Team notebook

BUET Berserkers

June 2, 2022

Contents

1 Onsite Template Hasan

1.1 Convex Hull Trick

```
/*****************************************/
/**
Fully Generalized implementation for
    Monotone slope , Arbitrary query
runtime insert() O(1)
        query() O(logn)
**/
class MonotoneCHT{
    deque<Line> Q;
    int type;
    void insertBack(Line nl){
        //handle parallel line insertion ,
            there cannot be more than
        //one parallel line to new line
            currently inside Q;
        if(!Q.empty() &&
            Q.back().parallel(nl)){
            if(type<2){
                if(Q.back().c>nl.c)
                    Q.pop_back();
                else
                    return;
            }
            else{
                if(Q.back().c<nl.c)
                    Q.pop_back();
                else
```

```
                return;
            }
        }
        while(Q.size()>1 &&
            Q.back().intersect(nl) <
            Q[Q.size()-2].intersect(nl))
            Q.pop_back();
        Q.push_back(nl);
    }
    void insertFront(Line nl){
        //handle parallel line insertion ,
            there cannot be more than one
        //parallel line to new line currently
            inside Q;
        if(!Q.empty() && Q[0].parallel(nl)){
            if(type<2){
                if(Q[0].c>nl.c)
                    Q.pop_front();
                else
                    return;
            }
            else{
                if(Q[0].c<nl.c)
                    Q.pop_front();
                else
                    return;
            }
        }
        while(Q.size()>1 &&
            Q[0].intersect(nl) >
            Q[1].intersect(nl))
            Q.pop_front();
        Q.push_front(nl);
    }
    pii bSearch(ll x){
        if(Q.size()==1 ||
            Q[0].intersect(Q[1]).first >= x)
            return {0 , 0};
        int l=1 , r=(int)Q.size()-1;
        while(r>l+1){
            int mid=(l+r)/2;
            if(Q[mid].intersect(Q[mid-1]).first
                < x)
                l=mid;
            else
                r=mid;
        }
        return {l , r};
    }
public:
    //slope increasing or decreasing
    //(not query point , query point is
        arbitrary) , querying for maximum or
        minimum
    MonotoneCHT(bool increasing , bool
        maximum){
        type=increasing;
        if(maximum)
            type|=2;
    }
    void insert(Line nl){
        if(type==3 || type==0)
            insertBack(nl);
        else
            insertFront(nl);
    }
    //if monotone query satisfied //not
        tested , although it should be ok
    ll fastQuery(ll x){
        #ifdef INCREASING_QUERY
        while(Q.size()>1 &&
            Q[0].intersect(Q[1]).first<x)
            Q.pop_front();
        return Q[0](x);
        #else
        while(Q.size()>1 && Q.back(
            ).intersect( Q[(int)Q.size()-2]
            ).first > x)
            Q.pop_back();
        return Q.back()(x);
        #endif
    }
    ll query(ll x){
        pii indx=bSearch(x);
        if(type<2)
            return min(Q[indx.first](x) ,
                Q[indx.second](x));
        return max(Q[indx.first](x) ,
            Q[indx.second](x));
    }
    void clear(){
        Q.clear();
    }
};
/*********************************************/

/***********set based implementation
source:https://github.com/kth-competitive-programmi
kactl/blob/main/content/data-structures/LineContain

**************************/
struct Line {
        mutable ll k , m , p;
        bool operator<(const Line& o) const {
            return k < o.k; }
        bool operator<(ll x) const { return p
            < x; }
};

struct LineContainer : multiset<Line ,
    less<>> {
        // (for doubles , use inf = 1/.0 ,
            div(a , b) = a/b)
        static const ll inf = LLONG_MAX;
        ll div(ll a , ll b) { // floored
            division
                return a / b - ((a ^ b) < 0 &&
                    a % b); }
        bool isect(iterator x , iterator y) {
```

```
            if (y == end()) return x->p =
                inf , 0;
            if (x->k == y->k) x->p = x->m
                > y->m ? inf : -inf;
            else x->p = div(y->m - x->m ,
                x->k - y->k);
            return x->p >= y->p;
        }
        void add(ll k , ll m) {
            auto z = insert({k , m , 0}) ,
                y = z++ , x = y;
            while (isect(y , z)) z =
                erase(z);
            if (x != begin() && isect(--x
                , y)) isect(x , y =
                erase(y));
            while ((y = x) != begin() &&
                (--x)->p >= y->p)
                    isect(x , erase(y));
        }
        ll query(ll x) {
            assert(!empty());
            auto l = *lower_bound(x);
            return l.k * x + l.m;
        }
};

/*********************************************/
```

1.2 Convex Hull

```
/**********convex Hull (no boundary
    points)***********/
#define xx first
#define yy second
template<class T>
bool cw(pair<T , T>& a , pair<T , T>&b ,
    pair<T , T>&c){
    return a.xx*(b.yy-c.yy) + b.xx *
        (c.yy-a.yy) + c.xx*(a.yy-b.yy)<0;
}
template<class T>
bool ccw(pair<T , T>& a , pair<T , T>&b ,
    pair<T , T>&c){
    return a.xx*(b.yy-c.yy) + b.xx *
        (c.yy-a.yy) + c.xx*(a.yy-b.yy)>0;
}
template<class T>
void convex_hull(vector<pair<T , T>
    >&a){//O(n) a is assumed to be sorted
    if(a.size()<2)return;
    pair<T , T> p1 , p2;
    vector<pair<T , T> > up , down;
    p1=a[0];
    p2=a.back();
    up.push_back(p1);
    down.push_back(p1);
    for(int i=1;i<(int)a.size();i++){
        if(i==a.size()-1||cw(p1 , a[i] , p2)){
            while(up.size()>=2 &&
                !cw(up[up.size()-2] ,
                up[up.size()-1] ,
                a[i]))up.pop_back();
            up.push_back(a[i]);
        }
        if(i==a.size()-1||ccw(p1 , a[i] ,
            p2)){
            while(down.size()>=2 &&
                !ccw(down[down.size()-2] ,
                down[down.size()-1] ,
                a[i]))down.pop_back();
            down.push_back(a[i]);
        }
    }
    a.clear();
    for(int i=0;i<(int)up.size();i++ )
        a.push_back(up[i]);
    for(int i=down.size()-2;i>0;i--)
        a.push_back(down[i]);
    return;
}
```

1.3 Ex GCD

```
//returns <u,v> such that nu+mv=(n,m)
template <class T>pair<T , T>
    extended_euclid(T n,T m){
    T rn_1,rn,sn_1,sn,tn_1,tn,tr,ts,tt,q;
    rn_1=n;sn_1=1;tn_1=0;
    rn=m;sn=0;tn=1;
    while(1){
        tr=rn_1%rn;
        q=(rn_1-tr)/rn;
        ts=sn_1-(q*sn);
        tt=tn_1-(q*tn);
        if(tr==0){
            return mp(sn,tn);
            //return (sn+m)%m;//n^-1 mod m
        }
        sn_1=sn;sn=ts;
        tn_1=tn;tn=tt;
        rn_1=rn;rn=tr;
    }
}
```

