Team notebook

Team: Dead_lock

December 23, 2024

Contents

1 BasicNumberTheory

```
void extendgcd(ll a, ll b, ll *v)
{
   if (b == 0)
   {
      v[0] = 1;
      v[1] = 0;
      v[2] = a;
      return;
   }
   extendgcd(b, a % b, v);
   ll x = v[1];
   v[1] = v[0] - v[1] * (a / b);
   v[0] = x;
   return;
} // pass an arry of size1 3
ll mminv(ll a, ll b)
{
   ll arr[3];
   extendgcd(a, b, arr);
   return arr[0];
} // for non prime b
ll mminvprime(ll a, ll b) { return expo(a, b - 2, b); }

int phi[LIM];
void calculatePhi() {
      rep(i,0,LIM) phi[i] = i&1 ? i : i/2;
      for (int i = 3; i < LIM; i += 2) if(phi[i] == i)
            for (int j = i; j < LIM; j += i)
         phi[j] -= phi[j] / i;
}
```

2 DataStructures

2.1 BIT

```
struct BIT{
ll N; vll bit;
void init(ll n){
N = n; bit.assign(n+1 , 0);
}
void add(int x, int k) {
for (; x <= N; x += x & -x) bit[x] += k;
}
int rsum(int l, int r) {
int res = 0;
for (int x = l - 1; x; x -= x & -x) res -= bit[x];
for (int x = r; x; x -= x & -x) res += bit[x];
return res;
}
ll find(ll val){
ll curr = 0 , prevsum = 0;
for(int i = log2(N) ; i >= 0 ; i --){
if(curr + (1 << i) < N && prevsum + bit[curr + (1 << i)] < val){
   prevsum += bit[curr + (1 << i)];
   curr += (1 << i);
}
}
return curr + 1;
}
void prints(void){
printv(bit);
}
};
```

2.2 CHT

```
struct pt {
   double x, y;
   bool operator == (pt const& t) const {
      return x == t.x && y == t.y;
   }
};

int orientation(pt a, pt b, pt c) {
   double v = a.x*(b.y-c.y)+b.x*(c.y-a.y)
   +c.x*(a.y-b.y);
   if (v < 0) return -1; // clockwise
   if (v > 0) return +1; // counter-clockwise
   return 0;
}

bool cw(pt a, pt b, pt c, bool include_collinear)
{
int o = orientation(a, b, c);
return o < 0 || (include_collinear && o == 0);
}
bool collinear(pt a, pt b, pt c)
 { return orientation(a, b, c) == 0; }
```

```
void convex_hull(vector<pt>& a, bool
   include_collinear
 = false) {
pt p0 = *min_element(a.begin(), a.end(),
[](pt a, pt b) {
return make_pair(a.y, a.x) < make_pair(b.y, b.x);
});
sort(a.begin(), a.end(), [&p0](const pt& a, const
   pt& b) {
   int o = orientation(p0, a, b);
   if (o == 0)
      return (p0.x-a.x)*(p0.x-a.x) +
         (p0.y-a.y)*(p0.y-a.y)
         < (p0.x-b.x)*(p0.x-b.x) +
            (p0.y-b.y)*(p0.y-b.y);
   return o < 0;
});
if (include_collinear) {
   int i = (int)a.size()-1;
   while (i >= 0 && collinear(p0, a[i],
      a.back())) i--;
   reverse(a.begin()+i+1, a.end());
}

vector<pt> st;
for (int i = 0; i < (int)a.size(); i++) {
   while (st.size() > 1 && !cw(st[st.size()-2],
    st.back(), a[i], include_collinear))
      st.pop_back();
   st.push_back(a[i]);
}
if (include_collinear == false && st.size() == 2
 && st[0] == st[1])
   st.pop_back();

a = st;
}
```

2.3 DSUrollback

```
int n , q;
const int maxN = 3e5+1;
vll sol;
struct DSU{
vector<pll> st[4*maxN];
vll p;
vll rank;
vector<pair<int&, int>> e;
vll op;
int ans = 0;

void init(int n){
   p.resize(n+1); rank.assign(n+1 , 1);
   for(int i = 0 ; i<= n ; i++) p[i] = i;
   ans = n;
}

int get(int u){
   if(u == p[u]){
      return u;
   }
   return get(p[u]);
}

void add(int u , int v){
   u = get(u); v = get(v);
   if(u == v){
      op.pb(0);
      return;
   }
   if(rank[u] > rank[v]) swap(u , v);
   ans--;
   op.pb(-1);
   e.pb({p[u] , p[u]});
   p[u] = v;
   e.pb({rank[v] , rank[v]});
   rank[v] += rank[u];
}
// update the range of queries from the index it
   starts to the index it ends [l,r] and total
   range will be [0,Q]
void upd(int node , int l , int r , int lx, int
   rx, pll p){
   if(lx >= r || rx <= l){
      return;
   }
   if(lx >= l && rx <= r){
      st[node].pb(p);
   }else{
      int mid = (lx+rx)/2;
      upd(2*node+1 , l , r , lx , mid , p);
      upd(2*node+2 , l , r , mid , rx , p);
   }
}
void undo(){
   if(!op.back()){
      op.pop_back();
      return;
   }else{
      ans++;
      op.pop_back();
      for(int i = 0 ; i <2 ; i ++){
         e.back().first = e.back().second;
         e.pop_back();
      }
   }
}

//dfs in the interval tree

void build(int node, int l , int r){
   for(auto it: st[node]){
      add(it.first , it.second);
   }
   if(r-l == 1){
      sol.pb(ans);
   }else{
      int mid = (l+r)/2;
      build(2*node+1 , l , mid);
      build(2*node+2 , mid ,r);
   }
   for(auto it: st[node]){
      undo();
   }
}
};
```

