ACM 模板

Aria
徐墨凡、徐铭扬、陈姿颖

目录

1 数学

1.1 多项式全家桶

```cpp
#include <bits/stdc++.h>
using namespace std;
const int D = 18, mod = 998244353, G = 3;
typedef unsigned long long ull;
typedef unsigned long long ll;

int power(int a, int b) {
    long long res = a, ans = 1;
    for (; b; b >>= 1, res = res * res % mod) if (b & 1) ans = ans * res % mod;
    return ans;
}
int Mod(int x) {return x >= mod ? x - mod : x;}
void SMod(int &x) { if (x >= mod) x -= mod; }
struct mint {
    int x;
    mint() {x = 0;}
    mint(int y) {x = y;}
    mint inv() const { return mint{power(x, mod - 2)}; }
    explicit operator int() const { return x; }
    int operator == (const mint &b) const { return x == b.x; }
    int operator != (const mint &b) const { return x != b.x; }
};
mint operator + (mint a, mint b) { return Mod(a.x + b.x); }
mint operator - (mint a, mint b) { return Mod(a.x + mod - b.x); }
mint operator * (mint a, mint b) { return 1ll * a.x * b.x % mod; }
mint operator - (mint a) { return Mod(mod - a.x); }
mint power(mint a, int b) {
    mint ans = 1;
    for (; b; b >>= 1, a = a * a) if (b & 1) ans = ans * a;
    return ans;
}

mint msqrt(mint x) {     // 二次剩余
    if (power(x, (mod - 1) / 2) != 1) return -1;
    while (1) {
        mint cur = rand() % mod;
```

```
37          if (power(cur * cur - x, (mod - 1) / 2) == 1) continue;
38          pair < mint, mint > res(cur, 1), ans(1, 0);
39          cur = cur * cur - x;
40          auto mult = [&](pair <mint, mint> a, pair <mint, mint> b) {
41              return make_pair(a.first * b.first + a.second * b.second * cur, a.
                    second * b.first + a.first * b.second);
42          };
43          for (int b = (mod + 1) / 2; b; b >>= 1, res = mult(res, res)) if (b &
                1) ans = mult(ans, res);
44          return min(ans.first.x, mod - ans.first.x);
45      }
46  }
47
48  mint fac[1 << D | 10], facinv[1 << D | 10], inv[1 << D | 10];
49  namespace Poly {
50      typedef vector <mint> poly;
51      vector <int> gpower[D];
52      ull nttf[1 << D | 10];
53      int rev[1 << D | 10];
54      int len(const poly &x) {return x.size();}
55      void init() {
56          for (int i = 0; i < D; i++) {
57              gpower[i].resize(2 << i);
58              int wn = power(G, (mod - 1) >> i + 1);
59              for (int j = 0, w = 1; j < 2 << i; j++, w = 1ll * w * wn % mod)
                    gpower[i][j] = w;
60          }
61          inv[1] = 1;
62          for (int i = 2; i <= 1 << D; i++) inv[i] = (mod - mod / i) * inv[mod %
                i];
63          fac[0] = facinv[0] = 1;
64          for (int i = 1; i <= 1 << D; i++) fac[i] = fac[i - 1] * i, facinv[i] =
                facinv[i - 1] * inv[i];
65      }
66      void get_rev(int l) {
67          static int lstl = -1;
68          if (l == lstl) return;
69          lstl = l;
70          for (int i = 1; i < 1 << l; i++) rev[i] = (rev[i >> 1] >> 1) | ((i & 1)
```

```
            << l - 1);
    }
    void NTT(ull *a, int n, int type) {
        for (int i = 0; i < n; i++) if (rev[i] < i) swap(a[i], a[rev[i]]);
        for (int p = 1, d = 0; p < n; p <<= 1, d++) {
            if (d == 17) for (int i = 0; i < n; i++) a[i] %= mod;
            for (int s = 0; s < n; s += p << 1) {
                for (int *w = gpower[d].data(), i = s; i < s + p; i++) {
                    int h1 = *w++ * a[i + p] % mod;
                    a[i + p] = a[i] + mod - h1;
                    a[i] += h1;
                }
            }
        }
        if (type == -1) {
            int inv = power(n, mod - 2);
            for (int i = 0; i < n; i++) a[i] = a[i] * inv % mod;
            reverse(a + 1, a + n);
        }
        else for (int i = 0; i < n; i++) a[i] %= mod;
    }
    void NTT(poly &a, int l, int type) {
        get_rev(l);
        for (int i = 0; i < 1 << l; i++) nttf[i] = a[i].x;
        NTT(nttf, 1 << l, type);
        for (int i = 0; i < 1 << l; i++) a[i].x = nttf[i];
    }
    poly operator * (poly a, poly b) {
        int n = len(a), m = len(b);
        int l = 0;
        for (; 1 << l < n + m - 1; l++);
        a.resize(1 << l), b.resize(1 << l);
        NTT(a, l, 1), NTT(b, l, 1);
        for (int i = 0; i < 1 << l; i++) a[i] = a[i] * b[i];
        NTT(a, l, -1);
        a.resize(n + m - 1);
        return a;
    }
    // 48 - 107 必抄
```

```
    poly operator + (poly a, const poly &b) {
        a.resize(max(len(a), len(b)));
        for (int i = 0; i < len(b); i++) a[i] = a[i] + b[i];
        return a;
    }
    poly operator - (poly a, const poly &b) {
        a.resize(max(len(a), len(b)));
        for (int i = 0; i < len(b); i++) a[i] = a[i] - b[i];
        return a;
    }
    poly operator * (poly a, mint b) {
        for (auto &i : a) i = i * b;
        return a;
    }

    poly Inv(const poly &a, int size) { // 逆
        poly ans;
        ans.push_back(a[0].inv());
        for (int l = 0; 1 << l < size; l++) {
            poly b = ans;
            b.resize(2 << l);
            poly tmp(a.begin(), min(a.end(), a.begin() + (2 << l)));
            tmp.resize(2 << l);
            NTT(tmp, l + 1, 1), NTT(b, l + 1, 1);
            for (int i = 0; i < 2 << l; i++) tmp[i] = tmp[i] * b[i];
            NTT(tmp, l + 1, -1);
            for (int i = 0; i < 1 << l; i++) tmp[i] = 0;
            NTT(tmp, l + 1, 1);
            for (int i = 0; i < 2 << l; i++) tmp[i] = tmp[i] * b[i];
            NTT(tmp, l + 1, -1);
            ans.resize(2 << l);
            for (int i = 1 << l; i < 2 << l; i++) ans[i] = 0 - tmp[i];
        }
        ans.resize(size);
        return ans;
    }

    poly Der(poly a) { // 求导
        for (int i = 1; i < len(a); i++) a[i - 1] = i * a[i];
```

```
        a.pop_back();
        return a;
    }
    poly Int(poly a) {
        a.push_back(0);
        for (int i = len(a); i --> 1; ) a[i] = a[i - 1] * inv[i];
        a[0] = 0;
        return a;
    }
    poly Ln(const poly &a, int size) {
        poly ans = Int(Inv(a, size) * Der(a));
        ans.resize(size);
        return ans;
    }

    poly Exp(const poly &a, int size) {
        poly ans;
        ans.push_back(1);
        for (int l = 0; 1 << l < size; l++) {
            poly b = ans, tmp(a.begin(), min(a.end(), a.begin() + (2 << l)));
            b.resize(2 << l), tmp.resize(2 << l);
            tmp = tmp - Ln(ans, 2 << l);
            NTT(b, l + 1, 1), NTT(tmp, l + 1, 1);
            for (int i = 0; i < 2 << l; i++) tmp[i] = tmp[i] * b[i];
            NTT(tmp, l + 1, -1);
            ans.resize(2 << l);
            for (int i = 1 << l; i < 2 << l; i++) ans[i] = tmp[i];
        }
        ans.resize(size);
        return ans;
    }
    poly T(poly a) {
        reverse(a.begin(), a.end());
        return a;
    }
    poly Div(const poly &a, const poly &b) {
        if (len(a) < len(b)) return poly();
        int l = len(a) - len(b) + 1;
        poly ans = T(a) * Inv(T(b), l);
```

```
        ans.resize(l);
        return T(ans);
    }
    poly Mod(const poly &a, const poly &b) {
        poly ans = a - Div(a, b) * b;
        ans.resize(len(b) - 1);
        return ans;
    }
    mint calcVal(const poly &a, mint b) {
        mint ans = 0;
        for (int i = len(a); i --> 0; ) ans = ans * b + a[i];
        return ans;
    }
    poly Pow(poly a, mint b, int size) {
        int n = len(a);
        return Exp(Ln(a, size) * b, size);
    }
    poly Sqrt(poly a, int size) {
        mint st = msqrt(a[0]), sti = a[0].inv();
        for (auto &i : a) i = i * sti;
        a = Pow(a, inv[2], size);
        for (auto &i : a) i = i * st;
        return a;
    }
    namespace QuickCalc {
        mint *x, *y;
        poly *p;
        void build(int cur, int l, int r) {
            if (l == r) return void(p[cur] = poly{1, -x[l]});
            int mid = l + r >> 1;
            build(cur << 1, l, mid);
            build(cur << 1 | 1, mid + 1, r);
            p[cur] = p[cur << 1] * p[cur << 1 | 1];
        }
        poly Tmult(poly a, poly b) {
            b = T(b);
            int l = 0, n = len(a), m = len(b);
            for (; 1 << l < n; l++);
            a.resize(1 << l), b.resize(1 << l);
```

```
226             NTT(a, l, 1), NTT(b, l, 1);
227             for (int i = 0; i < 1 << l; i++) a[i] = a[i] * b[i];
228             NTT(a, l, -1);
229             int ansl = n - m + 1;
230             poly ans(ansl);
231             for (int i = 0; i < ansl; i++) ans[i] = a[m - 1 + i];
232             return ans;
233         }
234         void calc(const poly &cur, int x, int l, int r) {
235             if (l == r) return void(y[l] = cur[0]);
236             int mid = l + r >> 1;
237             calc(Tmult(cur, p[x << 1 | 1]), x << 1, l, mid);
238             calc(Tmult(cur, p[x << 1]), x << 1 | 1, mid + 1, r);
239         }
240         void quickeva(poly f, int m, mint *X, mint *Y) {
241             if (m == 0) return;
242             int l = 0, n = len(f);
243             for (; 1 << l < m; l++);
244             p = new poly[(2 << l) + 1];
245             x = X, y = Y;
246             build(1, 0, m - 1);
247             f.resize(n + m - 1);
248             calc(Tmult(f, Inv(p[1], n)), 1, 0, m - 1);
249             delete[] p;
250             x = y = nullptr;
251         }
252         poly calc2(int x, int l, int r) {
253             if (l == r) return poly({y[l]});
254             int mid = l + r >> 1;
255             return calc2(x << 1, l, mid) * p[x << 1 | 1] + calc2(x << 1 | 1,
                    mid + 1, r) * p[x << 1];
256         }
257         poly quickint(int n, mint *X, mint *Y) {
258             if (n == 0) return poly();
259             int l = 0;
260             for (; 1 << l < n; l++);
261             p = new poly[(2 << l) + 1];
262             x = X;
263             y = new mint[n];
```

```
            build(1, 0, n - 1);
            poly f = Der(T(p[1]));
            f.resize(n + n - 1);
            calc(Tmult(f, Inv(p[1], n)), 1, 0, n - 1);
            for (int i = 0; i < n; i++) y[i] = Y[i] * y[i].inv();
            poly ans = calc2(1, 0, n - 1);
            delete[] p;
            delete[] y;
            x = y = nullptr;
            return T(ans);
        }
    }
    using QuickCalc :: quickeva;
    using QuickCalc :: quickint;
}
using Poly :: poly;
using Poly :: operator *;
using Poly :: operator +;
using Poly :: operator -;

int main() {
    Poly :: init(); // 必写
    int n, k;
    scanf("%d%d", &n, &k);
    Poly :: poly x;
    x.resize(++n);
    for (int i = 0; i < n; i++) scanf("%d", &x[i].x);
    Poly :: poly s = Poly :: Exp(Poly :: Int(Poly :: Inv(Poly :: Sqrt(x, n), n)
       ), n);
    x[0] = 2;
    x = Poly :: Ln(x - s, n);
    x[0] = 1;
    x = Poly :: Der(Poly :: Pow(x, k, n));
    for (int i = 0; i < n - 1; i++) printf("%d%c", x[i].x, " \n"[i == n - 2]);
}
```

