)
{
    dfn[u] = low[u] = ++cnt;
    for (int i = 0; i < edge[u].size(); i++)
    {
        int v = edge[u][i];
        if (!dfn[v])    //未访问节点v
        {
            Tarjan(v, u);
            low[u] = min(low[u], low[v]);
            if (low[v] > dfn[u]) //节点v到达祖先必须经过(u,v)
                bridge[u][v] = bridge[v][u] = true;    //(u,v)是桥
        }
        else if (fa != v)    //u的父节点不是v, (u,v)不存在重边
            low[u] = min(low[u], dfn[v]);
    }
}
```

2.3.3 强连通分量

```
const int maxn=1000+10;

vector<int> edge[maxn];

int dfn[maxn], low[maxn];
int stack[maxn], index, tot;
int belong[maxn], inde[maxn], outde[maxn], scc;
bool vis[maxn];

void add(int u, int v)
{
    edge[u].push_back(v);
    edge[v].push_back(u);
}

void Tarjan(int u)
{
    dfn[u] = low[u] = ++tot;
    stack[++index] = u;
    vis[u] = true;
```

```
    int v;
    for(int i = 0;i < edge[u].size(); i++)
    {
        v=edge[u][i];
        if(!dfn[v])
        {
            Tarjan(v);
            low[u] = min(low[v], low[u]);
        }
        else if(vis[v]) low[u] = min(low[v], dfn[u]);
    }
    if(dfn[u] == low[u])
    {
        scc++;
        do
        {
            v = stack[index--];
            vis[v] = false;
            belong[v] = scc;
        }while(v != u);
    }
}
```

2.4 二分图匹配

1. 二分图中的最大匹配数 = 最小点覆盖数
2. 最小路径覆盖 = 最小路径覆盖= | G | - 最大匹配数
3. 二分图最大独立集 = 顶点数-最小点覆盖
4. 二分图的最大团 = 补图的最大独立集

2.4.1 Hungary Algorithm

```
const int maxn = 150;

int n;
int edge[maxn][maxn];
int linker[maxn];
bool vis[maxn];

bool path(int u)
{
    for (int v = 1; v <= n; v++)
    {
        if (edge[u][v] && !vis[v])
        {
            vis[v] = true;
            if (linker[v] == -1 || path(linker[v]))
            {
                linker[v] = u;
                return true;
            }
        }
    }
    return false;
}

int hungary()
```

```
{
    int res = 0;
    memset(linker, 0xff, sizeof(linker));
    for (int i = 1; i <= n; i++)
    {
        memset(vis, false, sizeof(vis));
        res += path(i);
    }
    return res;
}
```

2.4.2 Hopcroft-karp Algorithm

```
//复杂度O(n^0.5*m),注意这个板子的下标是从0开始的

const int MAXN = 3010;//左边节点数量、右边节点数量
const int MAXM = 3010 * 3010;//边的数量
const int INF = 0x3f3f3f3f;

struct Edge
{
    int v;
    int next;
} edge[MAXM];

int nx, ny;
int cnt;
int dis;

int first[MAXN];
int xlink[MAXN], ylink[MAXN];
/*xlink[i]表示左集合顶点所匹配的右集合顶点序号，ylink[i]表示右集合i顶点匹配到的左集合顶点序号。*/
int dx[MAXN], dy[MAXN];
/*dx[i]表示左集合i顶点的距离编号，dy[i]表示右集合i顶点的距离编号*/
int vis[MAXN]; //寻找增广路的标记数组

void init()
{
    cnt = 0;
    memset(first, -1, sizeof(first));
    memset(xlink, -1, sizeof(xlink));
    memset(ylink, -1, sizeof(ylink));
}

void read_graph(int u, int v)
{
    edge[cnt].v = v;
    edge[cnt].next = first[u], first[u] = cnt++;
}

int bfs()
{
    queue<int> q;
    dis = INF;
    memset(dx, -1, sizeof(dx));
    memset(dy, -1, sizeof(dy));
    for (int i = 0; i < nx; i++)
    {
```

```
46          if (xlink[i] == -1)
47          {
48              q.push(i);
49              dx[i] = 0;
50          }
51      }
52      while (!q.empty())
53      {
54          int u = q.front();
55          q.pop();
56          if (dx[u] > dis) break;
57          for (int e = first[u]; e != -1; e = edge[e].next)
58          {
59              int v = edge[e].v;
60              if (dy[v] == -1)
61              {
62                  dy[v] = dx[u] + 1;
63                  if (ylink[v] == -1) dis = dy[v];
64                  else
65                  {
66                      dx[ylink[v]] = dy[v] + 1;
67                      q.push(ylink[v]);
68                  }
69              }
70          }
71      }
72      return dis != INF;
73  }
74
75  int find(int u)
76  {
77      for (int e = first[u]; e != -1; e = edge[e].next)
78      {
79          int v = edge[e].v;
80          if (!vis[v] && dy[v] == dx[u] + 1)
81          {
82              vis[v] = 1;
83              if (ylink[v] != -1 && dy[v] == dis) continue;
84              if (ylink[v] == -1 || find(ylink[v]))
85              {
86                  xlink[u] = v, ylink[v] = u;
87                  return 1;
88              }
89          }
90      }
91      return 0;
92  }
93
94  int MaxMatch()
95  {
96      int ans = 0;
97      while (bfs())
98      {
99          memset(vis, 0, sizeof(vis));
100         for (int i = 0; i < nx; i++)
101             if (xlink[i] == -1)
102                 ans += find(i);
103     }
104     return ans;
```

```
}
```

2.4.3 二分图多重匹配

```
const int maxn = 1e2 + 5;//左边最大点数
const int maxm = 1e2 + 5;//右边最大点数
int graph[maxn][maxm], vis[maxm];//图G和增广路访问标记
int match[maxm][maxn];//左边元素与右边元素第n次匹配
int nx, ny, m;//左边点数, 右边点数,边数
int vol[maxm];//右边点多重匹配可容纳值
int cnt[maxm];//右边点已匹配值

bool find_path(int u)//找增广路
{
    for (int i = 0; i < ny; i++)//注意, 这里节点是从0开始编号, 题目有时是从1开始编号
    {
        if (graph[u][i] && !vis[i])//不在增广路
        {
            vis[i] = 1;//放进增广路
            if (cnt[i] < vol[i])//如果当前已匹配数量小于可容纳量, 则直接匹配
            {
                match[i][cnt[i]++] = u;
                return true;
            }
            for (int j = 0; j < cnt[i]; j++)
            {
                if (find_path(match[i][j]))//如果先前已匹配右边的点能另外找到增广路, 则此点仍可匹配
                {
                    match[i][j] = u;
                    return true;
                }
            }
        }
    }
    return false;
}

int max_match()//计算多重匹配的最大匹配数
{
    int res = 0;
    memset(match, -1, sizeof(match));
    memset(cnt, 0, sizeof(cnt));
    for (int i = 0; i < nx; i++)
    {
        memset(vis, 0, sizeof(vis));
        if (find_path(i)) res++;
    }
    return res;
}

bool all_match()//判断左边的点是否都与右边的点匹配了
{
    memset(cnt, 0, sizeof(cnt));
    for (int i = 0; i < nx; i++)
    {
        memset(vis, 0, sizeof(vis));
        if (!find_path(i)) return false;
    }
```

```
55      return true;
56  }
```

2.4.4 二分图最大权匹配 (KM 算法)

```
1   const int maxn=1000+10;
2   const int inf=0x3f3f3f3f;
3
4   int n;
5   int lx[maxn],ly[maxn],edge[maxn][maxn];
6   int match[maxn],delta;
7   bool vx[maxn],vy[maxn];
8
9   bool dfs(int x) //DFS增广, 寻找相等子图的完备匹配
10  {
11      vx[x]=true;
12      for(int y=1;y<=n;y++)
13      {
14          if(!vy[y])
15          {
16              int tmp=lx[x]+ly[y]-edge[x][y];
17              if(!tmp)    //edge(x,y)为可行边
18              {
19                  vy[y]=true;
20                  if(!match[y]||dfs(match[y]))
21                  {
22                      match[y]=x;
23                      return true;
24                  }
25              }
26              else delta=min(delta,tmp);
27          }
28      }
29      return false;
30  }
31
32  void KM()
33  {
34      for(int i=1;i<=n;i++)   //初始化可行顶标的值
35      {
36          lx[i]=-inf;
37          ly[i]=0;
38          for(int j=1;j<=n;j++)
39              lx[i]=max(lx[i],edge[i][j]);
40      }
41      memset(match,0,sizeof(match));
42      for(int x=1;x<=n;x++)
43      {
44          for(;;)
45          {
46              delta=inf;
47              memset(vx,0,sizeof(vx));
48              memset(vy,0,sizeof(vy));
49              if(dfs(x)) break;
50              for(int i=1;i<=n;i++)   //修改顶标
51              {
52                  if(vx[i]) lx[i]-=delta;
53                  if(vy[i]) ly[i]+=delta;
```

```
            }
        }
    }
}
```

2.4.5 一般图匹配带花树

```
//一般图匹配，带花树算法
const int maxn = 1000 + 10;

vector<int> edge[maxn];
queue<int> que;

int n, pre[maxn], type[maxn], link[maxn], nex[maxn], vis[maxn];

void add(int u, int v)
{
    edge[u].push_back(v);
    edge[v].push_back(u);
}

int Find(int x)
{
    return x == pre[x] ? x : pre[x] = Find(pre[x]);
}

void combine(int x, int lca)    //如果找到奇环，对当前点x和找到的
{
    while (x != lca)
    {
        int u = link[x], v = nex[u];
        if (Find(v) != lca) nex[v] = u;
        if (type[u] == 1) type[u] = 2, que.push(u);
        pre[Find(x)] = Find(u);
        pre[Find(u)] = Find(v);
        x = v;
    }
}

void contrack(int x, int y)
{
    int lca = x;
    memset(vis, 0, sizeof(vis));
    for (int i = x; i; i = nex[link[i]])
    {
        i = Find(i);
        vis[i] = 1;
    }
    for (int i = y; i; i = nex[link[i]])
    {
        i = Find(i);
        if (vis[i])
        {
            lca = i;
            break;
        }
    }
    if (lca != Find(x)) nex[x] = y;
```

```
52      if (lca != Find(y)) nex[y] = x;
53      combine(x, lca);
54      combine(y, lca);
55  }
56
57  void bfs(int s)
58  {
59      memset(type, 0, sizeof(type));
60      memset(nex, 0, sizeof(nex));
61      for (int i = 1; i <= n; i++) pre[i] = i;
62      while (!que.empty()) que.pop();
63      que.push(s);
64      type[s] = 2;
65      while (!que.empty())
66      {
67          int x = que.front();
68          que.pop();
69          for (int i = 0; i < edge[x].size(); i++)
70          {
71              int y = edge[x][i];
72              if (Find(x) == Find(y) || link[x] == y || type[y] == 1) continue;
73              if (type[y] == 2) contrack(x, y);
74              else if (link[y])
75              {
76                  nex[y] = x;
77                  type[y] = 1;
78                  type[link[y]] = 2;
79                  que.push(link[y]);
80              } else
81              {
82                  nex[y] = x;
83                  int pos = y, u = nex[pos], v = link[u];
84                  while (pos)
85                  {
86                      link[pos] = u;
87                      link[u] = pos;
88                      pos = v;
89                      u = nex[pos];
90                      v = link[u];
91                  }
92                  return;
93              }
94          }
95      }
96  }
97
98  int maxmatch()
99  {
100     for (int i = 1; i <= n; i++) if (!link[i]) bfs(i);
101     int ans = 0;
102     for (int i = 1; i <= n; i++) if (link[i]) ans++;
103     return ans / 2;
104 }
105
106 void init()
107 {
108     for (int i = 1; i <= n; i++) edge[i].clear();
109     memset(link, 0, sizeof(link));
110 }
```

2.5 网络流

2.5.1 Dinic

```
const int MAX_V = 1000 + 10;
const int INF = 0x3f3f3f3f;

//用于表示边的结构体（终点，流量，反向边）
struct edge{int to, cap, rev;};

vector<edge> G[MAX_V];  //图的邻接表表示
int level[MAX_V];   //顶点到源点的距离标号
int iter[MAX_V];    //当前弧

void add(int from, int to, int cap)
{
    G[from].push_back((edge){to, cap, G[to].size()});
    G[to].push_back((edge){from, 0, G[from].size() - 1});
}

//计算从源点出发的距离标号
void bfs(int s)
{
    memset(level, -1, sizeof(level));
    queue<int> que;
    level[s] = 0;
    que.push(s);
    while(!que.empty())
    {
        int v = que.front(); que.pop();
        for(int i = 0; i < G[v].size(); i++)
        {
            edge &e = G[v][i];
            if(e.cap > 0 && level[e.to] < 0)
            {
                level[e.to] = level[v] + 1;
                que.push(e.to);
            }
        }
    }
}

//通过DFS寻找增广路
int dfs(int v, int t, int f)
{
    if(v == t) return f;
    for(int &i = iter[v]; i<G[v].size(); i++)
    {
        edge &e = G[v][i];
        if(e.cap > 0 && level[v] < level[e.to])
        {
            int d = dfs(e.to, t, min(f, e.cap));
            if(d > 0)
            {
                e.cap -= d;
                G[e.to][e.rev].cap += d;
                return d;
            }
        }
```

```
56      }
57      return 0;
58  }
59
60  //求解从s到t的最大流
61  int max_flow(int s, int t)
62  {
63      int flow = 0;
64      for(;;)
65      {
66          bfs(s);
67          if(level[t] < 0) return flow;
68          memset(iter, 0, sizeof(iter));
69          int f;
70          while((f = dfs(s,t,INF)) > 0) flow += f;
71      }
72  }
```

2.5.2 ISAP

```
1   struct Edge {
2     int from, to, cap, flow;
3     Edge(int u, int v, int c, int f) : from(u), to(v), cap(c), flow(f) {}
4   };
5
6   bool operator<(const Edge& a, const Edge& b) {
7     return a.from < b.from || (a.from == b.from && a.to < b.to);
8   }
9
10  struct ISAP {
11    int n, m, s, t;
12    vector<Edge> edges;
13    vector<int> G[maxn];
14    bool vis[maxn];
15    int d[maxn];
16    int cur[maxn];
17    int p[maxn];
18    int num[maxn];
19
20    void AddEdge(int from, int to, int cap) {
21      edges.push_back(Edge(from, to, cap, 0));
22      edges.push_back(Edge(to, from, 0, 0));
23      m = edges.size();
24      G[from].push_back(m - 2);
25      G[to].push_back(m - 1);
26    }
27
28    bool BFS() {
29      memset(vis, 0, sizeof(vis));
30      queue<int> Q;
31      Q.push(t);
32      vis[t] = 1;
33      d[t] = 0;
34      while (!Q.empty()) {
35        int x = Q.front();
36        Q.pop();
37        for (int i = 0; i < G[x].size(); i++) {
38          Edge& e = edges[G[x][i] ^ 1];
```

```
39          if (!vis[e.from] && e.cap > e.flow) {
40            vis[e.from] = 1;
41            d[e.from] = d[x] + 1;
42            Q.push(e.from);
43          }
44        }
45      }
46      return vis[s];
47    }
48
49    void init(int n) {
50      this->n = n;
51      for (int i = 0; i < n; i++) G[i].clear();
52      edges.clear();
53    }
54
55    int Augment() {
56      int x = t, a = INF;
57      while (x != s) {
58        Edge& e = edges[p[x]];
59        a = min(a, e.cap - e.flow);
60        x = edges[p[x]].from;
61      }
62      x = t;
63      while (x != s) {
64        edges[p[x]].flow += a;
65        edges[p[x] ^ 1].flow -= a;
66        x = edges[p[x]].from;
67      }
68      return a;
69    }
70
71    int Maxflow(int s, int t) {
72      this->s = s;
73      this->t = t;
74      int flow = 0;
75      BFS();
76      memset(num, 0, sizeof(num));
77      for (int i = 0; i < n; i++) num[d[i]]++;
78      int x = s;
79      memset(cur, 0, sizeof(cur));
80      while (d[s] < n) {
81        if (x == t) {
82          flow += Augment();
83          x = s;
84        }
85        int ok = 0;
86        for (int i = cur[x]; i < G[x].size(); i++) {
87          Edge& e = edges[G[x][i]];
88          if (e.cap > e.flow && d[x] == d[e.to] + 1) {
89            ok = 1;
90            p[e.to] = G[x][i];
91            cur[x] = i;
92            x = e.to;
93            break;
94          }
95        }
96        if (!ok) {
97          int m = n - 1;
```

```
98          for (int i = 0; i < G[x].size(); i++) {
99            Edge& e = edges[G[x][i]];
100           if (e.cap > e.flow) m = min(m, d[e.to]);
101         }
102         if (--num[d[x]] == 0) break;
103         num[d[x] = m + 1]++;
104         cur[x] = 0;
105         if (x != s) x = edges[p[x]].from;
106       }
107     }
108     return flow;
109   }
110 };
```

2.5.3 MCMF

```
1  const int maxn = 10000 + 10;
2  const int inf = 0x3f3f3f3f;
3
4  struct Edge { int from, to, cap, flow, cost; };
5
6  struct MCMF
7  {
8      int n, m;
9      vector<Edge> edges;
10     vector<int> G[maxn];
11     bool inq[maxn];
12     int dis[maxn], path[maxn], a[maxn];
13
14     void init(int n)
15     {
16         this->n = n;
17         for(int i = 0;i <= n;i ++)
18             G[i].clear();
19         edges.clear();
20     }
21
22     void addEdge(int from, int to, int cap, int cost)
23     {
24         edges.push_back(Edge{from, to, cap, 0, cost});
25         edges.push_back(Edge{to, from, 0, 0, -cost});
26         m = edges.size();
27         G[from].push_back(m - 2);
28         G[to].push_back(m - 1);
29     }
30
31     bool Bellman_Ford(int s, int t, int& flow, int& cost)
32     {
33         for(int i = 0; i<= n; i++) dis[i] = inf;
34         memset(inq, 0, sizeof inq);
35         dis[s]=0, inq[s]=true, path[s]=0, a[s]=inf;
36         queue<int> Q;
37         Q.push(s);
38         while(!Q.empty())
39         {
40             int u = Q.front(); Q.pop();
41             inq[u] = false;
42             for(int i = 0; i < G[u].size(); i++)
```