2.4 LineContainer

```
vector<pll> all_lines;

lld intersection(pll l1 , pll l2){
return ((lld)l1.second -
   l2.second)/(l2.first-l1.first);
}

bool can_delete(pll l1 , pll l2 , pll l3){

return intersection(l1 , l2) < intersection(l2 ,
   l3); // min
//return intersection(l1 , l2) > intersection(l2
   , l3); // max
}

void add_line(ll k , ll b){
pll nl = {k,b};
while(all_lines.size() >= 2 &&
   can_delete(all_lines[all_lines.size()-2] ,
   all_lines.back() , nl)){
   all_lines.pop_back();
}
all_lines.pb(nl);
}

int n;

ll vall(int pos , ll x){
return all_lines[pos].first*x +
   all_lines[pos].second;
}

ll compute_min(ll x){
ll l = -1;
ll r = all_lines.size()-1;
while(r-l > 1){
   ll mid = (l+r)/2;
   // vall(mid , x) < vall(mid+1 , x) // max
   if(vall(mid , x) > vall(mid+1 , x)){ // min
      l = mid;
   }else{
      r = mid;
   }
}
return vall(r , x);
}
```

2.5 LineContainerDynamic

```
//y =kx+m
// LineContainer hull;
// for min for(int i = 0 ; i < n ; i ++){
//   dp[i] = -hull.query(s[i]);
//   hull.add(-f[i] , -dp[i]);
//   }
// for max , no change

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;
        }
};
```

2.6 Mo's

```
const int N = 2e5 + 5;
const int Q = 2e5 + 5;
const int M = 1e6 + 5;
const int SZ = sqrt(N) + 1;
struct var{
        ll l , r , idx;
} qr[Q];
int n , q , a[N]; ll freq[M];
ll ans[Q];ll cur = 0;
bool comp(var &d1, var &d2){
  int b1 = d1.l / SZ;
  int b2 = d2.l / SZ;
  if(b1 != b2){
    return b1 < b2;
  }else{
    return (b1 & 1) ? d1.r < d2.r : d1.r > d2.r;
  }
}
inline void add(ll x){...}
inline void del(ll x){...}

void mo(){
  cin >> n >> q;
  for(int i = 1; i<= n ; i++)cin >> a[i];
  for(int i = 1; i<= q ; i++){
    cin >> qr[i].l >> qr[i].r;
    qr[i].idx = i;
  }
  sort(qr+1, qr+q+1 , comp);
  for(int i = 1; i<= q ; i ++){
    while(l < qr[i].l) remove(a[l++]);
    while(l > qr[i].l) add(a[--l]);
    while(r < qr[i].r) add(a[++r]);
    while(r > qr[i].r) remove(a[r--]);
    ans[qr[i].idx] = cur;
  }
}
```

2.7 RMQ

```
template<class T>
struct RMQ {
vector<vector<T>> jmp;
RMQ(const vector<T>& V) : jmp(1, V) {
for (int pw = 1, k = 1; pw * 2 <= sz(V); pw *= 2,
    ++k) {
jmp.emplace_back(sz(V) - pw * 2 + 1);
rep(j,0,sz(jmp[k]))
jmp[k][j] = min(jmp[k - 1][j], jmp[k - 1][j +
    pw]);
}
}
T query(int a, int b) {
assert(a < b); // or return in f i f a == b
int dep = 31 - __builtin_clz(b - a);
return min(jmp[dep][a], jmp[dep][b - (1 << dep)]);
}
};
```