1.2 任意模数 NTT

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N=1<<18;//取大于输入/输出的最小二的幂次乘二
int M=1e9+7;
namespace In_Out{
    #define gc getchar
    inline int rd(){
        int x=0,fl=1;char ch=gc();
        for (;ch<48||ch>57;ch=gc())if(ch=='-')fl=-1;
        for (;48<=ch&&ch<=57;ch=gc())x=x*10+(ch^48);
        return x*fl;
    }
    #define pc putchar
    inline void wri(int x){
        static char st[19];
        int tp=0;
        if(x<0)pc(45),x=-x;
        if(!x)pc(48);
        while(x)st[tp++]=x%10|48,x/=10;
        while(tp)pc(st[--tp]);
    }
    inline void wln(int x){wri(x),pc(10);}
}
using namespace In_Out;
namespace Math{
    inline int pw(int x,int y,int M){int z=1;for(;y;y>>=1,x=1ll*x*x%M)if(y&1)z
       =1ll*z*x%M;return z;}
    inline int inv(int x,int M){return pw(x,M-2,M);}
}
const int m1=998244353,m2=1004535809,m3=469762049;
const int inv1=Math::inv(m1,m2),inv2=Math::inv(1ll*m1*m2%m3,m3),M12=1ll*m1*m2%M
   ;
struct MOD{
    int x,y,z;
    friend void rd(MOD &a){a.x=rd();}
    MOD(){}
    MOD(int x):x(x),y(x),z(x){}
```

```
37      MOD(int x,int y,int z):x(x),y(y),z(z){}
38      inline friend MOD reduce(const MOD &a){return MOD(a.x+(a.x>>31&m1),a.y+(a.y
            >>31&m2),a.z+(a.z>>31&m3));}
39      inline friend MOD operator+(MOD a,MOD b){return reduce(MOD(a.x+b.x-m1,a.y+b
            .y-m2,a.z+b.z-m3));}
40      inline friend MOD operator-(MOD a,MOD b){return reduce(MOD(a.x-b.x,a.y-b.y,
            a.z-b.z));}
41      inline friend MOD operator*(MOD a,MOD b){return MOD(1ll*a.x*b.x%m1,1ll*a.y*
            b.y%m2,1ll*a.z*b.z%m3);}
42      inline friend MOD operator/(MOD a,int b){return a*(MOD(b)^(M-2));}
43      inline MOD &operator+=(MOD a){*this=*this+a;return *this;}
44      inline MOD &operator-=(MOD a){*this=*this-a;return *this;}
45      inline MOD &operator*=(MOD a){*this=*this*a;return *this;}
46      inline friend MOD operator^(MOD a,int b){return Math::pw(a.get(),b,M);}
47      inline int get(){//负数在模意义下的同余方程组直接合并会出锅
48          ll t=1ll*(y-x+m2)%m2*inv1%m2*m1+x;
49          return (1ll*(z-t%m3+m3)%m3*inv2%m3*M12%M+t)%M;
50      }
51  };
52  namespace Prepare{
53      const int g=3;
54      int r[N],Lim,l;//Lim是全局lim
55      MOD Wn[N|1];
56      void init(int n){
57          for(Lim=1,l=-1;Lim<n;Lim<<=1,l++);
58          for(int i=1;i<Lim;i++) r[i]=r[i>>1]>>1|(i&1)<<l;
59          Wn[0]=MOD(1);
60          MOD t(Math::pw(g,(m1-1)/Lim,m1),Math::pw(g,(m2-1)/Lim,m2),Math::pw(g,(
                m3-1)/Lim,m3));
61          for (int i=1;i<=Lim;i++) Wn[i]=Wn[i-1]*t;
62      }
63      void ntt(MOD *A,int op){
64          for(int i=0;i<Lim;i++)
65          if (i<r[i]) swap(A[i],A[r[i]]);
66          for(int mid=1,pp=Lim>>1;mid<Lim;mid<<=1,pp>>=1){
67              for(int R=mid<<1,j=0;j<Lim;j+=R){
68                  for(int k=0;k<mid;k++){
69                      MOD w=(op==1?Wn[pp*k]:Wn[Lim-pp*k]),x=A[j|k],y=w*A[j|k|mid
                            ];
```

```
                    A[j|k]=x+y,A[j|k|mid]=x-y;
                }
            }
        }
        if (op==-1){
            MOD Inv(Math::inv(Lim,m1),Math::inv(Lim,m2),Math::inv(Lim,m3));
            for(int i=0;i<Lim;i++) A[i]*=Inv;
        }
    }
    MOD I;
    struct C{
        MOD x,y;
        friend C operator*(C a,C b){return {a.x*b.x+a.y*b.y*I,a.x*b.y+b.x*a.y};}
    }b;
    C pw(C x,int y){C z={1,0};for(;y;y>>=1,x=x*x)if(y&1)z=z*x;return z;}
    bool chk(MOD x){return (x^((M-1)>>1)).get()==1;}
    MOD Cipolla(MOD n){
        srand(time(0));
        MOD a;
        do a=rand(),I=a*a-n;while(a.get()==0 || chk(I));
        C b=pw({a,1},(M+1)>>1);
        if (n.get()==0) return 0;
        else{
            MOD x1=b.x,x2=M-x1;
            return x1.get()<x2.get()?x1.get():x2.get();
        }
    }
}
using namespace Prepare;
class Poly{//调用时要时刻保证没有多余项
    private:
    int n;MOD *a;//n代表项数，即最高次数-1
    public:
    Poly(){a=new MOD[N];}
    void input(int t){n=t;for(int i=0;i<n;i++)a[i]=MOD(rd());}
    void output(){for(int i=0;i<n;i++)wri(a[i].get()),putchar(i==n-1?'\n':' ');}
    void setn(int t){n=t;}
```

```
107     MOD &operator[](int x)const{return a[x];}
108     Poly&operator=(Poly A){
109         n=A.n;
110         MOD *tmp=a;
111         a=A.a;
112         delete tmp;
113         return *this;
114     }
115     void copy(Poly A){n=A.n;for (int i=0;i<n;i++) a[i]=A[i];}
116     void get(){for(int i=0;i<n;i++){int t=a[i].get();a[i]=MOD(t%m1,t%m2,t%m3)
            ;}}
117     friend Poly operator+(const Poly a,const Poly b){
118         Poly C;C.n=max(a.n,b.n);
119         for(int i=0;i<C.n;i++)
120         if (i>=a.n) C[i]=b[i];
121         else if (i>b.n) C[i]=a[i];
122         else C[i]=a[i]+b[i];
123         return C;
124     }
125     friend Poly operator-(const Poly a,const Poly b){
126         Poly C;C.n=max(a.n,b.n);int t;
127         for(int i=0;i<C.n;i++)
128         if (i>=a.n) t=M-b[i].get(),C[i]=MOD(t%m1,t%m2,t%m3);
129         else if (i>b.n) C[i]=a[i];
130         else t=a[i].get()+M-b[i].get(),C[i]=MOD(t%m1,t%m2,t%m3);
131         return C;
132     }
133     friend Poly operator*(const Poly a,MOD b){
134         Poly A;A=a;
135         for (int i=0;i<A.n;i++) A.a[i]*=b;
136         return A;
137     }
138     friend Poly operator*(const Poly a,const Poly b){
139         Poly A,B;
140         A.copy(a),B.copy(b);
141         init(A.n+B.n-1);
142         for(int i=A.n;i<Lim;i++) A.a[i]=0;
143         for(int i=B.n;i<Lim;i++) B.a[i]=0;
144         A.n+=B.n-1;
```

```
145         ntt(A.a,1),ntt(B.a,1);
146         for (int i=0;i<Lim;i++) A[i]*=B[i];
147         ntt(A.a,-1);
148         A.get();
149         delete B.a;
150         return A;
151     }
152     friend Poly operator/(const Poly a,const Poly b){
153         Poly A,B,C;
154         A.copy(a),B.copy(b);
155         reverse(A.a,A.a+A.n);
156         reverse(B.a,B.a+B.n);
157         A.n=A.n-B.n+1;
158         B.n=A.n;//模x^(n-m+1)的意义下求逆
159         B=B.Inv();
160         C=A*B;
161         C.n=A.n;
162         reverse(C.a,C.a+C.n);
163         delete A.a,delete B.a;
164         return C;
165     }
166     friend Poly operator%(const Poly a,const Poly b){
167         Poly C;
168         C=a/b;
169         C=C*b;
170         C=a-C;
171         C.n=b.n-1;
172         return C;
173     }
174     Poly Sqrt(){//b=sqrt(a),new_b=(b+a*b^-1)/2 会调用Inv里的A,B
175         Poly A,B,b;
176         b[0]=Cipolla(a[0]);
177         int len,lim;
178         MOD inv2=MOD(2)^(M-2);
179         for(len=2;len<n<<1;len<<=1){
180             lim=len<<1;
181             for(int i=0;i<len;i++) A[i]=a[i];
182             for(int i=len;i<lim;i++) A[i]=0;
183             b.n=len,B=b.Inv();
```

```
            A=A*B;
            for(int i=0;i<len;i++) b[i]=(b[i]+A[i])*inv2;
        }
        for(int i=n;i<len;i++) b[i]=0;
        b.n=n;
        delete A.a,delete B.a;
        return b;
    }
    Poly Inv(){//b=a^-1,new_b=(2-a*b)*b
        Poly A,B;
        B[0]=a[0]^(M-2);
        int len,lim;
        for(len=2;len<n<<1;len<<=1){//len相当于x^n,len/2为已计算长度,lim是预留空间
            lim=len<<1;
            for(int i=0;i<len;i++) A[i]=a[i];
            for(int i=len;i<lim;i++) A[i]=0;//无法解释,多项式求逆可过,exp不可过
            init(lim);
            ntt(A.a,1),ntt(B.a,1);
            for(int i=0;i<lim;i++) A[i]=A[i]*B[i];//三模数NTT本质是答案小于m1*m2*m3,故不能连乘
            ntt(A.a,-1);
            for(int i=0;i<lim;i++) A[i]=M-A[i].get();
            ntt(A.a,1);
            for (int i=0;i<lim;i++) B[i]=(A[i]+2)*B[i];
            ntt(B.a,-1);
            for(int i=0;i<len;i++) B[i]=B[i].get();
            for(int i=len;i<lim;i++) B[i]=0;
        }
        for(int i=n;i<len;i++) B[i]=0;//留给别的函数ntt的,后面要清0
        B.n=n;
        delete A.a;
        return B;
    }
    Poly Deri(){
        Poly A;
        A.n=n-1;
        for(int i=0;i<n-1;i++) A[i]=a[i+1]*(i+1);
        A.get();
```

```
        return A;
    }
    Poly Inte(){
        Poly A;
        A.n=n+1,A[0]=0;
        for(int i=1;i<=n;i++) A[i]=a[i-1]/i;
        A.get();
        return A;
    }
    Poly Ln(){//lnf=∫(f'(x)/f(x)) dx
        Poly A,B;
        A=this->Deri();
        B=this->Inv();
        A=A*B,A.n=n-1;
        delete B.a;
        A=A.Inte();
        return A;
    }
    Poly Exp(){//b=e^a,new_b=b*(1+a-ln(b))
        Poly A,B,tmp;
        B[0]=1;
        int len,lim;
        for(len=2;len<n<<1;len<<=1){
            lim=len<<1;
            for(int i=0;i<len;i++) A[i]=a[i];
            A.n=B.n=len;
            tmp=B.Ln();
            A=A-tmp,A[0]+=MOD(1);
            B=A*B;
            for(int i=len;i<lim;i++) B[i]=0;
        }
        for(int i=n;i<len;i++) B[i]=0;
        B.n=n;
        delete A.a,delete tmp.a;
        return B;
    }
    Poly Pow(char s[]){
        Poly A;
        int l=strlen(s),t=0,k=0,k1=0,k2=0;
```

```
        MOD t1,t2;
        for (t=0;t<n && !a[t].x;t++);//统计0的个数
        for (int i=0;i<l;i++){
            if (t) k=k*10+(s[i]^48);
            k1=(10ll*k1+(s[i]^48))%M;//F(x)/F(0)的次数是由e^(kln(F))来计算的,是乘数,不是指数
            k2=(10ll*k2+(s[i]^48))%(M-1);//F(0)的次数
            if (k*t>=n){
                A.n=n;
                for (int j=0;j<n;j++) A[j]=0;
                return A;
            }
        }
        A.n=n-t;
        for (int i=0;i<n-t;i++) A[i]=a[i+t];//多余项不清零没事
        t1=A[0]^(M-2),t2=A[0]^k2;
        for (int i=0;i<A.n;i++) A[i]*=t1;
        A=A.Ln();
        for (int i=0;i<A.n;i++) A[i]*=k1;
        A=A.Exp();
        for (int i=0;i<A.n;i++) A[i]*=t2;
        A.n=n;
        for (int i=n-1;i>=k*t;i--) A[i]=A[i-k*t];
        for (int i=0;i<k*t;i++) A[i]=0;
        return A;
    }
};
Poly F;
int n;
int main(){
    n=rd();
    F.input(n);
    F.Inv().output();
}
```

1.3 离散对数 (BSGS)

求 x，使得 $a^x\ b(modp)$