```
43              {
44                  Edge& e = edges[G[u][i]];
45                  if(e.cap > e.flow && dis[e.to] > dis[u] + e.cost)
46                  {
47                      dis[e.to] = dis[u] + e.cost;
48                      path[e.to] = G[u][i];
49                      a[e.to] = min(a[u], e.cap - e.flow);
50                      if(!inq[e.to])
51                      {
52                          Q.push(e.to);
53                          inq[e.to] = true;
54                      }
55                  }
56              }
57          }
58          if(dis[t] == inf) return false;     //求最小费用最大流
59          //if(1ll * dis[t] * a[t] > 0) return false; 求可行流最小费用，因此当费用增量大于0时不继续增加
        流量
60          flow += a[t];
61          cost += dis[t] * a[t];
62          for(int u = t; u != s; u = edges[path[u]].from)
63          {
64              edges[path[u]].flow += a[t];
65              edges[path[u] ^ 1].flow -= a[t];
66          }
67          return true;
68      }
69
70      int mincostMaxFlow(int s, int t)
71      {
72          int flow = 0, cost = 0;
73          while(Bellman_Ford(s, t, flow, cost));
74          return cost;
75      }
76  };
```

2.5.4 Trick

建模技巧

二分图带权最大独立集。给出一个二分图，每个结点上有一个正权值。要求选出一些点，使得这些点之间没有边相连，且权值和最大。

解： 在二分图的基础上添加源点 S 和汇点 T，然后从 S 向所有 X 集合中的点连一条边，所有 Y 集合中的点向 T 连一条边，容量均为该点的权值。X 结点与 Y 结点之间的边的容量均为无穷大。这样，对于图中的任意一个割，将割中的边对应的结点删掉就是一个符合要求的解，权和为所有权减去割的容量。因此，只需要求出最小割，就能求出最大权和。

公平分配问题。把 m 个任务分配给 n 个处理器。其中每个任务有两个候选处理器，可以任选一个分配。要求所有处理器中，任务数最多的那个处理器所分配的任务数尽量少。不同任务的候选处理器集 $\{p_1, p_2\}$ 保证不同。

解： 本题有一个比较明显的二分图模型，即 X 结点是任务，Y 结点是处理器。二分答案 x，然后构图，首先从源点 S 出发向所有的任务结点引一条边，容量等于 1，然后从每个任务结点出发引两条边，分别到达它所能分配到的两个处理器结点，容量为 1，最后从每个处理器结点出发引一条边到汇点 T，容量为 x，表示选择该处理器的任务不能超过 x。这样网络中的每个单位流量都是从 S 流到一个任务结点，再到处理器结点，最后到汇点 T。只有当网络中的总流量等于 m 时才意味着所有任务都选择了一个处理器。这样，我们通过 $O(\log m)$ 次最大流便算出了答案。

区间 k 覆盖问题。数轴上有一些带权值的左闭右开区间。选出权和尽量大的一些区间，使得任意一个数最多被 k 个区间覆盖。

解：本题可以用最小费用流解决，构图方法是把每个数作为一个结点，然后对于权值为 w 的区间 $[u,v)$ 加边 $u \to v$，容量为 1，费用为 $-w$。再对所有相邻的点加边 $i \to i+1$，容量为 k，费用为 0。最后，求最左点到最右点的最小费用最大流即可，其中每个流量对应一组互不相交的区间。如果数值范围太大，可以先进行离散化。

最大闭合子图。给定带权图 G（权值可正可负），求一个权和最大的点集，使得起点在该点集中的任意弧，终点也在该点集中。

解：新增附加源 s 和附加汇 t，从 s 向所有正权点引一条边，容量为权值；从所有负权点向汇点引一条边，容量为权值的相反数。求出最小割以后，$S-\{s\}$ 就是最大闭合子图。

最大密度子图。给出一个无向图，找一个点集，使得这些点之间的边数除以点数的值（称为子图的密度）最大。

解：如果两个端点都选了，就必然要选边，这就是一种推导。如果把每个点和每条边都看成新图中的结点，可以把问题转化为最大闭合子图。

无源汇有上下界可行流：附加源 S 和汇 T；对于边 (u,v,min,max)，记 $d[u]-=min, d[v]+=max$，并添加弧 $(u,v,max-min)$；对于流量不平衡的点 u，设多余流量为 W，如果 $W>0$，添加弧 $S->u:W$，否则若 $W<0$，添加弧 $u->T:-W$，求改造后的网络 $S-T$ 最大流即可，当且仅当所有附加弧满载时原图有可行流。

有源汇有上下界可行流：建 $t->s$，容量为 inf，然后和无源汇相同。

有源汇有上下界最大/最小流：与上面相同，跑完可行流 $S->T$ 后去掉边 $t->s$，最大流为加 $s->t$，最小流为 $G[s][t].cap - max_flow(t,s)$。

2.6 Others

2.6.1 拓扑排序

```
const int maxn = 1e5 + 10;

vector<int> edge[maxn];
int indegree[maxn];

void add(int u, int v)
{
    edge[u].push_back(v);
    indegree[v]++;
}

void Toposort(int n)
{
    queue<int> que;
    for (int i = 1; i <= n; i++)
        if (!indegree[i]) que.push(i);    //将图中没有前驱，即入度为0的点加入队列
    while (!que.empty())
    {
        int u = que.front();
        que.pop();
        indegree[u] = -1;    //从图中删去此顶点
        for (int i = 0; i < edge[u].size(); i++)
        {
            int v = edge[u][i];
            indegree[v]--;    //删去图中以u为尾的弧
            if (!indegree[v]) que.push(v);    //将新增的当前入度为0的点压入队列中
        }
    }
}
```

2.6.2 2-SAT

```cpp
const int maxn = 2e6 + 10;

int n, m, a, va, b, vb;
int low[maxn], dfn[maxn], color[maxn], cnt, scc_cnt;
bool instack[maxn];

vector<int> g[maxn];

void Tarjan(int u)
{
    low[u] = dfn[u] = ++cnt;
    st.push(u);
    instack[u] = true;
    for(const auto &v : g[u])
    {
        if(!dfn[v]) Tarjan(v), low[u] = min(low[u], low[v]);
        else if(instack[v]) low[u] = min(low[u], dfn[v]);
    }
    if(low[u] == dfn[u])
    {
        ++scc_cnt;
        do {
            color[u] = scc_cnt;
            u = st.top(); st.pop();
            instack[u] = false;
        } while(low[u] != dfn[u]);
    }
}

void 2_SAT()
{
    scanf("%d%d", &n, &m);
    for(int i = 0; i < m; i ++)
    {
        scanf("%d%d%d%d", &a, &va, &b, &vb);
        g[ a + n * (va & 1) ].push_back(b + n * (vb ^ 1));
        g[ b + n * (vb & 1) ].push_back(a + n * (va ^ 1));
    }
    cnt = scc_cnt = 0;
    for(int i = 1; i <= (n << 1); i ++) if(!dfn[i]) Tarjan(i);
}
```

2.6.3 差分约束系统

```cpp
const int maxn = 1000 + 10;
const int inf = 0x3f3f3f3f;

struct Edge
{
    int nex, to, w;
} edge[10 * maxn];

int head[maxn], cnt, dis[maxn], n;
bool vis[maxn];

void init()
```

```
13  {
14      cnt = 0;
15      memset(head, 0xff, sizeof head);
16  }
17
18  void add(int u, int v, int w)
19  {
20      edge[cnt].nex = head[u];
21      edge[cnt].to = v;
22      edge[cnt].w = w;
23      head[u] = ++cnt;
24  }
25
26  void spfa(int u)
27  {
28      int u, v, w;
29      for (int i = 1; i <= n; i++) dis[i] = inf, vis[i] = false;
30      dis[u] = 0;
31      queue<int> que;
32      que.push(u);
33      vis[u] = true;
34      while (!que.empty())
35      {
36          u = que.front();
37          que.pop();
38          vis[u] = false;
39          for (int i = head[u]; ~i; i = edge[i].nex)
40          {
41              v = edge[i].v, w = edge[i].w;
42              if (dis[u] + w < dis[v])
43              {
44                  dis[v] = dis[u] + w;
45                  if (!vis[v])
46                  {
47                      que.push(v);
48                      vis[v] = true;
49                  }
50              }
51          }
52      }
53  }
```

3 DataStructrue

3.1 SegmentTreeDS

3.1.1 SegmentTree

```
struct SegmentTree {
#define TYPE int
    TYPE val[maxn << 2];
    int sz;
    // check this type
    int lazy[maxn << 2];

    inline TYPE comb(const TYPE& a, const TYPE& b) {
        TYPE res;
        res = a + b;
        return res;
    }

    int le, re, k;

    inline void init(int sz_) {
        sz = sz_;
        memset(lazy, 0, sizeof lazy);
        memset(val, 0, sizeof val);
    }
    inline void pushup(int rt) {val[rt] = comb(val[rt << 1], val[rt << 1 | 1]);}
    inline void deal(int rt, int kt) {
        ;
    }
    inline void pushdown(int rt, int len) {
        if (lazy[rt]) {
            // check the lazy change
            lazy[rt << 1] += lazy[rt];
            lazy[rt << 1 | 1] += lazy[rt];
            deal(rt << 1, lazy[rt]);
            deal(rt << 1 | 1, lazy[rt]);
            lazy[rt] = 0;
        }
    }

    inline void build(int rt, int l, int r) {
        if (l == r) {
            val[rt] = a[l];
            return;
        }
        int mid = l + r >> 1;
        build(rt << 1, l, mid);
        build(rt << 1 | 1, mid + 1, r);
        pushup(rt);
    }
    inline void build() {build(1, 1, sz);}

    inline void update(int rt, int l, int r) {
        if (le <= l && r <= re) {
            deal(rt, k);
            return;
        }
        pushdown(rt, r - l + 1);
```

```
        int mid = l + r >> 1;
        if (le <= mid) update(rt << 1, l, mid);
        if (re > mid) update(rt << 1 | 1, mid + 1, r);
        pushup(rt);
    }

    inline TYPE query(int rt, int l, int r) {
        if (le <= l && r <= re) {
            return val[rt];
        }
        pushdown(rt, r - l + 1);
        // check the zero type
        TYPE res;
        int mid = l + r >> 1;
        if (le <= mid) res = comb(res, query(rt << 1, l, mid));
        if (re > mid) res = comb(res, query(rt << 1 | 1, mid + 1, r));
        return res;
    }

    // check return type
    inline int query(int l, int r) {
        le = l, re = r;
        return query(1, 1, sz);
    }
    inline void modify(int l, int r, int kt) {
        le = l, re = r, k = kt;
        update(1, 1, sz);
    }

    inline void pt(int rt, int l, int r) {
        if (l == r) {
            printf("%d ", val[l]);
            return;
        }
        pushdown(rt, r - l + 1);
        int mid = l + r >> 1;
        if (le <= mid) pt(rt << 1, l, mid);
        if (re > mid) pt(rt << 1 | 1, mid + 1, r);
    }

#undef TYPE
};
```

3.1.2 离散化区间

```
// 原题1e5个个区间有2e5个端点，离散化出来4e5个区间
// 然后线段树需要4e5*4=16e5的大小
// 注意三个数组要开离散化数量的四倍，如果不需要sz可以不用这个数组。
int val[maxn << 4];
int lpos[maxn << 2], rpos[maxn << 2], tot, sz[maxn << 2];
sort(xpos.begin(), xpos.end());
xpos.erase(unique(xpos.begin(), xpos.end()), xpos.end());
tot = 1;
lpos[1] = rpos[1] = xpos[0];
sz[1] = 1;
for (int i = 1; i < xpos.size(); ++i) {
    if (xpos[i] - xpos[i - 1] != 1) {
        lpos[++tot] = xpos[i - 1] + 1;
```

```
        rpos[tot] = xpos[i] - 1;
        sz[tot] = rpos[tot] - lpos[tot] + 1;
    }
    ++tot;
    lpos[tot] = rpos[tot] = xpos[i];
    sz[tot] = 1;
}
le = lower_bound(lpos + 1, lpos + 1 + tot, p[i].x) - lpos;
re = upper_bound(rpos + 1, rpos + 1 + tot, p[i].y) - rpos - 1;
```

3.1.3 动态区间最大子段和

```
namespace ST {
    struct node{
        ll ans,ls,rs,sum;
    }xx[maxn << 2];
    inline void pushdown(int x){
        xx[x].sum=xx[x<<1].sum+xx[x<<1|1].sum;
        xx[x].ls=max(xx[x<<1].ls,xx[x<<1].sum+xx[x<<1|1].ls);
        xx[x].rs=max(xx[x<<1|1].rs,xx[x<<1|1].sum+xx[x<<1].rs);
        xx[x].ans=max(xx[x<<1].ans,max(xx[x<<1|1].ans,xx[x<<1].rs+xx[x<<1|1].ls));
        return;
    }
    inline void build(int k,int l,int r){
        if(l==r){
            xx[k].ls=xx[k].rs=xx[k].ans=xx[k].sum=0;
            return;
        }
        int mid=l+r>>1;
        build(k<<1,l,mid),build(k<<1|1,mid+1,r);
        pushdown(k);
        return;
    }
    inline void change(int k,int l,int r,int x,int y,int w){ // 1, 1, n
        if(x<=l&&r<=y){
            xx[k].ls += w;
            xx[k].rs += w;
            xx[k].ans += w;
            xx[k].sum += w;
//            xx[k].ls=xx[k].rs=xx[k].ans=xx[k].sum=w;
            return;
        }
        int mid=l+r>>1;
        if(x<=mid) change(k<<1,l,mid,x,y,w);
        if(mid<y) change(k<<1|1,mid+1,r,x,y,w);
        pushdown(k);
        return;
    }
    inline node query(int k,int l,int r,int x,int y){
        if(x<=l&&r<=y) {
            return xx[k];
        }
        int mid=l+r>>1;
        if(x<=mid&&!(mid<y)) return query(k<<1,l,mid,x,y);
        else if(!(x<=mid)&&mid<y) return query(k<<1|1,mid+1,r,x,y);
        else{
            node st,t1=query(k<<1,l,mid,x,y),t2=query(k<<1|1,mid+1,r,x,y);
            st.sum=t1.sum+t2.sum;
```

```
            st.ls=max(t1.ls,t1.sum+t2.ls);
            st.rs=max(t2.rs,t2.sum+t1.rs);
            st.ans=max(t1.ans,max(t2.ans,t1.rs+t2.ls));
            return st;
        }
    }
}
```

3.1.4 动态开点权值线段树

```
int root[100005];
int ls[1800000], rs[1800000], sum[1800000];
int sz = 0;

void insert(int &k, int l, int r, int val){
    if (!k) k = ++sz;
    if (l == r) {
        sum[k] = 1;
        return;
    }
    int mid = (l + r) >> 1;
    if (val <= mid) insert(ls[k], l, mid, val);
    else insert(rs[k], mid + 1, r, val);
    sum[k] = sum[ls[k]] + sum[rs[k]];
}

int query(int k, int l, int r, int rank) {
    if (l == r) return l;
    int mid = (l + r) >> 1;
    if (sum[ls[k]] >= rank) return query(ls[k], l, mid, rank);
    else return query(rs[k], mid + 1, r, rank - sum[ls[k]]);
}
int merge(int x, int y)
{
    if (!x) return y;
    if (!y) return x;
    ls[x] = merge(ls[x], ls[y]);
    rs[x] = merge(rs[x], rs[y]);
    sum[x] = sum[ls[x]] + sum[rs[x]];
    return x;
}
insert(root[i], 1, n, a[i]);
query(root[p], 1, n, x);
```

3.2 HLD

3.2.1 HLD

```
#include <bits/stdc++.h>
#define ll long long
using namespace std;
/*
node 计算点权， path 下放后计算边权， edge 根据边的编号计算边权
work 中没有build需手动写
sz[]数组，以x为根的子树节点个数
top[]数组，当前节点的所在链的顶端节点
son[]数组，重儿子
deep[]数组，当前节点的深度
```

```
11  fa[]数组，当前节点的父亲
12  idx[]数组，树中每个节点剖分后的新编号
13  rnk[]数组，idx的逆，表示线段上中当前位置表示哪个节点
14  */
15
16  const int maxn = 1e5+5;
17
18  int sz[maxn], top[maxn], son[maxn], deep[maxn], fa[maxn], idx[maxn], rnk[maxn];
19  int tot;
20  int n, le, re;
21  ll k;
22
23  struct HLD {
24  #define type int
25
26      struct edge {
27          int a, b;
28          type v;
29
30          edge(int _a, int _b, type _v = 0) : a(_a), b(_b), v(_v) {}
31      };
32
33      struct node {
34          int to;
35          type w;
36
37          node() {}
38
39          node(int _to, type _w) : to(_to), w(_w) {}
40      };
41
42      vector<int> mp[maxn];
43      vector<edge> e;
44
45      void init(int _n) {
46          n = _n;
47          for (int i = 0; i <= n; i++) mp[i].clear();
48          e.clear();
49          e.push_back(edge(0, 0));
50      }
51
52      void add_edge(int a, int b, type v = 0) {
53  //        e.push_back(edge(a,b,v));
54          mp[a].push_back(b);
55          mp[b].push_back(a);
56      }
57
58      void dfs1(int x, int pre, int h) {
59          int i, to;
60          deep[x] = h;
61          fa[x] = pre;
62          sz[x] = 1;
63          for (i = 0; i < (int) (mp[x].size()); i++) {
64              to = mp[x][i];
65              if (to == pre) continue;
66              dfs1(to, x, h + 1);
67              sz[x] += sz[to];
68              if (son[x] == -1 || sz[to] > sz[son[x]]) son[x] = to;
69          }
```

```
70      }
71
72      void dfs2(int x, int tp) {
73          int i, to;
74          top[x] = tp;
75          idx[x] = ++tot;
76          rnk[idx[x]] = x;
77          if (son[x] == -1) return;
78          dfs2(son[x], tp);
79          for (i = 0; i < (int) (mp[x].size()); i++) {
80              to = mp[x][i];
81              if (to != son[x] && to != fa[x]) dfs2(to, to);
82          }
83      }
84
85      void work(int _rt = 1) {
86          memset(son, -1, sizeof son);
87          tot = 0;
88          dfs1(_rt, 0, 0);
89          dfs2(_rt, _rt);
90      }
91
92      int LCA(int x, int y) {
93          while (top[x] != top[y]) {
94              if (deep[top[x]] < deep[top[y]]) swap(x, y);
95              x = fa[top[x]];
96          }
97          if (deep[x] > deep[y]) swap(x, y);
98          return x;
99      }
100
101     void modify_node(int x, int y, type val) {
102         while (top[x] != top[y]) {
103             if (deep[top[x]] < deep[top[y]]) swap(x, y);
104             le = idx[top[x]], re = idx[x];
105             k = val;
106             update(1, 1, n);
107             x = fa[top[x]];
108         }
109         if (deep[x] > deep[y]) swap(x, y);
110         le = idx[x], re = idx[y];
111         k = val;
112         update(1, 1, n);
113     }
114
115     type query_node(int x, int y) {
116         type res = 0;
117         while (top[x] != top[y]) {
118             if (deep[top[x]] < deep[top[y]]) swap(x, y);
119             le = idx[top[x]], re = idx[x];
120             res += query(1, 1, n);
121             x = fa[top[x]];
122         }
123         if (deep[x] > deep[y]) swap(x, y);
124         le = idx[x], re = idx[y];
125         res += query(1, 1, n);
126         return res;
127     }
128
```

```
129     //path
130 //    void init_path()
131 //    {
132 //        v[idx[rt]]=0;
133 //        for(int i=1;i<n;i++)
134 //        {
135 //            if(deep[e[i].a]<deep[e[i].b]) swap(e[i].a,e[i].b);
136 //            a[idx[e[i].a]]=e[i].v;
137 //        }
138 //        build(n);
139 //    }
140     void modify_edge(int id, type val) {
141         if (deep[e[id].a] > deep[e[id].b]) {
142             le = idx[e[id].a], re = idx[e[id].a];
143             k = val;
144             update(1, 1, n);
145         } else {
146             le = idx[e[id].b], re = idx[e[id].b];
147             k = val;
148             update(1, 1, n);
149         }
150     }
151 
152     void modify_path(int x, int y, type val) {
153         while (top[x] != top[y]) {
154             if (deep[top[x]] < deep[top[y]]) swap(x, y);
155             le = idx[top[x]], re = idx[x];
156             k = val;
157             update(1, 1, n);
158             x = fa[top[x]];
159         }
160         if (deep[x] > deep[y]) swap(x, y);
161         if (x != y) {
162             le = idx[x] + 1, re = idx[y];
163             k = val;
164             update(1, 1, n);
165         }
166     }
167 
168     type query_path(int x, int y) {
169         type res = 0;
170         while (top[x] != top[y]) {
171             if (deep[top[x]] < deep[top[y]]) swap(x, y);
172             le = idx[top[x]], re = idx[x];
173             res += query(1, 1, n);
174             x = fa[top[x]];
175         }
176         if (deep[x] > deep[y]) swap(x, y);
177         if (x != y) {
178             le = idx[x] + 1, re = idx[y];
179             res += query(1, 1, n);
180         }
181         return res;
182     }
183 
184 #undef type
185 } hld;
```

3.3 RMQ

3.3.1 RMQ

```
//一维RMQ
//MAX=1e6时 第二维开22 内存(int型)占10w
int v[MAX],maxx[MAX][22],minn[MAX][22];
void RMQ(int n)
{
    int i,j;
    for(i=1;i<=n;i++)
    {
        maxx[i][0]=minn[i][0]=v[i];//下标rmq 初始化赋值成i
        for(j=1;1<<(j-1)<=n;j++)
        {
            maxx[i][j]=0;
            minn[i][j]=INF;
        }
    }
    for(j=1;1<<(j-1)<=n;j++)
    {
        for(i=1;i+(1<<j)-1<=n;i++)
        {
            int t=1<<(j-1);
            maxx[i][j]=max(maxx[i][j-1],maxx[i+t][j-1]);
            minn[i][j]=min(minn[i][j-1],minn[i+t][j-1]);
        }
    }
}
int query(int l,int r)
{
    int j=(int)(log10(r-l+1)/log10(2))+1;
    int i=r-(1<<(j-1))+1;
    return max(maxx[l][j-1],maxx[i][j-1]);
//  return min(minn[l][j-1],minn[i][j-1]);
}
```

3.3.2 RMQbyIndex