2.8 SegTree

```
 // 0-based indexed segment tree , with last
    element of range of element included
// struct var{
//     ll x;
// };
struct seg_tree{
      ll size;
      vector<var> a;
      vector<ll> lazy;
      vector<bool> clazy;
      var invariant = INF;
      void init(ll n){
            size = 1;
            while(size < n) size*=2;
            a.assign(2*size , INF);
            lazy.assign(2*size ,0);
            clazy.assign(2*size , false);
      }
      var merge( var b , var c){
            // minimum:var a = min(b , c);
            return a;
      }
      void apply_operation(ll &a , ll b){
            //addition:a +=b;
            //assignment: a=b;
      }
```

```
void propagate(ll node , ll lx , ll rx){
      apply_operation(a[node] , lazy[node]);
      if(lx != rx){
            if(lazy[node]){
                  apply_operation(lazy[2*node+1]
                     , lazy[node]);
                  apply_operation(lazy[2*node+2]
                     , lazy[node]);
                  clazy[2*node+1] = true;
                  clazy[2*node+2] = true;

            }
      }
      lazy[node] = 0;
      clazy[node]=false;
}

void build( vll &arr , ll l , ll r , ll
   node){
      if(l == r){
            if(l < (ll)arr.size()){
                  //set value;
                  a[node] = arr[l];
            }
            return;
      }
      ll mid = (l+r)/2;
      build(arr , l , mid , 2*node+1 );
      build(arr , mid+1 , r , 2*node+2);
      a[node] = merge( a[2*node+1],
         a[2*node+2]);
}
```

```
void modify(ll l , ll r , ll v , ll node ,
   ll lx , ll rx){
      if(clazy[node]){
            propagate(node , lx , rx);
      }
      if(lx > r || l > rx) return;
      if(lx >= l && rx <= r){
            //addition:lazy[node]+=v
            lazy[node]+=v;
            clazy[node] = true;
            propagate(node , lx , rx);
            return;
      }
      ll mid= (lx + rx)/2;
      modify(l , r , v , 2*node+1 , lx ,
         mid);
      modify(l , r ,v , 2*node+2 , mid+1 ,
         rx);
      a[node] = merge( a[2*node+1] ,
         a[2*node+2]);
}
var get(ll i , ll node, ll lx , ll rx){
      if(clazy[node]){
            propagate(node , lx , rx);
      }
      if(rx == lx){
            return a[node];
      }
      ll mid = (lx+rx)/2;
      ll res;
      if(i <= mid){
            return get(i , 2*node+1 , lx ,
               mid);
```

```
      }else{
            return get(i , 2*node+2 , mid+1
               , rx);
      }
}
void set(ll l , ll r , ll v , ll node , ll
   pos ){
      if(clazy[node]){
            propagate(node , l , r);
      }
      if(l == r){
            //assignment:lazy[node]=v;
            clazy[node] = 1;
            lazy[node] += v;
            propagate(node , l , r);
            return;
      }
      ll mid = (l+r)/2;
      if( pos <= mid){
            set(l , mid , v , 2*node+1 ,
               pos);
      }else{
            set(mid+1 , r , v , 2*node+2 ,
               pos);
      }
      a[node]= merge(a[2*node+1],
         a[2*node+2]);
}
var calc(ll l , ll r , ll lx , ll rx , ll
   node){
      if(clazy[node]){
            propagate(node , lx , rx);
      }
```

```
            if(r < lx || l > rx){
                  return INF;
            }
            if( l <= lx && r >= rx){
                  return a[node];
            }
            ll mid = (lx+rx)/2;
            var sum1 = calc(l , r , lx , mid ,
               2*node+1 );
            var sum2 = calc(l , r , mid+1 , rx ,
               2*node+2);
            return merge(sum1 , sum2);
      }
       void build( vll &arr ){
            build(arr , 0 , size-1 , 0);
      }
      var calc(ll l , ll r){
            var ans = calc(l , r , 0 , size-1 ,
               0);
            return ans;
      }
      void set(ll i , ll v){
            set(0 , size-1 , v, 0 , i);
      }
      void modify(ll l , ll r , ll v){
            modify(l , r ,v , 0 , 0 , size-1);
      }
      var get(ll i){
            return get(i , 0 , 0, size-1);
      }
};
```