1.4 FFT

```
/**
Iterative Implementation of FFT and
    FFTanymod. Complexity: O(N log N)
1. Whenever possible remove leading zeros.
2. Custom Complex class may slightly improve
    performance.
```

```
3. Use pairfft to do two ffts of real
    vectors at once, slightly less accurate
than doing two ffts, but faster by about 30%.
4. FFT accuracy depends on answer. x <= 5e14
    (double), x <= 1e18(long double)
   where x = max(ans[i]) for FFT, and x =
       N*mod for anymod
   Author: anachor
**/

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

//typedef complex<double> CD;
struct CD {
    double x, y;
    CD(double x=0, double y=0) :x(x), y(y) {}
    CD operator+(const CD& o) { return
        {x+o.x, y+o.y};}
    CD operator-(const CD& o) { return
        {x-o.x, y-o.y};}
    CD operator*(const CD& o) { return
        {x*o.x-y*o.y, x*o.y+o.x*y};}
    void operator /= (double d) { x/=d;
        y/=d;}
    double real() {return x;}
    double imag() {return y;}
};
CD conj(const CD &c) {return CD(c.x, -c.y);}

typedef long long LL;
const double PI = acos(-1.0L);

namespace FFT {
    int N;
    vector<int> perm;
    vector<CD> wp[2];

    void precalculate(int n) {
        assert((n & (n-1)) == 0);
        N = n;
        perm = vector<int> (N, 0);
        for (int k=1; k<N; k<<=1) {
            for (int i=0; i<k; i++) {
                perm[i] <<= 1;
                perm[i+k] = 1 + perm[i];
            }
        }

        wp[0] = wp[1] = vector<CD>(N);
        for (int i=0; i<N; i++) {
            wp[0][i] = CD( cos(2*PI*i/N),
                sin(2*PI*i/N) );
            wp[1][i] = CD( cos(2*PI*i/N),
                -sin(2*PI*i/N) );
        }
    }

    void fft(vector<CD> &v, bool invert =
        false) {
        if (v.size() != perm.size())
            precalculate(v.size());

        for (int i=0; i<N; i++)
            if (i < perm[i])
                swap(v[i], v[perm[i]]);

        for (int len = 2; len <= N; len *= 2)
            {
            for (int i=0, d = N/len; i<N;
                i+=len) {
                for (int j=0, idx=0; j<len/2;
                    j++, idx += d) {
                    CD x = v[i+j];
                    CD y = wp[invert][idx] *
                        v[i+j+len/2];
                    v[i+j] = x+y;
                    v[i+j+len/2] = x-y;
                }
            }

        if (invert) {
            for (int i=0; i<N; i++) v[i]/=N;
        }
    }

     void pairfft(vector<CD> &a, vector<CD>
         &b, bool invert = false) {
        int N = a.size();
        vector<CD> p(N);
        for (int i=0; i<N; i++) p[i] = a[i] +
            b[i] * CD(0, 1);
        fft(p, invert);
        p.push_back(p[0]);

        for (int i=0; i<N; i++) {
            if (invert) {
                a[i] = CD(p[i].real(), 0);
                b[i] = CD(p[i].imag(), 0);
            }
            else {
                a[i] =
                    (p[i]+conj(p[N-i]))*CD(0.5,
                    0);
                b[i] =
                    (p[i]-conj(p[N-i]))*CD(0,
                    -0.5);
            }
        }
    }

    vector<LL> multiply(const vector<LL> &a,
        const vector<LL> &b) {
        int n = 1;
        while (n < a.size()+ b.size()) n<<=1;

        vector<CD> fa(a.begin(), a.end()),
            fb(b.begin(), b.end());
        fa.resize(n); fb.resize(n);
```

```
//       fft(fa); fft(fb);
        pairfft(fa, fb);
        for (int i=0; i<n; i++) fa[i] = fa[i]
            * fb[i];
        fft(fa, true);

        vector<LL> ans(n);
        for (int i=0; i<n; i++)  ans[i] =
            round(fa[i].real());
        return ans;
    }

    const int M = 1e9+7, B = sqrt(M)+1;
    vector<LL> anyMod(const vector<LL> &a,
        const vector<LL> &b) {
        int n = 1;
        while (n < a.size()+ b.size()) n<<=1;
        vector<CD> al(n), ar(n), bl(n), br(n);

        for (int i=0; i<a.size(); i++) al[i]
            = a[i]%M/B, ar[i] = a[i]%M%B;
        for (int i=0; i<b.size(); i++) bl[i]
            = b[i]%M/B, br[i] = b[i]%M%B;

        pairfft(al, ar); pairfft(bl, br);
//       fft(al); fft(ar); fft(bl); fft(br);

        for (int i=0; i<n; i++) {
            CD ll = (al[i] * bl[i]), lr =
                (al[i] * br[i]);
            CD rl = (ar[i] * bl[i]), rr =
                (ar[i] * br[i]);
            al[i] = ll; ar[i] = lr;
            bl[i] = rl; br[i] = rr;
        }

        pairfft(al, ar, true); pairfft(bl,
            br, true);
//       fft(al, true); fft(ar, true);
        fft(bl, true); fft(br, true);

        vector<LL> ans(n);
        for (int i=0; i<n; i++) {
            LL right = round(br[i].real()),
                left = round(al[i].real());;
            LL mid =
                round(round(bl[i].real()) +
                round(ar[i].real()));
            ans[i] = ((left%M)*B*B +
                (mid%M)*B + right)%M;
        }
        return ans;
    }
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n, m;
    cin>>n>>m;

    vector<LL> a(n), b(m);
    for (int i=0; i<n; i++) cin>>a[i];
    for (int i=0; i<m; i++) cin>>b[i];

    vector<LL> ans = FFT::anyMod(a, b);
    ans.resize(n+m-1);
    for (LL x: ans) cout<<x<<" ";
}
///ntt
namespace NTT {
    vector<int> perm, wp[2];
    const int mod = 998244353, G = 3; ///G
        is the primitive root of M
    int root, inv, N, invN;

    int power(int a, int p) {
        int ans = 1;
        while (p) {
            if (p & 1) ans = (1LL*ans*a)%mod;
            a = (1LL*a*a)%mod;
            p >>= 1;
        }
        return ans;
    }

    void precalculate(int n) {
        assert( (n&(n-1)) == 0 &&
            (mod-1)%n==0);
        N = n;
        invN = power(N, mod-2);
        perm = wp[0] = wp[1] = vector<int>(N);

        perm[0] = 0;
        for (int k=1; k<N; k<<=1)
            for (int i=0; i<k; i++) {
                perm[i] <<= 1;
                perm[i+k] = 1 + perm[i];
            }

        root = power(G, (mod-1)/N);
        inv = power(root, mod-2);
        wp[0][0]=wp[1][0]=1;

        for (int i=1; i<N; i++) {
            wp[0][i] =
                (wp[0][i-1]*1LL*root)%mod;
            wp[1][i] =
                (wp[1][i-1]*1LL*inv)%mod;
        }
    }

    void fft(vector<int> &v, bool invert =
        false) {
        if (v.size() != perm.size())
            precalculate(v.size());
        for (int i=0; i<N; i++)
```

```
        if (i < perm[i])
            swap(v[i], v[perm[i]]);

    for (int len = 2; len <= N; len *= 2)
        {
        for (int i=0, d = N/len; i<N;
            i+=len) {
            for (int j=0, idx=0; j<len/2;
                j++, idx += d) {
                int x = v[i+j];
                int y = (wp[invert][idx] *
                    1LL *
                    v[i+j+len/2])%mod;
                v[i+j] = (x+y>=mod ?
                    x+y-mod : x+y);
                v[i+j+len/2] = (x-y>=0 ?
                    x-y : x-y+mod);
            }
        }
    }
    if (invert) {
        for (int &x : v) x =
            (x*1LL*invN)%mod;
    }
}

vector<int> multiply(vector<int> a,
    vector<int> b) {
    int n = 1;
    while (n < a.size()+ b.size()) n<<=1;
    a.resize(n);
    b.resize(n);

    fft(a);
    fft(b);
    for (int i=0; i<n; i++) a[i] = (a[i]
        * 1LL * b[i])%mod;
    fft(a, true);
    return a;
}
```

```
};
const int M = 998244353;
int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n, m;
    cin>>n>>m;

    vector<int> a(n), b(m);
    for (int i=0; i<n; i++)  cin>>a[i];
    for (int i=0; i<m; i++)  cin>>b[i];
    vector<int> c = NTT::multiply(a, b);
    c.resize(n+m-1);
    for (int x: c) cout<<x<<" ";
}
```

1.5 Graph

```
ll g[500][500];
void floyedWarshal(int n){
    //g[i][i] should be zero
for(int k=0;k<n;k++){
for(int i=0;i<n;i++){
for(int j=0;j<n;j++){
if(g[i][k]<LLONG_MAX&&g[k][j]<LLONG_MAX)
g[i][j]=min(g[i][j],g[i][k]+g[k][j]);}}}}
```

1.6 LCT

```
///*********************************////////

/**Class based implementation**/
class LCT{
    Line* lct;
    int n;
    bool mntree;
    void updateMin(int v , int vl , int vr ,
        Line nl){
        if(lct[v].m==infl){
            lct[v]=nl;
            return;
        }
        if(vl==vr){
            if(lct[v](vl)>nl(vl))
                lct[v]=nl;
            return;
        }
        if(lct[v].m>nl.m)
            swap(lct[v] , nl);
        int mid=(vl+vr)/2;
        if(nl(mid)>lct[v](mid)){
            updateMin(2*v , vl , mid , nl);
        }
        else{
            swap(lct[v] , nl);
            updateMin(2*v+1 , mid+1 , vr ,
                nl);
        }
    }
    void updateMax(int v , int vl , int vr ,
        Line nl)
    {
        if(lct[v].m==infl){
            lct[v]=nl;
            return;
        }
        if(vl==vr){
            if(lct[v](vl)<nl(vl))
                lct[v]=nl;
            return;
        }
        if(lct[v].m>nl.m){
            swap(lct[v] , nl);
        }
```

```
    int mid=(vl+vr)/2;
    if(nl(mid)>lct[v](mid)){
        swap(lct[v] , nl);
        updateMax(2*v , vl , mid , nl);
    }
    else{
        updateMax(2*v+1 , mid+1 , vr ,
            nl);
    }
}
ll queryMin(int v , int vl , int vr ,
    int indx)
{
    if(vl==vr){
        if(lct[v].m==infl)
            return infl;
        return lct[v](vl);
    }
    int mid=(vl+vr)/2;
    ll res=infl;
    if(lct[v].m!=infl)
        res=lct[v](indx);
    if(indx<=mid)
        return min(res , queryMin(2*v ,
            vl , mid , indx));
    else
        return min(res , queryMin(2*v+1 ,
            mid+1 , vr , indx));
}
ll queryMax(int v , int vl , int vr ,
    int indx)
{
    if(vl==vr){
        if(lct[v].m==infl)
            return -infl;
        return lct[v](vl);
    }
    int mid=(vl+vr)/2;
    ll res=-infl;
    if(lct[v].m!=infl)
        res=lct[v](indx);
    if(indx<=mid)
        return max(res , queryMax(2*v ,
            vl , mid , indx));
    else
        return max(res , queryMax(2*v+1 ,
            mid+1 , vr , indx));
}
public:
    LCT(int sz , bool minTree=true){
        n=sz;
        mntree=minTree;
        lct=new Line[4*n];
    }
    void update(Line nl){
        if(mntree)
            updateMin(1 , 0 , n-1 , nl);
        else
            updateMax(1 , 0 , n-1 , nl);
    }
    ll query(int x){
        if(mntree)
            return queryMin(1 , 0 , n-1 , x);
        return queryMax(1 , 0 , n-1 , x);
    }
    ~LCT(){
        if(lct!=NULL)
            delete[] lct;
    }

};
/*****************************/


/** super cool dynamic lct****/

#define LL_MAX 0x7fffffffffffffff
struct Line{
    long long m , c;//mx+c
    Line(){
        m=c=LL_MAX;
    }
    Line(ll m , ll c):m(m) , c(c){}
    long long operator()(long long x){
        return m*x+c;
    }
    bool parallel(Line l){
        return m==l.m;
    }
    //assuming not parallel
    pair<long double , long double>
        intersect(Line l){
        long double x , y;
        x=(long double)(l.c-c)/(m-l.m);
        y=(long double)m*x+c;
        return {x , y};
    }
};
struct LctElement{
    Line l;
    LctElement *lft , *rht;
    LctElement(){
        lft=rht=NULL;
    }
};
class DynamicLCT{
    LctElement* root;
    bool minTree;
    void updateMin(LctElement* v , int vl ,
        int vr , Line nl){
        if(v->l.m==LL_MAX){
            v->l=nl;
            return;
        }
        if(vl==vr){
            if(nl(vl)<v->l(vl)){
                v->l=nl;
            }
            return;
        }
```

```
    if(v->l.m>nl.m){
        swap(v->l , nl);
    }
    int mid=(vl+vr)/2;
    if(nl(mid)>v->l(mid)){
        if(v->lft==NULL)
            v->lft=new LctElement();
        updateMin(v->lft , vl , mid , nl);
    }
    else{
        swap(v->l , nl);
        if(v->rht==NULL)
            v->rht=new LctElement();
        updateMin(v->rht , mid+1 , vr ,
            nl);
    }
}
void updateMax(LctElement* v , int vl ,
    int vr , Line nl){
    if(v->l.m==LL_MAX){
        v->l=nl;
        return;
    }
    if(vl==vr){
        if(nl(vl)>v->l(vl)){
            v->l=nl;
        }
        return;
    }
    if(v->l.m==nl.m){
        if(nl(vl)>v->l(vl)){
            v->l=nl;
        }
        return;
    }
    if(v->l.m>nl.m){
        swap(v->l , nl);
    }
    int mid=(vl+vr)/2;
    if(nl(mid)>v->l(mid)){
        swap(v->l , nl);
        if(v->lft==NULL)
            v->lft=new LctElement();
        updateMax(v->lft , vl , mid , nl);
    }
    else{
        if(v->rht==NULL)
            v->rht=new LctElement();
        updateMax(v->rht , mid+1 , vr ,
            nl);
    }
}

ll queryMin(LctElement* v , int vl , int
    vr , ll x){
    if(v==NULL)
        return LL_MAX;
    if(vl==vr){
        if(v->l.m==LL_MAX)
            return LL_MAX;
        return v->l(x);
    }
    ll res=LL_MAX;
    if(v->l.m!=LL_MAX)
        res=v->l(x);
    int mid=(vl+vr)/2;
    if(x<=mid)
        return min(res , queryMin(v->lft
            , vl , mid , x));
    else
        return min(res , queryMin(v->rht
            , mid+1 , vr , x));
}
ll queryMax(LctElement* v , int vl , int
    vr , ll x){
    if(v==NULL)
        return -LL_MAX;
    if(vl==vr){
        if(v->l.m==LL_MAX)
            return -LL_MAX;
        return v->l(x);
    }
    ll res=-LL_MAX;
    if(v->l.m!=LL_MAX)
        res=v->l(x);
    int mid=(vl+vr)/2;
    if(x<=mid)
        return max(res , queryMax(v->lft
            , vl , mid , x));
    else
        return max(res , queryMax(v->rht
            , mid+1 , vr , x));
}
void freeMemory(LctElement* node){
    if(node==NULL)
        return;
    freeMemory(node->lft);
    freeMemory(node->rht);
    delete node;
}
public:
    DynamicLCT(bool MinTree=true){
        root=new LctElement();
        minTree=MinTree;
    }
    void update(int vl , int vr , Line nl){
        if(minTree)
            updateMin(root , vl , vr , nl);
        else
            updateMax(root , vl , vr , nl);
    }
    ll query(int vl , int vr , ll x){
        if(minTree)
            return queryMin(root , vl , vr ,
                x);
        else
            return queryMax(root , vl , vr ,
                x);
    }
    ~DynamicLCT(){
```

```
        freeMemory(root);
    }
};
/*******************************/
```