```
//下标RMQ
int v[MAX],maxx[MAX][22],minn[MAX][22];
int pmax(int a,int b){return v[a]>v[b]?a:b;}
int pmin(int a,int b){return v[a]<v[b]?a:b;}
void RMQ(int n)
{
    int i,j;
    for(i=1;i<=n;i++)
    {
        maxx[i][0]=minn[i][0]=i;
    }
    for(j=1;1<<(j-1)<=n;j++)
    {
        for(i=1;i+(1<<j)-1<=n;i++)
        {
            int t=1<<(j-1);
            maxx[i][j]=pmax(maxx[i][j-1],maxx[i+t][j-1]);
            minn[i][j]=pmin(minn[i][j-1],minn[i+t][j-1]);
        }
    }
```

```
}
int query(int l,int r)
{
    int j=(int)(log10(r-l+1)/log10(2))+1;
    int i=r-(1<<(j-1))+1;
    return pmax(maxx[l][j-1],maxx[i][j-1]);
//  return pmin(minn[l][j-1],minn[i][j-1]);
}
```

3.3.3 RMQinNM

```
//二维RMQ
int v[302][302];
int maxx[302][302][9][9],minn[302][302][9][9];
void RMQ(int n,int m)
{
    int i,j,ii,jj;
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=m;j++)
        {
            maxx[i][j][0][0]=minn[i][j][0][0]=v[i][j];
        }
    }
    for(ii=0;(1<<ii)<=n;ii++)
    {
        for(jj=0;(1<<jj)<=m;jj++)
        {
            if(ii+jj)
            {
                for(i=1;i+(1<<ii)-1<=n;i++)
                {
                    for(j=1;j+(1<<jj)-1<=m;j++)
                    {
                        if(ii)
                        {
                            minn[i][j][ii][jj]=min(minn[i][j][ii-1][jj],minn[i+(1<<(ii-1))][j][ii
    -1][jj]);
                            maxx[i][j][ii][jj]=max(maxx[i][j][ii-1][jj],maxx[i+(1<<(ii-1))][j][ii
    -1][jj]);
                        }
                        else
                        {
                            minn[i][j][ii][jj]=min(minn[i][j][ii][jj-1],minn[i][j+(1<<(jj-1))][ii][
    jj-1]);
                            maxx[i][j][ii][jj]=max(maxx[i][j][ii][jj-1],maxx[i][j+(1<<(jj-1))][ii][
    jj-1]);
                        }
                    }
                }
            }
        }
    }
}
int query(int x1,int y1,int x2,int y2)
{
    int k1=0;
    while((1<<(k1+1))<=x2-x1+1) k1++;
```

```
    int k2=0;
    while((1<<(k2+1))<=y2-y1+1) k2++;
    x2=x2-(1<<k1)+1;
    y2=y2-(1<<k2)+1;
    return max(max(maxx[x1][y1][k1][k2],maxx[x1][y2][k1][k2]),max(maxx[x2][y1][k1][k2],maxx[x2][y2][k1][k2]))
//  return min(min(minn[x1][y1][k1][k2],minn[x1][y2][k1][k2]),min(minn[x2][y1][k1][k2],minn[x2][y2][k1][k2]));
}
```

3.4 MO

3.4.1 MO

```
#include <bits/stdc++.h>
#define ll long long
using namespace std;
const int maxn = 200005;

struct MO {
    int l, r, id;
}q[maxn];

int n, m, col[maxn], block, belong[maxn];
ll vis[maxn * 10], ans;
ll res[maxn];
bool cmp(const MO& a, const MO& b) { return belong[a.l] == belong[b.l] ? a.r < b.r : a.l < b.l; }
void add(ll x) {
    vis[x] ++;
    ans += x * (vis[x] * vis[x] - (vis[x] - 1) * (vis[x] - 1));
}

void del(ll x) {
    vis[x] --;
    ans -= x * ((vis[x] + 1) * (vis[x] + 1) - vis[x] * vis[x]);
}

int main() {
    scanf("%d%d", &n, &m);
    block = sqrt(n);
    for (int i = 1; i <= n; ++i) {
        scanf("%d", &col[i]);
        belong[i] = i / block + 1;
    }
    for (int i = 1; i <= m; ++i) {
        scanf("%d%d", &q[i].l, &q[i].r);
        q[i].id = i;
    }
    sort(q + 1, q + 1 + m, cmp);
    int l = 1, r = 0;
    for (int i = 1; i <= m; ++i) {
        while(r < q[i].r) add(col[++r]);
        while(r > q[i].r) del(col[r--]);
        while(l < q[i].l) del(col[l++]);
        while(l > q[i].l) add(col[--l]);
        res[q[i].id] = ans;
    }
    for (int i = 1; i <= m; ++i) printf("%lld\n", res[i]);
    return 0;
```

```
}
```

3.4.2 MObyModify

```
#include <bits/stdc++.h>
#define ll long long
using namespace std;
const int maxn = 50005;

struct MO {
    int l, r, id, oppre;
}q[maxn];

int n, m, col[maxn], block, belong[maxn], colpre[maxn];
int changepos[maxn], changepre[maxn], changenow[maxn];
int vis[maxn * 20];
int ans;
int res[maxn];
bool cmp(const MO& a, const MO& b) {
    if (belong[a.l] != belong[b.l]) return a.l < b.l;
    if (belong[a.r] != belong[b.r]) return a.r < b.r;
    return a.oppre < b.oppre;
}
void add(int x) {}

void del(int x) {}

void unmodify(int pos, int now) {
    if (q[pos].l <= changepos[now] && changepos[now] <= q[pos].r) {
        del(changenow[now]);
        add(changepre[now]);
    }
    col[changepos[now]] = changepre[now];
}

void modify(int pos, int now) {
    if (q[pos].l <= changepos[now] && changepos[now] <= q[pos].r) {
        del(changepre[now]);
        add(changenow[now]);
    }
    col[changepos[now]] = changenow[now];
}

int main() {
    scanf("%d%d", &n, &m);
    block = pow(n, 0.66666);
    for (int i = 1; i <= n; ++i) {
        scanf("%d", &col[i]);
        colpre[i] = col[i];
        belong[i] = i / block + 1;
    }
    char s[2];
    int t = 0, t2 = 0;
    for (int i = 1; i <= m; ++i) {
        scanf("%s", s);
        if (s[0] == 'Q') {
            ++t;
            scanf("%d%d", &q[t].l, &q[t].r);
```

```
            q[t].oppre = t2;
            q[t].id = t;
        } else {
            ++t2;
            scanf("%d%d", &changepos[t2], &changenow[t2]);
            changepre[t2] = colpre[changepos[t2]];
            colpre[changepos[t2]] = changenow[t2];
        }
    }
    sort(q + 1, q + 1 + t, cmp);
    int l = 1, r = 0, now = 0;
    for (int i = 1; i <= t; ++i) {
        while(r < q[i].r) add(col[++r]);
        while(r > q[i].r) del(col[r--]);
        while(l < q[i].l) del(col[l++]);
        while(l > q[i].l) add(col[--l]);
        while (now < q[i].oppre) modify(i, ++now);
        while (now > q[i].oppre) unmodify(i, now--);
        res[q[i].id] = ans;
    }
    for (int i = 1; i <= t; ++i) printf("%d\n", res[i]);
    return 0;
}
```

3.5 VirtualTree

3.5.1 VirtualTree

```
const int maxn = "Edit";
vector<int> vtree[maxn];
void build(vector<int>& vec)
{
    sort(vec.begin(), vec.end(), [&](int x, int y) { return dfn[x] < dfn[y]; });
    static int s[maxn];
    int top = 0;
    s[top] = 0;
    vtree[0].clear();
    for (auto& u : vec)
    {
        int vlca = lca(u, s[top]);
        vtree[u].clear();
        if (vlca == s[top])
            s[++top] = u;
        else
        {
            while (top && dep[s[top - 1]] >= dep[vlca])
            {
                vtree[s[top - 1]].push_back(s[top]);
                top--;
            }
            if (s[top] != vlca)
            {
                vtree[vlca].clear();
                vtree[vlca].push_back(s[top--]);
                s[++top] = vlca;
            }
            s[++top] = u;
        }
    }
```

```
    for (int i = 0; i < top; ++i) vtree[s[i]].push_back(s[i + 1]);
}
```

3.6 PersistentDS

3.6.1 主席树区间 k 大

```
/*************************************************************************
    > File Name: a.cpp
    > Author: badcw
    > Mail: 952223482@qq.com
    > Created Time: 2018年07月21日 星期六 08时47分54秒
 ************************************************************************/

#include <bits/stdc++.h>
#define ll long long
using namespace std;

const int maxn = 100005;
int n, m;
int a[maxn];
int root[maxn];
int cnt = 0;
vector<int> b;
struct node {
    int l, r, val;
}p[maxn * 40];

void update(int l, int r, int pre, int &now, int pos) {
    now = ++cnt;
    p[now] = p[pre];
    p[now].val++;
    if (l == r) {
        return;
    }
    int mid = l + r >> 1;
    if (pos <= mid) update(l, mid, p[pre].l, p[now].l, pos);
    else update(mid + 1, r, p[pre].r, p[now].r, pos);
}

int query(int l, int r, int x, int y, int k) {
    if (l == r) return b[l - 1];
    int mid = l + r >> 1;
    int temp = p[p[y].l].val - p[p[x].l].val;
    if (k <= temp) return query(l, mid, p[x].l, p[y].l, k);
    return query(mid + 1, r, p[x].r, p[y].r, k - temp);
}

int main(int argc,char *argv[])
{
    while (scanf("%d%d", &n, &m) != EOF) {
        b.clear();
        cnt = 0;
        for (int i = 1; i <= n; ++i) scanf("%d", &a[i]), b.push_back(a[i]);
        sort(b.begin(), b.end());
        b.erase(unique(b.begin(), b.end()), b.end());
        for (int i = 1; i <= n; ++i) {
            update(1, b.size(), root[i - 1], root[i], lower_bound(b.begin(), b.end(), a[i]) - b.
    begin() + 1);
```

```
        }
        int L, R, k;
        while (m--) {
            scanf("%d%d%d", &L, &R, &k);
            printf("%d\n", query(1, b.size(), root[L - 1], root[R], k));
        }
    }
    return 0;
}
```

3.6.2 可持久化数组

```
/*1、操作将u, v合并 2、操作回退 */
const int maxn = 2e5+5;
int n, m, sz;
int root[maxn],ls[maxn*40],rs[maxn*40],v[maxn*40],deep[maxn*40];
int has[maxn];

void build(int &k, int l, int r) {
    if (!k)k = ++sz;
    if (l == r) {
        v[k] = l;
        return;
    }
    int mid = (l + r) >> 1;
    build(ls[k], l, mid);
    build(rs[k], mid + 1, r);
}

void modify(int l, int r, int x, int &y, int pos, int val) {
    y = ++sz;
    if (l == r) {
        v[y] = val;
        deep[y] = deep[x];
        return;
    }
    ls[y] = ls[x];
    rs[y] = rs[x];
    int mid = (l + r) >> 1;
    if (pos <= mid)
        modify(l, mid, ls[x], ls[y], pos, val);
    else modify(mid + 1, r, rs[x], rs[y], pos, val);
}

int query(int k, int l, int r, int pos) {
    if (l == r)return k;
    int mid = (l + r) >> 1;
    if (pos <= mid)return query(ls[k], l, mid, pos);
    else return query(rs[k], mid + 1, r, pos);
}

void add(int k, int l, int r, int pos) {
    if (l == r) {
        deep[k]++;
        return;
    }
    int mid = (l + r) >> 1;
    if (pos <= mid)add(ls[k], l, mid, pos);
```

```
47      else add(rs[k], mid + 1, r, pos);
48  }
49
50  int find(int k, int x) {
51      int p = query(k, 1, n, x);
52      if (x == v[p])return p;
53      return find(k, v[p]);
54  }
55
56  int main() {
57      int T = read();
58      while (T--) {
59          sz = 0;
60          memset(root, 0, sizeof root);
61          memset(ls, 0, sizeof ls);
62          memset(rs, 0, sizeof rs);
63          n = read();
64          has[0] = n;
65          m = read();
66          build(root[0], 1, n);
67          int f, k, a, b;
68          for (int i = 1; i <= m; i++) {
69              f = read();
70              if (f == 1) {
71                  root[i] = root[i - 1];
72                  has[i] = has[i - 1];
73                  a = read();
74                  b = read();
75                  int p = find(root[i], a), q = find(root[i], b);
76                  if (v[p] == v[q])continue;
77                  has[i]--;
78                  if (deep[p] > deep[q])swap(p, q);
79                  modify(1, n, root[i - 1], root[i], v[p], v[q]);
80                  if (deep[p] == deep[q])add(root[i], 1, n, v[q]);
81              } else if (f == 2) {
82                  k = read();
83                  root[i] = root[k];
84                  has[i] = has[k];
85              }
86              printf("%d\n", has[i]);
87          }
88      }
89      return 0;
90  }
```

3.7 Tree

3.7.1 LCA

```
1   const int maxn = 1e5 + 10;
2
3   int n, dep[maxn], fa[maxn][30];
4   vector<int> edge[maxn];
5
6   void dfs(int u, int pre)
7   {
8       dep[u] = dep[pre] + 1, fa[u][0] = pre;
9       for(int i = 1; (1 << i) <= n; i ++)
10          fa[u][i] = fa[fa[u][i - 1]][i - 1];
```

```
    for(auto v : edge[u]) if(v != pre) dfs(v, u);
}

int LCA(int u, int v)
{
    if(dep[u] < dep[v]) swap(u, v);
    int d = dep[u] - dep[v];
    for(int i = 0; (1 << i) <= d; i ++)
        if((1 << i) & d) u = fa[u][i];
    if(u == v) return u;
    for(int i = 20; i >= 0; i --)
        if(fa[u][i] != fa[v][i])
            u = fa[u][i], v = fa[v][i];
    return fa[u][0];
}
```

3.7.2 前向星

```
// 清零 head 和 tot
const int maxm = 4e5+5;
int ver[maxm], Next[maxm], head[maxn], edge[maxm];
void addEdge(int u, int v, int w){
    ver[++tot]=v;
    Next[tot]=head[u];
    head[u]=tot;
    edge[tot]=w;
}

for(int i = head[u]; i; i=Next[i])
```

3.7.3 点分治

```
int n, k;

// 清零 head 和 tot
const int maxm = 2e4+5;
int ver[maxm], Next[maxm], head[maxn], edge[maxm];
int tot;
void addEdge(int u, int v, int w){
    ver[++tot]=v;
    Next[tot]=head[u];
    head[u]=tot;
    edge[tot]=w;
}

int sz[maxn], vis[maxn];
int rt, mxsz, has;

void getrt(int u, int pre) {
    sz[u] = 1;
    int mxnow = 0;
    for (int i = head[u]; i; i = Next[i]) {
        int v = ver[i];
        if (v == pre || vis[v]) continue;
        getrt(v, u);
        sz[u] += sz[v];
        mxnow = max(mxnow, sz[v]);
    }
```

```
27      mxnow = max(mxnow, has - sz[u]);
28      if (mxnow < mxsz) {
29          mxsz = mxnow, rt = u;
30      }
31  }
32
33  int dl[maxn], r;
34  int val[maxn];
35
36  void getdis(int u, int pre) {
37      dl[r++] = val[u];
38      for (int i = head[u]; i; i = Next[i]) {
39          int v = ver[i];
40          if (v == pre || vis[v]) continue;
41          val[v] = val[u] + edge[i];
42          getdis(v, u);
43      }
44  }
45
46  ll cal(int u, int pre) {
47      r = 0;
48      val[u] = pre;
49      getdis(u, 0);
50      ll sum = 0;
51      sort(dl, dl + r);
52      r --;
53      int l = 0;
54      while (l < r) {
55          if (dl[l] + dl[r] > k) r --;
56          else sum += r - l, l ++;
57      }
58      return sum;
59  }
60
61  ll res = 0;
62  void dfs(int u) {
63      res += cal(u, 0);
64      vis[u] = 1;
65      for (int i = head[u]; i; i = Next[i]) {
66          int v = ver[i];
67          if (vis[v]) continue;
68          res -= cal(v, edge[i]);
69          has = sz[v];
70          mxsz = 0x3f3f3f3f;
71          getrt(v, 0);
72          dfs(rt);
73      }
74  }
75
76  int main(int argc, char* argv[]) {
77      while (scanf("%d%d", &n, &k) != EOF && (n || k)) {
78          tot = 0; memset(head, 0, sizeof head);
79          memset(vis, 0, sizeof vis);
80          res = 0;
81          for (int i = 1, u, v, w; i < n; ++i) {
82              scanf("%d%d%d", &u, &v, &w);
83              addEdge(u, v, w);
84              addEdge(v, u, w);
85          }
```

```
86          mxsz = 0x3f3f3f3f;
87          has = n;
88          getrt(1, 0);
89          dfs(rt);
90          printf("%lld\n", res);
91      }
92      return 0;
93  }
```

3.8 Others

3.8.1 BITinNM

```
1   struct Fenwick_Tree
2   {
3       #define type int
4       type bit[MAX][MAX];
5       int n,m;
6       void init(int _n,int _m){n=_n;m=_m;mem(bit,0);}
7       int lowbit(int x){return x&(-x);}
8       void update(int x,int y,type v)
9       {
10          int i,j;
11          for(i=x;i<=n;i+=lowbit(i))
12          {
13              for(j=y;j<=m;j+=lowbit(j))
14              {
15                  bit[i][j]+=v;
16              }
17          }
18      }
19      type get(int x,int y)
20      {
21          type i,j,res=0;
22          for(i=x;i>0;i-=lowbit(i))
23          {
24              for(j=y;j>0;j-=lowbit(j))
25              {
26                  res+=bit[i][j];
27              }
28          }
29          return res;
30      }
31      type query(int x1,int x2,int y1,int y2)
32      {
33          x1--;
34          y1--;
35          return get(x2,y2)-get(x1,y2)-get(x2,y1)+get(x1,y1);
36      }
37      #undef type
38  }tr;
```

3.8.2 静态区间 k 大划分树

```
1  // const int maxn = 100010;
2  int tree[20][maxn];
3  // 读入sorted并排序，赋值给tree的第0层
4  int sorted[maxn];
```

```
int toleft[20][maxn];
// 保存左子树的和
// ll sum[20][maxn];

// 1, n, 0
void build(int l, int r, int dep) {
    if (l == r) return;
    // sum[dep][0] = 0;
    toleft[dep][0] = 0;
    int mid = l + r >> 1;
    int same = mid - l + 1;
    for (int i = l; i <= r; ++i) {
        if (tree[dep][i] < sorted[mid]) same--;
    }
    int lpos = l, rpos = mid + 1;
    for (int i = l; i <= r; ++i) {
        // sum[dep][i] = sum[dep][i - 1];
        if (tree[dep][i] < sorted[mid]) {
            // sum[dep][i] += tree[dep][i];
            tree[dep + 1][lpos++] = tree[dep][i];
        }
        else if (tree[dep][i] == sorted[mid] && same > 0) {
            // sum[dep][i] += tree[dep][i];
            tree[dep + 1][lpos++] = tree[dep][i];
            same --;
        } else tree[dep + 1][rpos ++] = tree[dep][i];
        toleft[dep][i] = toleft[dep][l - 1] + lpos - l;
    }
    build(l, mid, dep + 1);
    build(mid + 1, r, dep + 1);
}

//(1~k-1)的数的和, 注意每次查询前初始化
// ll ress = 0;