3 Graph

3.1 dinic

```
strsaauct Dinic {
      struct Edge {
            int to, rev;
            ll c, oc;
            ll flow() { return max(oc - c, 0LL);
               } // if you need flows
      };
      vi lvl, ptr, q;
      vector<vector<Edge>> adj;
      Dinic(int n) : lvl(n), ptr(n), q(n), adj(n)
         {}
      void addEdge(int a, int b, ll c, ll rcap =
         0) {
            adj[a].push_back({b, sz(adj[b]), c,
               c});
            adj[b].push_back({a, sz(adj[a]) - 1,
               rcap, rcap});
      }
      ll dfs(int v, int t, ll f) {
            if (v == t || !f) return f;
            for (int& i = ptr[v]; i < sz(adj[v]);
               i++) {
                  Edge& e = adj[v][i];
                  if (lvl[e.to] == lvl[v] + 1)
                        if (ll p = dfs(e.to, t,
                           min(f, e.c))) {
                              e.c -= p,
                                 adj[e.to][e.rev].c
                                 += p;
                              return p;
                        }
            }
            return 0;
      }
      ll calc(int s, int t) {
            ll flow = 0; q[0] = s;
            rep(L,0,31) do { // 'int L=30' maybe
               faster for random data
                  lvl = ptr = vi(sz(q));
                  int qi = 0, qe = lvl[s] = 1;
                  while (qi < qe && !lvl[t]) {
                        int v = q[qi++];
                        for (Edge e : adj[v])
                              if (!lvl[e.to] &&
                                 e.c >> (30 - L))
                                    q[qe++] =
                                       e.to,
                                       lvl[e.to]
                                       = lvl[v] +
                                       1;
                  }
                  while (ll p = dfs(s, t,
                     LLONG_MAX)) flow += p;
            } while (lvl[t]);
            return flow;
      }
      bool leftOfMinCut(int a) { return lvl[a] !=
         0; }
};
```

3.2 lca

```
int n, l;
vector<vector<int>> adj;

int timer;
vector<int> tin, tout;
vector<vector<int>> up;

void dfs(int v, int p)
{
tin[v] = ++timer;
up[v][0] = p;
for (int i = 1; i <= l; ++i)
   up[v][i] = up[up[v][i-1]][i-1];

for (int u : adj[v]) {
   if (u != p)
      dfs(u, v);
}

tout[v] = ++timer;
}

bool is_ancestor(int u, int v)
{
return tin[u] <= tin[v] && tout[u] >= tout[v];
}

int lca(int u, int v)
{
```

```
if (is_ancestor(u, v))
   return u;
if (is_ancestor(v, u))
   return v;
for (int i = l; i >= 0; --i) {
   if (!is_ancestor(up[u][i], v))
      u = up[u][i];
}
return up[u][0];
}
```

4 Maths

4.1 CRT

```
ll crt(ll a, ll m, ll b, ll n) {
if (n > m) swap(a, b), swap(m, n);
ll x, y, g = euclid(m, n, x, y);
assert((a - b) % g == 0); // e lse no so lution
x = (b - a) % n * x % n / g * m + a;
return x < 0 ? x + m*n/g : x;
```

4.2 FFT

```
typedef double ld;
typedef complex<ld> cd;
#define pvll pair<ll,vll>
// const int SIZE = 1<<19;
 //inv = 1 (ifft) inv = 0 (fft)
void fft(vector<cd> &a, bool inv){
   int N = (int) a.size();
   //bit permutation reversal ->
      (0,1,2,3,4,5,6,7) ->
      ([{0,4},{2,6}],[{1,5},{3,7}])
   for(int i = 1, j = 0; i < N; i++){
      int bit = N>>1;
      for(; j&bit; bit >>= 1)
         j ^= bit;
      j ^= bit;

      if(i < j)
         swap(a[i], a[j]);
   }

   // omega(n,k) = (2*k*pi*i)/n; n'th roots of
      unity
   for(int len = 2; len <= N; len <<= 1){
      ld theta = 2*PI / len * (inv ? -1 : 1);
      cd wlen(cos(theta), sin(theta));
      for(int i = 0; i < N; i += len){
         cd w(1);
         for(int j = 0; j < len / 2; j++){
            cd u = a[i+j], v = a[i+j+len/2] * w;
            a[i+j] = u + v;
            a[i+j+len/2] = u - v;
            w *= wlen;
         }
      }
   }

   if(inv)
```

```
      for(cd &z : a)
         z /= N;
}

//a = multiply(a,b) means a = a * b

vll multiply(vll a , vll b){
   ll n=1; vll v;
   while(n<((ll)a.size())+((ll)b.size())) n <<=1;
   vector<cd> fa(n), fb(n);
   for(int i = 0 ; i < n ; i ++) fa[i] = fb[i] =
      cd(0);
   for(int i = 0 ; i <a.size() ; i ++) fa[i] =
      cd(a[i]);
   for(int i = 0 ; i <b.size() ; i ++) fb[i] =
      cd(b[i]);
   fft(fa,false);
   fft(fb,false);
   for(int i = 0 ; i < n ; i ++){
      fa[i]=(fa[i]*fb[i]);
   }
   fft(fa,true);
   for (int i = 0; i < a.size() + b.size() - 1;
      ++i) {
      v.push_back((long long)(fa[i].real() +
         0.5));
   }
   return v;
}

//exponentiation can be done, by resizing the
   inital array after 5n,
//and doing fft transformation,
//then exponentiating the values of points and
   then inverse fft
```