1.7 LIS

```
int LIS(vector<int> & a){
    int n=a.size();
    int inf=1e9+1;
    vector<int> dp(n+1 , inf);
    dp[0]=-inf;
    int i;
    for(i=0;i<n;i++){
        int j=upper_bound(dp.begin() ,
            dp.end() , a[i])-dp.begin();
        if(dp[j-1]<a[i]&&a[i]<dp[j])
        {
            dp[j]=a[i];
        }
    }
    int ans=0;
    for(i=0;i<=n;i++){
        if(dp[i]<inf)
            ans=i;
    }
    return ans;
}

/*******************************************/
///out[i] contain length of longest
    increasing sequence
///ending at index i(0 based)
/*******************************************/
void LIS(vector<int> & a , vector<int>& out){
    int n=a.size();
    int inf=1e9+1;
    vector<int> dp(n+1 , INT_MAX);
    dp[0]=-INT_MAX;
    int i;
    for(i=0;i<n;i++){
        int j=upper_bound(dp.begin() ,
            dp.end() , a[i])-dp.begin();
        int k=lower_bound(dp.begin() ,
            dp.end() , a[i])-dp.begin();
        out[i]=k;
        if(dp[j-1]<a[i]&&a[i]<dp[j])
        {
            dp[j]=a[i];
        }
    }
}
```

1.8 Linear Diophantine Equation

```
///asuming the line equation in form ax+by=c
///assuming at least one of a or b is not
    zero ,
///check it before passing to this function
ll shiftx(ll x , ll refx , ll q)
{
    if(x==refx)
        return x;
    if(x<refx){
        ll d=refx-x;
        if(d%q)
        {
            d=d-d%q+q;
        }
        return x+d;}
    else{
        ll d=x-refx;
        if(d%q)
            d-=d%q;
        return x-d;}
}

//ax+by=c
ll solveDiophantine(ll a , ll b , ll c , ll
    x1 , ll x2 , ll y1 , ll y2)
{
    if(a==0){
        if(abs(c)%abs(b))
            return 0;
        ll y=c/b;
        if(y>=y1 && y<=y2)
            return (x2-x1+1);
        return 0;
    }
    if(b==0){
        if(abs(c)%abs(a))
            return 0;
        ll x=c/a;
        if(x>=x1&&x<=x2)
            return (y2-y1+1);
        return 0;
    }
    ll g=__gcd(abs(a) , abs(b));
    if(abs(c)%g)
        return 0;
    pair<ll , ll> sol=extended_euclid(abs(a)
        , abs(b));
    if(a<0){
        a=-a;
        b=-b;
        c=-c;}
    if(b<0){
        sol.second=-sol.second;}
    sol.first*=(c/g);
    sol.second*=(c/g);
    ll x1p , x2p;
    if(b<0)//slope is positive
    {
        x1p=floor(1.0*(c-b*y1)/a);
        if((c-b*y1)%a!=0)
            x1p++;
        x2p=floor(1.0*(c-b*y2)/a);
```

```
    }
    else//slope is negative
    {
        x2p=floor(1.0*(c-b*y1)/a);
        x1p=floor(1.0*(c-b*y2)/a);
        if((c-b*y2)%a!=0)
            x1p++;}
    ll x11 , x22;
    x11=max(x1 , x1p);
    x22=min(x2 , x2p);
    if(x22<x11)
        return 0;
    ll lx , rx;
    lx=shiftx(sol.first , x11 , abs(b/g));
    rx=shiftx(sol.first , x22 , abs(b/g));
    if(rx!=x22){
        rx-=abs(b/g);}
    return max((rx-lx)/(abs(b/g))+1 , 0ll);
}
```

1.9 Math

```
///***********faster linear
    implementation*****///
#define MAXP 100000
#define MAXN 10000000
vector<int> prime;
int lp[MAXN];//lowest prime that divide i
void LinearSeive(int n){
    memset(lp , 0 , sizeof(int)*n);
    for(int i=2;i<n;++i){
        if(lp[i]==0){
            lp[i]=i;
            prime.push_back(i);
        }
        for(int j=0;j<(int)prime.size() &&
            prime[j]<=lp[i] &&
            i*prime[j]<n;++j)
        {
            lp[i*prime[j]]=prime[j];
        }
    }
}
///*******mod should be <32 bit**///
ll modular_exp(ll a , ll b , ll mod){
    ll ans=1;
    while(b>0){
        if(b&1)ans=(ans*a)%mod;
        b=b>>1;
        a=(a*a)%mod;
    }
    return ans;
}
/*********finding all factor below n in
    O(nlogn)*********/
int *pfactor;
void build(int n){
    pfactor=new int[n+1];
    memset(pfactor , 0 , sizeof(int)*(n+1));
    int i , j;
    for(i=2;i<=n;i++){
        if(pfactor[i]==0){
            for(j=i;j<=n;j+=i){
                pfactor[j]=i;
            }
        }
    }
    return;
}
//pf and pfp must have size>log(n) returns
    number of prime factor
int get_p_factor(vector<int>& pf ,
    vector<int>& pfp , int n){
    int i=0;
    int j , pw;
    while(n>1){
        j=pfactor[n];
        pw=0;
        while(!(n%j)){
            n/=j;
            pw++;
        }
        pf[i]=j;
        pfp[i]=pw;
        i++;
    }
    return i;
}
int get_all_factor(vector<int>& pf ,
    vector<int>& pfp , int sz , vector<int>
    &vct){
    vct[0]=1;
    int i , j , k , l , r , s=1;
    for(i=0;i<sz;i++){
        l=0;
        for(j=0;j<pfp[i];j++){
            r=s;
            for(k=l;k<r;k++ , s++){
                vct[s]=(vct[k]*pf[i]);
            }
            l=r;
        }
    }
    return s;
}
/***********General multiplicative
    function***************/
int *mf;
int base_case(int p , int k){//base case for
    p^k
    return k+1;
}
void comp_mult_func(int n){
    mf=new int[n+1];
    memset(mf , -1 , sizeof(int)*(n+1));
    int i , k , k2;
    ll l;
    mf[1]=1;
```

```
    for(i=2;i<=n;i++){
        if(mf[i]==-1){
            for(l=i+i;l<=n;l+=i)mf[l]=-i;
            l=i;
            k=1;
            while(l<=(ll)n){
                mf[l]=base_case(i , k);
                k++;
                l*=i;
            }
        }
    }
    for(i=2;i<=n;i++){
        if(mf[i]<0){
            mf[i]=-mf[i];
            k=i;
            while(!(k%mf[i])){
                k/=mf[i];
            }
            mf[i]=mf[k]*mf[i/k];
        }
    }
    return;
}
/*********mobius function***********/
int mob[1000001];
void mobius(){
    memset(mob , -1 , sizeof(mob));
    mob[1]=1;
    for(int i=2;i<1000001;++i)
        for(int j=i*2;j<1000001;j+=i)
            mob[j]-=mob[i];
}
```