// L = 1, R = n,  dep = 0, l,r是查询区间
int query(int L, int R, int l, int r, int dep, int k) {
    if (l == r) return tree[dep][l];
    int mid = (L + R) >> 1;
    int cnt = toleft[dep][r] - toleft[dep][l - 1];
    if (cnt >= k) {
        int newl = L + toleft[dep][l - 1] - toleft[dep][L - 1];
        int newr = newl + cnt - 1;
        return query(L, mid, newl, newr, dep + 1, k);
    } else {
        int newr = r + toleft[dep][R] - toleft[dep][r];
        int newl = newr - (r - l - cnt);
        // ress += sum[dep][r] - sum[dep][l - 1];
        return query(mid + 1, R, newl, newr, dep + 1, k - cnt);
    }
}


scan(n), scan(m);
for (int i = 1; i <= n; ++i) {
    scan(sorted[i]);
    tree[0][i] = sorted[i];
}
sort(sorted + 1, sorted + 1 + n);
```

```
64  build(1, n, 0);
65  int l, r, k;
66  while (m--) {
67      scan(l), scan(r), scan(k);
68      printf("%d\n", query(1, n, l, r, 0, k));
69  }
```

4 Geometry

4.1 Class

4.1.1 geo

```
#define mp make_pair
#define fi first
#define se second
#define pb push_back
typedef double db;
const db eps=1e-6;
const db pi=acos(-1);
int sign(db k){
    if (k>eps) return 1; else if (k<-eps) return -1; return 0;
}
int cmp(db k1,db k2){return sign(k1-k2);}
int inmid(db k1,db k2,db k3){return sign(k1-k3)*sign(k2-k3)<=0;}// k3 在 [k1,k2] 内
struct point{
    db x,y;
    point operator + (const point &k1) const{return (point){k1.x+x,k1.y+y};}
    point operator - (const point &k1) const{return (point){x-k1.x,y-k1.y};}
    point operator * (db k1) const{return (point){x*k1,y*k1};}
    point operator / (db k1) const{return (point){x/k1,y/k1};}
    int operator == (const point &k1) const{return cmp(x,k1.x)==0&&cmp(y,k1.y)==0;}
    // 逆时针旋转
    point turn(db k1){return (point){x*cos(k1)-y*sin(k1),x*sin(k1)+y*cos(k1)};}
    point turn90(){return (point){-y,x};}
    bool operator < (const point k1) const{
        int a=cmp(x,k1.x);
        if (a==-1) return 1; else if (a==1) return 0; else return cmp(y,k1.y)==-1;
    }
    db abs(){return sqrt(x*x+y*y);}
    db abs2(){return x*x+y*y;}
    db dis(point k1){return ((*this)-k1).abs();}
    point unit(){db w=abs(); return (point){x/w,y/w};}
    void scan(){double k1,k2; scanf("%lf%lf",&k1,&k2); x=k1; y=k2;}
    void print(){printf("%.11lf %.11lf\n",x,y);}
    db getw(){return atan2(y,x);}
    point getdel(){if (sign(x)==-1||(sign(x)==0&&sign(y)==-1)) return (*this)*(-1); else return (*
    this);}
    int getP() const{return sign(y)==1||(sign(y)==0&&sign(x)==-1);}
};
int inmid(point k1,point k2,point k3){return inmid(k1.x,k2.x,k3.x)&&inmid(k1.y,k2.y,k3.y);}
db cross(point k1,point k2){return k1.x*k2.y-k1.y*k2.x;}
db dot(point k1,point k2){return k1.x*k2.x+k1.y*k2.y;}
db rad(point k1,point k2){return atan2(cross(k1,k2),dot(k1,k2));}
// -pi -> pi
int compareangle (point k1,point k2){
    return k1.getP()<k2.getP()||(k1.getP()==k2.getP()&&sign(cross(k1,k2))>0);
}
point proj(point k1,point k2,point q){ // q 到直线 k1,k2 的投影
    point k=k2-k1; return k1+k*(dot(q-k1,k)/k.abs2());
}
point reflect(point k1,point k2,point q){return proj(k1,k2,q)*2-q;}
int clockwise(point k1,point k2,point k3){// k1 k2 k3 逆时针 1 顺时针 -1 否则 0
    return sign(cross(k2-k1,k3-k1));
}
int checkLL(point k1,point k2,point k3,point k4){// 求直线 (L) 线段 (S)k1,k2 和 k3,k4 的交点
```

```
53      return cmp(cross(k3-k1,k4-k1),cross(k3-k2,k4-k2))!=0;
54  }
55  point getLL(point k1,point k2,point k3,point k4){
56      db w1=cross(k1-k3,k4-k3),w2=cross(k4-k3,k2-k3); return (k1*w2+k2*w1)/(w1+w2);
57  }
58  int intersect(db l1,db r1,db l2,db r2){
59      if (l1>r1) swap(l1,r1); if (l2>r2) swap(l2,r2); return cmp(r1,l2)!=-1&&cmp(r2,l1)!=-1;
60  }
61  int checkSS(point k1,point k2,point k3,point k4){
62      return intersect(k1.x,k2.x,k3.x,k4.x)&&intersect(k1.y,k2.y,k3.y,k4.y)&&
63      sign(cross(k3-k1,k4-k1))*sign(cross(k3-k2,k4-k2))<=0&&
64      sign(cross(k1-k3,k2-k3))*sign(cross(k1-k4,k2-k4))<=0;
65  }
66  db disSP(point k1,point k2,point q){
67      point k3=proj(k1,k2,q);
68      if (inmid(k1,k2,k3)) return q.dis(k3); else return min(q.dis(k1),q.dis(k2));
69  }
70  db disSS(point k1,point k2,point k3,point k4){
71      if (checkSS(k1,k2,k3,k4)) return 0;
72      else return min(min(disSP(k1,k2,k3),disSP(k1,k2,k4)),min(disSP(k3,k4,k1),disSP(k3,k4,k2)));
73  }
74  int onS(point k1,point k2,point q){return inmid(k1,k2,q)&&sign(cross(k1-q,k2-k1))==0;}
75  struct circle{
76      point o; db r;
77      void scan(){o.scan(); scanf("%lf",&r);}
78      int inside(point k){return cmp(r,o.dis(k));}
79  };
80  struct line{
81      // p[0]->p[1]
82      point p[2];
83      line(point k1,point k2){p[0]=k1; p[1]=k2;}
84      point& operator [] (int k){return p[k];}
85      int include(point k){return sign(cross(p[1]-p[0],k-p[0]))>0;}
86      point dir(){return p[1]-p[0];}
87      line push(){ // 向外 ( 左手边 ) 平移 eps
88          const db eps = 1e-6;
89          point delta=(p[1]-p[0]).turn90().unit()*eps;
90          return {p[0]-delta,p[1]-delta};
91      }
92  };
93  point getLL(line k1,line k2){return getLL(k1[0],k1[1],k2[0],k2[1]);}
94  int parallel(line k1,line k2){return sign(cross(k1.dir(),k2.dir()))==0;}
95  int sameDir(line k1,line k2){return parallel(k1,k2)&&sign(dot(k1.dir(),k2.dir()))==1;}
96  int operator < (line k1,line k2){
97      if (sameDir(k1,k2)) return k2.include(k1[0]);
98      return compareangle(k1.dir(),k2.dir());
99  }
100 int checkpos(line k1,line k2,line k3){return k3.include(getLL(k1,k2));}
101 vector<line> getHL(vector<line> &L){ // 求半平面交 , 半平面是逆时针方向 , 输出按照逆时针
102     sort(L.begin(),L.end()); deque<line> q;
103     for (int i=0;i<(int)L.size();i++){
104         if (i&&sameDir(L[i],L[i-1])) continue;
105         while (q.size()>1&&!checkpos(q[q.size()-2],q[q.size()-1],L[i])) q.pop_back();
106         while (q.size()>1&&!checkpos(q[1],q[0],L[i])) q.pop_front();
107         q.push_back(L[i]);
108     }
109     while (q.size()>2&&!checkpos(q[q.size()-2],q[q.size()-1],q[0])) q.pop_back();
110     while (q.size()>2&&!checkpos(q[1],q[0],q[q.size()-1])) q.pop_front();
111     vector<line>ans; for (int i=0;i<q.size();i++) ans.push_back(q[i]);
```

```
112     return ans;
113 }
114 db closepoint(vector<point>&A,int l,int r){ // 最近点对 , 先要按照 x 坐标排序
115     if (r-l<=5){
116         db ans=1e20;
117         for (int i=l;i<=r;i++) for (int j=i+1;j<=r;j++) ans=min(ans,A[i].dis(A[j]));
118         return ans;
119     }
120     int mid=l+r>>1; db ans=min(closepoint(A,l,mid),closepoint(A,mid+1,r));
121     vector<point>B; for (int i=l;i<=r;i++) if (abs(A[i].x-A[mid].x)<=ans) B.push_back(A[i]);
122     sort(B.begin(),B.end(),[](point k1,point k2){return k1.y<k2.y;});
123     for (int i=0;i<B.size();i++) for (int j=i+1;j<B.size()&&B[j].y-B[i].y<ans;j++) ans=min(ans,B[i
        ].dis(B[j]));
124     return ans;
125 }
126 int checkposCC(circle k1,circle k2){// 返回两个圆的公切线数量
127     if (cmp(k1.r,k2.r)==-1) swap(k1,k2);
128     db dis=k1.o.dis(k2.o);  int w1=cmp(dis,k1.r+k2.r),w2=cmp(dis,k1.r-k2.r);
129     if (w1>0) return 4; else if (w1==0) return 3; else if (w2>0) return 2;
130     else if (w2==0) return 1; else return 0;
131 }
132 vector<point> getCL(circle k1,point k2,point k3){ // 沿着 k2->k3 方向给出 , 相切给出两个
133     point k=proj(k2,k3,k1.o); db d=k1.r*k1.r-(k-k1.o).abs2();
134     if (sign(d)==-1) return {};
135     point del=(k3-k2).unit()*sqrt(max((db)0.0,d)); return {k-del,k+del};
136 }
137 vector<point> getCC(circle k1,circle k2){// 沿圆 k1 逆时针给出 , 相切给出两个
138     int pd=checkposCC(k1,k2); if (pd==0||pd==4) return {};
139     db a=(k2.o-k1.o).abs2(),cosA=(k1.r*k1.r+a-k2.r*k2.r)/(2*k1.r*sqrt(max(a,(db)0.0)));
140     db b=k1.r*cosA,c=sqrt(max((db)0.0,k1.r*k1.r-b*b));
141     point k=(k2.o-k1.o).unit(),m=k1.o+k*b,del=k.turn90()*c;
142     return {m-del,m+del};
143 }
144 vector<point> TangentCP(circle k1,point k2){// 沿圆 k1 逆时针给出
145     db a=(k2-k1.o).abs(),b=k1.r*k1.r/a,c=sqrt(max((db)0.0,k1.r*k1.r-b*b));
146     point k=(k2-k1.o).unit(),m=k1.o+k*b,del=k.turn90()*c;
147     return {m-del,m+del};
148 }
149 vector<line> TangentoutCC(circle k1,circle k2){
150     int pd=checkposCC(k1,k2); if (pd==0) return {};
151     if (pd==1){point k=getCC(k1,k2)[0]; return {(line){k,k}};}
152     if (cmp(k1.r,k2.r)==0){
153         point del=(k2.o-k1.o).unit().turn90().getdel();
154         return {(line){k1.o-del*k1.r,k2.o-del*k2.r},(line){k1.o+del*k1.r,k2.o+del*k2.r}};
155     } else {
156         point p=(k2.o*k1.r-k1.o*k2.r)/(k1.r-k2.r);
157         vector<point>A=TangentCP(k1,p),B=TangentCP(k2,p);
158         vector<line>ans; for (int i=0;i<A.size();i++) ans.push_back((line){A[i],B[i]});
159         return ans;
160     }
161 }
162 vector<line> TangentinCC(circle k1,circle k2){
163     int pd=checkposCC(k1,k2); if (pd<=2) return {};
164     if (pd==3){point k=getCC(k1,k2)[0]; return {(line){k,k}};}
165     point p=(k2.o*k1.r+k1.o*k2.r)/(k1.r+k2.r);
166     vector<point>A=TangentCP(k1,p),B=TangentCP(k2,p);
167     vector<line>ans; for (int i=0;i<A.size();i++) ans.push_back((line){A[i],B[i]});
168     return ans;
169 }
```

```
vector<line> TangentCC(circle k1,circle k2){
    int flag=0; if (k1.r<k2.r) swap(k1,k2),flag=1;
    vector<line>A=TangentoutCC(k1,k2),B=TangentinCC(k1,k2);
    for (line k:B) A.push_back(k);
    if (flag) for (line &k:A) swap(k[0],k[1]);
    return A;
}
db getarea(circle k1,point k2,point k3){
    // 圆 k1 与三角形 k2 k3 k1.o 的有向面积交
    point k=k1.o; k1.o=k1.o-k; k2=k2-k; k3=k3-k;
    int pd1=k1.inside(k2),pd2=k1.inside(k3);
    vector<point>A=getCL(k1,k2,k3);
    if (pd1>=0){
        if (pd2>=0) return cross(k2,k3)/2;
        return k1.r*k1.r*rad(A[1],k3)/2+cross(k2,A[1])/2;
    } else if (pd2>=0){
        return k1.r*k1.r*rad(k2,A[0])/2+cross(A[0],k3)/2;
    }else {
        int pd=cmp(k1.r,disSP(k2,k3,k1.o));
        if (pd<=0) return k1.r*k1.r*rad(k2,k3)/2;
        return cross(A[0],A[1])/2+k1.r*k1.r*(rad(k2,A[0])+rad(A[1],k3))/2;
    }
}
circle getcircle(point k1,point k2,point k3){
    db a1=k2.x-k1.x,b1=k2.y-k1.y,c1=(a1*a1+b1*b1)/2;
    db a2=k3.x-k1.x,b2=k3.y-k1.y,c2=(a2*a2+b2*b2)/2;
    db d=a1*b2-a2*b1;
    point o=(point){k1.x+(c1*b2-c2*b1)/d,k1.y+(a1*c2-a2*c1)/d};
    return (circle){o,k1.dis(o)};
}
circle getScircle(vector<point> A){
    random_shuffle(A.begin(),A.end());
    circle ans=(circle){A[0],0};
    for (int i=1;i<A.size();i++)
        if (ans.inside(A[i])==-1){
            ans=(circle){A[i],0};
            for (int j=0;j<i;j++)
                if (ans.inside(A[j])==-1){
                    ans.o=(A[i]+A[j])/2; ans.r=ans.o.dis(A[i]);
                    for (int k=0;k<j;k++)
                        if (ans.inside(A[k])==-1)
                            ans=getcircle(A[i],A[j],A[k]);
                }
        }
    return ans;
}
db area(vector<point> A){ // 多边形用 vector<point> 表示 , 逆时针
    db ans=0;
    for (int i=0;i<A.size();i++) ans+=cross(A[i],A[(i+1)%A.size()]);
    return ans/2;
}
int checkconvex(vector<point>A){
    int n=A.size(); A.push_back(A[0]); A.push_back(A[1]);
    for (int i=0;i<n;i++) if (sign(cross(A[i+1]-A[i],A[i+2]-A[i]))==-1) return 0;
    return 1;
}
int contain(vector<point>A,point q){ // 2 内部 1 边界 0 外部
    int pd=0; A.push_back(A[0]);
    for (int i=1;i<A.size();i++){
```

```
229         point u=A[i-1],v=A[i];
230         if (onS(u,v,q)) return 1; if (cmp(u.y,v.y)>0) swap(u,v);
231         if (cmp(u.y,q.y)>=0||cmp(v.y,q.y)<0) continue;
232         if (sign(cross(u-v,q-v))<0) pd^=1;
233     }
234     return pd<<1;
235 }
236 vector<point> ConvexHull(vector<point>A,int flag=1){ // flag=0 不严格 flag=1 严格
237     int n=A.size(); vector<point>ans(n*2);
238     sort(A.begin(),A.end()); int now=-1;
239     for (int i=0;i<A.size();i++){
240         while (now>0&&sign(cross(ans[now]-ans[now-1],A[i]-ans[now-1]))<flag) now--;
241         ans[++now]=A[i];
242     } int pre=now;
243     for (int i=n-2;i>=0;i--){
244         while (now>pre&&sign(cross(ans[now]-ans[now-1],A[i]-ans[now-1]))<flag) now--;
245         ans[++now]=A[i];
246     } ans.resize(now); return ans;
247 }
248 db convexDiameter(vector<point>A){
249     int now=0,n=A.size(); db ans=0;
250     for (int i=0;i<A.size();i++){
251         now=max(now,i);
252         while (1){
253             db k1=A[i].dis(A[now%n]),k2=A[i].dis(A[(now+1)%n]);
254             ans=max(ans,max(k1,k2)); if (k2>k1) now++; else break;
255         }
256     }
257     return ans;
258 }
259 vector<point> convexcut(vector<point>A,point k1,point k2){
260     // 保留 k1,k2,p 逆时针的所有点
261     int n=A.size(); A.push_back(A[0]); vector<point>ans;
262     for (int i=0;i<n;i++){
263         int w1=clockwise(k1,k2,A[i]),w2=clockwise(k1,k2,A[i+1]);
264         if (w1>=0) ans.push_back(A[i]);
265         if (w1*w2<0) ans.push_back(getLL(k1,k2,A[i],A[i+1]));
266     }
267     return ans;
268 }
269 int checkPoS(vector<point>A,point k1,point k2){
270     // 多边形 A 和直线 ( 线段 )k1->k2 严格相交 , 注释部分为线段
271     struct ins{
272         point m,u,v;
273         int operator < (const ins& k) const {return m<k.m;}
274     }; vector<ins>B;
275     //if (contain(A,k1)==2||contain(A,k2)==2) return 1;
276     vector<point>poly=A; A.push_back(A[0]);
277     for (int i=1;i<A.size();i++) if (checkLL(A[i-1],A[i],k1,k2)){
278         point m=getLL(A[i-1],A[i],k1,k2);
279         if (inmid(A[i-1],A[i],m)/*&&inmid(k1,k2,m)*/) B.push_back((ins){m,A[i-1],A[i]});
280     }
281     if (B.size()==0) return 0; sort(B.begin(),B.end());
282     int now=1; while (now<B.size()&&B[now].m==B[0].m) now++;
283     if (now==B.size()) return 0;
284     int flag=contain(poly,(B[0].m+B[now].m)/2);
285     if (flag==2) return 1;
286     point d=B[now].m-B[0].m;
287     for (int i=now;i<B.size();i++){
```

```
        if (!(B[i].m==B[i-1].m)&&flag==2) return 1;
        int tag=sign(cross(B[i].v-B[i].u,B[i].m+d-B[i].u));
        if (B[i].m==B[i].u||B[i].m==B[i].v) flag+=tag; else flag+=tag*2;
    }
    //return 0;
    return flag==2;
}
int checkinp(point r,point l,point m){
    if (compareangle(l,r)){return compareangle(l,m)&&compareangle(m,r);}
    return compareangle(l,m)||compareangle(m,r);
}
int checkPosFast(vector<point>A,point k1,point k2){ // 快速检查线段是否和多边形严格相交
    if (contain(A,k1)==2||contain(A,k2)==2) return 1; if (k1==k2) return 0;
    A.push_back(A[0]); A.push_back(A[1]);
    for (int i=1;i+1<A.size();i++)
        if (checkLL(A[i-1],A[i],k1,k2)){
            point now=getLL(A[i-1],A[i],k1,k2);
            if (inmid(A[i-1],A[i],now)==0||inmid(k1,k2,now)==0) continue;
            if (now==A[i]){
                if (A[i]==k2) continue;
                point pre=A[i-1],ne=A[i+1];
                if (checkinp(pre-now,ne-now,k2-now)) return 1;
            } else if (now==k1){
                if (k1==A[i-1]||k1==A[i]) continue;
                if (checkinp(A[i-1]-k1,A[i]-k1,k2-k1)) return 1;
            } else if (now==k2||now==A[i-1]) continue;
            else return 1;
        }
    return 0;
}
// 拆分凸包成上下凸壳 凸包尽量都随机旋转一个角度来避免出现相同横坐标
// 尽量特判只有一个点的情况 凸包逆时针
void getUDP(vector<point>A,vector<point>&U,vector<point>&D){
    db l=1e100,r=-1e100;
    for (int i=0;i<A.size();i++) l=min(l,A[i].x),r=max(r,A[i].x);
    int wherel,wherer;
    for (int i=0;i<A.size();i++) if (cmp(A[i].x,l)==0) wherel=i;
    for (int i=A.size();i;i--) if (cmp(A[i-1].x,r)==0) wherer=i-1;
    U.clear(); D.clear(); int now=wherel;
    while (1){D.push_back(A[now]); if (now==wherer) break; now++; if (now>=A.size()) now=0;}
    now=wherel;
    while (1){U.push_back(A[now]); if (now==wherer) break; now--; if (now<0) now=A.size()-1;}
}
// 需要保证凸包点数大于等于 3,2 内部 ,1 边界 ,0 外部
int containCoP(const vector<point>&U,const vector<point>&D,point k){
    db lx=U[0].x,rx=U[U.size()-1].x;
    if (k==U[0]||k==U[U.size()-1]) return 1;
    if (cmp(k.x,lx)==-1||cmp(k.x,rx)==1) return 0;
    int where1=lower_bound(U.begin(),U.end(),(point){k.x,-1e100})-U.begin();
    int where2=lower_bound(D.begin(),D.end(),(point){k.x,-1e100})-D.begin();
    int w1=clockwise(U[where1-1],U[where1],k),w2=clockwise(D[where2-1],D[where2],k);
    if (w1==1||w2==-1) return 0; else if (w1==0||w2==0) return 1; return 2;
}
// d 是方向 , 输出上方切点和下方切点
pair<point,point> getTangentCow(const vector<point> &U,const vector<point> &D,point d){
    if (sign(d.x)<0||(sign(d.x)==0&&sign(d.y)<0)) d=d*(-1);
    point whereU,whereD;
    if (sign(d.x)==0) return mp(U[0],U[U.size()-1]);
    int l=0,r=U.size()-1,ans=0;
```