4.3 fftconvmod

```
const int mod = 998244353;
typedef double ld;
typedef complex<double> cd;
typedef vector<double> vd;
void fft(vector<cd>& a) {
int n = sz(a), L = 31 - __builtin_clz(n);
static vector<complex<long double>> R(2, 1);
static vector<cd> rt(2, 1); // (^ 10% faster if
   double)
for (static int k = 2; k < n; k *= 2) {
R.resize(n); rt.resize(n);
auto x = polar(1.0L, acos(-1.0L) / k);
rep(i,k,2*k) rt[i] = R[i] = i&1 ? R[i/2] * x :
   R[i/2];
}
vi rev(n);
rep(i,0,n) rev[i] = (rev[i / 2] | (i & 1) << L) /
   2;
rep(i,0,n) if (i < rev[i]) swap(a[i], a[rev[i]]);
for (int k = 1; k < n; k *= 2)
for (int i = 0; i < n; i += 2 * k) rep(j,0,k) {
      // cd z = rt[j+k] * a[i+j+k]; // (25%
         faster if hand-rolled) /// include-line
      auto x = (double *)&rt[j+k], y = (double
         *)&a[i+j+k]; /// exclude-line
```

```
    cd z(x[0]*y[0] - x[1]*y[1], x[0]*y[1] +
      x[1]*y[0]); /// exclude-line
    a[i + j + k] = a[i + j] - z;
    a[i + j] += z;
}
}
vd conv(const vd& a, const vd& b) {
if (a.empty() || b.empty()) return {};
vd res(sz(a) + sz(b) - 1);
int L = 32 - __builtin_clz(sz(res)), n = 1 << L;
vector<cd> in(n), out(n);
copy(all(a), begin(in));
rep(i,0,sz(b)) in[i].imag(b[i]);
fft(in);
for (cd& x : in) x *= x;
rep(i,0,n) out[i] = in[-i & (n - 1)] -
  conj(in[i]);
fft(out);
rep(i,0,sz(res)) res[i] = imag(out[i]) / (4 * n);
return res;
}
const int M = mod;

vll convMod(const vll &a, const vll &b) {
if (a.empty() || b.empty()) return {};
vll res(a.size() + b.size() - 1);
int B=32-__builtin_clz(res.size()), n=1<<B,
  cut=int(sqrt(M));
vector<cd> L(n), R(n), outs(n), outl(n);
for(int i = 0 ; i < (int)a.size() ; i ++) L[i] =
  cd((int)a[i] / cut, (int)a[i] % cut);
for(int i = 0 ; i < (int)b.size() ; i ++) R[i] =
  cd((int)b[i] / cut, (int)b[i] % cut);
fft(L), fft(R);
for(int i = 0 ; i < n ; i ++) {
int j = -i & (n - 1);
outl[j] = (L[i] + conj(L[j])) * R[i] / (2.0 * n);
outs[j] = (L[i] - conj(L[j])) * R[i] / (2.0 * n)
  / 1i;
}
fft(outl), fft(outs);
for(int i = 0 ; i < (int)res.size() ; i ++){
ll av = ll(real(outl[i])+.5), cv =
  ll(imag(outs[i])+.5);
ll bv = ll(imag(outl[i])+.5) +
  ll(real(outs[i])+.5);
res[i] = ((av % M * cut + bv) % M * cut + cv) % M;
}
return res;
}

vll binpow(vll b,ll p){
vll ans=vll(1,1);
for(;p;p>>=1){
if(p&1)ans=convMod(ans,b);
b=convMod(b,b);
}
return ans;
}
```

4.4 Matrix

```
template<class T, int N> struct Matrix {
typedef Matrix M;
array<array<T, N>, N> d{};
M operator*(const M& m) const {
   M a;
   rep(i,0,N) rep(j,0,N)
      rep(k,0,N) a.d[i][j]+=d[i][k]*m.d[k][j];
   return a;
}
array<T, N> operator*(const array<T, N>&
   vec)const{
   array<T, N> ret{};
   rep(i,0,N) rep(j,0,N) ret[i]+=d[i][j]*vec[j];
   return ret;
}
M operator^(ll p) const {
   assert(p >= 0);
   M a, b(*this);
   rep(i,0,N) a.d[i][i] = 1;
   while (p) {
      if (p&1) a = a*b;
      b = b*b;
      p >>= 1;
   }
   return a;
}
};
```