```
#include<cstdio>
#include<cmath>
using namespace std;
const int M=1000003;
int p,a,b,i,s,m,t,mp[M],val[M],q[(1<<16)+10],cnt;
bool fl;
int ha(int x){
    int v=x%M;
    for (;mp[v] && mp[v]!=x;v=(v+1)%M);
    return v;
}
int pw(int x,int y){
    int z=1;
    for (;y;y>>=1,x=1ll*x*x%p)
    if (y&1) z=1ll*z*x%p;
    return z;
}
int main(){
    while (~scanf("%d%d%d",&p,&a,&b)){
        if (!(a%p)){
            puts("no solution");
            continue;
        }
        m=ceil(sqrt(p));
        s=b,cnt=0;
        for (i=0;i<=m;i++) val[ha(s)]=i,mp[ha(s)]=s,s=1ll*s*a%p,q[cnt++]=ha(s);
        s=1,t=pw(a,m),fl=0;
        for (i=1;i<=m && !fl;i++){
            s=1ll*s*t%p;
            if (val[ha(s)]) printf("%lld\n",1ll*i*m-val[ha(s)]),fl=1;
        }
        for (i=0;i<cnt;i++) mp[q[i]]=val[q[i]]=0;
        if (!fl) puts("no solution");
    }
}
```

1.4 高次剩余

求 x 使得 $x^a \equiv b(modp)$

```
#include<bits/stdc++.h>
using namespace std;
const int P=33331;
int T,x,y,r,G,g,k,phi,ans[P],pr[30],cnt,vis[P],val[P],Val[P],pos,i,p,a,b,j,num;
bool fl;
int pw(int x,int y){int z=1;for (;y;y>>=1,x=1ll*x*x%p)if (y&1) z=1ll*z*x%p;
   return z;}
int ex_gcd(int a,int b,int &x,int &y){
    if (!b){
        x=1,y=0;
        return a;
    }
    int G=ex_gcd(b,a%b,y,x);
    y-=a/b*x;
    return G;
}
int ha(int x){
    int v=x%P;
    for (;vis[v] && val[v]!=x;v=(v==P-1?0:v+1));
    return v;
}
int BSGS(int a,int b){
    int k=sqrt(p),pi=1;
    for (int i=0;i<=k;i++,pi=1ll*pi*a%p){
        int d=1ll*pi*b%p;
        pos=ha(d);
        vis[pos]=1,val[pos]=d,Val[pos]=i;
    }
    int d=pw(a,k);pi=d;
    for (int i=1;i<=phi/k;i++,pi=1ll*pi*d%p){
        pos=ha(pi);
        if (vis[pos]) return k*i-Val[pos];
    }
    return -1;
}
int main(){
    scanf("%d",&T);
```

```
37      for (;T--;){
38          memset(vis,0,sizeof(vis));
39          memset(val,0,sizeof(val));
40          memset(Val,0,sizeof(Val));
41          scanf("%d%d%d",&p,&a,&b);
42          phi=p-1;
43          cnt=0;
44          for (i=2;i*i<=phi;i++)//分解phi(p)，即p-1
45          if (phi%i==0){
46              pr[cnt++]=i;
47              while (phi%i==0) phi/=i;
48          }
49          if (phi>1) pr[cnt++]=phi;
50          phi=p-1;
51          for (i=1;i<=p;i++){//求原根
52              fl=1;
53              for (j=0;j<cnt && fl;j++)
54              if (pw(i,phi/pr[j])==1) fl=0;
55              if (fl) break;
56          }
57          g=i;
58          r=BSGS(g,b);//b≡g^r,x≡g^y
59          G=ex_gcd(phi,a,x,y);//g^(ay)≡g^r(mod p),ay≡r(mod phi)
60          if (r==-1 || r%G!=0){
61              puts("No Solution");
62              continue;
63          }
64          k=phi/G;
65          y=(1ll*r/G*y%k+k)%k;
66          num=0;
67          while (y<phi) ans[num++]=pw(g,y),y+=k;
68          sort(ans,ans+num);
69          for (i=0;i<num;i++) printf("%d%c",ans[i],i==num-1?'\n':' ');
70      }
71  }
```

1.5 类欧几里得算法

分别求 $\sum_{i=0}^{n}\lfloor\frac{ai+b}{c}\rfloor$, $\sum_{i=0}^{n}\lfloor\frac{ai+b}{c}\rfloor^2$, $\sum_{i=0}^{n}i\lfloor\frac{ai+b}{c}\rfloor$

```
#include<bits/stdc++.h>
using namespace std;
#define int long long
#define sqr(x) ((x)*(x)%M)
const int M=998244353,i2=499122177,i6=166374059;
struct num{
    int f,g,h;
    void print(){cout<<(f+M)%M<<' '<<(h+M)%M<<' '<<(g+M)%M<<endl;}
};
num calc(int n,int a,int b,int c){
    int ac=a/c,bc=b/c,n1=n+1,n2=n+n1;
    if (!a) return {bc*n1%M,
        n*n1/2%M*bc%M,
        sqr(bc)*n1%M};
    if (a>=c || b>=c){
        num s=calc(n,a%c,b%c,c);
        return {(s.f+n*n1/2%M*ac+bc*n1)%M,
            (s.g+ac*n%M*n1%M*n2%M*i6+n*n1/2%M*bc)%M,
            (s.h+sqr(ac)*n%M*n1%M*n2%M*i6+sqr(bc)*n1
            +n*n1%M*ac%M*bc+ac*2*s.g+bc*2*s.f)%M};
    }
    int m=(a*n+b)/c;
    num s=calc(m-1,c,c-b-1,a);
    int nowf=(n*m-s.f)%M;
    return {nowf,
        (m*n%M*n1-s.h-s.f)%M*i2%M,
        (n*m%M*(m+1)-s.g*2-s.f*2-nowf)%M};
}
int T,n,a,b,c;
signed main(){
    ios::sync_with_stdio(false),cin.tie(0),cout.tie(0);
    cin>>T;
    for (;T--;){
        cin>>n>>a>>b>>c;
        calc(n,a,b,c).print();
    }
}
```

1.6 Pollard_ Rho

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
int T;
ll ans,n;
ll mul(ll a,ll b,ll m){//快速乘
    ll r=a*b-(ll)((long double)a/m*b+1e-8)*m;
    if (r>=m) r-=m;
    if (r<0) r+=m;
    return r;
}
/*ll mul(ll x,ll y,ll M){
    ll z=0;
    for (;y;y>>=1,x=x*2%M)
    if (y&1) z=(z+x)%M;
    return z;
}*/
ll pw(ll x,ll y,ll M){
    ll z=1;
    for (;y;y>>=1,x=mul(x,x,M))
    if (y&1) z=mul(z,x,M);
    return z;
}
bool test(ll p,ll x){
    int r=0;ll d=x-1;
    while (!(d&1)) d>>=1,r++;
    for (ll i=pw(p,d,x);r--;){
        ll j=mul(i,i,x);
        if (j==1) return i==1 || i==x-1;
        i=j;
    }
    return 0;
}
bool miller_rabin(ll x){
    if (x==2 || x==61 || x==127) return 1;
    if (!test(2,x)) return 0;
    if (!test(61,x)) return 0;
    if (!test(127,x)) return 0;
```

```
    return 1;
}
#define ctz __builtin_ctzll
ll gcd(ll a,ll b){
    if (!a || !b) return a+b;
    int t=ctz(a|b);
    a>>=ctz(a);
    do{
        b>>=ctz(b);
        if (a>b) swap(a,b);
        b-=a;
    }while (b);
    return a<<t;
}
ll irand(ll x){return 1ll*((rand()<<15^rand())<<30^(rand()<<15^rand()))%x;}
ll f(ll x,ll c,ll M){
    ll t=mul(x,x,M)+c;
    if (t>=M) t-=M;
    return t;
}
/*ll work(ll x){
    ll c=irand(x),t1=irand(x),t2=f(t1,c,x),G=gcd(x,abs(t1-t2));
    while (G==1){
        t1=f(t1,c,x),t2=f(f(t2,c,x),c,x);
        G=gcd(x,abs(t1-t2));
    }
    return G;
}*/
ll work(ll n){
    if(!(n%2))return 2;
    if(!(n%3))return 3;
    ll x=0,y=x,t=1,q=1,c=irand(n-1)+1;
    for(int k=2;;k<<=1,y=x,q=1){
        for(int i=1;i<=k;++i){
            x=f(x,c,n);
            q=mul(q,abs(x-y),n);
            if(!(i&M)){
                t=gcd(q,n);
                if(t>1)break;
```

```
            }
        }
        if(t>1||(t=gcd(q,n))>1)break;
    }
    return t;
}
void pollard_rho(ll x){
    if (x==1) return;
    if (miller_rabin(x)){
        ans=max(ans,x);
        return;
    }
    ll y=x;
    while (x==y) y=work(x);
    pollard_rho(y),pollard_rho(x/y);
}
int main(){
    srand(time(0));
    scanf("%d",&T);
    for (;T--;){
        scanf("%lld",&n),ans=0;
        pollard_rho(n);
        if (ans==n) puts("Prime");
        else printf("%lld\n",ans);
    }
}
```

1.7 拉格朗日插值

$f(x)=\sum_{i=0}^{n-1} y_i \prod_{j=0,j\neq i}^{n-1} \frac{x-x_j}{x_i-x_j}$

```
void mul(int *f,int len,int t)
{//len为多项式的次数+1，函数让多项式f变成f*(x+t)
    for(int i=len;i>0;--i)
    temp[i]=f[i],f[i]=f[i-1];
    temp[0]=f[0],f[0]=0;
    for(int i=0;i<=len;++i)
    f[i]=(f[i]+1ll*t*temp[i]%P+P)%P;
}
void dev(int *f,int *r,int t)
```

```
{//f是被除多项式的系数，r保存f除以x+t的结果
    for(int i=0;i<=n;++i)
    temp[i]=f[i];
    for(int i=n;i>0;--i)
    {
        r[i-1]=temp[i];
        temp[i-1]=(temp[i-1]-1ll*t*temp[i]%P+P)%P;
    }
    return;
}
void lglr()
{
    memset(a,0,sizeof a);
    b[1]=1,b[0]=-x[1];
    for(int i=2;i<=n;++i)
    mul(b,i,-x[i]);
    //预处理(x-x1)*(x-x2)...*(x-xn)
    for(int i=1;i<=n;++i)
    {
        int fz=1;
        for(int j=1;j<=n;++j)
        {
            if(j==i)continue;
            fz=1ll*fz*(x[i]-x[j])%P;
        }
        fz=my_pow((fz+P)%P,P-2);
        fz=1ll*fz*y[i]%P;//得到多项式系数
        dev(b,c,-x[i]);//得到多项式，保存在b数组
        for(int j=0;j<n;++j)
        a[j]=(a[j]+1ll*fz*c[j])%P;
    }
    //1->n项得到x^0->x^(n-1)
}
```

1.8 分拆数

$O(n^2)$

```
void init()
{
    //g[i][j] -> i into j
    g[0][0]=1;
    for(int i=1;i<=1000;i++)
    for(int j=1;j<=i;j++)
    g[i][j]=(g[i-1][j-1]+g[i-j][j])%P;
    for(int i=0;i<=1000;i++)
    for(int j=0;j<=i;j++)
    (p[i]+=g[i][j])%=P;
    f[0]=1;
    for(int _=1;_<=4;_++)
    for(int i=1000;i>=0;i--)
    for(int j=1;j<=i;j++)
    (f[i]+=1ll*f[i-j]*p[j]%P)%=P;
}
```

考虑根号分治，设 $S = sqrt(n)$

对于 $> S$ 的数，设 $g_{i,j}$ 表示选了 i 个数总和为 $i(S+1)+j$ 的方案数，转移有两种：

1. 新加入一个数，初始值为 $S+1$: $g_{i,j} += g_{i-1,j}$

2. 给之前选的所有数 $+1$: $g_{i,j} += g_{i,j-1}$

不难发现，这样转移对于每一种拆分都有唯一的构造方式，并且第一维只有 $O(sqrt(n))$ 级别

求出 $\geq S$ 的数的 DP 数组后，对于 $\leq S$ 的数，暴力背包 DP 即可

1.9 第二类斯特林数

性质:$n^k = \sum_{i=0}^{k} S(k,i) * i! * C(n,i)$

1.10 第二类斯特林数·行

给定 n，对于所有的整数 $i \in [0,n]$，你要求出 $S(n,i)$。

```
Poly :: poly a, b;
a.resize(n + 1), b.resize(n + 1);
for(int i = 0; i <= n; ++i)
    a[i] = (i&1) ? (mod-facinv[i]) : facinv[i], b[i] = 1LL * power(i, n) *
       facinv[i];
a = a * b;
for (int i = 0; i <= n; i++) printf("%d ", a[i]);
// n = 3
// 0 1 3 1 固定元素数量
```