1.10 Minkowski sum

```
struct pt{
    ll x,y;
    pt operator+(const pt& p)const
    {return pt{x+p.x,y+p.y};}
    pt operator-(const pt& p)const
    {return pt{x-p.x,y-p.y};}
    ll cross(const pt& p)const{
        return x*p.y-y*p.x;}
};
void reorderPolygon(vector<pt>& P)//p must
    be ordered counterclockwise
{
    int pos=0;
    for(int i=1;i<(int)P.size();i++)
    {
        if(P[i].y < P[pos].y||(P[i].y ==
            P[pos].y && P[i].x < P[pos].x))
        {
            pos=i;
        }
    }
    rotate(P.begin(),P.begin()+pos,P.end());
}
vector<pt> minkowski(vector<pt> P,vector<pt>
    Q)//p and q is assumed to be sorted
{
    reorderPolygon(P);
    reorderPolygon(Q);
    P.push_back(P[0]);
    P.push_back(P[1]);
    Q.push_back(Q[0]);
    Q.push_back(Q[1]);
    vector<pt> res;
    int i=0,j=0;
    while(i<P.size()-2||j<Q.size()-2)
    {
        res.push_back(P[i]+Q[j]);
        auto cross = (P[i+1]-P[i]).cross(
            Q[j+1]-Q[j] );
        if(cross>=0)
            ++i;
        if(cross<=0)
            ++j;
    }
    return res;
}
```

1.11 Misc Geo

```
//number of point in a line segment
template <class T>
T npoint(pair<T , T> l , pair<T , T>
    r)//inclusive
{
    T x=abs(l.first - r.first);
    T y=abs(l.second - r.second);
    if(x==0){
        return y+1;
    }
    else if(y==0){
        return x+1;
    }
    T g=__gcd(x , y);
    return g+1;
}
//picks theorem s=i+b/2 - 1
//s area , i inside b boundary point
/// polar order
ll cross(pll a , pll b){
    return a.first * b.second - a.second *
        b.first;
}
int quarter(pll a){
    if(a.first >= 0){
        if(a.second >= 0)
            return 0;
        else
            return 3;}
    else{
        if(a.second >= 0)
```

```
            return 1;
        return 2;}
}
const long double pi=2 * acos(0.0);
bool polar_order(pair<pll , int> & a ,
    pair<pll , int>& b)
{
    if(quarter(a.first)==quarter(b.first)){
        return cross(a.first , b.first)>0; }
    else
        return quarter(a.first) <
            quarter(b.first);
}
//moderately tested on loj and cses
//crossing number from a point
//polygon[n]=polygon[0] where n= number of
    vertex in the original polygon
template<class T>
int cnPnPolygon(vector<pair<T , T> >&
    polygon , pair<T , T> pn)
{
    int n=(int)polygon.size();
    int cnt=0;
    for(int i=0;i<n - 1;i++)
    {
        if((polygon[i].second <= pn.second &&
            polygon[i+1].second > pn.second)
           ||(polygon[i].second > pn.second
              && polygon[i+1].second <=
              pn.second))
        {
            T tmp=(polygon[i+1].first -
                polygon[i].first) *
                (pn.second -
                polygon[i].second) -
                (pn.first - polygon[i].first)
                * (polygon[i+1].second -
                polygon[i].second);
            if(polygon[i+1].second -
                polygon[i].second<0)
                tmp= - tmp;
            if(tmp>0)
                cnt++;
        }
    }
    return cnt;
}
template<class T>
bool onBoundary(vector<pair<T , T> >&polygon
    , pair<T , T> pn)
{
    int n=(int)polygon.size();
    for(int i=0; i<n-1; i++){
        T tmp=(polygon[i].first - pn.first) *
            (polygon[i+1].second - pn.second)
                - (polygon[i+1].first -
                    pn.first) *
                    (polygon[i].second -
                    pn.second);
        if(tmp==0
            && ((pn.first >= polygon[i].first
                && pn.first <=
                polygon[i+1].first)||(pn.first
                <= polygon[i].first &&
                pn.first >=
                polygon[i+1].first))
            && ((pn.second >=
                polygon[i].second &&
                pn.second <=
                polygon[i+1].second)||(pn.second
                <= polygon[i].second &&
                pn.second >=
                polygon[i+1].second)))
        {
            return true;
        }
    }
    return false;
}
template <class T>
inline
T cross(pair<T , T> p0 , pair<T , T> p1 ,
    pair<T , T> p2)
{
    return (p1.first - p0.first) *
        (p2.second - p0.second) - (p1.second
        - p0.second) * (p2.first - p0.first);
}
//winding number from a point , tested on
    loj timus and cses
template<class T>
int wnPnPolygon(vector<pair<T , T> >&
    polygon , pair<T , T> pn)
{
    //assuming polygon[n]=polygon[0]
    int n=(int)polygon.size();
    int wn=0;
    for(int i=0;i<n - 1;i++){
        if(polygon[i].second <= pn.second){
            if(polygon[i+1].second>pn.second
                && cross(pn , polygon[i] ,
                polygon[i+1])>0){
                ++wn;
            }
        }
        else if(polygon[i+1].second <=
            pn.second && cross(pn ,
            polygon[i] , polygon[i+1])<0){
             --wn;
        }
    }
    return wn;
}
```

1.12 Point Inside Poly

```
template<class T>
T check_sign(vector2d<T> p1 , vector2d<T> p2
    , vector2d<T> p3){
    return (p1.x*p2.y + p2.x*p3.y + p3.x*p1.y
           -p1.y*p2.x - p2.y*p3.x -
               p3.y*p1.x);
}

template<class T>
ll check_sign(pair<T , T> p1 , pair<T , T>
    p2 , pair<T , T> p3){
    return (1ll*p1.first*p2.second + 1ll*
        p2.first *p3.second +
        1ll*p3.first*p1.second
           -1ll*p1.second*p2.first - 1ll*
               p2.second *p3.first - 1ll *
               p3.second*p1.first);
}
//points must be given in anticlockwise order
template<class T>
bool check_if_inside_polygon(vector<pair<T ,
    T>>& points , pair<T , T> qpoint){
    int n=points.size();

    if(check_sign(points[0] , points[1] ,
        qpoint)<0)
        return false;
    int l=1 , r=n-1;
    while(r>l+1)
    {
        int mid=(l+r)/2;
        if(check_sign(points[0] , points[mid]
            , qpoint)>=0)
            l=mid;
        else
            r=mid;
    }
    if(check_sign(points[0] , points[r] ,
        qpoint)>0||check_sign(points[l] ,
        points[r] , qpoint)<0)
        return false;
    /// add these line if u want strictly
        inside
//    if(l==1&&check_sign(points[0] ,
    points[1] , qpoint)==0)
//        return false;
//    if(r==n-1&&check_sign(points[0] ,
    points[r] , qpoint)==0)
//        return false;
//    if(check_sign(points[l] , points[r] ,
    qpoint)==0)
//        return false;
    return true;
}
```

1.13 Point

```
template<class T>
class vector2d{
private:
public:
    T x , y;
    vector2d(){x=y=0;}
    vector2d(T x , T y):x(x) , y(y){}
    vector2d operator+(T a){
        return {x+a , y+a};
    }
    vector2d operator+(vector2d v){return
        {x+v.x , y+v.y};}
    vector2d operator-(){return {-x , -y};}
    vector2d operator-(T a){return{x-a ,
        y-a};}
    vector2d operator-(vector2d v){return
        {x-v.x , y-v.y};}
    vector2d operator*(T a){return {x*a ,
        y*a};}
    vector2d operator*(vector2d v)//complex
        multiplication
{return {x*v.x-y*v.y , x*v.y+y*v.x};}
    vector2d& operator*=(vector2d
        v)//complex multiplication
    {   T tx=x*v.x-y*v.y;
        T ty=x*v.y+y*v.x;
        x=tx;
        y=ty;
        return *this;}
    vector2d& operator+=(vector2d v)
    {   x+=v.x;
        y+=v.y;
        return *this;}
    vector2d& operator-=(vector2d v){
        x-=v.x;
        y-=v.y;
        return *this;}
    vector2d& operator*=(T a){
        x*=a;
        y*=a;
        return *this;}
    vector2d& operator/=(T a){
        x/=a;
        y/=a;
        return *this;}
    T abs(){
        return sqrt(x*x+y*y);}
    T abs2(){
        return x*x+y*y;}
    vector2d operator/(T a){return {x/a ,
        y/a};}
    vector2d operator/(vector2d v)//complex
        division
    {
        return ((*this)*vector2d(v.x ,
            -v.y))/v.abs2();}
    T dot(vector2d v){
        return x*v.x+y*v.y;}
    vector2d<T> rotate(double
        angle)//counterclockwise rotation
    {   double cs=cos(angle);
```

```
        double sn=sin(angle);
        return {x*cs-y*sn , x*sn+y*cs};}
    vector2d<T> rot90()//counterclockwise 90
        degree rotation to work with integer
        points
    {return {-y , x};}
    T cross(vector2d b)
    {return x*b.y-y*b.x;}
};

//finds transform such that p1->p3 , p2->p4
//to find transform of another point r ,
    evaluate a*r+b;
template<class T>
pair<vector2d<T> , vector2d<T> >
    findLinearTransform
(vector2d<T> p1 , vector2d<T> p2 ,
    vector2d<T> p3 , vector2d<T> p4)
{
    vector2d<T> a=(p3-p4)/(p1-p2);
    vector2d<T> b=p3-a*p1;
    return {a , b};}
template<class T>
T abs(vector2d<T> v)
{
    return sqrt(v.x*v.x+v.y*v.y);}
template<class T>
T angle(vector2d<T> a , vector2d<T> b)
{
    return acos(a.dot(b)/(abs(a)*abs(b)));
}
template <class T>
ostream& operator<<(ostream& stream , const
    vector2d<T>& v)
{
    stream<<'('<<v.x<<' '<<v.y<<')';
    return stream;
}
template <class T>
istream& operator>>(istream& stream ,
    vector2d<T>& v)
{
    stream>>v.x>>v.y;
    return stream;
}
/*********************************************/
```