4.5 Miller Robin

```
bool composite(ll m, ll a, ll d, ll s) {
   ll x = binpow(a, d, m);
   if(x == 1 || x == m-1) return 0;
   for(ll r = 1; r <= s; r++) {
      x = (u128)x * x % m;
      if(x==m-1) return 0;
   }
   return 1;
}
bool miller_rabin(ll x) {
   if(x < 2) return 0;
   if(x==2) return 1;
   if(x % 2 == 0) return 0;
   ll s = 0, d = x-1;
   while((d & 1)==0) {
      s++;
      d >>= 1;
   }
   for(ll a:{2,3,5,7,11,13,17,19,23,29,31,37}){
      if(x==a) return 1;
      if(composite(x, a, d, s)) return 0;
   }
   return 1;
}
ll f(ll x, ll n) {
   x = binpow(x, 2, n);
   return x++ == n ? 0 : x;
}
ll pollard(ll n) {
   if(n%2==0) return 2;
   for(ll i = 2;; i++) {
      ll x = i, y = f(x, n), d;
      while((d = __gcd(n + y - x, n)) == 1) {
         x = f(x, n), y = f(f(y, n), n);
      }
      if(d != n) return d;
   }
}
void ff(ll x) {
   if(x < 2) return;
   if(miller_rabin(x)) {
      ppf.insert(x);
      return;
   }
   else {
      ll d = pollard(x);
      ff(d);
      ff(x/d);
}}
```

4.6 nCr

```
struct nCr{
   ll maxx , md;
   vll fact, ifact;
   inline ll mul(ll a, ll b) { return a *1LL* b %
      md ;}
   ll power(ll a, ll n) {
      if(n == 0) return 1 ;
      int p = power(a, n/2) % md ;
      p = mul(p, p) ;
      return n & 1 ? mul(p, a) : p ;
   }
   int invMod(int a) {return power(a,md-2);}
   void pre() {
      fact[0] = 1 ;
      for(int i = 1;i< maxx;++i) fact[i] = mul(i,
         fact[i-1]) ;
      ifact[maxx-1] = invMod(fact[maxx-1]) ;
      for(int i = maxx-1 ; i>0 ;--i) ifact[i-1] =
         mul(ifact[i], i) ;
   }
   nCr(int _mxN, int _M) {
      maxx = _mxN + 1;
      md = _M ;
      fact.resize(maxx) ;
      ifact.resize(maxx) ;
      pre() ;
   }
   ll C(ll n, ll r) {
      if (n < r || r < 0 || n < 0) return 0;
      return mul(fact[n], mul(ifact[r],
         ifact[n-r])) ;
   }
};
//maxx N we need
//const int N = 100;
// initialise nCr struct
// nCr comb(N , mod);
```

4.7 NTT

```
const ll mod = 998244353;


namespace getPrimitive{
   ll powmod (ll a, ll b, ll p) {
```

```
   ll res = 1;
   while (b)
      if (b & 1)
         res = ll (res * 1ll * a % p), --b;
      else
         a = ll (a * 1ll * a % p), b >>= 1;
   return res;
}

// to generate primitive root
ll generator (ll p) {
   vector<ll> fact;
   ll phi = p-1, n = phi;
   for (ll i=2; i*i<=n; ++i)
      if (n % i == 0) {
         fact.push_back (i);
         while (n % i == 0)
            n /= i;
      }
   if (n > 1)
      fact.push_back (n);

   for (ll res=2; res<=p; ++res) {
      bool ok = true;
      for (size_t i=0; i<fact.size() && ok;
         ++i)
         ok &= powmod (res, phi / fact[i], p)
            != 1;
      if (ok) return res;
   }
   return -1;
}
```

```
};


namespace NTT {
   vll perm, wp[2];
   const ll mod = 998244353, G = 3; ///G is the
      primitive root of M(can be calculated using
      generator)
   ll root, inv, N, invN;

   ll power(ll a, ll p) {
      ll ans = 1;
      while (p) {
         if (p & 1) ans = (1LL*ans*a)%mod;
         a = (1LL*a*a)%mod;
         p >>= 1;
      }
      return ans;
   }
   // (mod-1)%n == 0 , condition for NTT,
      otherwise use CRT
   void precalculate(ll n) {
      assert( (n&(n-1)) == 0 && (mod-1)%n==0);
      N = n;
      invN = power(N, mod-2);
      perm = wp[0] = wp[1] = vector<ll>(N);

      perm[0] = 0;
      for (ll k=1; k<N; k<<=1)
         for (ll 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 (ll 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 ntt(vector<ll> &v, bool invert = false) {
      if (v.size() != perm.size())
         precalculate(v.size());
      for (ll i=0; i<N; i++)
         if (i < perm[i])
            swap(v[i], v[perm[i]]);

      for (ll len = 2; len <= N; len *= 2) {
         for (ll i=0, d = N/len; i<N; i+=len) {
            for (ll j=0, idx=0; j<len/2; j++, idx
               += d) {
               ll x = v[i+j];
               ll 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 (ll &x : v) x = (x*1LL*invN)%mod;
      }
   }

   vector<ll> multiply(vector<ll> a, vector<ll>
      b) {
      ll n = 1;
      while (n < a.size()+ b.size()) n<<=1;
      a.resize(n);
      b.resize(n);
      ntt(a);
      ntt(b);
      for (ll i=0; i<n; i++) a[i] = (a[i] * 1LL *
         b[i])%mod;
      ntt(a, true);
      return a;
   }

   //if polynomial exponentiation needed, instead
      resize the size of polynomial to atleast 5n
      , then exponentiate the coefficients and
      then inverse transform
};


vll binpow(vll b,ll p){
   vll ans=vll(1,1);
   while(p > 0){
      if(p&1){
         ans = NTT::multiply(ans,b);
      }
      cout << b.size() << endl;
      b = NTT::multiply(b,b);
      cout << b.size() << " " << count(all(b) ,
         0) << endl;
      p = p >> 1;
   }
   return ans;
}
```