1.11 第二类斯特林数 · 列

给定 n,k，对于所有的整数 $i\in[0,n]$，你要求出 $S(i,k)$。

```
Poly :: poly st[27][2], f;
int n, k;
void solve(int l, int r, int lev, int dr)  {
    if (l == r) {
        st[lev][dr].resize(2);
        st[lev][dr][0] = 1, st[lev][dr][1] = mod - l;
        return; // 注意不是 mod - 1 ! ! !
    }
    solve(l, l + r >> 1, lev + 1, 0);
    solve((l + r >> 1) + 1, r, lev + 1, 1);
    st[lev][dr] = st[lev + 1][0] * st[lev + 1][1];
}
int main() {
    Poly :: init(); // 必写
    scanf("%d%d", &n, &k);
    if (k > n)
        for (int i = 0; i <= n; i++) printf("0 ");
    else {
        for (int i = 0; i <= k - 1; i++)
            printf("0 ");
        solve(1, k, 0, 0);
        f = Poly::Inv(st[0][0], n - k + 1);
        for (int i = 0; i <= n - k; i++) printf("%d ", f[i]);
    }
}
// n = 3, k = 2
// 0 0 1 3
```

1.12 第一类斯特林数・行

```
#include<bits/stdc++.h>
using namespace std;
#define int long long
typedef long long ll;
const ll mod=167772161;
ll G=3,invG;
const int N=1200000;
ll ksm(ll b,int n){
    ll res=1;
    while(n){
        if(n&1) res=res*b%mod;
        b=b*b%mod; n>>=1;
    }
    return res;
}
int read(){
    int x=0;char ch=getchar();
    while(!isdigit(ch))ch=getchar();
    while(isdigit(ch)) x=(x*10+(ch-'0'))%mod,ch=getchar();
    return x;
}
int tr[N];
void NTT(ll *f,int n,int fl){
    for(int i=0;i<n;++i)
        if(i<tr[i]) swap(f[i],f[tr[i]]);
    for(int p=2;p<=n;p<<=1){
        int len=(p>>1);
        ll w=ksm((fl==0)?G:invG,(mod-1)/p);
        for(int st=0;st<n;st+=p){
            ll buf=1,tmp;
            for(int i=st;i<st+len;++i)
                tmp=buf*f[i+len]%mod,
                f[i+len]=(f[i]-tmp+mod)%mod,
                f[i]=(f[i]+tmp)%mod,
                buf=buf*w%mod;
        }
    }
    if(fl==1){
```

```
        ll invN=ksm(n,mod-2);
        for(int i=0;i<n;++i)
            f[i]=(f[i]*invN)%mod;
    }
}
void Mul(ll *f,ll *g,int n,int m){
    m+=n;n=1;
    while(n<m) n<<=1;
    for(int i=0;i<n;++i)
        tr[i]=(tr[i>>1]>>1)|((i&1)?(n>>1):0);
    NTT(f,n,0);
    NTT(g,n,0);
    for(int i=0;i<n;++i) f[i]=f[i]*g[i]%mod;
    NTT(f,n,1);
}
ll inv[N],fac[N];
ll w[N],a[N],b[N],g[N];
void Solve(ll *f,int m){
    if(m==1) return f[1]=1,void(0);
    if(m&1){
        Solve(f,m-1);
        for(int i=m;i>=1;--i)
            f[i]=(f[i-1]+f[i]*(m-1)%mod)%mod;
        f[0]=f[0]*(m-1)%mod;
    }
    else{
        int n=m/2;ll res=1;
        Solve(f,n);
        for(int i=0;i<=n;++i)
            a[i]=f[i]*fac[i]%mod,b[i]=res*inv[i]%mod,res=res*n%mod;
        reverse(a,a+n+1);
        Mul(a,b,n+1,n+1);
        for(int i=0;i<=n;++i)
            g[i]=inv[i]*a[n-i]%mod;
        Mul(f,g,n+1,n+1);
        int limit=1;
        while(limit<(n+1)<<1) limit<<=1;
        for(int i=n+1;i<limit;++i) a[i]=b[i]=g[i]=0;
        for(int i=m+1;i<limit;++i) f[i]=0;
```

```
    }
}
ll f[N];
void init(int n){
    fac[0]=1;
    for(int i=1;i<=n;++i)
        fac[i]=1ll*fac[i-1]*i%mod;
    inv[n]=ksm(fac[n],mod-2);
    for(int i=n-1;i>=0;--i)
        inv[i]=1ll*inv[i+1]*(i+1)%mod;
}
signed main(){
    invG=ksm(G,mod-2);
    int n,k=0;
    cin>>n;
    init(n+n);
    Solve(f,n);
    for(int i=0;i<=n;++i)
        printf("%lld ",f[i]);
    return 0;
}
//n = 3
//0 2 3 1
```

1.13 常用生成函数公式

01 背包:$1+x^a$

01 可逆背包:$1-x^a$

完全背包:$\sum_{i=0}^{\infty} x^{a*i}=\frac{1}{1-x^a}$

多重背包:$\sum_{i=0}^{b} x^{a*i}=\frac{1-x^{a*(b+1)}}{1-x^a}$

完全背包逆背包:$\frac{1}{1+x^a}$

1.14 高斯消元

```
#include<bits/stdc++.h>
using namespace std;
const double eps=1e-8;
int n,i,j,w[15][15],st,pos[15];
double Ans,a[15][15],tot,ans[15];
bool fl;
int sgn(double a){return a<-eps?-1:a>eps;}
bool gauss(int n){
    int c=1,r=1;
    for (;c<=n;c++){
        int tmp=r;
        for (int i=r;i<=n;i++)
        if (sgn(a[i][c])) tmp=i;
        if (sgn(a[tmp][c])==0){
            ans[c]=0;//随意定
            continue;
        }
        pos[r]=c;
        for (int i=c;i<=n+1;i++) swap(a[r][i],a[tmp][i]);
        for (int i=n+1;i>=c;i--) a[r][i]/=a[r][c];
        for (int i=r+1;i<=n;i++)
        if (sgn(a[i][c]))
        for (int j=n+1;j>=c;j--) a[i][j]-=a[r][j]*a[i][c];
        r++;
    }
    r--;
    for (int i=r+1;i<=n;i++)
    if (sgn(a[i][n+1])) return 0;
    //以下一定有解
    for (int i=r;i;i--){
        ans[pos[i]]=a[i][n+1];
        for (int j=pos[i]+1;j<=n;j++) ans[pos[i]]-=ans[j]*a[i][j];
    }
    return 1;
}
int main(){
    cin>>n;
    for (i=1;i<=n;i++)
```

```
    for (j=1;j<=n;j++) cin>>w[i][j];
    for (st=0;st<(1<<n);st++){
        for (i=1;i<=n+1;i++)
        for (j=1;j<=n+2;j++) a[i][j]=0;
        for (i=1;i<=n;i++){
            for (j=1;j<=n;j++)
            if (j!=i) a[i][j]=w[i][j];
            a[i][n+1]=1;
            a[i][n+2]=0;
        }
        for (j=1;j<=n;j++) a[n+1][j]=1;
        a[n+1][n+2]=1;
        for (i=1;i<=n;i++)
        if (((st>>(i-1))&1)){
            a[n+1][i]=0;
            a[i][n+1]=0;
            for (j=1;j<=n;j++)
            if (j!=i) a[i][j]=a[j][i]=0;
        }
        if (gauss(n+1)){
            tot=0;
            fl=1;
            for (i=1;i<=n;i++)
            if (ans[i]<-eps) fl=0;
            if (!fl) continue;
            for (i=1;i<=n;i++)
            for (j=i+1;j<=n;j++) tot+=w[i][j]*ans[i]*ans[j];
            Ans=max(Ans,tot);
        }
    }
    printf("%.6f",Ans);
}
```

2 数据结构

2.1 01Trie

```
//从低位到高位
#include<bits/stdc++.h>
#define pb push_back
#define ll long long
using namespace std;
const int N=550010;
const int MAXH=20;
const int M=MAXH+3;
struct node
{
    int s[2],w,ans;
}tr[N*M];
int cnt;
#define ls tr[nw].s[0]
#define rs tr[nw].s[1]
int np()
{
    cnt++;
    tr[cnt].s[0]=tr[cnt].s[1]=tr[cnt].w=tr[cnt].ans=0;
    return cnt;
}
void update(int nw)
{
    tr[nw].w=tr[ls].w+tr[rs].w;
    tr[nw].ans=((tr[ls].ans^tr[rs].ans)<<1)|(tr[rs].w&1);
}
void insert(int &nw,int x,int d)
{
    if(!nw)nw=np();
    if(d>MAXH){tr[nw].w++;return;}
    insert(tr[nw].s[x&1],x>>1,d+1);
    update(nw);
}
void erase(int nw,int x,int d)
{
    if(d>MAXH){tr[nw].w--;return;}
```

```
    erase(tr[nw].s[x&1],x>>1,d+1);
    update(nw);
}
void addall(int nw)
{
    swap(ls,rs);
    if(ls)addall(ls);
    update(nw);
}
void merge(int &nw,int x)
{
    if(!nw){nw=x;return;}
    if(!x){return;}
    tr[nw].w+=tr[x].w;
    tr[nw].ans^=tr[x].ans;
    merge(ls,tr[x].s[0]);
    merge(rs,tr[x].s[1]);
}
```