```
347     while (l<r){int mid=l+r>>1; if (sign(cross(U[mid+1]-U[mid],d))<=0) l=mid+1,ans=mid+1; else r=
        mid;}
348     whereU=U[ans]; l=0,r=D.size()-1,ans=0;
349     while (l<r){int mid=l+r>>1; if (sign(cross(D[mid+1]-D[mid],d))>=0) l=mid+1,ans=mid+1; else r=
        mid;}
350     whereD=D[ans]; return mp(whereU,whereD);
351 }
352 // 先检查 contain, 逆时针给出
353 pair<point,point> getTangentCoP(const vector<point>&U,const vector<point>&D,point k){
354     db lx=U[0].x,rx=U[U.size()-1].x;
355     if (k.x<lx){
356         int l=0,r=U.size()-1,ans=U.size()-1;
357         while (l<r){int mid=l+r>>1; if (clockwise(k,U[mid],U[mid+1])==1) l=mid+1; else ans=mid,r=
        mid;}
358         point w1=U[ans]; l=0,r=D.size()-1,ans=D.size()-1;
359         while (l<r){int mid=l+r>>1; if (clockwise(k,D[mid],D[mid+1])==-1) l=mid+1; else ans=mid,r=
        mid;}
360         point w2=D[ans]; return mp(w1,w2);
361     } else if (k.x>rx){
362         int l=1,r=U.size(),ans=0;
363         while (l<r){int mid=l+r>>1; if (clockwise(k,U[mid],U[mid-1])==-1) r=mid; else ans=mid,l=mid
        +1;}
364         point w1=U[ans]; l=1,r=D.size(),ans=0;
365         while (l<r){int mid=l+r>>1; if (clockwise(k,D[mid],D[mid-1])==1) r=mid; else ans=mid,l=mid
        +1;}
366         point w2=D[ans]; return mp(w2,w1);
367     } else {
368         int where1=lower_bound(U.begin(),U.end(),(point){k.x,-1e100})-U.begin();
369         int where2=lower_bound(D.begin(),D.end(),(point){k.x,-1e100})-D.begin();
370         if ((k.x==lx&&k.y>U[0].y)||(where1&&clockwise(U[where1-1],U[where1],k)==1)){
371             int l=1,r=where1+1,ans=0;
372             while (l<r){int mid=l+r>>1; if (clockwise(k,U[mid],U[mid-1])==1) ans=mid,l=mid+1; else
        r=mid;}
373             point w1=U[ans]; l=where1,r=U.size()-1,ans=U.size()-1;
374             while (l<r){int mid=l+r>>1; if (clockwise(k,U[mid],U[mid+1])==1) l=mid+1; else ans=mid,
        r=mid;}
375             point w2=U[ans]; return mp(w2,w1);
376         } else {
377             int l=1,r=where2+1,ans=0;
378             while (l<r){int mid=l+r>>1; if (clockwise(k,D[mid],D[mid-1])==-1) ans=mid,l=mid+1; else
         r=mid;}
379             point w1=D[ans]; l=where2,r=D.size()-1,ans=D.size()-1;
380             while (l<r){int mid=l+r>>1; if (clockwise(k,D[mid],D[mid+1])==-1) l=mid+1; else ans=mid
        ,r=mid;}
381             point w2=D[ans]; return mp(w1,w2);
382         }
383     }
384 }
385 struct P3{
386     db x,y,z;
387     P3 operator + (P3 k1){return (P3){x+k1.x,y+k1.y,z+k1.z};}
388     P3 operator - (P3 k1){return (P3){x-k1.x,y-k1.y,z-k1.z};}
389     P3 operator * (db k1){return (P3){x*k1,y*k1,z*k1};}
390     P3 operator / (db k1){return (P3){x/k1,y/k1,z/k1};}
391     db abs2(){return x*x+y*y+z*z;}
392     db abs(){return sqrt(x*x+y*y+z*z);}
393     P3 unit(){return (*this)/abs();}
394     int operator < (const P3 k1) const{
395         if (cmp(x,k1.x)!=0) return x<k1.x;
```

```
396         if (cmp(y,k1.y)!=0) return y<k1.y;
397         return cmp(z,k1.z)==-1;
398     }
399     int operator == (const P3 k1){
400         return cmp(x,k1.x)==0&&cmp(y,k1.y)==0&&cmp(z,k1.z)==0;
401     }
402     void scan(){
403         double k1,k2,k3; scanf("%lf%lf%lf",&k1,&k2,&k3);
404         x=k1; y=k2; z=k3;
405     }
406 };
407 P3 cross(P3 k1,P3 k2){return (P3){k1.y*k2.z-k1.z*k2.y,k1.z*k2.x-k1.x*k2.z,k1.x*k2.y-k1.y*k2.x};}
408 db dot(P3 k1,P3 k2){return k1.x*k2.x+k1.y*k2.y+k1.z*k2.z;}
409 //p=(3,4,5),l=(13,19,21),theta=85 ans=(2.83,4.62,1.77)
410 P3 turn3D(db k1,P3 l,P3 p){
411     l=l.unit(); P3 ans; db c=cos(k1),s=sin(k1);
412     ans.x=p.x*(l.x*l.x*(1-c)+c)+p.y*(l.x*l.y*(1-c)-l.z*s)+p.z*(l.x*l.z*(1-c)+l.y*s);
413     ans.y=p.x*(l.x*l.y*(1-c)+l.z*s)+p.y*(l.y*l.y*(1-c)+c)+p.z*(l.y*l.z*(1-c)-l.x*s);
414     ans.z=p.x*(l.x*l.z*(1-c)-l.y*s)+p.y*(l.y*l.z*(1-c)+l.x*s)+p.z*(l.x*l.x*(1-c)+c);
415     return ans;
416 }
417 typedef vector<P3> VP;
418 typedef vector<VP> VVP;
419 db Acos(db x){return acos(max(-(db)1,min(x,(db)1)));}
420 // 球面距离 , 圆心原点 , 半径 1
421 db Odist(P3 a,P3 b){db r=Acos(dot(a,b)); return r;}
422 db r; P3 rnd;
423 vector<db> solve(db a,db b,db c){
424     db r=sqrt(a*a+b*b),th=atan2(b,a);
425     if (cmp(c,-r)==-1) return {0};
426     else if (cmp(r,c)<=0) return {1};
427     else {
428         db tr=pi-Acos(c/r); return {th+pi-tr,th+pi+tr};
429     }
430 }
431 vector<db> jiao(P3 a,P3 b){
432     // dot(rd+x*cos(t)+y*sin(t),b) >= cos(r)
433     if (cmp(Odist(a,b),2*r)>0) return {0};
434     P3 rd=a*cos(r),z=a.unit(),y=cross(z,rnd).unit(),x=cross(y,z).unit();
435     vector<db> ret = solve(-(dot(x,b)*sin(r)),-(dot(y,b)*sin(r)),-(cos(r)-dot(rd,b)));
436     return ret;
437 }
438 db norm(db x,db l=0,db r=2*pi){ // change x into [l,r)
439     while (cmp(x,l)==-1) x+=(r-l); while (cmp(x,r)>=0) x-=(r-l);
440     return x;
441 }
442 db disLP(P3 k1,P3 k2,P3 q){
443     return (cross(k2-k1,q-k1)).abs()/(k2-k1).abs();
444 }
445 db disLL(P3 k1,P3 k2,P3 k3,P3 k4){
446     P3 dir=cross(k2-k1,k4-k3); if (sign(dir.abs())==0) return disLP(k1,k2,k3);
447     return fabs(dot(dir.unit(),k1-k2));
448 }
449 VP getFL(P3 p,P3 dir,P3 k1,P3 k2){
450     db a=dot(k2-p,dir),b=dot(k1-p,dir),d=a-b;
451     if (sign(fabs(d))==0) return {};
452     return {(k1*a-k2*b)/d};
453 }
454 VP getFF(P3 p1,P3 dir1,P3 p2,P3 dir2){// 返回一条线
```

```
455     P3 e=cross(dir1,dir2),v=cross(dir1,e);
456     db d=dot(dir2,v); if (sign(abs(d))==0) return {};
457     P3 q=p1+v*dot(dir2,p2-p1)/d; return {q,q+e};
458 }
459 // 3D Covex Hull Template
460 db getV(P3 k1,P3 k2,P3 k3,P3 k4){ // get the Volume
461     return dot(cross(k2-k1,k3-k1),k4-k1);
462 }
463 db rand_db(){return 1.0*rand()/RAND_MAX;}
464 VP convexHull2D(VP A,P3 dir){
465     P3 x={(db)rand(),(db)rand(),(db)rand()}; x=x.unit();
466     x=cross(x,dir).unit(); P3 y=cross(x,dir).unit();
467     P3 vec=dir.unit()*dot(A[0],dir);
468     vector<point>B;
469     for (int i=0;i<A.size();i++) B.push_back((point){dot(A[i],x),dot(A[i],y)});
470     B=ConvexHull(B); A.clear();
471     for (int i=0;i<B.size();i++) A.push_back(x*B[i].x+y*B[i].y+vec);
472     return A;
473 }
474 namespace CH3{
475     VVP ret; set<pair<int,int> >e;
476     int n; VP p,q;
477     void wrap(int a,int b){
478         if (e.find({a,b})==e.end()){
479             int c=-1;
480             for (int i=0;i<n;i++) if (i!=a&&i!=b){
481                 if (c==-1||sign(getV(q[c],q[a],q[b],q[i]))>0) c=i;
482             }
483             if (c!=-1){
484                 ret.push_back({p[a],p[b],p[c]});
485                 e.insert({a,b}); e.insert({b,c}); e.insert({c,a});
486                 wrap(c,b); wrap(a,c);
487             }
488         }
489     }
490     VVP ConvexHull3D(VP _p){
491         p=q=_p; n=p.size();
492         ret.clear(); e.clear();
493         for (auto &i:q) i=i+(P3){rand_db()*1e-4,rand_db()*1e-4,rand_db()*1e-4};
494         for (int i=1;i<n;i++) if (q[i].x<q[0].x) swap(p[0],p[i]),swap(q[0],q[i]);
495         for (int i=2;i<n;i++) if ((q[i].x-q[0].x)*(q[1].y-q[0].y)>(q[i].y-q[0].y)*(q[1].x-q[0].x))
    swap(q[1],q[i]),swap(p[1],p[i]);
496         wrap(0,1);
497         return ret;
498     }
499 }
500 VVP reduceCH(VVP A){
501     VVP ret; map<P3,VP> M;
502     for (VP nowF:A){
503         P3 dir=cross(nowF[1]-nowF[0],nowF[2]-nowF[0]).unit();
504         for (P3 k1:nowF) M[dir].pb(k1);
505     }
506     for (pair<P3,VP> nowF:M) ret.pb(convexHull2D(nowF.se,nowF.fi));
507     return ret;
508 }
509 //  把一个面变成 ( 点 , 法向量 ) 的形式
510 pair<P3,P3> getF(VP F){
511     return mp(F[0],cross(F[1]-F[0],F[2]-F[0]).unit());
512 }
```

```
// 3D Cut 保留 dot(dir,x-p)>=0 的部分
VVP ConvexCut3D(VVP A,P3 p,P3 dir){
    VVP ret; VP sec;
    for (VP nowF: A){
        int n=nowF.size(); VP ans; int dif=0;
        for (int i=0;i<n;i++){
            int d1=sign(dot(dir,nowF[i]-p));
            int d2=sign(dot(dir,nowF[(i+1)%n]-p));
            if (d1>=0) ans.pb(nowF[i]);
            if (d1*d2<0){
                P3 q=getFL(p,dir,nowF[i],nowF[(i+1)%n])[0];
                ans.push_back(q); sec.push_back(q);
            }
            if (d1==0) sec.push_back(nowF[i]); else dif=1;
            dif|=(sign(dot(dir,cross(nowF[(i+1)%n]-nowF[i],nowF[(i+1)%n]-nowF[i])))==-1);
        }
        if (ans.size()>0&&dif) ret.push_back(ans);
    }
    if (sec.size()>0) ret.push_back(convexHull2D(sec,dir));
    return ret;
}
db vol(VVP A){
    if (A.size()==0) return 0; P3 p=A[0][0]; db ans=0;
    for (VP nowF:A)
        for (int i=2;i<nowF.size();i++)
            ans+=abs(getV(p,nowF[0],nowF[i-1],nowF[i]));
    return ans/6;
}
VVP init(db INF) {
    VVP pss(6,VP(4));
    pss[0][0] = pss[1][0] = pss[2][0] = {-INF, -INF, -INF};
    pss[0][3] = pss[1][1] = pss[5][2] = {-INF, -INF, INF};
    pss[0][1] = pss[2][3] = pss[4][2] = {-INF, INF, -INF};
    pss[0][2] = pss[5][3] = pss[4][1] = {-INF, INF, INF};
    pss[1][3] = pss[2][1] = pss[3][2] = {INF, -INF, -INF};
    pss[1][2] = pss[5][1] = pss[3][3] = {INF, -INF, INF};
    pss[2][2] = pss[4][3] = pss[3][1] = {INF, INF, -INF};
    pss[5][0] = pss[4][0] = pss[3][0] = {INF, INF, INF};
    return pss;
}
```

4.1.2 3D 凸包

```
#include<stdio.h>
#include<algorithm>
#include<string.h>
#include<math.h>
#include<stdlib.h>
using namespace std;
const int MAXN=1050;
const double eps=1e-8;

struct Point
{
    double x,y,z;
    Point() {}
    Point(double xx,double yy,double zz):x(xx),y(yy),z(zz) {}
    //两向量之差
```

```
16      Point operator -(const Point p1)
17      {
18          return Point(x-p1.x,y-p1.y,z-p1.z);
19      }
20      //两向量之和
21      Point operator +(const Point p1)
22      {
23          return Point(x+p1.x,y+p1.y,z+p1.z);
24      }
25      //叉乘
26      Point operator *(const Point p)
27      {
28          return Point(y*p.z-z*p.y,z*p.x-x*p.z,x*p.y-y*p.x);
29      }
30      Point operator *(double d)
31      {
32          return Point(x*d,y*d,z*d);
33      }
34      Point operator / (double d)
35      {
36          return Point(x/d,y/d,z/d);
37      }
38      //点乘
39      double  operator ^(Point p)
40      {
41          return (x*p.x+y*p.y+z*p.z);
42      }
43  };
44  struct CH3D
45  {
46      struct face
47      {
48          //表示凸包一个面上的三个点的编号
49          int a,b,c;
50          //表示该面是否属于最终凸包上的面
51          bool ok;
52      };
53      //初始顶点数
54      int n;
55      //初始顶点
56      Point P[MAXN];
57      //凸包表面的三角形数
58      int num;
59      //凸包表面的三角形
60      face F[8*MAXN];
61      //凸包表面的三角形
62      int g[MAXN][MAXN];
63      //向量长度
64      double vlen(Point a)
65      {
66          return sqrt(a.x*a.x+a.y*a.y+a.z*a.z);
67      }
68      //叉乘
69      Point cross(const Point &a,const Point &b,const Point &c)
70      {
71          return Point((b.y-a.y)*(c.z-a.z)-(b.z-a.z)*(c.y-a.y),
72                       (b.z-a.z)*(c.x-a.x)-(b.x-a.x)*(c.z-a.z),
73                       (b.x-a.x)*(c.y-a.y)-(b.y-a.y)*(c.x-a.x)
74                      );
```

```
75      }
76      //三角形面积*2
77      double area(Point a,Point b,Point c)
78      {
79          return vlen((b-a)*(c-a));
80      }
81      //四面体有向体积*6
82      double volume(Point a,Point b,Point c,Point d)
83      {
84          return (b-a)*(c-a)^(d-a);
85      }
86      //正：点在面同向
87      double dblcmp(Point &p,face &f)
88      {
89          Point m=P[f.b]-P[f.a];
90          Point n=P[f.c]-P[f.a];
91          Point t=p-P[f.a];
92          return (m*n)^t;
93      }
94      void deal(int p,int a,int b)
95      {
96          int f=g[a][b];//搜索与该边相邻的另一个平面
97          face add;
98          if(F[f].ok)
99          {
100             if(dblcmp(P[p],F[f])>eps)
101                 dfs(p,f);
102             else
103             {
104                 add.a=b;
105                 add.b=a;
106                 add.c=p;//这里注意顺序，要成右手系
107                 add.ok=true;
108                 g[p][b]=g[a][p]=g[b][a]=num;
109                 F[num++]=add;
110             }
111         }
112     }
113     void dfs(int p,int now)//递归搜索所有应该从凸包内删除的面
114     {
115         F[now].ok=0;
116         deal(p,F[now].b,F[now].a);
117         deal(p,F[now].c,F[now].b);
118         deal(p,F[now].a,F[now].c);
119     }
120     bool same(int s,int t)
121     {
122         Point &a=P[F[s].a];
123         Point &b=P[F[s].b];
124         Point &c=P[F[s].c];
125         return fabs(volume(a,b,c,P[F[t].a]))<eps &&
126                fabs(volume(a,b,c,P[F[t].b]))<eps &&
127                fabs(volume(a,b,c,P[F[t].c]))<eps;
128     }
129     //构建三维凸包
130     void create()
131     {
132         int i,j,tmp;
133         face add;
```

```
134         num=0;
135         if(n<4)return;
136         //********************************************
137         //此段是为了保证前四个点不共面
138         bool flag=true;
139         for(i=1; i<n; i++)
140             if(vlen(P[0]-P[i])>eps)
141             {
142                 swap(P[1],P[i]);
143                 flag=false;
144                 break;
145             }
146         if(flag)return;
147         flag=true;
148         //使前三个点不共线
149         for(i=2; i<n; i++)
150             if(vlen((P[0]-P[1])*(P[1]-P[i]))>eps)
151             {
152                 swap(P[2],P[i]);
153                 flag=false;
154                 break;
155             }
156         if(flag)return;
157         flag=true;
158         //使前四个点不共面
159         for(int i=3; i<n; i++)
160             if(fabs((P[0]-P[1])*(P[1]-P[2])^(P[0]-P[i]))>eps)
161             {
162                 swap(P[3],P[i]);
163                 flag=false;
164                 break;
165             }
166         if(flag)return;
167         //*****************************************
168         for(i=0; i<4; i++)
169         {
170             add.a=(i+1)%4;
171             add.b=(i+2)%4;
172             add.c=(i+3)%4;
173             add.ok=true;
174             if(dblcmp(P[i],add)>0)swap(add.b,add.c);
175             g[add.a][add.b]=g[add.b][add.c]=g[add.c][add.a]=num;
176             F[num++]=add;
177         }
178         for(i=4; i<n; i++)
179             for(j=0; j<num; j++)
180                 if(F[j].ok&&dblcmp(P[i],F[j])>eps)
181                 {
182                     dfs(i,j);
183                     break;
184                 }
185         tmp=num;
186         for(i=num=0; i<tmp; i++)
187             if(F[i].ok)
188                 F[num++]=F[i];
189 
190     }
191     //表面积
192     double area()
```