4.8 SQRT

```
ll sqrt(ll a, ll p) {
a %= p; if (a < 0) a += p;
if (a == 0) return 0;
assert(modpow(a, (p-1)/2, p) == 1); // e lse no
   so lution
if (p % 4 == 3) return modpow(a, (p+1)/4, p);
// a^(n+3)/8 or 2^(n+3)/8  2^(n1)/4 works i f p %
   8 == 5
ll s = p - 1, n = 2;
int r = 0, m;
while (s % 2 == 0)
++r, s /= 2;
while (modpow(n, (p - 1) / 2, p) != p - 1) ++n;
ll x = modpow(a, (s + 1) / 2, p);
ll b = modpow(a, s, p), g = modpow(n, s, p);
for (;; r = m) {
ll t = b;
for (m = 0; m < r && t != 1; ++m)
t = t * t % p;
if (m == 0) return x;
ll gs = modpow(g, 1LL << (r - m - 1), p);
g = gs * gs % p;
x = x * gs % p;
b = b * g % p;
}
}
```

5 Runflag

```
code -r ~/.bashrc
source ~/.bashrc

run(){
   g++ $1.cpp -std=c++17 -O2 -wall -O $1.out &&
      ./$1.out< in.txt > out.txt && rm $1.out
}
```

6 Strings

6.1 KMP

```
vll kmp ( string &s){
   ll n = s.size();
   vll pi(n , 0);
   for(int i = 1 ; i < n ; i ++){
      ll j = pi[i-1];
      while(j > 0 && s[i] != s[j]){
         j = pi[j-1];
      }
      if(s[i] == s[j]) j++;
      pi[i] = j;
   }
   return pi;
}
vector<vll>aut;
void compute_automaton(string s){
   s += '#';
   vll pi = kmp(s);
   ll n = s.size();
   for(int i = 0 ; i < n ;i++ ){
      for(int j = 0 ; j < 26 ; j ++ ){
         if(i > 0 && s[i]!='a'+j){
            aut[i][j] = aut[pi[i-1]][j];
         }else{
            aut[i][j] = i + ('a'+j == s[i]);
         }
      }
   }
}
vector<int> zFunction(string &str){
   int n = str.length();
   vector<int>ans(n);int l = 0,r = 0;
   for(int i=1;i<n;i++){
      if(i <= r){
         ans[i] = min(ans[i-l],r-i+1);
      }
      while((i+ans[i])<n and (str[ans[i]] ==
         str[i+ans[i]])){
         ans[i]++;
      }
      if((i+ans[i]-1)>r){
         l = i;
         r = i+ans[i]-1;
      }
   }
   return ans;
}
```

6.2 Manachar

```
struct manacher{
   vector<int>p;
   void run_manacher(string s){
      int n = s.length();
      p.assign(n,1);
      int l=1,r=1;
      for(int i=1;i<n;i++){
         p[i] = max(0ll,min(r-i,p[l+r-i]));
         while(i+p[i]<n and i-p[i]>=0 and
            s[i+p[i]]==s[i-p[i]]){
            p[i]++;
         }

         if((i+p[i])>r){
            l = i-p[i];
            r = i+p[i];
         }
      }
   }
   void build(string s){
      string t;
      for(auto i:s){
         t.push_back('#');
      t.push_back(i);
    }
    t.push_back('#');
    run_manacher(t);
  }

  int get_longest(int index,bool odd){
    if(odd){
      return (p[(2*index)+1])-1;
    }else{
      return (p[2*(index+1)])-1;
    }
  }

  bool check_palindrome(int l,int r){
    int l1 = l,r1=r;
    l = (2*l+1);
    r = (2*r+1);
    int index = (l+r)>>1;
    if(p[index]-1>=(r1-l1+1)){
      return true;
    }else{
      return false;
    }

  }

}m;
```

7 template

```
mt19937
   rng(chrono::steady_clock::now().time_since_epoch()
.count());
ll uid(ll l, ll r) {return
   uniform_int_distribution<ll>(l, r)(rng);}
ios::sync_with_stdio(0);
cin.tie(0);
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
typedef tree<ll, null_type, less<ll>,
   rb_tree_tag,
   tree_order_statistics_node_update> pbds; //
   find_by_order, order_of_key
```