2020ICPC 济南 K Kth Query

令 $f(a,S,K)$ 表示数组 b 中第 K 小的元素，其中 $\forall i\in[1,n],b_i=a_i\oplus S$

现在有数组 a 和 q 次询问，每次询问给出 L,R, 求 $min_{S=L}^{R}f(a,S,K)$

其中，$n,q\leq 10^5,0\leq a_i<2^{30},0\leq L\leq R<2^{30}$

```
//从高位到低位
#include<bits/stdc++.h>
#define pb push_back
#define ll long long
using namespace std;
const int N=100010;
const int MAXH=31;
const int M=MAXH+3;
int ch[N*M][2],val[N*M],cnt;
void clear()
{
    //注意没有多组数据也需要clear
    for(int i=1;i<=cnt;i++)
    ch[i][0]=ch[i][1]=val[i]=0;
    cnt=1;
}
void insert(int x)
```

```
{
    for(int i=MAXH,u=1;i>=0;i--)
    {
        int c=((x>>i)&1);
        if(!ch[u][c])ch[u][c]=++cnt;
        u=ch[u][c];
        val[u]++;
    }
}
void erase(int x)
{
    for(int i=MAXH,u=1;i>=0;i--)
    {
        int c=((x>>i)&1);
        if(!ch[u][c])ch[u][c]=++cnt;
        u=ch[u][c];
        val[u]--;
    }
}
int query(int x)
{
    int res=0;
    for(int i=MAXH,u=1;i>=0;i--)
    {
        int c=((x>>i)&1);
        if(ch[u][c^1]&&val[ch[u][c^1]])
        {
            u=ch[u][c^1];
            res|=(1<<i);
        }
        else u=ch[u][c];
    }
    return res;
}
int S;
int get_mex(int u,int d)
{
    //求数组中所有数 xor S 后的 mex
    //注意不能重复插入数
```

```
    if(!u)return 0;
    int c=((S>>d)&1);
    if(val[ch[u][c]]==(1<<d))
    return (1<<d)+get_mex(ch[u][c^1],d-1);
    return get_mex(ch[u][c],d-1);
}
vector<int>kth[N*M];
void dfs(int u,int d)
{
    if(!ch[u][0]&&!ch[u][1])
    {
        kth[u].resize(val[u]+1);
        for(int i=1;i<=val[u];i++)
        kth[u][i]=0;
        return;
    }
    if(ch[u][0])
    dfs(ch[u][0],d-1);
    if(ch[u][1])
    dfs(ch[u][1],d-1);
    kth[u].resize(val[u]+1);
    for(int i=1,lnum;i<=val[u];i++)
    {
        int ans1,ans2;
        lnum=val[ch[u][0]];
        if(lnum>=i)ans1=kth[ch[u][0]][i];
        else ans1=(1<<d)+kth[ch[u][1]][i-lnum];
        lnum=val[ch[u][1]];
        if(lnum>=i)ans2=kth[ch[u][1]][i];
        else ans2=(1<<d)+kth[ch[u][0]][i-lnum];
        kth[u][i]=min(ans1,ans2);
    }
}
int solve_l(int u,int d,int l,int k)
{
    if(!ch[u][0]&&!ch[u][1])return 0;
    int bit=(l>>d)&1;
    if(bit==1)
    {
```

```
96          int lnum=val[ch[u][1]];
97          if(lnum>=k)return solve_l(ch[u][1],d-1,l,k);
98          else return (1<<d)+solve_l(ch[u][0],d-1,l,k-lnum);
99      }
100     else
101     {
102         int lnum,ans1,ans2;
103         lnum=val[ch[u][0]];
104         if(lnum>=k)ans1=solve_l(ch[u][0],d-1,l,k);
105         else ans1=(1<<d)+solve_l(ch[u][1],d-1,l,k-lnum);
106         lnum=val[ch[u][1]];
107         if(lnum>=k)ans2=kth[ch[u][1]][k];
108         else ans2=(1<<d)+kth[ch[u][0]][k-lnum];
109         return min(ans1,ans2);
110     }
111 }
112 int solve_r(int u,int d,int r,int k)
113 {
114     if(!ch[u][0]&&!ch[u][1])return 0;
115     int bit=(r>>d)&1;
116     if(bit==0)
117     {
118         int lnum=val[ch[u][0]];
119         if(lnum>=k)return solve_r(ch[u][0],d-1,r,k);
120         else return (1<<d)+solve_r(ch[u][1],d-1,r,k-lnum);
121     }
122     else
123     {
124         int lnum,ans1,ans2;
125         lnum=val[ch[u][0]];
126         if(lnum>=k)ans1=kth[ch[u][0]][k];
127         else ans1=(1<<d)+kth[ch[u][1]][k-lnum];
128         lnum=val[ch[u][1]];
129         if(lnum>=k)ans2=solve_r(ch[u][1],d-1,r,k);
130         else ans2=(1<<d)+solve_r(ch[u][0],d-1,r,k-lnum);
131         return min(ans1,ans2);
132     }
133 }
134 int solve(int u,int d,int l,int r,int k)
```

```
{
    //将[L,R]区间拆分成无限区间
    if(!ch[u][0]&&!ch[u][1])return 0;
    int lbit=(l>>d)&1,rbit=(r>>d)&1;
    if(lbit==rbit)
    {
        int c=lbit,lnum=val[ch[u][c]];
        if(lnum>=k)return solve(ch[u][c],d-1,l,r,k);
        else return (1<<d)+solve(ch[u][c^1],d-1,l,r,k-lnum);
    }
    else
    {
        int lnum,ans1,ans2;
        lnum=val[ch[u][0]];
        if(lnum>=k)ans1=solve_l(ch[u][0],d-1,l,k);
        else ans1=(1<<d)+solve_l(ch[u][1],d-1,l,k-lnum);
        lnum=val[ch[u][1]];
        if(lnum>=k)ans2=solve_r(ch[u][1],d-1,r,k);
        else ans2=(1<<d)+solve_r(ch[u][0],d-1,r,k-lnum);
        return min(ans1,ans2);
    }
}
int main()
{
    clear();
    int n,Q;scanf("%d%d",&n,&Q);
    for(int i=1;i<=n;i++)
    {
        int x;scanf("%d",&x);
        insert(x);
    }
    dfs(1,MAXH);
    while(Q--)
    {
        int L,R,K;scanf("%d%d%d",&L,&R,&K);
        printf("%d\n",solve(1,MAXH,L,R,K));
    }
    return 0;
}
```

2.2 李超线段树

```
//要求在平面直角坐标系下维护两个操作：
//1.在平面上加入一条线段。记第 i 条被插入的线段的标号为 i。
//2.给定一个数 k，询问与直线 x=k 相交的线段中，交点纵坐标最大的线段的编号。
#include<bits/stdc++.h>
const int P1=39989;
const int P2=1e9;
const int N=1e5+10;
const double eps=1e-9;
using namespace std;
int cmp(double x,double y)
{
    if(x-y>eps)return 1;
    if(y-x>eps)return -1;
    return 0;
}
struct line{double k,b;}l[N];
double calc(int id,int x){return l[id].k*x+l[id].b;}
int n,m,lastans,tr[N<<2];
void add(int x0,int y0,int x1,int y1)
{
    m++;
    if(x0==x1)
    l[m].k=0,l[m].b=max(y0,y1);
    else
    l[m].k=1.0*(y1-y0)/(x1-x0),
    l[m].b=y0-l[m].k*x0;
}
#define ls (nw<<1)
#define rs (ls+1)
#define mid (l+r>>1)
void upd(int nw,int l,int r,int x)
{
    int &y=tr[nw];
    if(cmp(calc(x,mid),calc(y,mid))==1)swap(x,y);
    int L=cmp(calc(x,l),calc(y,l));
    int R=cmp(calc(x,r),calc(y,r));
    if(L==1||(L==0&&x<y))upd(ls,l,mid,x);
    if(R==1||(R==0&&x<y))upd(rs,mid+1,r,x);
```

```
}
void update(int nw,int l,int r,int ql,int qr,int x)
{
    if(ql<=l&&r<=qr)
    {
        upd(nw,l,r,x);
        return;
    }
    if(ql<=mid)update(ls,l,mid,ql,qr,x);
    if(qr>mid)update(rs,mid+1,r,ql,qr,x);
}
int query(int nw,int l,int r,int x)
{
    if(l==r)return tr[nw];
    int res;
    if(x<=mid)res=query(ls,l,mid,x);
    else res=query(rs,mid+1,r,x);
    int tmp=cmp(calc(res,x),calc(tr[nw],x));
    if(tmp==1||(tmp==0&&res<tr[nw]))return res;
    return tr[nw];
}
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0),cout.tie(0);
    cin >> n;
    while(n--)
    {
        int op;
        cin >> op;
        if(op)
        {
            int x0,y0,x1,y1;
            cin >> x0 >> y0 >> x1 >> y1;
            x0=(x0+lastans-1+P1)%P1+1;
            x1=(x1+lastans-1+P1)%P1+1;
            y0=(y0+lastans-1+P2)%P2+1;
            y1=(y1+lastans-1+P2)%P2+1;
            if(x0>x1)swap(x0,x1),swap(y0,y1);
```

```
78              add(x0,y0,x1,y1);
79              update(1,1,P1,x0,x1,m);
80          }
81          else
82          {
83              int k;
84              cin >> k;
85              k=(k+lastans-1+P1)%P1+1;
86              lastans=query(1,1,P1,k);
87              cout << lastans << endl;
88          }
89      }
90      return 0;
91  }
```

2.3 珂朵莉树

```
1   #include<bits/stdc++.h>
2   #define ll long long
3   #define pb push_back
4   #define fir first
5   #define sec second
6   const int P=1e9+7;
7   using namespace std;
8   int my_pow(ll x,int y,int p)
9   {
10      x%=p;
11      int z=1;
12      while(y)
13      {
14          if(y&1)z=1ll*x*z%p;
15          x=1ll*x*x%p,y/=2;
16      }
17      return z;
18  }
19  struct node
20  {
21      int l,r;
22      mutable ll v;
```

```
23      node(int l,int r,ll v):l(l),r(r),v(v){}
24      bool operator<(const node&o)const{return l<o.l;}
25  };
26  set<node>tr;
27  auto split(int pos)
28  {
29      auto it=tr.lower_bound(node(pos,0,0));
30      if(it!=tr.end()&&it->l==pos)return it;
31      it--;
32      int l=it->l,r=it->r;
33      ll v=it->v;
34      tr.erase(it);
35      tr.insert(node(l,pos-1,v));
36      return tr.insert(node(pos,r,v)).fir;
37  }
38  int n,m,seed,vmax,ret;
39  int rnd()
40  {
41      ret=seed;
42      seed=(1ll*seed*7+13)%P;
43      return ret;
44  }
45  void assign(int l,int r,int v)
46  {
47      auto end=split(r+1),it=split(l),begin=it;
48      //  for(;it!=end;it++)n-=(it->r-it->l+1)*it->v;
49      tr.erase(begin,end);
50      tr.insert(node(l,r,v));
51      //  n+=(r-l+1)*v;
52  }
53  void add(int l,int r,int v)
54  {
55      auto end=split(r+1);
56      for(auto it=split(l);it!=end;it++)
57      it->v+=v;
58  }
59  ll kth(int l,int r,int k)
60  {
61      auto end=split(r+1);
```

```
62      vector<pair<ll,int>>v;
63      for(auto it=split(l);it!=end;it++)
64      v.pb({it->v,it->r-it->l+1});
65      sort(v.begin(),v.end());
66      for(auto e:v)
67      {
68          k-=e.sec;
69          if(k<=0)return e.fir;
70      }
71  }
72  int sum_of_pow(int l,int r,int x,int y)
73  {
74      int ans=0;
75      auto end=split(r+1);
76      for(auto it=split(l);it!=end;it++)
77      ans=(ans+1ll*my_pow(it->v,x,y)*(it->r-it->l+1))%y;
78      return ans;
79  }
80  int main()
81  {
82      ios::sync_with_stdio(false);
83      cin.tie(0),cout.tie(0);
84      cin >> n >> m >> seed >> vmax;
85      for(int i=1;i<=n;i++)
86      tr.insert(node(i,i,rnd()%vmax+1));
87      for(int i=1;i<=m;i++)
88      {
89          int op,l,r,x,y;
90          op=rnd()%4+1;
91          l=rnd()%n+1;
92          r=rnd()%n+1;
93          if(l>r)swap(l,r);
94          if(op==3)x=rnd()%(r-l+1)+1;
95          else x=rnd()%vmax+1;
96          if(op==4)y=rnd()%vmax+1;
97
98          if(op==1)add(l,r,x);
99          else if(op==2)assign(l,r,x);
100         else if(op==3)cout << kth(l,r,x) << endl;
```

```
        else cout << sum_of_pow(l,r,x,y) << endl;
    }
    return 0;
}
```

2.4 扫描线

```
#include<bits/stdc++.h>
#define ll long long
const int N=1e5+10;
using namespace std;
int n,X[N<<1],tr[N<<3],c[N<<3],s[N<<3];
bool fl[N<<3],fr[N<<3];
struct node{int t,l,r,x;}p[N<<1];
bool cmp(node a,node b){return a.t==b.t?a.x>b.x:a.t<b.t;}
#define ls (nw<<1)
#define rs (ls+1)
#define mid (l+r>>1)
void up(int nw,int l,int r)
{
    if(c[nw])
    {
        tr[nw]=X[r+1]-X[l];
        s[nw]=1;
        fl[nw]=true;
        fr[nw]=true;
    }
    else if(l!=r)
    {
        tr[nw]=tr[ls]+tr[rs];
        s[nw]=s[ls]+s[rs];
        fl[nw]=fl[ls];
        fr[nw]=fr[rs];
        if(fr[ls]&&fl[rs])s[nw]--;
    }
    else
    {
        tr[nw]=0;
        s[nw]=0;
```

```
        fl[nw]=false;
        fr[nw]=false;
    }
}
void update(int nw,int l,int r,int ql,int qr,int x)
{
    if(ql<=X[l]&&X[r+1]<=qr)
    {
        c[nw]+=x;
        up(nw,l,r);
        return;
    }
    if(ql<X[mid+1])update(ls,l,mid,ql,qr,x);
    if(qr>X[mid+1])update(rs,mid+1,r,ql,qr,x);
    up(nw,l,r);
}
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0),cout.tie(0);
    cin >> n;
    for(int i=1;i<=n;i++)
    {
        int x1,x2,y1,y2;
        cin >> x1 >> y1 >> x2 >> y2;
        p[i*2-1]={y1,x1,x2,1};
        p[i*2]={y2,x1,x2,-1};
        X[i*2-1]=x1,X[i*2]=x2;
    }
    n<<=1;
    sort(p+1,p+1+n,cmp);
    sort(X+1,X+1+n);
    int tot=unique(X+1,X+1+n)-X-1;
    ll ans=0,lst=0;
    for(int i=1;i<n;i++)
    {
        update(1,1,tot-1,p[i].l,p[i].r,p[i].x);
        //      ans+=1ll*tr[1]*(p[i+1].t-p[i].t);
        //      求面积
```

```
72          ans+=abs(lst-tr[1]),lst=tr[1];
73          ans+=2ll*s[1]*(p[i+1].t-p[i].t);
74          //      求周长
75      }
76      ans+=p[n].r-p[n].l;
77      cout << ans << endl;
78      return 0;
79  }
```

2.5 摩尔投票

线段树/随机/根号分治/主席树找 size>l/k 的段

普通摩尔投票

```
1   struct Node
2   {
3       int m,cnt;
4       Node operator+(const Node &o)const
5       {
6           if(m==o.m)return {m,cnt+o.cnt};
7           else if(cnt>o.cnt)return {m,cnt-o.cnt};
8           else return {o.m,o.cnt-cnt};
9       }
10  }
```