```
193     {
194         double res=0;
195         if(n==3)
196         {
197             Point p=cross(P[0],P[1],P[2]);
198             res=vlen(p)/2.0;
199             return res;
200         }
201         for(int i=0; i<num; i++)
202             res+=area(P[F[i].a],P[F[i].b],P[F[i].c]);
203         return res/2.0;
204     }
205     //体积
206     double volume()
207     {
208         double res=0;
209         Point tmp(0,0,0);
210         for(int i=0; i<num; i++)
211             res+=volume(tmp,P[F[i].a],P[F[i].b],P[F[i].c]);
212         return fabs(res/6.0);
213     }
214     //表面三角形个数
215     int triangle()
216     {
217         return num;
218     }
219     //表面多边形个数
220     int polygon()
221     {
222         int i,j,res,flag;
223         for(i=res=0; i<num; i++)
224         {
225             flag=1;
226             for(j=0; j<i; j++)
227                 if(same(i,j))
228                 {
229                     flag=0;
230                     break;
231                 }
232             res+=flag;
233         }
234         return res;
235     }
236     //三维凸包重心
237     Point barycenter()
238     {
239         Point ans(0,0,0),o(0,0,0);
240         double all=0;
241         for(int i=0; i<num; i++)
242         {
243             double vol=volume(o,P[F[i].a],P[F[i].b],P[F[i].c]);
244             ans=ans+(o+P[F[i].a]+P[F[i].b]+P[F[i].c])/4.0*vol;
245             all+=vol;
246         }
247         ans=ans/all;
248         return ans;
249     }
250     //点到面的距离
251     double ptoface(Point p,int i)
```

```
252     {
253         return fabs(volume(P[F[i].a],P[F[i].b],P[F[i].c],p)/vlen((P[F[i].b]-P[F[i].a])*(P[F[i].c]-P
        [F[i].a])));
254     }
255 };
256 CH3D hull;
257 int main()
258 {
259     while(~scanf("%d",&hull.n))
260     {
261         for(int i=0; i<hull.n; i++)
262             scanf("%lf%lf%lf",&hull.P[i].x,&hull.P[i].y,&hull.P[i].z);
263         hull.create();
264         printf("%d\n",hull.polygon());
265     }
266     return 0;
267 }
```

5 String

5.1 KMP

5.1.1 KMP

```
const int maxn=1e6+10;

char a[maxn],b[maxn];
int nex[maxn];

void getNext()
{
    int n = strlen(b), i = 0, j = -1;
    nex[i] = j;
    while(i < n)
    {
        if(j == -1 || b[i] == b[j]) nex[++ i] = ++j;
        else j = nex[j];
    }
}

int KMP()
{
    int n = strlen(a), m = strlen(b);
    getNext(b);
    int i = 0, j = 0;
    while(i < n && j < m)
    {
        if(j == -1 || a[i] == b[j]) i ++, j ++;
        else j = nex[j];
    }
}
```

5.1.2 exKMP

```
const int maxn = 1e5 + 10;
int nex[maxn], extend[maxn];

//预处理计算Next数组
void getNext(char *str)
{
    int i = 0, j, po, len = strlen(str);
    nex[0] = len;     //初始化nex[0]
    while (str[i] == str[i + 1] && i + 1 < len) i++;   //计算nex[1]
    nex[1] = i;
    po = 1;   //初始化po的位置
    for (int i = 2; i < len; i++)
    {
        if (nex[i - po] + i < nex[po] + po)  //第一种情况，可以直接得到nex[i]的值
            nex[i] = nex[i - po];
        else    //第二种情况，要继续匹配才能得到nex[i]的值
        {
            j = nex[po] + po - i;
            if (j < 0) j = 0;    //如果i>po+nex[po],则要从头开始匹配
            while (i + j < len && str[j] == str[j + i]) j++;
            nex[i] = j;
            po = i;   //更新po的位置
        }
```

```
    }
}

void EXKMP(char *s1, char *s2)
{
    int i = 0, j, po, len = strlen(s1), l2 = strlen(s2);
    getNext(s2);
    while (s1[i] == s2[i] && i < l2 && i < len) i++;
    extend[0] = i;
    po = 0;
    for (int i = 1; i < len; i++)
    {
        if (nex[i - po] + i < extend[po] + po)
            extend[i] = nex[i - po];
        else
        {
            j = extend[po] + po - i;
            if (j < 0) j = 0;
            while (i + j < len && j < l2 && s1[j + i] == s2[j]) j++;
            extend[i] = j;
            po = i;
        }
    }
}
```

5.2 Trie

5.2.1 Trie

```
const int maxn = 2e6 + 10;

int trie[maxn][30], tot;
bool flag[maxn];

void insert_ch(char *str)
{
    int len = strlen(str);
    int root = 0;
    for (int i = 0; i < len; i++)
    {
        int id = str[i] - 'a';
        if (!trie[root][id]) trie[root][id] = ++tot;
        root = trie[root][id];
    }
    flag[root] = true;
}

bool find_ch(char *str)
{
    int len = strlen(str);
    int root = 0;
    for (int i = 0; i < len; i++)
    {
        int id = str[i] - 'a';
        if (!trie[root][id]) return false;
        root = trie[root][id];
    }
    return true;
}
```

5.2.2 Persistence Trie

```
const int maxn = 1e5 + 10;

int a[maxn], rt[maxn], n;

struct Trie
{
    int tot;
    int child[maxn * 32][2], sum[maxn *32];
    int insert(int x, int val)
    {
        int tmp, y;
        tmp = y= ++tot;
        for(int i = 30; i >= 0; --i)
        {
            child[y][0] = child[x][0];
            child[y][1] = child[x][1];
            sum[y] = sum[x] + 1;
            int t = val >> i & 1;
            x = child[x][t];
            child[y][t] = ++tot;
            y = child[y][t];
        }
        sum[y] = sum[x] + 1;
        return tmp;
    }
    int query(int l, int r, int val)
    {
        int tmp = 0;
        for(int i =30; i >= 0; --i)
        {
            int t = val >> i & 1;
            if(sum[child[r][t^1]] - sum[child[l][t^1]]) tmp += (1<<i), r = child[r][t^1], l = child
    [l][t ^ 1];
            else r = child[r][t], l = child[l][t];
        }
        return tmp;
    }
}trie;
```

5.2.3 01Trie

```
struct Trie {
    int tree[maxn*20][2], tot;
    int flag[maxn*20];

    void insert_ch(int x) {
        int root = 0;
        flag[0]++;
        for (int i = 30; i >= 0; --i) {
            int id = (x >> i) & 1;
            if (!tree[root][id]) {
                tree[root][id] = ++tot;
                tree[tree[root][id]][0] = tree[tree[root][id]][1] = 0;
                flag[tree[root][id]] = flag[tree[tree[root][id]][0]] = flag[tree[tree[root][id
    ]][1]] = 0;
            }
```

```
            root = tree[root][id];
            flag[root]++;
        }
    }

    void del(int x) {
        int root = 0;
        flag[0]--;
        for (int i = 30; i >= 0; --i) {
            int id = (x >> i) & 1;
            assert(tree[root][id]);
            if (flag[tree[root][id]] == 1) {
                flag[tree[root][id]] = 0;
                tree[root][id] = 0;
                return;
            }
            root = tree[root][id];
            flag[root]--;
        }
    }

    int find_ch(int x, int flag = 0) { // flag 0 最小异或值， 1 最大异或值
        int root = 0;
        int res = 0;
        for (int i = 30; i >= 0; --i) {
            int id = ((x >> i) & 1);
            if (flag) id = !id;
            if (tree[root][id]) {
                root = tree[root][id];
                res = res << 1 | id;
            } else {
                root = tree[root][!id];
                res = res << 1 | (!id);
            }
        }
        return res;
    }

    void init() {
        tree[0][0] = tree[0][1] = 0;
        tot = 0;
    }
};
```

5.3 Manachar

5.3.1 Manacher

```
const int maxn = 1e5 + 10;

char s[maxn];

char tmp[maxn << 1];
int Len[maxn << 1];

int init(char *str)
{
    int len = strlen(str);
    tmp[0] = '@';
```

```
    for (int i = 1; i <= 2 * len; i += 2)
    {
        tmp[i] = '#';
        tmp[i + 1] = str[i / 2];
    }
    tmp[2 * len + 1] = '#';
    tmp[2 * len + 2] = '$';
    tmp[2 * len + 3] = 0;
    return 2 * len + 1;
}

int manacher(char *str)
{
    int mx = 0, ans = 0, pos = 0;
    int len = init(str);
    for (int i = 1; i <= len; i++)
    {
        if (mx > i) Len[i] = min(mx - i, Len[2 * pos - i]);
        else Len[i] = 1;
        while (tmp[i - Len[i]] == tmp[i + Len[i]]) Len[i]++;
        if (Len[i] + i > mx) mx = Len[i] + i, pos = i;
    }
}
```

5.4 Aho-Corasick Automation

5.4.1 AC Automation

```
const int maxn = 5e5 + 10;

class AC_automation
{
public:
    int trie[maxn][26], cnt;
    int tag[maxn];
    int fail[maxn];

    void init()
    {
        memset(trie, 0, sizeof trie);
        memset(tag, 0, sizeof tag);
        memset(fail, 0, sizeof fail);
        cnt = 0;
    }

    void insert(char *str)
    {
        int root = 0;
        for (int i = 0; str[i]; i++)
        {
            int id = str[i] - 'a';
            if (!trie[root][id]) trie[root][id] = ++cnt;
            root = trie[root][id];
        }
        tag[root]++;
    }

    void build()
    {
```

```
        queue<int> que;
        for (int i = 0; i < 26; i++) if (trie[0][i]) que.push(trie[0][i]);
        while (!que.empty())
        {
            int k = que.front();
            // tag[k] += tag[fail[k]];
            que.pop();
            for (int i = 0; i < 26; i++)
            {
                if (trie[k][i])
                {
                    fail[trie[k][i]] = trie[fail[k]][i];
                    que.push(trie[k][i]);
                } else trie[k][i] = trie[fail[k]][i];
            }
        }
    }

    int query(char *str)
    {
        int p = 0, res = 0;
        for (int i = 0; str[i]; i++)
        {
            p = trie[p][str[i] - 'a'];
            for (int j = p; j && ~tag[j]; j = fail[j]) res += tag[j], tag[j] = -1;
        }
        return res;
    }

    void query(string str, ll *res) { // 查询所有前缀的匹配串个数, build时把fail指针上的tag加到当前tag
        int p = 0;
        for (int i = 0; i < (int)str.length(); i++)
        {
            p = trie[p][str[i] - 'a'];
            res[i] = tag[p];
        }
    }
} AC;
```

5.5 Suffix Array

5.5.1 Suffix Array

```
char s[maxn];
int sa[maxn], t[maxn], t2[maxn], c[maxn], n;

//build_sa(n + 1, 130), sa, height下标从1开始,rk下标从0开始
void build_sa(int n, int m)
{
    int *x = t, *y = t2;
    for(int i = 0; i < m; i++) c[i] = 0;
    for(int i = 0; i < n; i++) c[x[i] = s[i]]++;
    for(int i = 1; i < m; i++) c[i] += c[i - 1];
    for(int i = n - 1; i >= 0; i--) sa[--c[x[i]]] = i;
    for(int k = 1; k <= n; k <<= 1)
    {
        int p = 0;
        for(int i = n - k; i < n; i++) y[p++] = i;
        for(int i = 0; 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 = 0; i < n; i++) c[x[y[i]]]++;
        for(int i = 0; i < m; i++) c[i] += c[i - 1];
        for(int i = n - 1; i >= 0; i--) sa[--c[x[y[i]]]] = y[i];
        swap(x, y);
        p = 1; x[sa[0]] = 0;
        for(int i = 1; i < n; i++)
            x[sa[i]] = y[sa[i - 1]] == y[sa[i]] && y[sa[i - 1] + k] == y[sa[i] + k] ? p - 1 : p++;
        if(p >= n) break;
        m = p;
    }
}

int rk[maxn], height[maxn];

void getHeight()
{
    for(int i = 1; i <= n; i++) rk[sa[i]] = i;
    for(int i = 0, k = 0; i < n; i++)
    {
        if(k) k--;
        int j = sa[rk[i] - 1];
        while(s[i + k] == s[j + k]) k++;
        height[rk[i]] = k;
    }
}

int dp[maxn][20];

void RMQ()
{
    for(int i = 1; i <= n; i ++) dp[i][0] = height[i];
    for(int j = 1; (1 << j) < maxn; j ++)
        for(int i = 1; i + (1 << j) - 1 <= n; i ++)
            dp[i][j] = min(dp[i][j - 1], dp[i + (1 << (j - 1))][j - 1]);
}

int query(int l, int r)
{
    int k = 0;
    while((1 << (k + 1)) <= r - l + 1) k ++;
    return min(dp[l][k], dp[r - (1 << k) + 1][k]);
}

int lcp(int x, int y)
{
    x = rk[x], y = rk[y];
    if(x > y) swap(x, y);
    return query(x + 1, y);
}
```

5.6 PalindromicTree

5.6.1 PalindromicTree

```
// 求相交回文串数量

#include<bits/stdc++.h>

```

```cpp
#define ll long long
using namespace std;

const int maxn = 2e6+6;
const int N = 26;
const int mod = 51123987;

struct Palindromic_Tree {
    vector<pair<int, int> > next[maxn];
//    int next[maxn][N];//next指针，next指针和字典树类似，指向的串为当前串两端加上同一个字符构成
    int fail[maxn]{};//fail指针，失配后跳转到fail指针指向的节点
    int cnt[maxn]{}; //表示节点i表示的本质不同的串的个数（建树时求出的不是完全的，最后count()函数跑一遍以后才是正确的）
    int num[maxn]{}; //表示以节点i表示的最长回文串的最右端点为回文串结尾的回文串个数
    int len[maxn]{};//len[i]表示节点i表示的回文串的长度（一个节点表示一个回文串）
    int S[maxn]{};//存放添加的字符
    int last{};//指向新添加一个字母后所形成的最长回文串表示的节点。
    int n{};//表示添加的字符个数。
    int p{};//表示添加的节点个数。
    //0向前加，1向后加字符
    //int last[2];
    //int lpos, rpos;

    int newnode(int l) {//新建节点
        next[p].clear();
//        for (int i = 0; i < N; ++i) next[p][i] = 0;
//        cnt[p] = 0;
        num[p] = 0;
        len[p] = l;
        return p++;
    }

    void init() {//初始化
        n = last = p = 0;
        newnode(0);
        newnode(-1);
        S[n] = -1;//开头放一个字符集中没有的字符，减少特判
        fail[0] = 1;
        // lpos 为字符串最大长度
        // last[0] = last[1] = 0;
        // lpos = 100000, rpos = lpos - 1;
        // S[lpos - 1] = S[rpos + 1] = -1;
    }

    int get_fail(int x) {//和KMP一样，失配后找一个尽量最长的
        // op 0 向前， 1 向后
        // if (op == 0) while (S[lpos + len[x] + 1] != S[lpos]) x = fail[x];
        // else while(S[rpos - len[x] - 1] != S[rpos]) x = fail[x];
        while (S[n - len[x] - 1] != S[n]) x = fail[x];
        return x;
    }

    int find(int u, int c) {
        vector<pair<int, int> > & x = next[u];
        int sz = x.size();
        for(int i = 0; i < sz; ++i) {
            if(x[i].first == c) return x[i].second;
        }
        return 0;
```

```
63      }
64
65      int add(int c) {
66          // 注意清空左右字符
67          // if (op == 0) S[--lpos] = c, S[lpos - 1] = -1;
68          // else S[++rpos] = c, S[rpos + 1] = -1;
69          S[++n] = c;
70          int cur = get_fail(last);//通过上一个回文串找这个回文串的匹配位置
71          int x = find(cur, c);
72          if (!x) {
73  //          if (!next[cur][c]) {//如果这个回文串没有出现过，说明出现了一个新的本质不同的回文串
74              int now = newnode(len[cur] + 2);//新建节点
75              x = now;
76              fail[now] = find(get_fail(fail[cur]), c);
77              next[cur].emplace_back(make_pair(c, now));
78  //            fail[now] = next[get_fail(fail[cur])][c];//和AC自动机一样建立fail指针，以便失配后跳转
79  //            next[cur][c] = now;
80              num[now] = num[fail[now]] + 1;
81          }
82          last = x;
83          // 修改最终长度
84          // if (len[last[op]] == rpos - lpos + 1) last[op ^ 1] = last[op];
85  //          last = next[cur][c];
86  //          cnt[last]++;
87          return num[last];
88      }
89
90      void count() {
91          for (int i = p - 1; i >= 0; --i) cnt[fail[i]] += cnt[i];
92          //父亲累加儿子的cnt，因为如果fail[v]=u，则u一定是v的子回文串!
93      }
94  } solve;
95
96  char s[maxn];
97
98  ll a[maxn], b[maxn];
99  int main() {
100     solve.init();
101     int n;
102     scanf("%d", &n);
103     scanf("%s", s);
104     for (int i = 0; i < n; ++i) {
105         a[i] = solve.add(s[i] - 'a');
106     }
107     solve.init();
108     for (int i = n - 1; i >= 0; --i) {
109         b[i] = (b[i + 1] + solve.add(s[i] - 'a')) % mod;
110     }
111     ll res = (b[0] * (b[0] - 1) / 2) % mod;
112     for (int i = 0; i < n; ++i) {
113         res = ((res - (a[i] * b[i + 1]) + mod) % mod) % mod;
114     }
115     printf("%lld\n", res);
116     return 0;
117 }
```

5.7 Hash

5.7.1 hash

```
// hash常用素数
// 61, 83, 113, 151, 211
// 91815541, 38734667, 68861641
// 917120411, 687840301, 386910137, 515880193
// 1222827239, 1610612741

typedef unsigned long long ull;
struct mhash {
    // 自然溢出无模数 805306457
    ull base[maxn];
    ull hash_index[maxn];
    ull seed; //31, 131
    void inithash(ull seedt = 31) {
        base[0] = 1;
        seed = seedt;
        for (int i = 1; i < maxn; ++i) base[i] = base[i - 1] * seed;
    }
    void hash(char *p, int n) { // from 1 to n
        hash_index[0] = 0;
        for (int i = 1; i <= n; ++i) hash_index[i] = hash_index[i - 1] * seed + p[i] - 'a';
    }
    ull gethash(int s, int e) {
        return hash_index[e] - hash_index[s - 1] * base[e - s + 1];
    }
};
```

5.7.2 doubleHash

```
namespace Hash{

    template<class __A,class __B>
    class Hash{
    private:
        static const int size=2000000;
        __B *hash; __A *O; int sz;
    public:
        Hash(int hash_size=size){ sz=hash_size;
            hash=(__B *)malloc(sizeof(__B)*sz);
            O=(__A *)malloc(sizeof(__A)*sz);
            memset(O,0xff,sizeof(__A)*sz);
        }~Hash(){free(O);free(hash);}
        __B &operator [](const __A &_O){
            int loc=_O%sz;
            while(~O[loc]&&O[loc]!=_O){
                ++loc;
                if(loc>sz)loc=0;
            }if(!~O[loc])O[loc]=_O;
            return hash[loc];
        }
        void clear(){memset(O,0xff,sizeof(__A)*sz);}
    };

    struct StringDoubleHashResult{
        int32_t *H1,*H2,c_len,len;
        StringDoubleHashResult(int32_t sz=0){
```