1.14 Primality Test

```
//able to compute a*b%mod a , b<2^63
ll moduloMultiplication(ll a , ll b , ll
    mod){
    ll res = 0;a %= mod;
    while (b){
        if(b&1)res=(res+a)%mod;
        a=(2*a)%mod;b>>=1;
    }
    return res;
}
ll modular_exp(ll a , ll b , ll mod){
    ll ans=1;
    while(b>0){
        if(b&1)ans=moduloMultiplication(ans ,
            a , mod);
        b=b>>1;
        a=moduloMultiplication(a , a , mod);
    }
    return ans;
}
bool farmatsTest(ll n , int it=10)
{
    if(n<4)return n==2||n==3;
    for(int i=0;i<it;i++)
    {
        ll a=2+rand()%(n-3);
        if(modular_exp(a , n-1 , n)!=1)
            return false;
    }
    return true;
}
/**********miller rabin********************/
using u64 = uint64_t;
using u128=__uint128_t;
u64 modular_exp(u64 a , u64 b , u64 mod){
    ll ans=1;
    while(b>0){
        if(b&1)ans=(__uint128_t)ans*a%mod;
        b=b>>1;
        a=(__uint128_t)a*a%mod;
    }
    return ans;
}
bool check_composite(u64 n , u64 a , u64 d ,
    int s)
{
    u64 x=modular_exp(a , d , n);
    if(x==1||x==n-1)
        return false;
    for(int r=1;r<s;r++)
    {
        x=(u128)x*x%n;
        if(x==n-1)
            return false;
    }
    return true;
}
bool millerRabin(u64 n)
{
    if(n<2)return false;
    u64 d=n-1;
    int r=0;
    while(!(d&1))
    {
        d=d>>1;
        r++;
    }
```

```
    for(int a:{2 , 3 , 5 , 7 , 11 , 13 , 17
        , 19 , 23 , 29 , 31 , 37})
    {
        if(n==a)
            return true;
        if(check_composite(n , a , d , r))
            return false;
    }
    return true;
}
```

1.15 Segment Tree

```
///lazy propagation
template<class T>
class SegTree{
    T* sgt;
    ///combine must clear out any
        unpropagated value
    T (*combine)(T , T);
    T (*propagate)(T to , T from , int);
    int n;
public:
    SegTree(int sz , T(*combine)(T , T) ,
        T(*propagate)(T , T , int) , T*
        data=NULL)
    {
        this->combine=combine;
        this->propagate=propagate;
        n=sz;
        sgt=new T[4*sz];
        if(data!=NULL)
            build(1 , 0 , n-1 , data);
    }
    void build(int v , int vl , int vr , T*
        data)
    {
        if(vl==vr)
        {
            sgt[v]=data[vl];
            return;
        }
        int mid=(vl+vr)/2;
        build(2*v , vl , mid , data);
        build(2*v+1 , mid+1 , vr , data);
        sgt[v]=combine(sgt[2*v] , sgt[2*v+1]);
    }
    void update(int v , int vl , int vr ,
        int l , int r , T unp)
    {
        if(vl==l&&vr==r)
        {
            sgt[v]=propagate(sgt[v] , unp ,
                r-l+1);
            return;
        }
        int mid=(vl+vr)/2;
        sgt[2*v]=propagate(sgt[2*v] , sgt[v]
            , mid-vl+1);
        sgt[2*v+1]=propagate(sgt[2*v+1] ,
            sgt[v] , vr-mid);
        if(r<=mid)
            update(2*v , vl , mid , l , r ,
                unp);
        else if(l>mid)
            update(2*v+1 , mid+1 , vr , l , r
                , unp);
        else
        {
            update(2*v , vl , mid , l , mid ,
                unp);
            update(2*v+1 , mid+1 , vr , mid+1
                , r , unp);
        }
        sgt[v]=combine(sgt[2*v] , sgt[2*v+1]);
    }
    void update(int l , int r , T unp)
    {
        update(1 , 0 , n-1 , l , r , unp);
    }
    T query(int v , int vl , int vr , int l
        , int r)
    {
        if(vl==l&&vr==r)
        {
            return sgt[v];
        }
        int mid=(vl+vr)/2;
        sgt[2*v]=propagate(sgt[2*v] , sgt[v]
            , mid-vl+1);
        sgt[2*v+1]=propagate(sgt[2*v+1] ,
            sgt[v] , vr-mid);
        sgt[v]=combine(sgt[2*v] , sgt[2*v+1]);
        if(r<=mid)
            return query(2*v , vl , mid , l ,
                r);
        else if(l>mid)
            return query(2*v+1 , mid+1 , vr ,
                l , r);
        else
            return combine(query(2*v , vl ,
                mid , l , mid) , query(2*v+1
                , mid+1 , vr , mid+1 , r));
    }
    T query(int l , int r)
    {
        return query(1 , 0 , n-1 , l , r);
    }
    ~SegTree()
    {
        if(n!=0&&sgt!=NULL)
            delete[] sgt;
    }
};
```

1.16 Trianglulation Ear Cliping

```
///O(n^3) v bad brute force implementation ,
    implement better algorithm later
template<class T>
int area(pair<T , T>& p1 , pair<T , T>& p2 ,
    pair<T , T>& p3){
    return (p1.first * p2.second + p2.first
        * p3.second + p3.first * p1.second
            -p1.second * p2.first-p2.second *
                p3.first-p3.second *
                p1.first);
}
template<class T>
bool inside(pair<T , T>& a , pair<T , T>& b
    , pair<T , T>& c , pair<T , T>& p)
{
    int ar=abs(area(a , b , c));
    int t=abs(area(a , b , p)) + abs(area(b
        , c , p)) + abs(area(c , a , p));
    return ar==t;
}
template<class T>
void triangulate(vector<pair<T , T> > p ,
    vector<pair<T , T> >&out)
{
    int pindx=0;
    if((int)p.size()<=3)
    {
        out.resize(p.size());
        copy(p.begin() , p.end() ,
            out.begin());
        return;
    }
    while(p.size()>3)
    {
        int n=(int)p.size();
        int i , j , k;
        for(i=0;i<n;i + + )
        {
            j=i + 1;
            k=i + 2;
            j=j>=n?j-n:j;
            k=k>=n?k-n:k;
            if(area(p[i] , p[j] , p[k])<0)
                continue;
            bool chk=true;
            for(int l=0;l<n;l + + )
            {
                if(l==i||l==j||l==k)
                    continue;
                if(inside(p[i] , p[j] , p[k] ,
                    p[l]))
                {
                    chk=false;
                    break;
                }}
            if(chk)
                break;
        }
        out[pindx++]=p[i];
        out[pindx++]=p[j];
        out[pindx++]=p[k];
        p.erase(p.begin() + j);
    }
    for(auto e:p)
        out[pindx + + ]=e;
}
```

1.17 aho

```
///aho corasick///
namespace AC{
    const int AlphabetSize = 26;
    struct Vertex{
        int next[AlphabetSize];
        int go[AlphabetSize];
        int leaf = -1;
        int p = -1;
        char pch;
        int elink = -1;//link to the maximum
            proper suffix of current string
        int vlink = -1;//link to the maximum
            proper suffix of current string
            which is also a string in the
            given set
        Vertex(int p = -1,char pch =
            '$'):p(p),pch(pch){
            memset(next,-1,sizeof(next));
            memset(go,-1,sizeof(go));}
        void reset(){
            memset(next,-1,sizeof(next));
            memset(go,-1,sizeof(go));
            leaf = p = elink = vlink = -1;}};
const int MaxLength = 1e6;
Vertex trie[MaxLength];
int curSize = 1;
//returns identifier of the string
    currently at the end vertex of s
int addString(const string& s,int
    id)//id should be unique identifier
    of the string
{   int v = 0;
    for(char ch:s){
        int c = ch-'a';
        if(trie[v].next[c] = = -1)
        {
            trie[curSize].p = v;
            trie[curSize].pch = ch;
            trie[v].next[c] = curSize++;
        }
        v = trie[v].next[c];
    }
    if(trie[v].leaf = = -1)
        trie[v].leaf = id;
    return trie[v].leaf;}
int addString(char* s,int id){
    int v = 0;
```

```
    while(*s! = '\0')
    {   int c = *s-'a';
        if(trie[v].next[c] = = -1)
        {
            trie[curSize].p = v;
            trie[curSize].pch = *s;
            trie[v].next[c] = curSize++;
        }
        v = trie[v].next[c];
        ++s;}
    if(trie[v].leaf = = -1)
        trie[v].leaf = id;
    return trie[v].leaf;
}
//should be called after adding all
    string to the trie
void constructAutomation(void)
{
    trie[0].elink = 0;
    trie[0].vlink = 0;//assume that empty
        string always exist in the set
    queue<int> Q;
    for(int i = 0;i<AlphabetSize;++i)
    {   trie[0].go[i] = (trie[0].next[i]
        = = -1?0:trie[0].next[i]);
        if(trie[0].next[i] = = -1)
        {trie[0].go[i] = 0;}
        else{
            trie[0].go[i] =
                trie[0].next[i];
            Q.push(trie[0].next[i]);}
    }
    while(!Q.empty())
    {   int v = Q.front();
        Q.pop();
        int p = trie[v].p;
        char pch = trie[v].pch;
        trie[v].elink = trie[
            trie[p].elink ].go[pch-'a'];
        if(trie[v].elink = = v)
            trie[v].elink = 0;
        if(trie[trie[v].elink].leaf! = -1)
            trie[v].vlink = trie[v].elink;
        else
            trie[v].vlink =
                trie[trie[v].elink].vlink;
        for(int i = 0;i<AlphabetSize;++i)
        {
            if(trie[v].next[i] = = -1)
            {
                trie[v].go[i] = trie[
                    trie[v].elink ].go[i];
            }
            else
            {
                trie[v].go[i] =
                    trie[v].next[i];
                Q.push(trie[v].next[i]);
            }}}}
int go(int v,char ch)//implementation of
    function that older code used
{return trie[v].go[ch-'a'];}
int getLink(int v)//implementation of
    function that older code used
{return trie[v].elink;}
int nMatch[MaxLength];
const int MaxNumberOfString = 1e6;
int ans[MaxNumberOfString];
//O(n+m^2) where m is the number of
    distinct length of string in the set
    added to the structure
void dfs(int v)
{   if(trie[v].leaf! = -1){
        int vv = v;
        while(vv>0){
            ans[trie[vv].leaf]+ =
                nMatch[v];
            vv = trie[vv].vlink;}}
    for(int i = 0;i<AlphabetSize;++i){
        if(trie[v].next[i] = = -1)
            continue;
        dfs(trie[v].next[i]);}}
//O(n)
void reverseBFS(int v)//should be faster
    than dfs,but in loj its runtime was
    slowar than about 100 ms than dfs,but
    was faster in cses
{   stack<int> st;
    queue<int> q;
    q.push(v);
    while(!q.empty())
    {
        v = q.front();
        q.pop();
        if(trie[v].leaf! = -1)
            st.push(v);
        for(int i = 0;i <
            AlphabetSize;++i)
        {
            if(trie[v].next[i]! = -1)
            {
                q.push(trie[v].next[i]);
            }}}
    while( ! st.empty())
    {   v = st.top();
        st.pop();
        ans[trie[v].leaf]+ = nMatch[v];
        nMatch[trie[v].vlink]+ =
            nMatch[v];}}
void computeAllMatch(const string& s){
    int v = 0;
    for(char ch:s)
    {
        v = trie[v].go[ch-'a'];
        if( trie[v].leaf ! = -1)
        {++nMatch[v];}
        else
            ++nMatch[trie[v].vlink];
    }
    //dfs(0);
```

```
        reverseBFS(0);
    }
    void reset(){
        for(int i = 0;i<curSize;++i)
        {
            trie[i].reset();
            nMatch[i] = ans[i] = 0;
        }
        curSize = 1;
    }
}
```