拓展摩尔投票

```
1   struct Node
2   {
3       int m[K],cnt[K];
4       void insert(int m_,int cnt_)
5       {
6           for(int i=0;i<K;i++)
7           if(m[i]==m_)
8           {
9               cnt[i]+=cnt_;
10              return;
11          }
12          for(int i=0;i<K;i++)
13          if(!cnt[i])
14          {
```

```
            m[i]=m_;
            cnt[i]=cnt_;
            return;
        }
        int minn=cnt_;
        for(int i=0;i<K;i++)
        minn=min(minn,cnt[i]);
        cnt_-=minn;
        for(int i=0;i<K;i++)
        cnt[i]-=minn;
        if(cnt_)
        for(int i=0;i<K;i++)
        if(!cnt[i])
        {
            m[i]=m_;
            cnt[i]=cnt_;
            return;
        }
    }
    Node operator+(const Node &o)const
    {
        auto ret=*this;
        for(int i=0;i<K;i++)
        if(o.cnt[i]>0)
        ret.insert(o.m[i],o.cnt[i]);
        return ret;
    }
}
```

不带修询问

```
auto tmp=query(1,1,n,l,r);
int ans=inf;
for(int i=0;i<K;i++)
{
    if(!tmp.cnt[i])continue;
    auto itr=upper_bound(pos[tmp.m[i]].begin(),pos[tmp.m[i]].end(),r);
    auto itl=lower_bound(pos[tmp.m[i]].begin(),pos[tmp.m[i]].end(),l);
    if((itr-itl)*k>r-l+1)ans=min(ans,tmp.m[i]);
}
if(ans==inf)ans=-1;
```

带修询问

```
Node tmp=query(1,1,n,l,r);
if(!tmp.cnt)return -1;
auto itr=pos[tmp.m].order_of_key(r+1);
auto itl=pos[tmp.m].order_of_key(l);
int cnt=itr-itl,sum=r-l+1;
if(cnt>sum/2)return tmp.m;
else return -1;
```

修改

```
#include<ext/pb_ds/assoc_container.hpp>
typedef tree<int,null_type,less<int>,rb_tree_tag,
   tree_order_statistics_node_update>Set;
Set pos[N];

int x=read();
pos[a[x]].erase(x);
a[x]=ans;
update(1,1,n,x,a[x]);
pos[a[x]].insert(x);
```

3 图论

3.1 欧拉路

欧拉路径判定（是否存在）：

有向图欧拉路径：图中恰好存在 1 个点出度比入度多 1（这个点即为起点 S），1 个点入度比出度多 1（这个点即为终点 T），其余节点出度 = 入度。

有向图欧拉回路：所有点的入度 = 出度（起点 S 和终点 T 可以为任意点）。

无向图欧拉路径：图中恰好存在 2 个点的度数是奇数，其余节点的度数为偶数，这两个度数为奇数的点即为欧拉路径的起点 S 和终点 T。

无向图欧拉回路：所有点的度数都是偶数（起点 S 和终点 T 可以为任意点）。

注：存在欧拉回路（即满足存在欧拉回路的条件），也一定存在欧拉路径。

当然，一副图有欧拉路径，还必须满足将它的有向边视为无向边后它是连通的（不考虑度为 0 的孤立点），连通性的判断我们可以使用并查集或 dfs 等。

```
#include <bits/stdc++.h>
using namespace std;
const int MAX=100010;
int n,m,u,v,del[MAX];
int du[MAX][2];//记录入度和出度
stack <int> st;
vector <int> G[MAX];
void dfs(int now)
{
    for(int i=del[now];i<G[now].size();i=del[now])
    {
        del[now]=i+1;
        dfs(G[now][i]);
    }
    st.push(now);
}
int main()
{
    scanf("%d%d",&n,&m);
    for(int i=1;i<=m;i++) scanf("%d%d",&u,&v),G[u].push_back(v),du[u][1]++,du[v
       ][0]++;
    for(int i=1;i<=n;i++) sort(G[i].begin(),G[i].end());
    int S=1,cnt[2]={0,0}; //记录
    bool flag=1; //flag=1表示,所有的节点的入度都等于出度,
    for(int i=1;i<=n;i++)
    {
```

```
        if(du[i][1]!=du[i][0])
        {
            flag=0;
            if(du[i][1]-du[i][0]==1/*出度比入度多1*/) cnt[1]++,S=i;
            else if(du[i][0]-du[i][1]==1/*入度比出度多1*/) cnt[0]++;
            else return puts("No"),0;
        }
    }
    if((!flag)&&!(cnt[0]==cnt[1]&&cnt[0]==1)) return !puts("No"),0;
    //不满足欧拉回路的判定条件，也不满足欧拉路径的判定条件，直接输出"No"
    dfs(S);
    while(!st.empty()) printf("%d ",st.top()),st.pop();
    return 0;
}
```

3.2 三四元环计数

```
#include<bits/stdc++.h>
#define ll long long
#define pb push_back
const int P=1e9+7;
const int N=5e5+10;
using namespace std;
int n,m,d[N],x[N],y[N],vis[N],tag[N];
vector<int>T[N],e[N];
ll ans;
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0),cout.tie(0);
    int t;cin >> t;
    while(t--)
    {
        for(int i=1;i<=n;i++)
        T[i].clear(),e[i].clear(),d[i]=0;
        cin >> n >> m,ans=0;
        for(int i=1;i<=m;i++)
        {
            cin >> x[i] >> y[i];
```

```
            d[x[i]]++,d[y[i]]++;
            T[x[i]].pb(y[i]);
            T[y[i]].pb(x[i]);
        }
        for(int i=1;i<=m;i++)
        {
            if(d[x[i]]<d[y[i]]||(d[x[i]]==d[y[i]]&&x[i]<y[i]))
            e[x[i]].pb(y[i]);
            else e[y[i]].pb(x[i]);
        }
        for(int u=1;u<=n;u++)
        {
            ans+=1ll*d[u]*(d[u]-1)/2;
            for(auto v:e[u])
            {
                vis[v]=1;
                for(auto w:T[v])
                if(d[w]>d[u]||(d[w]==d[u]&&w>u))
                ans+=tag[w],tag[w]++;
            }
            for(auto v:e[u])
            for(auto w:e[v])
            if(vis[w])ans+=3;
            for(auto v:e[u])
            {
                vis[v]=0;
                for(auto w:T[v])
                if(d[w]>d[u]||(d[w]==d[u]&&w>u))
                tag[w]=0;
            }
        }
        if(n+m>3)ans+=1ll*m*(n+m-3);
        cout << ans%P << endl;
    }
    return 0;
}
```

3.3 仙人掌 + 四元环判断

```
#include<bits/stdc++.h>
#define pb push_back
using namespace std;
const int N=2e5+10;
int n,m;
int dis[N];
int idx,dfn[N],low[N];
int top,st[N];
int tot=1,head[N],nxt[N<<1],to[N<<1];
bool used[N<<1];
bool flag1,flag2,flag3;
int d[N];
int sz;
void add(int u,int v)
{
    tot++;
    to[tot]=v;
    nxt[tot]=head[u];
    head[u]=tot;
}
int dfs(int u)
{
    int tmp=1;
    for(int i=head[u];i;i=nxt[i])
    {
        int v=to[i];
        if(dis[v])
        {
            if((dis[u]&1)+(dis[v]&1)!=1)
            flag1=true;
        }
        else dis[v]=dis[u]+1,tmp+=dfs(v);
    }
    return tmp;
}
void tarjan(int u)
{
    int f=0;
```

```
    dfn[u]=low[u]=++idx;
    for(int i=head[u];i;i=nxt[i])
    {
        int v=to[i];
        if(used[i])continue;
        used[i]=used[i^1]=1;
        st[++top]=i;
        if(!dfn[v])
        {
            tarjan(v);
            low[u]=min(low[u],low[v]);
            if(dfn[u]<=low[v])
            {
                int j,num=0;
                do{
                    j=st[top--];
                    num++;
                }while(i!=j);
                if(num%2==0)flag2=true;
                if(num%2==1&&num>3)flag3=true;
            }
        }
        else low[u]=min(low[u],dfn[v]);
        if(low[v]<dfn[u])f++;
    }
    if(f>1)flag2=true;
}
bool check1()
{
    return flag3;
}
bool check2()
{
    for(int u=1;u<=n;u++)
    {
        if(d[u]==2)
        {
            int v1=to[head[u]],v2=to[nxt[head[u]]];
            if(d[v1]>2||d[v2]>2)return true;
```

```
            if(d[v1]==2)
            {
                int v3=to[head[v1]]+to[nxt[head[v1]]]-u;
                if(v3!=v2)return true;
            }
            if(d[v2]==2)
            {
                int v3=to[head[v2]]+to[nxt[head[v2]]]-u;
                if(v3!=v1)return true;
            }
        }
        else if(d[u]>=3)
        {
            for(int i=head[u];i;i=nxt[i])
            {
                int v=to[i];
                if(d[v]>1)return true;
            }
        }
    }
    return false;
}
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0),cout.tie(0);
    cin >> n >> m;
    for(int i=1;i<=m;i++)
    {
        int x,y;
        cin >> x >> y;
        add(x,y);
        add(y,x);
        d[x]++;
        d[y]++;
    }
    for(int i=1;i<=n;i++)
    if(!dfn[i])
    dis[i]=1,sz=max(sz,dfs(i)),tarjan(i);
```

```
    if(n<=3)cout << "-1\n";
    else if(flag1&&flag2)cout << "0\n";
    else if(flag2)cout << "1\n";
    else if(flag1)
    {
        if(check1()||check2())cout << "1\n";
        else cout << "2\n";
    }
    else if(sz>=4)cout << "2\n";
    else cout << 5-min(m,2) << endl;
    return 0;
}
```

3.4 KM 算法

若求最大权完美匹配，则将不存在的边设为-inf

若求最大权匹配，则将负权边和不存在的边设为 0，点数补至相等

```
#include<iostream>
#include<cstdio>
#include<cmath>
#include<cstring>
using namespace std;
typedef long long ll;
const ll Maxn=505;
const ll inf=1e18;
ll n,m,map[Maxn][Maxn],matched[Maxn];
ll slack[Maxn],pre[Maxn],ex[Maxn],ey[Maxn];//ex,ey顶标
bool visx[Maxn],visy[Maxn];
void match(ll u)
{
    ll x,y=0,yy=0,delta;
    memset(pre,0,sizeof(pre));
    for(ll i=1;i<=n;i++)slack[i]=inf;
    matched[y]=u;
    while(1)
    {
        x=matched[y];delta=inf;visy[y]=1;
        for(ll i=1;i<=n;i++)
        {
```

```
            if(visy[i])continue;
            if(slack[i]>ex[x]+ey[i]-map[x][i])
            {
                slack[i]=ex[x]+ey[i]-map[x][i];
                pre[i]=y;
            }
            if(slack[i]<delta){delta=slack[i];yy=i;}
        }
        for(ll i=0;i<=n;i++)
        {
            if(visy[i])ex[matched[i]]-=delta,ey[i]+=delta;
            else slack[i]-=delta;
        }
        y=yy;
        if(matched[y]==-1)break;
    }
    while(y){matched[y]=matched[pre[y]];y=pre[y];}
}
ll KM()
{
    memset(matched,-1,sizeof(matched));
    memset(ex,0,sizeof(ex));
    memset(ey,0,sizeof(ey));
    for(ll i=1;i<=n;i++)
    {
        memset(visy,0,sizeof(visy));
        match(i);
    }
    ll res=0;
    for(ll i=1;i<=n;i++)
    if(matched[i]!=-1)res+=map[matched[i]][i];
    return res;
}
int main()
{
    scanf("%lld%lld",&n,&m);
    for(int i=1;i<=n;i++)
    for(int j=1;j<=n;j++)
    map[i][j]=-inf;
```

```
    for(ll i=1;i<=m;i++)
    {
        ll u,v,w;
        scanf("%lld%lld%lld",&u,&v,&w);
        map[u][v]=w;
    }
    printf("%lld\n",KM());
    for(ll i=1;i<=n;i++)
    printf("%lld ",matched[i]);
    printf("\n");
    return 0;
}
```

4 树上问题

4.1 树哈希

```
const ull mask = std::chrono::steady_clock::now().time_since_epoch().count();
ull shift(ull x) {
    x ^= mask;
    x ^= x << 13;
    x ^= x >> 7;
    x ^= x << 17;
    x ^= mask;
    return x;
}
const int N = 1e6 + 10;
int n;
ull hash[N];
std::vector<int> edge[N];
std::set<ull> trees;
void getHash(int x, int p) {
    hash[x] = 1;
    for (int i : edge[x]) {
        if (i == p) {
            continue;
        }
        getHash(i, x);
        hash[x] += shift(hash[i]);
    }
    trees.insert(hash[x]);
}
```