```
28              len=sz; c_len=0; //cur_len;
29              if(len<=0){
30                  H1=H2=0;
31                  return;
32              }
33              H1=(int32_t *)malloc(sizeof(int32_t)*sz);
34              H2=(int32_t *)malloc(sizeof(int32_t)*sz);
35          }
36          ~StringDoubleHashResult(){}
37          void clear(){free(H1);free(H2);len=0;H1=H2=0;}
38          void resize(int new_len){
39              int32_t *T1=(int32_t *)malloc(sizeof(int32_t)*new_len);
40              int32_t *T2=(int32_t *)malloc(sizeof(int32_t)*new_len);
41              for(int i=0;i<c_len;++i)T1[i]=H1[i],T2[i]=H2[i];
42              free(H1);free(H2); H1=T1; H2=T2; len=new_len;
43          }
44          void erase(int ers_len){//erase suffix
45              c_len-=ers_len;
46              if(c_len<0)c_len=0;
47          }
48          //erase prefix not better than reculc
49      };
50
51      namespace hash_random{
52          const int mod_tot=5;
53          const int mod[]={1000000009,1000000007,998244353,917120411,515880193};
54      };
55
56      class StringDoubleHash{
57      private:
58          static const int enable_random=1;
59          int32_t sz,HA1,HA2;
60          long long B,C;
61          int32_t *H1,*H2;
62      public:
63          StringDoubleHash(int32_t SZ=2e6+5,int32_t ha1=-1,int32_t ha2=-1,int32_t b=-1,int32_t c=-1){
64              sz=SZ;
65              if(enable_random){
66                  std::mt19937 rnd(time(0)+19990630);
67                  int z1= rnd() % hash_random::mod_tot;
68                  int z2= (z1 +rnd()%(hash_random::mod_tot - 1) + 1) % hash_random::mod_tot;
69                  if(ha1<0)ha1=hash_random::mod[z1];
70                  if(ha2<0)ha2=hash_random::mod[z2];
71                  if(b<0)b=rnd()%114514+23333;
72                  if(c<0)c=rnd()%1919810+23333;
73              } else {
74                  if(ha1<0)ha1=1e9+7;
75                  if(ha2<0)ha2=1e9+9;
76                  if(b<0)b=114514;
77                  if(c<0)c=1919810;
78              }
79              HA1=ha1; HA2=ha2; B=b; C=c;
80              //cerr<<HA1<<" "<<HA2<<" "<<B<<" "<<C<<endl;
81              H1=(int32_t *)malloc(sizeof(int32_t)*sz);
82              H2=(int32_t *)malloc(sizeof(int32_t)*sz);
83              init_hash_val();
84          }
85          ~StringDoubleHash(){free(H1);free(H2);}
86          void init_hash_val(){
```

```
87              H1[0]=H2[0]=1;
88              for(int32_t i=1;i<sz;++i){
89                  H1[i]=(H1[i-1]*B)%HA1;
90                  H2[i]=(H2[i-1]*B)%HA2;
91              }
92          }
93          template <class _Tp>
94          StringDoubleHashResult culc_hash(const _Tp &s,int32_t len,int32_t tot_len=-1){
95              if(tot_len<0)tot_len=len;
96              StringDoubleHashResult R(tot_len);
97              if(len<=0)return R;
98              R.H1[0]=(s[0]+C)%HA1;
99              R.H2[0]=(s[0]+C)%HA2;
100             for(int32_t i=1;i<len;++i){
101                 R.H1[i]=(R.H1[i-1]*B+s[i]+C)%HA1;
102                 R.H2[i]=(R.H2[i-1]*B+s[i]+C)%HA2;
103             }
104             R.c_len=len;
105             return R;
106         }
107         // s is the char* first, len is the append length
108         template <class _Tp>
109         void append(StringDoubleHashResult &R,const _Tp &s,int32_t len){
110             if(len<=0)return;
111             int t_len=R.len;
112             while(R.c_len+len>t_len)t_len<<=1;
113             if(t_len>R.len)R.resize(t_len);
114             for(int32_t i=R.c_len;i<R.c_len+len;++i){
115                 if(i==0){
116                     R.H1[i]=(s[i-R.c_len]+C)%HA1;
117                     R.H2[i]=(s[i-R.c_len]+C)%HA2;
118                 } else {
119                     R.H1[i]=(R.H1[i-1]*B+s[i-R.c_len]+C)%HA1;
120                     R.H2[i]=(R.H2[i-1]*B+s[i-R.c_len]+C)%HA2;
121                 }
122             }
123             R.c_len+=len;
124         }
125         void append(StringDoubleHashResult &R, char s){
126             int t_len=R.len;
127             while(R.c_len+1>t_len)t_len<<=1;
128             if(t_len>R.len)R.resize(t_len);
129             for(int32_t i=R.c_len;i<R.c_len+1;++i){
130                 if(i==0){
131                     R.H1[i]=(s+C)%HA1;
132                     R.H2[i]=(s+C)%HA2;
133                 } else {
134                     R.H1[i]=(R.H1[i-1]*B+s+C)%HA1;
135                     R.H2[i]=(R.H2[i-1]*B+s+C)%HA2;
136                 }
137             }
138             R.c_len+=1;
139         }
140         //return hash [l,r)
141         ll gethash(const StringDoubleHashResult &R, int32_t l,int32_t r){
142             if(l>r||l<0||r-->R.c_len)return -1;//fail
143             ll v1=l>0?R.H1[l-1]*(long long)H1[r-l+1]%HA1:0;
144             ll v2=l>0?R.H2[l-1]*(long long)H2[r-l+1]%HA2:0;
145             v1=R.H1[r]-v1; v2=R.H2[r]-v2;
```

```
            if(v1<0)v1+=HA1; if(v2<0)v2+=HA2;
            return v1<<32|v2;
        }
        //merge two hashes as one(s1+s2), but need s2's length
        ll merge_hash(const long long &hs1,const long long &hs2,int lenr){
            int32_t m1=hs1>>32,m2=hs1&0xffffffffLL;
            int32_t m3=hs2>>32,m4=hs2&0xffffffffLL;
            m1=m1*(long long)H1[lenr]%HA1+m3;
            if(m1>=HA1)m1-=HA1;
            m2=m2*(long long)H2[lenr]%HA2+m4;
            if(m2>=HA2)m2-=HA2;
            return (long long)m1<<32|m2;
        }
    };
};
```

5.7.3 二维 hash

```
#define ull unsigned long long
const int maxn = 1005;
ull hs[maxn][maxn];
char a[maxn][maxn];
int n, m;
ull base1 = 131, base2 = 13331;
ull pwb1[maxn] = {1}, pwb2[maxn] = {1};

void init() {
    for (int i = 1; i < maxn; ++i) {
        pwb1[i] = pwb1[i - 1] * base1;
        pwb2[i] = pwb2[i - 1] * base2;
    }
}

void Hash() {
    for(int i=1;i<=n;i++)
        for(int j=1;j<=m;j++)
            hs[i][j]=hs[i][j-1]*base1+a[i][j] - 'a';
    for(int i=1;i<=n;i++)
        for(int j=1;j<=m;j++)
            hs[i][j]+=hs[i-1][j]*base2;
}

// 右下角(i,j), 行列长度n,m
ull getHs(int i, int j, int lenn, int lenm) {
    return hs[i][j] - hs[i - lenn][j] * pwb2[lenn] -
            hs[i][j - lenm] * pwb1[lenm] +
            hs[i - lenn][j - lenm] * pwb2[lenn] * pwb1[lenm];
}
```

5.7.4 树 hash 同构

```
// n=1e5的话base开2e6+9, 可以输出看到top不比n小即可
const int base = 2e6+9;
// vis大小要开到素数大小, turn表示当前树的编号, p是预处理数组
int vis[base + 1], top, turn, p[base + 1];
// 程序开头调用一次
void init() {
    top = 0;
```

```
    for (int i = 2; i <= base; ++i) {
        if (!vis[i]) {
            p[++top] = i;
        }
        for (int j = 1; j <= top && i * p[j] <= base; ++j) {
            vis[i * p[j]] = 1;
            if (i % p[j] == 0) break;
        }
    }
    assert(top >= maxn);
}

vector<int> edge[maxn];
// h[x]表示x这棵子树的hash值， g[x]表示以x为根的hash值
int h[maxn], g[maxn], sz[maxn];

struct TreeHash {
    int n;
    // 如果树比较多，在类内部开edge可能会炸内存，可以改到外面做前向星
    // 除了hs是答案其他都可以改到外部，只有edge需要清零
    // vector<int> edge[maxn];
    // int h[maxn], g[maxn], sz[maxn];
    vector<int> hs;

    void init(int n_ = 0) {
        n = n_;
        hs.clear();
    }

    void dfs1(int u, int pre) {
        sz[u] = 1;
        h[u] = 1;
        for (auto v : edge[u]) {
            if (v == pre) continue;
            dfs1(v, u);
            h[u] = (h[u] + 1ll * h[v] * p[sz[v]] % mod) % mod;
            sz[u] += sz[v];
        }
    }

    void dfs2(int u, int pre, int V, int needres = 1) {
        g[u] = (h[u] + 1ll * V * p[n - sz[u]] % mod) % mod;
        if (needres) hs.push_back(g[u]);
        for (auto v : edge[u]) {
            if (v == pre) continue;
            dfs2(v, u, (g[u] - 1ll * h[v] * p[sz[v]] % mod + mod) % mod);
        }
    }

    void work(int needres = 1) {
        // 无根树选一个不存在的点当pre即可，当多棵无根树判重时需要sort
        dfs1(1, 0);
        dfs2(1, 0, 0, needres);
        sort(hs.begin(), hs.end());
    }
};

// 获取删掉某叶子节点后以与该叶子节点相邻点开头的hash值
// int res = (hs[edge[i][0]] - 2 + mod) % mod;
```

5.8 Suffix Automation

5.8.1 SAM

```
const int maxn = 2e4 + 10;

struct SuffixAutomation
{
    int last, cnt;
    int ch[maxn << 1][26], fa[maxn << 1], len[maxn << 1], pos[maxn << 1];
    int sz[maxn << 1], a[maxn << 1], c[maxn << 1];

    void init()
    {
        last = cnt = 1;
        memset(ch[1], 0, sizeof ch[1]);
        fa[1] = len[1] = 0;
    }

    int inline newnode(int idx)
    {
        ++cnt;
        memset(ch[cnt], 0, sizeof ch[cnt]);
        fa[cnt] = len[cnt] = 0;
        pos[cnt] = idx;
        return cnt;
    }

    void ins(int c)
    {
        int p = last , np = newnode(pos[last] + 1);
        last = np, len[np] = len[p] + 1;
        for(; p && !ch[p][c]; p = fa[p]) ch[p][c] = np;
        if(!p) fa[np] = 1;
        else
        {
            int q = ch[p][c];
            if(len[p] + 1 == len[q]) fa[np] = q;
            else
            {
                int nq = newnode(pos[p] + 1);
                len[nq] = len[p] + 1;
                memcpy(ch[nq], ch[q], sizeof ch[q]);
                fa[nq] = fa[q], fa[q] = fa[np] = nq;
                for(; ch[p][c] == q; p = fa[p]) ch[p][c] = nq;
            }
        }
        sz[np] = 1;
    }

    int solve(int n)
    {
        /*求两个串的LCS:
            对一个字符串建立SAM, 记录一个当前匹配的长度Len和当前节点v, 枚举另一个字符串的每个字符;
            如果p有字符v的转移边出边, 则使Len加一, 并使p转移到出边指向的节点上;
            否则不断向父节点上跳, 直到当前节点有字符p的转移出边, 或者跳到根节点;
        */
        int p = 1, ans = 0, now_len = 0;
        for(int i = 0; s2[i]; i ++)
```

```
56          {
57              if(ch[p][s2[i] - 'a']) p = ch[p][s2[i] - 'a'], now_len ++;
58              else
59              {
60                  for(;p && !ch[p][s2[i] -'a'] ; p = fa[p]) ;
61                  if(p == 0) now_len = 0, p = 1;
62                  else now_len = len[p] + 1, p = ch[p][s2[i] - 'a'];
63              }
64              ans = max(now_len, ans);
65          }
66      }
67
68      void Toposort()
69      {
70          long long ans = 0;
71          for(int i = 1; i <= cnt; i ++) c[len[i]] ++;
72          for(int i = 1; i <= cnt; i ++) c[i] += c[i - 1];
73          for(int i = 1; i <= cnt; i ++) a[c[len[i]] --] = i;
74          for(int i = cnt; i; i --) sz[fa[a[i]]] += sz[a[i]];
75      }
76  }sam;
```

5.9 Others

5.9.1 最小表示法

```
1  // 0起始
2  int Gao(char a[], int len) {
3      int i = 0, j = 1, k = 0;
4      while (i < len && j < len && k < len) {
5          int cmp = a[(j + k) % len] - a[(i + k) % len];
6          if (cmp == 0) k++;
7          else {
8              if (cmp > 0) j += k + 1;
9              else i += k + 1;
10             if (i == j) j ++;
11             k = 0;
12         }
13     }
14     return min(i, j);
15 }
```

6 dp

6.1 BitDP

6.1.1 数位 dp 计和

```cpp
#include <bits/stdc++.h>
#define ll long long
using namespace std;
const int mod = 998244353;
pair<ll, ll> dp[20][1<<10];
bool vis[20][1<<10];
int k;
int t[20];
ll base[20];

pair<ll, ll> dfs(int pos, int state, bool limit, bool lead) {
    if (pos == -1) return __builtin_popcount(state) <= k ? make_pair(1, 0) : make_pair(0, 0);
    if (!limit && !lead && vis[pos][state]) return dp[pos][state];
    int up = limit ? t[pos] : 9;
    pair<ll, ll> res = {0, 0};
    for (int i = 0; i <= up; ++i) {
        int n_s = state;
        if (lead && i == 0) n_s = 0;
        else n_s = state | (1 << i);
        auto tmp = dfs(pos - 1, n_s, limit && i == t[pos], lead && i == 0);
        ll pre = 1ll * i * base[pos] % mod;
        (res.first += tmp.first) %= mod;
        (res.second += tmp.second + pre * tmp.first) %= mod;
    }
    if (!limit && !lead) dp[pos][state] = res, vis[pos][state] = 1;
    return res;
}

ll solve(ll x) {
    int pos = 0;
    do {
        t[pos ++] = x % 10;
    } while (x /= 10);
    return dfs(pos - 1, 0, true, true).second;
}

int main(int argc,char *argv[])
{
    base[0] = 1;
    for (int i = 1; i < 20; ++i) base[i] = base[i - 1] * 10;
    ll l, r;
    scanf("%lld%lld%d", &l, &r, &k);
    printf("%lld\n", (solve(r) - solve(l - 1) + mod) % mod);
    return 0;
}
```

6.1.2 两个数数位 dp

```cpp
// 二进制数位dp, 求a $\in$ 1~x 和 b $\in$ 1~y, 满足 $a & b > c || a ^ b < c$的对数
ll dp[maxn][2][2][2][2];
int a[maxn], b[maxn], c[maxn];


```

```
6  void cal(int *xt, ll x) {
7      int has = 0;
8      while (x) {
9          xt[has++] = x % 2;
10         x /= 2;
11     }
12 }
13
14 ll dfs(int pos, int o1, int o2, int lim1, int lim2) {
15     if (pos < 0) return 1;
16     ll &t = dp[pos][o1][o2][lim1][lim2];
17     if (t != -1) return t;
18     int up1 = o1 ? a[pos] : 1;
19     int up2 = o2 ? b[pos] : 1;
20     ll res = 0;
21     for (int i = 0; i <= up1; ++i) {
22         for (int j = 0; j <= up2; ++j) {
23             int t1 = i & j;
24             int t2 = i ^ j;
25             if (lim1 && t1 > c[pos]) continue;
26             if (lim2 && t2 < c[pos]) continue;
27             res += dfs(pos - 1, o1 && i == up1, o2 && j == up2, lim1 && t1 == c[pos], lim2 && t2 ==
       c[pos]);
28         }
29     }
30     return t = res;
31 }
32
33 ll solve(ll x, ll y, ll z) {
34     memset(dp, -1ll, sizeof dp);
35     for (int i = 0; i < 33; ++i) a[i] = b[i] = c[i] = 0;
36     cal(a, x);
37     cal(b, y);
38     cal(c, z);
39     return dfs(32, 1, 1, 1, 1);
40 }
41
42 int main(int argc, char *argv[]) {
43     int T;
44     scanf("%d", &T);
45     ll x, y, z;
46     for (int kase = 1; kase <= T; ++kase) {
47         scanf("%lld%lld%lld", &x, &y, &z);
48         ll res = solve(x, y, z);
49         res -= max(0ll, y - z + 1);
50         res -= max(0ll, x - z + 1);
51         printf("%lld\n", x * y - res);
52     }
53     return 0;
54 }
```

6.2 Subsequence

6.2.1 MaxSum

```
1 // 传入序列a和长度n，返回最大子序列和
2 int MaxSeqSum(int a[], int n)
3 {
4     int rt = 0, cur = 0;
```

```
    for (int i = 0; i < n; i++)
        cur += a[i], rt = max(cur, rt), cur = max(0, cur);
    return rt;
}
```

6.2.2 LIS

```
// 简单写法(下标从0开始,只返回长度)
int dp[N];
int LIS(int a[], int n)
{
    memset(dp, 0x3f, sizeof(dp));
    for (int i = 0; i < n; i++) *lower_bound(dp, dp + n, a[i]) = a[i];
    return lower_bound(dp, dp + n, INF) - dp;
}

// 小常数nlogn求序列用树状数组维护dp即可
// dp[i] = max(dp[j]) + 1 (j < i && a[j] < a[i])
```

6.2.3 LongestCommonIncrease

```
// 序列下标从1开始
int LCIS(int a[], int b[], int n, int m)
{
    memset(dp, 0, sizeof(dp));
    for (int i = 1; i <= n; i++)
    {
        int ma = 0;
        for (int j = 1; j <= m; j++)
        {
            dp[i][j] = dp[i - 1][j];
            if (a[i] > b[j]) ma = max(ma, dp[i - 1][j]);
            if (a[i] == b[j]) dp[i][j] = ma + 1;
        }
    }
    return *max_element(dp[n] + 1, dp[n] + 1 + m);
}
```

6.2.4 LCS

```
#include <stdio.h>
#include <string.h>

#define M 30005
#define SIZE 128
#define WORDMAX 3200
#define BIT 32

char s1[M], s2[M];
int nword;
unsigned int str[SIZE][WORDMAX];
unsigned int tmp1[WORDMAX], tmp2[WORDMAX];

void pre(int len)
{
    int i, j;
    memset(str, 0, sizeof(str));
```

```
18      for(i = 0; i < len; i ++)
19          str[s1[i]][i / BIT] |= 1 << (i % BIT);
20  }
21
22  void cal(unsigned int *a, unsigned int *b, char ch)
23  {
24      int i, bottom = 1, top;
25      unsigned int x, y;
26      for(i = 0; i < nword; i ++)
27      {
28          y = a[i];
29          x = y | str[ch][i];
30          top = (y >> (BIT - 1)) & 1;
31          y = (y << 1) | bottom;
32          if(x < y) top = 1;
33          b[i] = x & ((x - y) ^ x);
34          bottom = top;
35      }
36  }
37
38  int bitcnt(unsigned int *a)
39  {
40      int i, j, res = 0, t;
41      unsigned int b[5] = {0x55555555, 0x33333333, 0x0f0f0f0f, 0x00ff00ff, 0x0000ffff}, x;
42      for(i = 0; i < nword; i ++)
43      {
44          x = a[i];
45          t = 1;
46          for(j = 0; j < 5; j ++, t <<= 1)
47              x = (x & b[j]) + ((x >> t) & b[j]);
48          res += x;
49      }
50      return res;
51  }
52
53  void process()
54  {
55      int i, j, len1, len2;
56      unsigned int *a, *b, *t;
57      len1 = strlen(s1);
58      len2 = strlen(s2);
59      nword = (len1 + BIT - 1) / BIT;
60      pre(len1);
61      memset(tmp1, 0, sizeof(tmp1));
62      a = &tmp1[0];
63      b = &tmp2[0];
64      for(i = 0; i < len2; i ++)
65      {
66          cal(a, b, s2[i]);
67          t = a; a = b; b = t;
68      }
69      printf("%d\n", bitcnt(a));
70  }
71
72  int main()
73  {
74      while(scanf("%s%s", s1, s2) != EOF)
75          process();
76      return 0;
```