1.18 misc$_{m}ath$

```
int s = m;
while (s > 0) {
 ... you can use s ...
 s = (s-1) & m;
}
```

2 Onsite Template Shafi

2.1 BIT

```
#include<bits/stdc++.h>
using namespace std;
const int MaxIdx=1e+5;
int tree[MaxIdx+1];
int read(int idx) {int sum = 0;
  while (idx > 0) {sum += tree[idx];idx -=
      (idx & -idx);}
  return sum;}
void update(int idx, int val) {while (idx <=
    MaxIdx) {
    tree[idx] += val;idx += (idx & -idx);}}
int readSingle(int idx) {
  int sum = tree[idx];
  if (idx > 0) { int z = idx - (idx &
      -idx);idx--;
    while (idx != z) { sum -= tree[idx];idx
        -= (idx & -idx);}}
  return sum;}
int findG(int cumFre) {int idx = 0;int
    bitMask=(1<<16);
    while (bitMask != 0) {int tIdx = idx +
        bitMask;bitMask >>= 1;
    if (tIdx > MaxIdx)continue;
    if (cumFre >= tree[tIdx]) {idx =
        tIdx;cumFre -= tree[tIdx];}}
    if (cumFre != 0)return -1;else return
        idx;}
```

2.2 CRT

```
const ll siz=1000000;
ll moduli[siz],rem[siz];
pii egcd(ll a,ll b)
{
    ll r0=max(a,b),r1=min(a,b), s0=1, s1=0,
        t0=0, t1=1, q, tmp;
    while(r1){
        q=r0/r1;
        r0-=r1* q,s0-=s1* q,t0-=t1* q;
        tmp=r1,r1=r0,r0=tmp;
        tmp=s1,s1=s0,s0=tmp;
        tmp=t1,t1=t0,t0=tmp;
    }
    if(a<b)
        swap(s0,t0);
    return {s0,t0};
    //r0 is GCD
}
ll crt(ll n)
{
        ll n1, a1, m1, n2, a2, d, x; //x=n1*
            m1+a1=n2* m2+a2
    n1=moduli[0],a1=rem[0];
    for(ll i=1;i<n;i++){
                n2=moduli[i], a2=rem[i];
        d=a2-a1; //n1* m1-n2* m2=a2-a1=d
        m1=(egcd(n1, n2).fi * d/__gcd(n1,
            n2))%n2;
        x=n1* m1+a1;
        n1=n1/__gcd(n1, n2)* n2;
        a1=x%n1;
        if(a1<0)
            a1+=n1;
    }
    return a1;
}
```

2.3 Dijkstra

```
#define inf 1000000000
#define pii pair<int,int>
using namespace std;
int dist[10001],vis[10001];
vector<pii> graph[10001];
void dijkstra(int src,int n)
{priority_queue<pii, vector<pii>,
    greater<pii>> cur_dist;
for(int i=1;i<=n;i++) dist[i]=inf;
for(int i=1;i<=n;i++)
    cur_dist.push(make_pair(dist[i],i));
dist[src]=0;cur_dist.push(make_pair(0,src));
while(!cur_dist.empty()){int
    centre=cur_dist.top().second;
cur_dist.pop(); if(vis[centre]) continue;
for(auto itr: graph[centre])
    {if(dist[centre]+itr.second<
    dist[itr.first]){
dist[itr.first]= dist[centre]+ itr.second;
```

```
cur_dist.push( make_pair( dist[itr.first],
    itr.first));}} vis[centre]=1;}}
```

2.4 Dinic

```
struct FlowEdge {
    int v, u;
    long long cap, flow = 0;
    FlowEdge(int v, int u, long long cap) :
        v(v), u(u), cap(cap) {}
};

class Dinic {
    const long long flow_inf = 1e18;
    vector<FlowEdge> edges;
    vector<vector<int>> adj;
    int n, m = 0;
    int s, t;
    int *level, *ptr;
    queue<int> q;
    // 0 based index
    Dinic(int n, int s, int t) : n(n), s(s),
        t(t) {
        adj.resize(n);
        level=new int[n];
        ptr=new int[n];
    }
    void add_edge(int v, int u, long long
        cap) {
        edges.emplace_back(v, u, cap);
        edges.emplace_back(u, v, 0);
        adj[v].push_back(m);
        adj[u].push_back(m + 1);
        m += 2;
    }
    bool bfs() { //builds level graph
        while (!q.empty()) {
            int v = q.front();
            q.pop();
            for (int id : adj[v]) {
                if (edges[id].cap -
                    edges[id].flow < 1)
                    continue;
                if (level[edges[id].u] != -1)
                    continue;
                level[edges[id].u] = level[v]
                    + 1;
                q.push(edges[id].u);}}
        return level[t] != -1;}
    long long dfs(int v, long long pushed) {
        //pushes flow through level graph
        if (pushed == 0)
            return 0;
        if (v == t)
            return pushed;
        for (int& cid = ptr[v]; cid <
            (int)adj[v].size(); cid++) {
            int id = adj[v][cid];
            int u = edges[id].u;
            if (level[v] + 1 != level[u] ||
                edges[id].cap -
                edges[id].flow < 1)
                continue;
            long long tr = dfs(u, min(pushed,
                edges[id].cap -
                edges[id].flow));
            if (tr == 0)
                continue;
            edges[id].flow += tr;
            edges[id ^ 1].flow -= tr;
            return tr;}
        return 0;}
    long long flow() {
        long long f = 0;
        while (true) {
            memset(level, -1, sizeof level);
            level[s] = 0;
            q.push(s);
            if (!bfs()) //sink not reachable
                means end
                break;
            memset(ptr, 0, sizeof ptr);
            while (long long pushed = dfs(s,
                flow_inf)) {
                f += pushed;
            }
        }
        return f;}
};
```

2.5 HLD

```
#include<bits/stdc++.h>
using namespace std;
const int sz= 3000001;
int cpos= 0;
int tree[4*sz];
int head[sz], heavy[sz], pos[sz],
    parent[sz], depth[sz], val[sz];
vector<int> g[sz];
inline swap(int &a, int &b){a^= b, b^= a,
    a^= b;}
int dfs(int a)
{int max_size= 0, sum= 1;for(auto v: g[a]){
        if(v!= parent[a]){parent[v]=
            a;depth[v]= 1+depth[a];
                int siz= dfs(v);sum+= siz;
                if(siz> max_size){max_size=
                    siz;heavy[a]= v;}}}
    return sum;}
void decompose(int a, int h)
{head[a]= h;pos[a]= cpos++;
        if(heavy[a]) decompose(heavy[a], h);
        for(auto v: g[a]){if(v!= heavy[a] &&
            v!= parent[a])decompose(v, v);}}
int query(int a, int b, int n)
```

```
{int res= 0;
    while(head[a]!=
        head[b]){if(depth[head[a]]>
        depth[head[b]]) swap(a, b);
        res+= squery(1, 0, n-1, pos[head[b]],
            pos[b]);b= parent[head[b]];}
    if(depth[a]> depth[b]) swap(a, b);res+=
        squery(1, 0, n-1, pos[a], pos[b]);
    return res;}
```

2.6 KMP

```
int failure[1000001];
void build_failure(string &str){
int i,j,k;int cur;
memset(failure,0,sizeof failure);
failure[0]=0;failure[1]=0;
for(i=2;i<=str.length();i++){
cur=i-1;
while(cur!=0){
if(str[failure[cur]]==str[i-1]){
failure[i]=failure[cur]+1;break;}
cur=failure[cur];}}}
int kmp(string &text,string &pat){
int i,j,k;
int cur=0;int ocur=0;
for(i=0;i<text.length();i++){
if(cur==pat.length()){
ocur++;}
while(cur and text[i]!=pat[cur])
cur=failure[cur];
if(text[i]==pat[cur])cur++;}
if(cur==pat.length())
ocur++;
return ocur;}
```