4.2 次小生成树

```
#include <algorithm>
#include <iostream>
const int INF = 0x3fffffff;
const long long INF64 = 0x3fffffffffffffffLL;
struct Edge {
    int u, v, val;
    bool operator<(const Edge &other) const { return val < other.val; }
};
```

```
Edge e[300010];
bool used[300010];
int n, m;
long long sum;
class Tr {
    private:
    struct Edge {
        int to, nxt, val;
    } e[600010];
    int cnt, head[100010];
    int pnt[100010][22];
    int dpth[100010];
    // 到祖先的路径上边权最大的边
    int maxx[100010][22];
    // 到祖先的路径上边权次大的边，若不存在则为 -INF
    int minn[100010][22];
    public:
    void addedge(int u, int v, int val) {
        e[++cnt] = (Edge){v, head[u], val};
        head[u] = cnt;
    }
    void insedge(int u, int v, int val) {
        addedge(u, v, val);
        addedge(v, u, val);
    }
    void dfs(int now, int fa) {
        dpth[now] = dpth[fa] + 1;
        pnt[now][0] = fa;
        minn[now][0] = -INF;
        for (int i = 1; (1 << i) <= dpth[now]; i++) {
            pnt[now][i] = pnt[pnt[now][i - 1]][i - 1];
            int kk[4] = {maxx[now][i - 1], maxx[pnt[now][i - 1]][i - 1],
                minn[now][i - 1], minn[pnt[now][i - 1]][i - 1]};
            // 从四个值中取得最大值
            std::sort(kk, kk + 4);
            maxx[now][i] = kk[3];
            // 取得严格次大值
            int ptr = 2;
            while (ptr >= 0 && kk[ptr] == kk[3]) ptr--;
```

```
48              minn[now][i] = (ptr == -1 ? -INF : kk[ptr]);
49          }
50          for (int i = head[now]; i; i = e[i].nxt) {
51              if (e[i].to != fa) {
52                  maxx[e[i].to][0] = e[i].val;
53                  dfs(e[i].to, now);
54              }
55          }
56      }
57      int lca(int a, int b) {
58          if (dpth[a] < dpth[b]) std::swap(a, b);
59          for (int i = 21; i >= 0; i--)
60          if (dpth[pnt[a][i]] >= dpth[b]) a = pnt[a][i];
61          if (a == b) return a;
62          for (int i = 21; i >= 0; i--) {
63              if (pnt[a][i] != pnt[b][i]) {
64                  a = pnt[a][i];
65                  b = pnt[b][i];
66              }
67          }
68          return pnt[a][0];
69      }
70      int query(int a, int b, int val) {
71          int res = -INF;
72          for (int i = 21; i >= 0; i--) {
73              if (dpth[pnt[a][i]] >= dpth[b]) {
74                  if (val != maxx[a][i])
75                  res = std::max(res, maxx[a][i]);
76                  else
77                  res = std::max(res, minn[a][i]);
78                  a = pnt[a][i];
79              }
80          }
81          return res;
82      }
83  } tr;
84  int fa[100010];
85  int find(int x) { return fa[x] == x ? x : fa[x] = find(fa[x]); }
86  void Kruskal() {
```

```
87      int tot = 0;
88      std::sort(e + 1, e + m + 1);
89      for (int i = 1; i <= n; i++) fa[i] = i;
90      for (int i = 1; i <= m; i++) {
91          int a = find(e[i].u);
92          int b = find(e[i].v);
93          if (a != b) {
94              fa[a] = b;
95              tot++;
96              tr.insedge(e[i].u, e[i].v, e[i].val);
97              sum += e[i].val;
98              used[i] = 1;
99          }
100         if (tot == n - 1) break;
101     }
102 }
103 int main() {
104     std::ios::sync_with_stdio(0);
105     std::cin.tie(0);
106     std::cout.tie(0);
107     std::cin >> n >> m;
108     for (int i = 1; i <= m; i++) {
109         int u, v, val;
110         std::cin >> u >> v >> val;
111         e[i] = (Edge){u, v, val};
112     }
113     Kruskal();
114     long long ans = INF64;
115     tr.dfs(1, 0);
116     for (int i = 1; i <= m; i++) {
117         if (!used[i]) {
118             int _lca = tr.lca(e[i].u, e[i].v);
119             // 找到路径上不等于 e[i].val 的最大边权
120             long long tmpa = tr.query(e[i].u, _lca, e[i].val);
121             long long tmpb = tr.query(e[i].v, _lca, e[i].val);
122             // 这样的边可能不存在，只在这样的边存在时更新答案
123             if (std::max(tmpa, tmpb) > -INF)
124             ans = std::min(ans, sum - std::max(tmpa, tmpb) + e[i].val);
125         }
```

```
    }
    // 次小生成树不存在时输出 -1
    std::cout << (ans == INF64 ? -1 : ans) << '\n';
    return 0;
}
```

4.3 点分治

```
#include<bits/stdc++.h>
#define pb push_back
#define fir first
#define sec second
using namespace std;
const int inf=0x3f3f3f3f;
const int N=1e4+10,M=1e7+10;
typedef pair<int,int> pii;
int n,m,s,rt,q[N],d[N],sz[N],mx[N];
bool tf[M],vis[N],ans[N];
vector<pii>e[N];
void calc1(int u,int f)
{
    sz[u]=1;
    mx[u]=0;
    for(auto E:e[u])
    {
        int v=E.fir,w=E.sec;
        if(v==f||vis[v])continue;
        calc1(v,u);
        mx[u]=max(mx[u],sz[v]);
        sz[u]+=sz[v];
    }
    mx[u]=max(mx[u],s-sz[u]);
    if(mx[u]<mx[rt])rt=u;
}
queue<int>tmp,tag;
void calc2(int u,int f)
{
    for(int i=1;i<=m;i++)
    if(q[i]>=d[u])
```

```
    ans[i]|=tf[q[i]-d[u]];
    if(d[u]<M)tmp.push(d[u]);
    for(auto E:e[u])
    {
        int v=E.fir,w=E.sec;
        if(v==f||vis[v])continue;
        d[v]=d[u]+w,calc2(v,u);
    }
}
void dfz(int u,int f)
{
    tf[0]=true;
    tag.push(0);
    vis[u]=true;
    for(auto E:e[u])
    {
        int v=E.fir,w=E.sec;
        if(v==f||vis[v])continue;
        d[v]=w,calc2(v,u);
        while(!tmp.empty())
        {
            int t=tmp.front();tmp.pop();
            tag.push(t),tf[t]=true;
        }
    }
    while(!tag.empty())
    tf[tag.front()]=false,tag.pop();
    for(auto E:e[u])
    {
        int v=E.fir,w=E.sec;
        if(v==f||vis[v])continue;
        rt=0,s=sz[v];
        calc1(v,u);
        calc1(rt,0);
        dfz(rt,0);
    }
}
int main()
{
```

```
71      ios::sync_with_stdio(false);
72      cin.tie(0),cout.tie(0);
73      cin >> n >> m;
74      for(int i=1;i<n;i++)
75      {
76          int x,y,z;
77          cin >> x >> y >> z;
78          e[x].pb({y,z});
79          e[y].pb({x,z});
80      }
81      for(int i=1;i<=m;i++)
82      cin >> q[i];
83      rt=0,s=n;
84      mx[rt]=inf;
85      calc1(1,0);
86      calc1(rt,0);
87      dfz(rt,0);
88      for(int i=1;i<=m;i++)
89      if(ans[i])
90      cout << "AYE\n";
91      else cout << "NAY\n";
92      return 0;
93  }
```

5 动态规划

5.1 数位 dp

Version 1

```
#include<bits/stdc++.h>
using namespace std;
#define int long long
int T,l,r,f[35][2][2][2][2];
int dfs(int pos,bool f1,bool f2,bool f3,bool f4){
    if (pos==-1) return 1;
    if (f[pos][f1][f2][f3][f4]!=-1) return f[pos][f1][f2][f3][f4];
    int &t=f[pos][f1][f2][f3][f4];t=0;
    int d1=f1?l>>pos&1:0,d2=f2?r>>pos&1:1,d3=f3?l>>pos&1:0,d4=f4?r>>pos&1:1;
    for (int x=d1;x<=d2;x++)
    for (int y=d3;y<=d4;y++)
    if (!x || !y) t+=dfs(pos-1,f1&(x==d1),f2&(x==d2),f3&(y==d3),f4&(y==d4));
    return t;
}
int cal(int l,int r){
    int pos=0;
    for (int i=31;i;i--)
    if (r>>i&1){
        pos=i;
        break;
    }
    return dfs(pos,1,1,1,1);
}
signed main(){
    scanf("%lld",&T);
    for (;T--;){
        memset(f,-1,sizeof(f));
        scanf("%lld%lld",&l,&r);
        printf("%lld\n",cal(l,r));
    }
}
```

Version 2

```
//数位dp+记忆化搜索
#include<bits/stdc++.h>
using namespace std;
int n,cnt,a[20],dp[20][20][5];
int dfs(int pos,int sum,int state,int limit)
{
    //state 0--剩余状态，1--上一位是1，2--出现过13
    if(pos==0)return state==2&&sum==0;//做完了
    if(!limit && dp[pos][sum][state]!=-1)
    return dp[pos][sum][state];//无限制记忆化
    int x=limit?a[pos]:9;//能到达的最大位数
    int ret=0;
    for(int i=0;i<=x;i++)
    {
        int nxt_sum=(sum*10+i)%13;
        int nxt_state=state;
        if(state==0&&i==1)nxt_state=1;
        if(state==1&&i!=1)nxt_state=0;
        if(state==1&&i==3)nxt_state=2;
        //状态转化
        ret+=dfs(pos-1,nxt_sum,nxt_state,limit&&i==x);
    }
    if(!limit)dp[pos][sum][state]=ret;
    //无限制记忆化
    return ret;
}
int main()
{
    memset(dp,-1,sizeof(dp));
    while(scanf("%d",&n)!=EOF)
    {
        cnt=0;while(n)a[++cnt]=n%10,n/=10;
        //初始化
        printf("%d\n",dfs(cnt,0,0,1));
    }
    return 0;
}
```

6 计算几何

6.1 凸包

```
#include<bits/stdc++.h>
using namespace std;
typedef double D;
const D pi=acos(-1.0),eps=1e-7;
const int N=100002;
int n,k,i,top,st[N];
D L,ans;
struct P{
    D x,y;
    P(D xx=0,D yy=0):x(xx),y(yy){}
    friend P operator -(P a,P b){return P(a.x-b.x,a.y-b.y);}
    friend D operator ^(P a,P b){return a.x*b.y-a.y*b.x;}
}p[N];
D dir(P p,P p1,P p2){return (p1-p)^(p2-p);}
D dis (P p1,P p2){return sqrt((p1.x-p2.x)*(p1.x-p2.x)+(p1.y-p2.y)*(p1.y-p2.y))
   ;}
bool cmp(P p1,P p2){
    D t=dir(p[0],p1,p2);
    return t>eps || abs(t)<eps && dis(p[0],p1)<dis(p[0],p2);
}
void graham(int n){
    if (n==1) top=0,st[0]=0;
    else{
        top=1;
        st[0]=0;st[1]=1;
        if (n>2){
            for (int i=2;i<n;i++){
                while (top && dir(p[st[top-1]],p[st[top]],p[i])<=0) top--;
                st[++top]=i;
            }
        }
    }
}
int main(){
    scanf("%d",&n);
    k=0;
```

```
    for (i=0;i<n;i++){
        scanf("%lf%lf",&p[i].x,&p[i].y);
        if (p[i].y<p[k].y || p[i].y==p[k].y && p[i].x<p[k].x) k=i;
    }
    swap(p[0],p[k]);
    sort(p+1,p+n,cmp);
    graham(n);
    for (i=0;i<top;i++) ans+=dis(p[st[i]],p[st[i+1]]);
    ans+=dis(p[st[0]],p[st[top]]);
    printf("%.2f",ans);
}
```

6.2 两圆面积并

```
double AREA(point a, double r1, point b, double r2)
{
    double d = sqrt((a.x-b.x)*(a.x-b.x) + (a.y-b.y)*(a.y-b.y));
    if (d >= r1+r2)
    return 0;
    if (r1>r2)
    {
        double tmp = r1;
        r1 = r2;
        r2 = tmp;
    }
    if(r2 - r1 >= d)
    return pi*r1*r1;
    double ang1=acos((r1*r1+d*d-r2*r2)/(2*r1*d));
    double ang2=acos((r2*r2+d*d-r1*r1)/(2*r2*d));
    return ang1*r1*r1 + ang2*r2*r2 - r1*d*sin(ang1);
}
```