```
77 }
```

6.3 Others

问题 设 $f(i)=\min(y[k]-s[i]\times x[k]),k\in[1,i-1]$, 现在要求出所有 $f(i),i\in[1,n]$

考虑两个决策 j 和 k，如果 j 比 k 优，则

$$y[j]-s[i]\times x[j]<y[k]-s[i]\times x[k]$$

化简得:

$$\frac{y_j-y_k}{x_j-x_k}<s_i$$

不等式左边是个斜率，我们把它设为 $\text{slope}(j,k)$

我们可以维护一个单调递增的队列，为什么呢?

因为如果 $\text{slope}(q[i-1],q[i])>\text{slope}(q[i],q[i+1])$，那么当前者成立时，后者必定成立。即 $q[i]$ 决策优于 $q[i-1]$ 决策时，$q[i+1]$ 必然优于 $q[i]$，因此 $q[i]$ 就没有存在的必要了。所以我们要维护递增的队列。

那么每次的决策点 i，都要满足

$$\begin{cases}\text{slope}(q[i-1],q[i])<s[i]\\ \text{slope}(q[i],q[i+1])\geq s[i]\end{cases}$$

一般情况去二分这个 i 即可。

如果 $s[i]$ 是单调不降的，那么对于决策 j 和 $k(j<k)$ 来说，如果决策 k 优于决策 j，那么对于 $i\in[k+1,n]$，都存在决策 k 优于决策 j，因此决策 j 就可以舍弃了。这样的话我们可以用单调队列进行优化，可以少个 log。

单调队列滑动窗口最大值

```cpp
// k为滑动窗口的大小，数列下标从1开始，d为序列长度+1
deque<int> q;
for (int i = 0, j = 0; i + k <= d; i++)
{
    while (j < i + k)
    {
        while (!q.empty() && a[q.back()] < a[j]) q.pop_back();
        q.push_back(j++);
    }
    while (q.front() < i) q.pop_front();
    // a[q.front()]为当前滑动窗口的最大值
}
```

6.3.1 矩阵快速幂

```cpp
struct Matrix {
    int sz;
    // int n, m;
    ll a[maxn][maxn];
    Matrix(int sz_ = 0):sz(sz_) {
        memset(a, 0, sizeof a);
    }
    void pr() {
        printf("*\n");
        for(int i = 0; i < sz; ++i) {
            for (int j = 0; j < sz; ++j) {
                printf("%lld ", a[i][j]);
            }
            printf("\n");
        }
    }
    void tr() {
```

```
18          for (int i = 0; i < sz; ++i) {
19              for (int j = i + 1; j < sz; ++j) {
20                  swap(a[i][j], a[j][i]);
21              }
22          }
23      }
24  }res, t1;
25
26  void init() {
27      ;
28  }
29
30  Matrix mul(Matrix a, Matrix b)
31  {
32      Matrix res(a.sz);
33      // if (a.m != b.n) return res;
34      for(int i = 0; i < res.sz; i++) // a.n
35          for(int j = 0; j < res.sz; j++) // b.m
36              for(int k = 0; k < res.sz; k++) // a.m, b.n
37                  (res.a[i][j] +=a.a[i][k] * b.a[k][j] % mod) %= mod;
38      return res;
39  }
40
41  Matrix pow(ll n)
42  {
43      init();
44      //for(int i = 0; i < cur; i++) res.a[i][i] = 1;
45      while(n > 0) {
46          if(n & 1) res = mul(res, t1);
47          t1 = mul(t1, t1);
48          n >>= 1;
49      }
50      return res;
51  }
```

7 Others

7.1 mint 类

```
const int mod = 998244353;

struct mint {
  int n;
  mint(int n_ = 0) : n(n_) {}
};

mint operator+(mint a, mint b) { return (a.n += b.n) >= mod ? a.n - mod : a.n; }
mint operator-(mint a, mint b) { return (a.n -= b.n) < 0 ? a.n + mod : a.n; }
mint operator*(mint a, mint b) { return 1LL * a.n * b.n % mod; }
mint &operator+=(mint &a, mint b) { return a = a + b; }
mint &operator-=(mint &a, mint b) { return a = a - b; }
mint &operator*=(mint &a, mint b) { return a = a * b; }
ostream &operator<<(ostream &o, mint a) { return o << a.n; }
```

7.2 不重叠区间贪心

```
#include <bits/stdc++.h>
#define ll long long
using namespace std;

const int maxn = 5e5+5;
pair<int, int> a[maxn];
int main() {
    int n;
    cin >> n;
    for (int i = 1; i <= n; ++i) {
        cin >> a[i].second >> a[i].first;
    }
    sort(a + 1, a + 1 + n);
    int res = 1;
    int tmp = a[1].first;
//    printf("%d %d\n", a[1].second, a[1].first);
    for (int i = 2; i <= n; ++i) {
        if (a[i].second > tmp) {
            res ++;
//            printf("%d %d\n", a[i].second, a[i].first);
            tmp = a[i].first;
        }
    }
    printf("%d\n", res);
    return 0;
}
```

7.3 BigInt 类

```
const double PI = acos(-1.0);
struct Complex{
    double x,y;
    Complex(double _x = 0.0,double _y = 0.0){
        x = _x;
        y = _y;
    }
```

```
8       Complex operator-(const Complex &b)const{
9           return Complex(x - b.x,y - b.y);
10      }
11      Complex operator+(const Complex &b)const{
12          return Complex(x + b.x,y + b.y);
13      }
14      Complex operator*(const Complex &b)const{
15          return Complex(x*b.x - y*b.y,x*b.y + y*b.x);
16      }
17  };
18  void change(Complex y[],int len){
19      int i,j,k;
20      for(int i = 1,j = len/2;i<len-1;i++){
21          if(i < j)    swap(y[i],y[j]);
22          k = len/2;
23          while(j >= k){
24              j = j - k;
25              k = k/2;
26          }
27          if(j < k)    j+=k;
28      }
29  }
30  void fft(Complex y[],int len,int on){
31      change(y,len);
32      for(int h = 2;h <= len;h<<=1){
33          Complex wn(cos(on*2*PI/h),sin(on*2*PI/h));
34          for(int j = 0;j < len;j += h){
35              Complex w(1,0);
36              for(int k = j;k < j + h/2;k++){
37                  Complex u = y[k];
38                  Complex t = w*y[k + h/2];
39                  y[k] = u + t;
40                  y[k + h/2] = u - t;
41                  w = w*wn;
42              }
43          }
44      }
45      if(on == -1){
46          for(int i = 0;i < len;i++){
47              y[i].x /= len;
48          }
49      }
50  }
51  class BigInt
52  {
53  #define Value(x, nega) ((nega) ? -(x) : (x))
54  #define At(vec, index) ((index) < vec.size() ? vec[(index)] : 0)
55      static int absComp(const BigInt &lhs, const BigInt &rhs)
56      {
57          if (lhs.size() != rhs.size())
58              return lhs.size() < rhs.size() ? -1 : 1;
59          for (int i = lhs.size() - 1; i >= 0; --i)
60              if (lhs[i] != rhs[i])
61                  return lhs[i] < rhs[i] ? -1 : 1;
62          return 0;
63      }
64      using Long = long long;
65      const static int Exp = 9;
66      const static Long Mod = 1000000000;
```

```
67      mutable std::vector<Long> val;
68      mutable bool nega = false;
69      void trim() const
70      {
71          while (val.size() && val.back() == 0)
72              val.pop_back();
73          if (val.empty())
74              nega = false;
75      }
76      int size() const { return val.size(); }
77      Long &operator[](int index) const { return val[index]; }
78      Long &back() const { return val.back(); }
79      BigInt(int size, bool nega) : val(size), nega(nega) {}
80      BigInt(const std::vector<Long> &val, bool nega) : val(val), nega(nega) {}
81
82  public:
83      friend std::ostream &operator<<(std::ostream &os, const BigInt &n)
84      {
85          if (n.size())
86          {
87              if (n.nega)
88                  putchar('-');
89              for (int i = n.size() - 1; i >= 0; --i)
90              {
91                  if (i == n.size() - 1)
92                      printf("%lld", n[i]);
93                  else
94                      printf("%0*lld", n.Exp, n[i]);
95              }
96          }
97          else
98              putchar('0');
99          return os;
100     }
101     friend BigInt operator+(const BigInt &lhs, const BigInt &rhs)
102     {
103         BigInt ret(lhs);
104         return ret += rhs;
105     }
106     friend BigInt operator-(const BigInt &lhs, const BigInt &rhs)
107     {
108         BigInt ret(lhs);
109         return ret -= rhs;
110     }
111     BigInt(Long x = 0)
112     {
113         if (x < 0)
114             x = -x, nega = true;
115         while (x >= Mod)
116             val.push_back(x % Mod), x /= Mod;
117         if (x)
118             val.push_back(x);
119     }
120     BigInt(const char *s)
121     {
122         int bound = 0, pos;
123         if (s[0] == '-')
124             nega = true, bound = 1;
125         Long cur = 0, pow = 1;
```

```
126         for (pos = strlen(s) - 1; pos >= Exp + bound - 1; pos -= Exp, val.push_back(cur), cur = 0,
    pow = 1)
127             for (int i = pos; i > pos - Exp; --i)
128                 cur += (s[i] - '0') * pow, pow *= 10;
129         for (cur = 0, pow = 1; pos >= bound; --pos)
130             cur += (s[pos] - '0') * pow, pow *= 10;
131         if (cur)
132             val.push_back(cur);
133     }
134     BigInt &operator=(const char *s){
135         BigInt n(s);
136         *this = n;
137         return n;
138     }
139     BigInt &operator=(const Long x){
140         BigInt n(x);
141         *this = n;
142         return n;
143     }
144     friend std::istream &operator>>(std::istream &is, BigInt &n){
145         string s;
146         is >> s;
147         n=(char*)s.data();
148         return is;
149     }
150     BigInt &operator+=(const BigInt &rhs)
151     {
152         const int cap = std::max(size(), rhs.size()) + 1;
153         val.resize(cap);
154         int carry = 0;
155         for (int i = 0; i < cap - 1; ++i)
156         {
157             val[i] = Value(val[i], nega) + Value(At(rhs, i), rhs.nega) + carry, carry = 0;
158             if (val[i] >= Mod)
159                 val[i] -= Mod, carry = 1;
160             else if (val[i] < 0)
161                 val[i] += Mod, carry = -1;
162         }
163         if ((val.back() = carry) == -1) //assert(val.back() == 1 or 0 or -1)
164         {
165             nega = true, val.pop_back();
166             bool tailZero = true;
167             for (int i = 0; i < cap - 1; ++i)
168             {
169                 if (tailZero && val[i])
170                     val[i] = Mod - val[i], tailZero = false;
171                 else
172                     val[i] = Mod - 1 - val[i];
173             }
174         }
175         trim();
176         return *this;
177     }
178     friend BigInt operator-(const BigInt &rhs)
179     {
180         BigInt ret(rhs);
181         ret.nega ^= 1;
182         return ret;
183     }
```

```
184     BigInt &operator-=(const BigInt &rhs)
185     {
186         rhs.nega ^= 1;
187         *this += rhs;
188         rhs.nega ^= 1;
189         return *this;
190     }
191     friend BigInt operator*(const BigInt &lhs, const BigInt &rhs)
192     {
193         int len=1;
194         BigInt ll=lhs,rr=rhs;
195         ll.nega = lhs.nega ^ rhs.nega;
196         while(len<2*lhs.size()||len<2*rhs.size())len<<=1;
197         ll.val.resize(len),rr.val.resize(len);
198         Complex x1[len],x2[len];
199         for(int i=0;i<len;i++){
200             Complex nx(ll[i],0.0),ny(rr[i],0.0);
201             x1[i]=nx;
202             x2[i]=ny;
203         }
204         fft(x1,len,1);
205         fft(x2,len,1);
206         for(int i = 0 ; i < len; i++)
207             x1[i] = x1[i] * x2[i];
208         fft( x1 , len , -1 );
209         for(int i = 0 ; i < len; i++)
210             ll[i] = int( x1[i].x + 0.5 );
211         for(int i = 0 ; i < len; i++){
212             ll[i+1]+=ll[i]/Mod;
213             ll[i]%=Mod;
214         }
215         ll.trim();
216         return ll;
217     }
218     friend BigInt operator*(const BigInt &lhs, const Long &x){
219         BigInt ret=lhs;
220         bool negat = ( x < 0 );
221         Long xx = (negat) ? -x : x;
222         ret.nega ^= negat;
223         ret.val.push_back(0);
224         ret.val.push_back(0);
225         for(int i = 0; i < ret.size(); i++)
226             ret[i]*=xx;
227         for(int i = 0; i < ret.size(); i++){
228             ret[i+1]+=ret[i]/Mod;
229             ret[i] %= Mod;
230         }
231         ret.trim();
232         return ret;
233     }
234     BigInt &operator*=(const BigInt &rhs) { return *this = *this * rhs; }
235     BigInt &operator*=(const Long &x) { return *this = *this * x; }
236     friend BigInt operator/(const BigInt &lhs, const BigInt &rhs)
237     {
238         static std::vector<BigInt> powTwo{BigInt(1)};
239         static std::vector<BigInt> estimate;
240         estimate.clear();
241         if (absComp(lhs, rhs) < 0)
242             return BigInt();
```

```
243         BigInt cur = rhs;
244         int cmp;
245         while ((cmp = absComp(cur, lhs)) <= 0)
246         {
247             estimate.push_back(cur), cur += cur;
248             if (estimate.size() >= powTwo.size())
249                 powTwo.push_back(powTwo.back() + powTwo.back());
250         }
251         if (cmp == 0)
252             return BigInt(powTwo.back().val, lhs.nega ^ rhs.nega);
253         BigInt ret = powTwo[estimate.size() - 1];
254         cur = estimate[estimate.size() - 1];
255         for (int i = estimate.size() - 1; i >= 0 && cmp != 0; --i)
256             if ((cmp = absComp(cur + estimate[i], lhs)) <= 0)
257                 cur += estimate[i], ret += powTwo[i];
258         ret.nega = lhs.nega ^ rhs.nega;
259         return ret;
260     }
261     friend BigInt operator/(const BigInt &num,const Long &x){
262         bool negat = ( x < 0 );
263         Long xx = (negat) ? -x : x;
264         BigInt ret;
265         Long k = 0;
266         ret.val.resize( num.size() );
267         ret.nega = (num.nega ^ negat);
268         for(int i = num.size() - 1 ;i >= 0; i--){
269             ret[i] = ( k * Mod + num[i]) / xx;
270             k = ( k * Mod + num[i]) % xx;
271         }
272         ret.trim();
273         return ret;
274     }
275     bool operator==(const BigInt &rhs) const
276     {
277         return nega == rhs.nega && val == rhs.val;
278     }
279     bool operator!=(const BigInt &rhs) const { return nega != rhs.nega || val != rhs.val; }
280     bool operator>=(const BigInt &rhs) const { return !(*this < rhs); }
281     bool operator>(const BigInt &rhs) const { return !(*this <= rhs); }
282     bool operator<=(const BigInt &rhs) const
283     {
284         if (nega && !rhs.nega)
285             return true;
286         if (!nega && rhs.nega)
287             return false;
288         int cmp = absComp(*this, rhs);
289         return nega ? cmp >= 0 : cmp <= 0;
290     }
291     bool operator<(const BigInt &rhs) const
292     {
293         if (nega && !rhs.nega)
294             return true;
295         if (!nega && rhs.nega)
296             return false;
297         return (absComp(*this, rhs) < 0) ^ nega;
298     }
299     void swap(const BigInt &rhs) const
300     {
301         std::swap(val, rhs.val);
```

```
302         std::swap(nega, rhs.nega);
303     }
304 };
305 BigInt ba,bb;
306 int main(){
307     cin>>ba>>bb;
308     cout << ba + bb << '\n';//和
309     cout << ba - bb << '\n';//差
310     cout << ba * bb << '\n';//积
311     BigInt d;
312     cout << (d = ba / bb) << '\n';//商
313     cout << ba - d * bb << '\n';//余
314     return 0;
315 }
```

7.4 date

```
1  string dayOfWeek[] = {"Mo", "Tu", "We", "Th", "Fr", "Sa", "Su"};
2  // converts Gregorian date to integer (Julian day number)
3  int DateToInt (int m, int d, int y){
4      return
5          1461 * (y + 4800 + (m - 14) / 12) / 4 +
6          367 * (m - 2 - (m - 14) / 12 * 12) / 12 -
7          3 * ((y + 4900 + (m - 14) / 12) / 100) / 4 +
8          d - 32075;
9  }
10
11 // converts integer (Julian day number) to Gregorian date: month/day/year
12 void IntToDate (int jd, int &m, int &d, int &y){
13     int x, n, i, j;
14     x = jd + 68569;
15     n = 4 * x / 146097;
16     x -= (146097 * n + 3) / 4;
17     i = (4000 * (x + 1)) / 1461001;
18     x -= 1461 * i / 4 - 31;
19     j = 80 * x / 2447;
20     d = x - 2447 * j / 80;
21     x = j / 11;
22     m = j + 2 - 12 * x;
23     y = 100 * (n - 49) + i + x;
24 }
25 // converts integer (Julian day number) to day of week
26 string IntToDay (int jd){
27     return dayOfWeek[jd % 7];
28 }
```

7.5 Frac 类

```
1  struct Frac {
2      ll a, b;
3      void getJian() {
4          ll gcd = abs(__gcd(a, b));
5          a /= gcd;
6          b /= gcd;
7          if (b < 0) {
8              a = -a;
9              b = -b;
```

```
        }
    }
    Frac(ll a_ = 1, ll b_ = 1) {
        a = a_;
        b = b_;
        getJian();
    }
    Frac add(const Frac& oth) {
        ll bt = b * oth.b;
        ll at = a * oth.b + oth.a * b;
        return Frac(at, bt);
    }
    Frac multi(const Frac& oth) {
        a *= oth.a;
        b *= oth.b;
        getJian();
        return *this;
    }
    bool operator < (const Frac& oth) const {
        return a * oth.b < b * oth.a;
    }
    bool operator == (const Frac& oth) const {
        return a * oth.b == b * oth.a;
    }
    bool operator <= (const Frac& oth) const {
        return a * oth.b <= b * oth.a;
    }
};
```

7.6 模拟退火 (最小圆覆盖)

```
const int maxn = 1e5 + 10;
const double eps = 1e-8;
const double delta = 0.98;
const double inf = 1e18;

struct Point { double x, y; } p[maxn];

double dis(Point A, Point B) { return sqrt((A.x - B.x) * (A.x - B.x) + (A.y - B.y) * (A.y - B.y));
    };

double Simulate_Annea(int n)
{
    Point S;
    S.x = S.y = 0;
    double t = 1000;
    double res = inf;
    while(t > eps)
    {
        int k = 0;
        for(int i = 0; i < n; i ++) if(dis(S, p[i]) > dis(S, p[k])) k = i;
        double d = dis(S, p[k]);
        res = min(res, d);
        S.x += (p[k].x - S.x) / d * t;
        S.y += (p[k].y - S.y) / d * t;
        t *= delta;
    }
    return res;
```

```
}

int main()
{
    int n;
    scanf("%d", &n);
    for(int i = 0; i < n; i ++) scanf("%lf%lf", &p[i].x, &p[i].y);
    printf("%.3f\n", Simulate_Annea(n));
    return 0;
}
```

7.7 string 类

```
const int maxn = 1005;
struct String{
    int nex[maxn];
    char x[maxn];
    int len;
    int getLength() {
        return len;
    }
    void getNext() {
        int n = len, i = 0, j = -1;
        nex[0] = -1;
        while (i < n) {
            if (j == -1 || x[i] == x[j]) nex[++i] = ++j;
            else j = -1;
        }
    }
    void input() {
        scanf("%s", x);
        len = strlen(x);
    }
    void inputAndCal() {
        scanf("%s", x);
        len = strlen(x);
        getNext();
    }
    void show() {
        printf("%s\n", x);
    }
    bool operator < (const String&oth) const {
        return strcmp(x, oth.x) < 0;
    }
    char operator [] (const int a) const {
        return x[a];
    }
    bool substring(String b) {//b is the substring of a
        int m = len, n = b.getLength();
        int i = 0, j = 0;
        while (i < m && j < n) {
            if (j == -1 || x[i] == b[j]) ++i, ++j;
            else j = b.nex[j];
            if (j == n) return true;
        }
        return false;
    }
};
```

7.8 前缀异或和

```
ll xor_sum(ll n) {
    ll t=n&3;
    if (t&1) return t/2ull^1;
    return t/2ull^n;
}
```