2.7 LCA

```
#include<bits/stdc++.h>
using namespace std;
const int MAXN= 100001, LOG= 20;
vector<vector<int> > g;
int n;int bparent[MAXN][LOG], depth[MAXN],
    vis[MAXN];
void dfs(int a){vis[a]= 1;
for(auto v: g[a]){if(!vis[v]){
bparent[v][0]= a;depth[v]=
    1+depth[a];dfs(v);}}}
void build_ancestor(){dfs(1);
for(int i= 1;(1<<i)<n;i++)for(int j= 1;j<=
    n;j++)
bparent[j][i]=
    bparent[bparent[j][i-1]][i-1];}
int pth_ancestor(int a, int p)
{for(int i= 0;(1<<i)<= p;i++)if((1<<i)&p)a=
    bparent[a][i];return a;}
int lca(int u, int v){if(depth[v]>
    depth[u])v= pth_ancestor(v,
    depth[v]-depth[u]);
if(depth[u]> depth[v])u= pth_ancestor(u,
    depth[u]-depth[v]);
if(u= = v)return u;
for(int i= log2(n-1);i> = 0;i--){
if(bparent[u][i]!= bparent[v][i]){u=
    bparent[u][i];v= bparent[v][i];}}
return bparent[u][0];}
```

2.8 Manacher

```
int pal[100001];
string str_mod(string str)
{string mstr;mstr.push_back('#');
for(int
    i=0;i<str.length();i++){mstr.push_back(str[i]);
mstr.push_back('#');}
return mstr;}
void manacher(string str)
{int l,r;int lc=0,rc=0;int lt,rt;int mir=0;
string mstr=str_mod(str);pal[0]=0;
for(r=1;r<mstr.length();r++){
l=2*mir-r;if(l>=0 &&
    pal[l]<(rc-r))pal[r]=pal[l];
else {mir=r;pal[mir]=max(0,rc-mir);
for(rt=max(r+1,rc+1),lt=2*mir-rt;lt>=0 &&
    rt<mstr.size() &&
    mstr[lt]==mstr[rt];rt++,lt=2*mir-rt){
pal[mir]++;}rc=rt-1;lc=lt+1;}}}
```

2.9 MaxFlowMinCost

```
namespace mcmf {
    typedef long long F; typedef long long C;
    const F infF = 1e18; const C infC = 1e18;
    const int N = 5005;
    typedef pair<C, F> PCF;
    struct Edge {int frm, to; C cost; F cap,
        flow;};
    int n, s, t;
    vector<Edge> edges;
    vector<int> adj[N];
    C pi[N], dis[N];
    F fl[N];
    int prv[N], vis[N];
    void init(int nodes, int source, int
        sink) {
        n = nodes, s = source, t = sink;
        for (int i=0; i<n; i++) pi[i] = 0,
            adj[i].clear();
        edges.clear();
    }
    void addEdge(int u, int v, F cap,C cost)
        {
```

```
    edges.push_back({u, v, cost, cap, 0});
    edges.push_back({v, u, -cost, 0, 0});
    adj[u].push_back(edges.size()-2);
    adj[v].push_back(edges.size()-1);
}
bool SPFA() {
    for (int i=0; i<n; i++) {
        dis[i] = infC; fl[i] = 0;
        vis[i] = 0; prv[i] = -1;
    }
    queue<int> q;
    q.push(s);
    dis[s] = 0; fl[s] = infF; vis[s] = 1;
    while (!q.empty()) {
        int u = q.front(); q.pop();
        vis[u] = 0;
        for (int eid : adj[u]) {
            Edge &e = edges[eid];
            if (e.cap == e.flow) continue;

            if (dis[u] + e.cost <
                dis[e.to]) {
                dis[e.to] = dis[u] +
                    e.cost;
                fl[e.to] = min(fl[u],
                    e.cap - e.flow);
                prv[e.to] = eid^1;
                if (!vis[e.to])
                    q.push(e.to);
            }
        }
    }
    return fl[t] > 0;
}
PCF solveSPFA() {
    C cost = 0; F flow = 0;
    while (SPFA()) {
        C pathcost = dis[t];
        cost += pathcost*fl[t]; flow +=
            fl[t];
        for (int u=t, e=prv[u]; e!=-1;
            u=edges[e].to, e=prv[u]) {
            edges[e].flow -= fl[t];
            edges[e^1].flow += fl[t];
        }
    }
    return {cost, flow};
}
void normalize() {
    SPFA();
    for (int i=0; i<n; i++) pi[i] =
        dis[i];
}
bool Dijkstra() {
    for (int i=0; i<n; i++) {
        dis[i] = infC; fl[i] = 0;
        vis[i] = 0; prv[i] = -1;
    }
    priority_queue<pair<C, int>> pq;
    pq.emplace(0, s);
    dis[s] = 0; fl[s] = infF;
    while (!pq.empty()) {
        int u = pq.top().second; pq.pop();
        if (vis[u]) continue;
        vis[u] = 1;
        for (int eid : adj[u]) {
            Edge &e = edges[eid];
            if (vis[e.to] || e.cap ==
                e.flow) continue;
            C nw = dis[u] + e.cost -
                pi[e.to] + pi[u];
            if (nw < dis[e.to]) {
                dis[e.to] = nw;
                fl[e.to] = min(fl[u],
                    e.cap - e.flow);
                prv[e.to] = eid^1;
                pq.emplace(-dis[e.to],
                    e.to);
            }
        }
    }
    return fl[t] > 0;
}
PCF solveDijkstra() {
    normalize();
    C cost = 0; F flow = 0;
    while (Dijkstra()) {
        for (int i=0; i<n; i++)
            if (fl[i])   pi[i] += dis[i];
        C pathcost = pi[t]-pi[s];
        cost += pathcost*fl[t]; flow +=
            fl[t];

        for (int u=t, e=prv[u]; e!=-1;
            u=edges[e].to, e=prv[u]) {
            edges[e].flow -= fl[t];
            edges[e^1].flow += fl[t];
        }
    }
    return {cost, flow};
}
}
```

2.10 OOP String Hash

```
mt19937 rng(chrono:: steady_clock::
    now().time_since_epoch().count());
const ll MAXN= 1000001 , hp= 31+rng()%20 ,
    mod= 1000004023;
ll power[MAXN] , ipower[MAXN];
void power_build();
ll bigmod(ll a , ll b){ll ans= 1;
    while(b>0){if(b&1)ans= (ans* a)%mod;b=
        b>>1;a= (a* a)%mod;}
    return ans;}
void ipower_build();
class Hush
{
```

```
public:
    ll * hf;string str;
    Hush(string &str): str(str)
    {hf= new ll[str.length()];hf[0]=
        str[0]-'a'+1;
    for(ll i= 1;i<str.length();i++)hf[i]=
        (hf[i-1] +(str[i]-'a'+1)*
        power[i])%mod;}
    ll getHash(){return hf[str.length()-1];}
    ll getHash(ll i1 , ll i2){if(i1= =
        0)return hf[i2];
        return ((hf[i2]-hf[i1-1]+ mod)*
            ipower[i1])%mod;}};

unsigned seed = chrono:: system_clock::
    now().time_since_epoch().count();
linear_congruential_engine<uint_fast32_t ,
    1002517UL , 1001593UL , 2147483647UL>
    lcg(seed);
int primes_for_mod[]= {1000019353 ,
    1000001087 , 1000020353 , 1000003267 ,
    1000000439 , 1000018001 , 1000019569 ,
    1000020701 , 1000016929 , 1000007521 ,
1000007773 , 1000013323 , 1000018379 ,
    1000017203 , 1000006211 , 1000004693 ,
    1000013011 , 1000020829 , 1000011277 ,
    1000007147};
int primes_for_base[]= {1831 , 1061 , 5927 ,
    6689 , 7529 , 9719 , 3917 , 271 , 6029 ,
    6091 , 9719 , 2819 , 4877 , 9679 , 6373
    , 6101 , 1039 , 4591 , 5531};

struct PairHash
{
    template <class T1 , class T2>
    std::size_t operator() (const
        std::pair<T1 , T2> &pair) const
    {
        return std::hash<T1>()(pair.first) ^
            std::hash<T2>()(pair.second);
    }
};
```

2.11 Suffix array+LCP

```
#include<bits/stdc++.h>
using namespace std;
char str[1000002];
int c[1000002], temp[1000002];
vector<int> sa, csort[1000002];
int lcp[1000002];
void build_suffix_array(int n) ///O(nlogn)
{vector<pair<int, int>> ipos;
for(int i= 0;i<n;i++) ipos.push_back(
    make_pair(str[i], i));
sort(ipos.begin(), ipos.end());
for(auto itr: ipos) sa.push_back(itr.second);
c[sa[0]]= 0;
for(int i= 1;i<n;i++){ c[sa[i]]= c[sa[i-1]];
if(str[sa[i]]!= str[sa[i-1]])c[sa[i]]++;}
for(int k= 0;(1<<k)<= n;k++){ for(int i=
    0;i<n;i++){
sa[i]= (sa[i]-(1<<k)+n)%n;
csort[c[sa[i]]].push_back(sa[i]);}
sa.clear();
for(int i= 0;i<n;i++){for(auto itr: csort[i])
sa.push_back(itr);csort[i].clear();}
temp[sa[0]]= 0;
for(int i= 1;i<n;i++){temp[sa[i]]=
    temp[sa[i-1]];
if(!(c[sa[i]]= = c[sa[i-1]] &&
    c[(sa[i]+(1<<k))%n]= =
    c[(sa[i-1]+(1<<k))%n]))
temp[sa[i]]++;}
for(int i= 0;i<n;i++)c[i]= temp[i];}}
void kasai(int n, vector<int> &sa)
{int k= 0;vector<int> ran(n, 0);
for(int i= 0; i<n; i++) ran[sa[i]]= i;
for(int i= 0; i<n; i++, k?k--:0){
if(ran[i]= = n-1) {k= 0; continue;}
int j= sa[ran[i]+1];
while(i+k<n && j+k<n && str[i+k]== str[j+k])
    k++;lcp[ran[i]]= k;}}
```