6.3 闵可夫斯基和

```
//询问tmp是否能成为三点分别位于三个凸包中的三角形的重心
#include<bits/stdc++.h>
#define ll long long
using namespace std;
const ll N=2e5+10;
```

```
struct Node
{
    ll x,y;
    Node operator - (Node A) {return (Node){x-A.x,y-A.y};}
    Node operator + (Node A) {return (Node){x+A.x,y+A.y};}
    Node operator * (ll A) {return (Node){x*A,y*A};}
    ll operator * (Node A) const {return x*A.y-y*A.x;}
    ll len() const {return x*x+y*y;}
}A[N],C1[N],C2[N],C3[N],s1[N],s2[N];
ll cmp1(const Node&A,const Node&B) {return A.y<B.y||(A.y==B.y&&A.x<B.x);}
ll cmp2(const Node&A,const Node&B) {return A*B>0||(A*B==0&&A.len()<B.len());}
ll n,m,k,sta[N],top,q,totA,totB,totC;
void Convex(Node *A,ll &n)
{
    sort(A+1,A+n+1,cmp1);
    A[0]=A[1];sta[top=1]=1;
    for(ll i=1;i<=n;i++) A[i]=A[i]-A[0];
    sort(A+2,A+n+1,cmp2);
    for(ll i=2;i<=n;sta[++top]=i,i++)
    while(top>=2&&(A[i]-A[sta[top-1]])*(A[sta[top]]-A[sta[top-1]])>=0) top--;
    for(ll i=1;i<=top;i++) A[i]=A[sta[i]]+A[0];
    n=top;A[n+1]=A[1];
}
void Minkowski(Node *C1,Node *C2,ll n,ll m,Node *A,ll &tot)
{
    for(ll i=1;i<n;i++) s1[i]=C1[i+1]-C1[i];s1[n]=C1[1]-C1[n];
    for(ll i=1;i<m;i++) s2[i]=C2[i+1]-C2[i];s2[m]=C2[1]-C2[m];
    A[tot=1]=C1[1]+C2[1];
    ll p1=1,p2=1;
    while(p1<=n&&p2<=m) ++tot,A[tot]=A[tot-1]+(s1[p1]*s2[p2]>=0?s1[p1++]:s2[p2
        ++]);
    while(p1<=n) ++tot,A[tot]=A[tot-1]+s1[p1++];
    while(p2<=m) ++tot,A[tot]=A[tot-1]+s2[p2++];
}
ll in(Node a,Node *A,ll tot)
{
    if(a*A[2]>0||A[tot]*a>0||(a*A[2]==0&&a.len()>A[2].len())||(A[tot]*a==0&&a.
        len()>A[tot].len())) return 0;
    ll ps=lower_bound(A+1,A+tot+1,a,cmp2)-A-1;
```

```
    return (a-A[ps])*(A[ps%tot+1]-A[ps])<=0;
}
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0),cout.tie(0);
    cin >> n;
    for(ll i=1;i<=n;i++)
    cin >> C1[i].x >> C1[i].y;
    Convex(C1,n);
    cin >> m;
    for(ll i=1;i<=m;i++)
    cin >> C2[i].x >> C2[i].y;
    Convex(C2,m);
    cin >> k;
    for(ll i=1;i<=k;i++)
    cin >> C3[i].x >> C3[i].y;
    Convex(C3,k);
    Minkowski(C1,C2,n,m,A,totA),Convex(A,totA);
    Minkowski(A,C3,totA,k,A,totA),Convex(A,totA);
    A[0]=A[1];for(ll i=totA;i>=1;i--)A[i]=A[i]-A[0];
    cin >> q;
    while(q--)
    {
        Node tmp;cin >> tmp.x >> tmp.y;
        cout << (in(tmp*3-A[0],A,totA)?"YES":"NO") << endl;
    }
    return 0;
}
```

7 离线算法

7.1 整体二分

```
void solve(int l,int r,vector<point>p)
{
    if(l==r)
    {
        for(auto o:p)
        if(cmp(calc(l,o.x),o.y)==1)
        ans[l].pb(o.id);
        return;
    }
    vector<point>p1,p2;
    del(1),cnt=0,rt=0;
    for(int i=l;i<=mid;i++)
    update(rt,-inf,inf,i);
    for(auto o:p)
    cmp(calc(query(rt,-inf,inf,o.x),o.x),o.y)==1?p1.pb(o):p2.pb(o);
    solve(l,mid,p1),solve(mid+1,r,p2);
}
```

7.2 回滚莫队

```
#include <bits/stdc++.h>
using namespace std;
#define gc getchar
inline int rd(){
    int x=0,fl=1;char ch=gc();
    for (;ch<48||ch>57;ch=gc())if(ch=='-')fl=-1;
    for (;48<=ch&&ch<=57;ch=gc())x=x*10+(ch^48);
    return x*fl;
}
typedef long long ll;
#define fr(i, l, r) for (int i=(l); i<=(r); ++i)
#define fo(i, l, r) for (int i=(l); i<(r); ++i)
struct Query{
    int l,r,id;
}q[5002];
ll ans[5002],now;
```

```
#define fi first
#define se second
vector<pair<int,int> > all;
const int N=1e6+5, M=1e9+7, PHI=M-1, B=31624, g=5,T=(1<<15)-1;
int n,m,a[N],b[N],f[B+5],p[4000],pre[N],nxt[N];
bool vis[B+5];
unordered_map<int,int>mp;
void init(){
    ll pw=1, tmp=1;
    int lim=10000;
    fr(i,1, lim) pw=pw*5%M;
    fr(i,1, int(1e5)) tmp=tmp*pw%M,mp[tmp]=i*lim;
}
int BSGS(ll x){
    for (int i=0;; x=x*g%M, ++i)
    if (mp.count(x)) return mp[x]-i;
}
ll dlog(int x){
    if (x<=B) return f[x];
    int q=M/x, r=M%x;
    return r*2<=x ? dlog(r)+(M>>1)-f[q] : dlog(x-r)-f[q+1];
}
struct FPOW{
    int bs[T+3], gs[T+3];
    void init(){
        *bs=*gs=1;
        fr(i,1,1<<15) bs[i]=1ll*bs[i-1]*g%M;
        fr(i,1,1<<15) gs[i]=1ll*gs[i-1]*bs[1<<15]%M;
    }
    int operator()(const int &i){return 1ll*bs[i&T]*gs[i>>15]%M;}
} G;
void rebuild(int L, int R){
    fr(i, L, R) all.push_back({a[i],i});
    auto ptr=all.end()-(R-L+1);
    sort(ptr, all.end());
    inplace_merge(all.begin(), ptr, all.end());
    int lst=0;
    for (auto p:all){
        pre[p.se]=lst;
```

```
        nxt[lst]=p.se;
        lst=p.se;
    }
    pre[0]=all.back().se;
    nxt[all.back().se]=0;
    now=1;
    fo(i,1, all.size()) now=now*G(1ll*a[all[i].se]*b[all[i-1].se]%PHI)%M;
}
void del(int i){
    if (pre[i])
    if (nxt[i]) now=now*G(1ll*b[pre[i]]*(a[nxt[i]]-a[i])%PHI)%M;
    else now=now*G(PHI-1ll*b[pre[i]]*a[i]%PHI)%M;
    if (nxt[i]) now=now*G(PHI-1ll*b[i]*a[nxt[i]]%PHI)%M;
    pre[nxt[i]]=pre[i];
    nxt[pre[i]]=nxt[i];
}
void add(int i){pre[nxt[i]]=nxt[pre[i]]=i;}
int main(){
    freopen("input.txt","r",stdin);
    init();
    f[1]=0;
    fr(i,2, B){
        if (!vis[i]) p[++p[0]]=i, f[i]=BSGS(i);
        for (int j=1,t; j<=p[0] && (t=i*p[j])<=B; ++j){
            vis[t]=1;
            f[t]=(f[i]+f[p[j]])%PHI;
            if (!(i%p[j])) break;
        }
    }
    n=rd(),m=rd();
    fr(i,1,n) b[i]=(dlog(a[i]=rd())%PHI+PHI)%PHI;
    fr(i,1,m) q[i].l=rd(),q[i].r=rd(),q[i].id=i;
    int bls=max(1, int(n/sqrt(m+1)));
    sort(q+1, q+m+1, [&](Query &a,Query &b){
        if(a.l/bls!=b.l/bls) return a.l/bls>b.l/bls;
        return a.r>b.r;
    });
    G.init();
    for (int L=n/bls*bls,R=n,pt=1;R;R=L-1,L-=bls){
```

```
        L=max(L,1);
        rebuild(L, R);
        int l=L,r=n;
        for (;pt<=m && q[pt].l>=L;++pt){
            while (r>q[pt].r) del(r--);
            ll tmp=now;
            while (l<q[pt].l) del(l++);
            ans[q[pt].id]=now;
            while (l>L) add(--l);
            now=tmp;
        }
    }
    fr(i,1,m) printf("%lld\n", ans[i]);
}
```

8 其他

8.1 pbds 平衡树

```
#include<ext/pb_ds/assoc_container.hpp> // 使用优先队列不需要
#include<ext/pb_ds/hash_policy.hpp> // 使用哈希表需要
#include <ext/pb_ds/tree_policy.hpp>
#include<bits/extc++.h> // 包含 pd_ds 和 rope
using namespace __gnu_pbds;
using namespace std;
#define int long long
typedef tree<int,null_type,less<int>,rb_tree_tag,
   tree_order_statistics_node_update>Set;
// 如果低版本编译器 null_type 也可能是 null_mapped_type
Set rbt, b;
signed main() {
    int n = read();
    for (int i = 1; i <= n; i++) {
        int opt = read(), x = read();
        if (opt == 1) rbt.insert((x << 20) + i); // 插入 x
        else if (opt == 2) rbt.erase(rbt.lower_bound(x << 20));
        // 删除 x，如果没有这个数，则什么都不会做
        else if (opt == 3) printf("%lld\n", rbt.order_of_key(x << 20) + 1);
        // 查询一个数 x 的排名，即返回比这个数小的个数，原树中没这个数也可查询
        else if (opt == 4) printf("%lld\n", (*rbt.find_by_order(x - 1)) >> 20);
        //查询排名为 x 的数，但树里排名以 0 开始编号，所以要查询 x - 1
        else if (opt == 5)
        printf("%lld\n", ((*--rbt.upper_bound(x << 20))) >> 20);
        else if (opt == 6)
        printf("%lld\n", (*rbt.upper_bound((x + 1) << 20)) >> 20);
    }
    // 因为这里的树会自动去重，所以以 (x << 20) + i 的方式存储
    // 如果定义中写的是 less<int>，lower_bound 查找第一个大于等于 x 的数
    // 且原树中没 x 也可查询。
    // 如果定义中写的是 greater<int>，lower_bound 查找第一个小于等于 x 的数
    rbt.join(b); // b 并入 rbt，前提是两棵树的 key 的取值范围不相交
    //即两棵树里面没有相同的数
    int v; rbt.split(v, b); //小于等于 v 的元素属于 rbt，其余的属于 b
}
```

8.2 pbds 哈希表

```
#include<ext/pb_ds/assoc_container.hpp> // 使用优先队列不需要
#include<ext/pb_ds/hash_policy.hpp> // 使用哈希表需要
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
using namespace std;
#define int long long
cc_hash_table<int,int> cc;//链表
gp_hash_table<int,int> gp;//开放寻址，gp 效率高
signed main() {
    gp.clear(); // 清空操作
    gp.erase(1);
    gp[1] = 0; // 一个新映射载入
    gp.insert({1,2}); // {key, mapped}，返回 bool
    int x; auto at = gp.find(x); // O(1)，找不到返回 end()
    // 用来替换 unorded_map
}
```

8.3 pbds trie

```
#include<ext/pb_ds/assoc_container.hpp> // 使用优先队列不需要
#include<ext/pb_ds/hash_policy.hpp> // 使用哈希表需要
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
using namespace std;
#define int long long
__gnu_pbds::trie<string,__gnu_pbds::null_type,
__gnu_pbds::trie_string_access_traits<>,
__gnu_pbds::pat_trie_tag,
__gnu_pbds::trie_prefix_search_node_update>tr, b;
char c[10010];
signed main() {
    tr.insert(c); tr.erase(c);
    tr.join(b); // 将 b 树加入并清空 b 树
    auto t = tr.find(c); // 找不到返回 end
    auto range = tr.prefix_range(c); // 返回一个 pair 表示前缀为 c 的迭代器范围
    for(auto it = range.first; it != range.second;it++);
}
```