8 ZGeometry

8.1 Centroid

```
int nodes = 0;
int subtree[N], parentcentroid[N];
set<int> g[N];
void dfs(int u, int par)
{
nodes++;
subtree[u] = 1;
for(auto &it:g[u])
{
if(it == par)
continue;
dfs(it, u);
subtree[u] += subtree[it];
}
}

int centroid(int u, int par)
{
for(auto &it:g[u])
{
if(it == par)
continue;
if(subtree[u] > (nodes >> 1))
return centroid(u, it);
}
return u;
}

void decompose(int u, int par)
{
nodes = 0;
dfs(u, u);
int node = centroid(u, u);
parentcentroid[node] = par;
for(auto &it:g[node])
{
g[it].erase(node);
decompose(it, node);
}
}
```

8.2 Primitives

```
template <class T> int sgn(T x) { return (x > 0)
   - (x < 0); }
template<class T>
struct Point {
typedef Point P;
T x, y;
explicit Point(T x=0, T y=0) : x(x), y(y) {}
bool operator<(P p) const { return tie(x,y) <
   tie(p.x,p.y); }
bool operator==(P p) const { return
   tie(x,y)==tie(p.x,p.y); }
P operator+(P p) const { return P(x+p.x, y+p.y); }
P operator-(P p) const { return P(x-p.x, y-p.y); }
P operator*(T d) const { return P(x*d, y*d); }
P operator/(T d) const { return P(x/d, y/d); }
T dot(P p) const { return x*p.x + y*p.y; }
T cross(P p) const { return x*p.y - y*p.x; }
T cross(P a, P b) const { return
   (a-*this).cross(b-*this); }
T dist2() const { return x*x + y*y; }
double dist() const { return
   sqrt((double)dist2()); }
// angle to x-axis in interval [-pi, pi]
double angle() const { return atan2(y, x); }
P unit() const { return *this/dist(); } // makes
   dist()=1
P perp() const { return P(-y, x); } // rotates
   +90 degrees
P normal() const { return perp().unit(); }
// returns point rotated ’a’ radians ccw around
   the origin
P rotate(double a) const {
```

```cpp
    return P(x*cos(a)-y*sin(a),x*sin(a)+y*cos(a)); }
friend ostream& operator<<(ostream& os, P p) {
    return os << "(" << p.x << "," << p.y << ")"; }
};

template<class P>
double lineDist(const P& a, const P& b, const P& p) {
return (double)(b-a).cross(p-a)/(b-a).dist();
}

/*
Returns the shortest distance between point p and the line segment from point s to e.
Point<double> a, b(2,2), p(1,1);
bool onSegment = segDist(a,b,p) < 1e-10;
*/
typedef Point<double> P;
double segDist(P& s, P& e, P& p) {
if (s==e) return (p-s).dist();
auto d = (e-s).dist2(), t = min(d,max(.0,(p-s).dot(e-s)));
return ((p-s)*d-(e-s)*t).dist()/d;
}

template<class P> bool onSegment(P s, P e, P p) {
return p.cross(s, e) == 0 && (s - p).dot(e - p) <= 0;
}

/*
Usage:
vector<P> inter = segInter(s1,e1,s2,e2);
if (sz(inter)==1)
cout << "segments intersect at " << inter[0] << endl;
*/
template<class P> vector<P> segInter(P a, P b, P c, P d) {
auto oa = c.cross(d, a), ob = c.cross(d, b),
        oc = a.cross(b, c), od = a.cross(b, d);
// Checks if intersection is single non-endpoint point.
if (sgn(oa) * sgn(ob) < 0 && sgn(oc) * sgn(od) < 0)
    return {(a * ob - b * oa) / (ob - oa)};
set<P> s;
if (onSegment(c, d, a)) s.insert(a);
if (onSegment(c, d, b)) s.insert(b);
if (onSegment(a, b, c)) s.insert(c);
if (onSegment(a, b, d)) s.insert(d);
return {all(s)};
}

template<class P>
pair<int, P> lineInter(P s1, P e1, P s2, P e2) {
auto d = (e1 - s1).cross(e2 - s2);
if (d == 0) // if parallel
    return {-(s1.cross(e1, s2) == 0), P(0, 0)};
auto p = s2.cross(e1, e2), q = s2.cross(e2, s1);
return {1, (s1 * p + e1 * q) / d};
}
```

```
template<class P>
int sideOf(P s, P e, P p) { return sgn(s.cross(e,
   p)); }

/** Usage:
 *     bool left = sideOf(p1,p2,q)==1;
 * */
template<class P>
```

```
int sideOf(const P& s, const P& e, const P& p,
   double eps) {
auto a = (e-s).cross(p-s);
double l = (e-s).dist()*eps;
return (a > l) - (a < -l);
}
```