2.12 cnt$_d$ec

```
int n;
const int sz=200001;
vector<int> tree[sz],ctree[sz];
int sub[sz],vis[sz],cparent[sz];
void dfs(int a,int p){
    sub[a]=1;
    for(auto v: tree[a]){
        if(v!=p && vis[v]==0){
            dfs(v,a);
            sub[a]+=sub[v];}}}
int findCentroid(int a,int p,int num){
    bool sig=true;
    while(sig){
    sig=false;
    for(auto v: tree[a]){
        if(v!=p && vis[v]==0 && 2*sub[v]>num){
            p=a;a=v;sig=true;
            break;}}}
    return a;}
int Decompose(int a,int p,int num){
    dfs(a,p);
    int centroid=findCentroid(a,p,num);
    vis[centroid]=1;
    for(auto v: tree[centroid]){
        if(vis[v]==0){
        if(sub[v]>sub[centroid])
ctree[centroid].pb(
    Decompose(v,centroid,num-sub[centroid]));
else
```

```
ctree[centroid].pb(
    Decompose(v,centroid,sub[v]));
cparent[ctree[centroid].back()]=centroid;}}
    return centroid;}
```

3 Onsite Template Tamim

3.1 Berlekamp

```
vector<ll> berlekampMassey(vector<ll> s) {
int n = sz(s), L = 0, m = 0;
vector<ll> C(n), B(n), T;
C[0] = B[0] = 1;
ll b = 1;
rep(i,0,n) { ++m;
ll d = s[i] % mod;
rep(j,1,L+1) d = (d + C[j] * s[i - j]) % mod;
if (!d) continue;
T = C; ll coef = d * modpow(b, mod-2) % mod;
rep(j,m,n) C[j] = (C[j] - coef * B[j - m]) %
    mod;
if (2 * L > i) continue;
L = i + 1 - L; B = T; b = d; m = 0;
}
C.resize(L + 1); C.erase(C.begin());
for (ll& x : C) x = (mod - x) % mod;
return C;
}
```

3.2 FWHT

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

const int N = 3e5 + 9, mod = 1e9 + 7;
int POW(long long n, long long k) {
    int ans = 1 % mod; n %= mod; if (n < 0)
        n += mod;
    while (k) {
        if (k & 1) ans = (long long) ans * n
            % mod;
        n = (long long) n * n % mod;
        k >>= 1;
    }
    return ans;
}
const int inv2 = (mod + 1) >> 1;
#define M (1 << 20)
#define OR 0
#define AND 1
#define XOR 2
struct FWHT{
    int P1[M], P2[M];
    void wt(int *a, int n, int flag = XOR) {
        if (n == 0) return;
        int m = n / 2;
        wt(a, m, flag); wt(a + m, m, flag);
        for (int i = 0; i < m; i++){
            int x = a[i], y = a[i + m];
            if (flag == OR) a[i] = x, a[i +
                m] = (x + y) % mod;
            if (flag == AND) a[i] = (x + y) %
                mod, a[i + m] = y;
            if (flag == XOR) a[i] = (x + y) %
                mod, a[i + m] = (x - y + mod)
                % mod;
        }
    }
    void iwt(int* a, int n, int flag = XOR) {
        if (n == 0) return;
        int m = n / 2;
        iwt(a, m, flag); iwt(a + m, m, flag);
        for (int i = 0; i < m; i++){
            int x = a[i], y = a[i + m];
            if (flag == OR) a[i] = x, a[i +
                m] = (y - x + mod) % mod;
            if (flag == AND) a[i] = (x - y +
                mod) % mod, a[i + m] = y;
            if (flag == XOR) a[i] = 1LL * (x
                + y) * inv2 % mod, a[i + m] =
                1LL * (x - y + mod) * inv2 %
                mod; // replace inv2 by >>1
                if not required
        }
    }
    // Finds the pairwise XOR / AND / OR of
        two vector A and B
    vector<int> multiply(int n, vector<int>
        A, vector<int> B, int flag = XOR) {
        assert(__builtin_popcount(n) == 1);
        A.resize(n); B.resize(n);
        for (int i = 0; i < n; i++) P1[i] =
            A[i];
        for (int i = 0; i < n; i++) P2[i] =
            B[i];
        wt(P1, n, flag); wt(P2, n, flag);
        for (int i = 0; i < n; i++) P1[i] =
            1LL * P1[i] * P2[i] % mod;
        iwt(P1, n, flag);
        return vector<int> (P1, P1 + n);
    }
    // Finds the presence of a value in all
        possible subset XOR / AND / OR of A
    vector<int> pow(int n, vector<int> A,
        long long k, int flag = XOR) {
        assert(__builtin_popcount(n) == 1);
        A.resize(n);
        for (int i = 0; i < n; i++) P1[i] =
            A[i];
        wt(P1, n, flag);
        for(int i = 0; i < n; i++) P1[i] =
            POW(P1[i], k);
        iwt(P1, n, flag);
        return vector<int> (P1, P1 + n);
```

```
    }
}t;
int32_t main() {
    int n; cin >> n;
    vector<int> a(M, 0);
    for(int i = 0; i < n; i++) {
        int k; cin >> k; a[k]++;
    }
    // Finds the pairwise XOR / AND / OR of
        two vector A and B
    vector<int> v2 = t.multiply(M,a,a,XOR);
    // Finds the presence of a value in all
        possible subset XOR / AND / OR of A
    vector<int> v = t.pow(M, a, n, AND);
    int ans = 1;
    for(int i = 1; i < M; i++) ans += v[i] >
        0;
    cout << ans << '\n';;
    return 0;
}
//
    https://csacademy.com/contest/archive/task/and-closure/statement/
```

3.3 Treap

```
// https://cses.fi/problemset/task/2072/
#include "bits/stdc++.h"
using namespace std;
#define pii pair<int,int>
#define pb push_back
#define mp make_pair
#define F first
#define S second

typedef long long LL;
mt19937
    rng(chrono::steady_clock::now().time_since_epoch().count());

typedef struct item * pitem;
struct item {
    int prior, value, cnt;
    LL sum;
    bool rev;

    item(int value):prior(rng()),
        value(value) {
        cnt = 0;
        rev = 0;
        sum = value;
        l = r = nullptr;
    }

    pitem l, r;
};

namespace Treap {
    int cnt (pitem it) {
        return it != nullptr? it->cnt : 0;
    }
    LL sum(pitem it) {
        return it != nullptr? it->sum : 0;
    }
    void upd_cnt (pitem it) {
        if (it!=nullptr) {
            it->cnt = cnt(it->l) + cnt(it->r)
                + 1;
            it->sum = sum(it->l) + sum(it->r)
                + it->value;
        }
    }
    void push (pitem it) {
        if (it != nullptr && it->rev == true)
            {
            it->rev = false;
            swap (it->l, it->r);
            if (it->l) it->l->rev ^= true;
            if (it->r) it->r->rev ^= true;
        }
    }
    void merge (pitem & t, pitem l, pitem r)
        {
        push (l);
        push (r);
        if (l==nullptr || r==nullptr)
            t = (l!=nullptr) ? l : r;
        else if (l->prior > r->prior)
            merge (l->r, l->r, r), t = l;
        else
            merge (r->l, l, r->l), t = r;
        upd_cnt (t);
    }
    void split (pitem t, pitem & l, pitem &
        r, int key, int add = 0) {
        if (t==nullptr) {
            l = r = nullptr;
            return;
        }
        push(t);
        int cur_key = add + cnt(t->l);

        if (key <= cur_key)
            split (t->l, l, t->l, key, add),
                r = t;
        else
            split (t->r, t->r, r, key, add +
                1 + cnt(t->l)), l = t;
        upd_cnt (t);
    }
    void reverse (pitem &t, int l, int r) {
        pitem t1, t2, t3;
        split (t, t1, t2, l); //< split t
            into t1 and t2 at position l,
            thus we have [..,l] and [l+1,..]
        split (t2, t2, t3, r-l+1); //< split
            t2 into t2 and t3
        // so we have now 3 tree t1,t2 and t3
        assert(t2 != NULL);
        t2->rev ^= true;
```

```
    merge (t, t1, t2); // merge t1 and t2
        into t
    // so now we have t and t3
    merge (t, t, t3); // merge t and t3
        into t
}
void cut (pitem &t, int l, int r) {
    pitem L, mid, R;
    split(t, L, mid, l );
    split(mid, mid, R, r - l+1);
    merge(t, L, R);
    merge(t, t, mid);
}
LL query (pitem &t, int l, int r) {
    pitem t1, t2, t3;
    split (t, t1, t2, l);
    split (t2, t2, t3, r-l+1);
    LL ans = t2->sum;
    merge (t, t1, t2);
    merge (t, t, t3);
    return ans;
}
void insert (pitem & t, int key, int
    value) {
    pitem x = new item(value);
    pitem L, R;
    split(t, L, R, key);
    merge(L, L, x);
    merge(t, L, R);
    upd_cnt(t);
}
int erase (pitem & t, int key) {
    assert(cnt(t) > key);
    pitem L, MID, R;
    split(t, L, MID, key);
    split(MID, MID, R, 1);
    merge(t, L, R);
        upd_cnt(t);
        int rt = MID->value;
        delete MID;
        return rt;
    }
    void output (pitem t, string &v) {
        if (t==nullptr) return;
        push (t);
        output (t->l, v);
        v.push_back(t->value);
        output (t->r, v);
    }
    void output2 (pitem t) {
        if (t==nullptr) return;
        push (t);
//        cout << "(";
        output2 (t->l);
        cout << (t->value) << " ";
        output2 (t->r);
//        cout << ")";
    }
}


int main(){
   // ifstream cin("test_input.txt");

    ios_base::sync_with_stdio(false);
    cin.tie(NULL); cout.tie(NULL);
    pitem root = nullptr;
    int n,q;
    cin>>n>>q;

    string s;
    cin>>s;
    for(int i=0;i<n;i++){
        Treap::insert(root,i,s[i]);
    }

    while(q--){
        int l,r;
        cin>>l>>r;
        l--;r--;
        Treap::cut(root,l,r);
    }
    string ans;
    Treap::output(root,ans);
    cout<<ans;


}
```

3.4 Z function

```
vector<int> z_function(string s) {
    int n = (int) s.length();
    vector<int> z(n);
    for (int i = 1, l = 0, r = 0; i < n;
        ++i) {
        if (i <= r)
            z[i] = min (r - i + 1, z[i - l]);
        while (i + z[i] < n && s[z[i]] == s[i
            + z[i]])
            ++z[i];
        if (i + z[i] - 1 > r)
            l = i, r = i + z[i] - 1;
    }
    return z;
}
```