Team Notebook

April 5, 2023

Contents

1 BasicNumberTheory

```
ll gcd(ll a, ll b)
{
    if (b > a)
    {
        return gcd(b, a);
    }
    if (b == 0)
    {
        return a;
    }
    return gcd(b, a % b);
}
ll expo(ll a, ll b, ll mod)
{
    ll res = 1;
    while (b > 0)
    {
        if (b & 1)
            res = (res * a) % mod;
        a = (a * a) % mod;
        b = b >> 1;
    }
    return res;
}
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); }
bool revsort(ll a, ll b) { return a > b; }

ll combination(ll n, ll r, ll m, ll *fact, ll *ifact)
{
    ll val1 = fact[n];
    ll val2 = ifact[n - r];
    ll val3 = ifact[r];
    return (((val1 * val2) % m) * val3) % m;
}
void google(int t) { cout << "Case #" << t << ": "; }
vector<ll> sieve(int n)
{
    int *arr = new int[n + 1]();
    vector<ll> vect;
    for (int i = 2; i <= n; i++)
        if (arr[i] == 0)
        {
            vect.push_back(i);
            for (int j = 2 * i; j <= n; j += i)
                arr[j] = 1;
        }
    return vect;
}
ll mod_add(ll a, ll b, ll m)
{
    a = a % m;
    b = b % m;
    return (((a + b) % m) + m) % m;
}
ll mod_mul(ll a, ll b, ll m)
{
    a = a % m;
    b = b % m;
    return (((a * b) % m) + m) % m;
}
ll mod_sub(ll a, ll b, ll m)
{
    a = a % m;
    b = b % m;
    return (((a - b) % m) + m) % m;
}
ll mod_div(ll a, ll b, ll m)
{
    a = a % m;
    b = b % m;
    return (mod_mul(a, mminvprime(b, m), m) + m) % m;
} // only for prime m
ll phin(ll n)
{
    ll number = n;
    if (n % 2 == 0)
    {
        number /= 2;
        while (n % 2 == 0)
            n /= 2;
    }
    for (ll i = 3; i <= sqrt(n); i += 2)
    {
        if (n % i == 0)
        {
            while (n % i == 0)
                n /= i;
            number = (number / i * (i - 1));
        }
    }
    if (n > 1)
        number = (number / n * (n - 1));
    return number;
} // O(sqrt(N))
```

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 DSU

```
// 1 based indexing

struct DSU{
    vll p;
    ll n , connected;
    vll sz;

    void init(ll n){
        p.resize(n+1);
        iota(p.begin(), p.end(), 0);
        sz.assign(n+1, 1);
        connected = n;
    }

    ll get(ll x) {
        if(x == p[x]){
            return x;
        }
        return p[x]=get(p[x]);
    }

    ll getsz(ll u)
 {
  return sz[get(u)];
 }

    bool unite(int x, int y) {
        x = get(x);
        y = get(y);
        if(x == y) return false;

        connected--;

        if(sz[x] > sz[y])
   swap(x, y);

  sz[y] += sz[x];
  sz[x] = 0;
  p[x] = p[y];
        return true;
    }
};
```

2.3 DSUrollback

```
int n , q;

const int maxN = 3e5+1;
vll sol;

struct DSU{
    vector<pll> st[4*maxN];
    vll p;
    //path compression wont work during rollbacks, so rank
        compression
    vll rank;
    // e is basically storing the states, where .first is
        storing the present, and .second is storing the past
         to the moment where it was changed(cool)
    vector<pair<int&, int>> e;
    // op is basically storing by how much or how the value
        changed
    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

```
//Decomposing queries in blocks of sqrt(N) size and storing
    the results in a vector, while increasing l and r in
    such a way, that it is most optimal.
```

```
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 SegTree

```
 // 0-based indexed segment tree , with last element of
     range of element included
#define var ll

// struct var{
//   //var as per needed in question (var present in the node
     of the segment tree)
```

```
//  ll x;
// };

struct seg_tree{

 ll size;
 vector<var> a;
 vector<ll> lazy;
 vector<bool> clazy;
 //declare invariant for calc
 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){
  //merge nodes of the seg_tree
  // minimum:var a = min(b , c);
  return a;
 }

    // apply operation defines what we are doing in range
         update , for adding a+=operation(a,b); , assignment:
          a = operation(a,b);
 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
   //assignment: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){
   //set value
   //addition:lazy[node]+=v
   //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);
 }
 //0 BASED INDEXED , QUERY, AND STUFF WILL BE FROM 0 to n-1,
       if q is 1 to n, then l--, r--, REMEMBER U DUMB
  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);
}



void prints(void){
 cout << size << endl;
 printv(a);
 printv(lazy);
 printv(clazy);
}

};
```

3 Graph

3.1 Djikstra

```
int bfs(int source){
   vector<int>vis(N,0);
   vector<int>dist(N,INF);

   set<pair<int,int>>st;
   st.insert({0,source});
   dist[source]=0;

   while(!st.empty()){
      pair<int,int>p = *st.begin();
      st.erase(st.begin());

      int dis = p.first;
      int curr_vec = p.second;
      if(vis[curr_vec]==0){
        for(auto child:graph[curr_vec]){
            int tempdist = dis+child.second;
            if(tempdist<dist[child.first]){
                st.insert({tempdist,child.first});
                dist[child.first] = tempdist;
            }
        }
      }
      vis[curr_vec]=1;
   }



}
```

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];
}

void preprocess(int root) {
    tin.resize(n);
    tout.resize(n);
    timer = 0;
    l = ceil(log2(n));
    up.assign(n, vector<int>(l + 1));
    dfs(root, root);
}
```

3.3 primMST

```
ll n , m , tot_wt , e;
vector<vector<pll>>graph , tree;
vll parent , dist;


void prim_mst(ll source ){
    set<pll>st;
    st.insert({0,source});
    dist[source] = 0;
    vector<bool>vis(n+1 , false);
    while(!st.empty()){
        auto x = *st.begin();
        st.erase(x);
        ll u = x.second;
        if(vis[u]){
            continue;
        }
        ll v = parent[u];
        ll w = x.first;
        tree[v].pb({w , u});
        tree[u].pb({w , v});
        e++;
        vis[u] = true;
        tot_wt += x.first;
        for(auto edge: graph[u]){
            if(!vis[edge.second] && edge.first < dist[edge.
                 second]){
                st.erase({dist[edge.second] , edge.second});
                dist[edge.second] = edge.first;
                parent[edge.second] = u;
                st.insert({edge.first , edge.second});
            }
        }
    }
}
```

```
// if e != n , then it is impossible to form the mst

void init( ){
    graph.resize(n+1);
    parent.assign(n+1 ,0);
    dist.assign(n+1 , INF);
    tree.resize(n+1);
}
```

4 Maths

4.1 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
```

4.2 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;
}
```

4.3 ixclu

```
ll ixclu (ll num, ll lim) {
    vector<ll> p;
    for (ll i=2; i*i<=num; ++i)
        if (num % i == 0) {
            p.push_back (i);
            while (num % i == 0)
                num /= i;
        }
    if (num > 1)
        p.push_back (num);

    ll sum = 0;
    for (ll msk=1; msk<(1<<p.size()); ++msk) {
        ll mult = 1, bits = 0;
        for (ll i=0; i<(ll)p.size(); ++i)
            if (msk & (1<<i)) {
                ++bits;
                mult *= p[i];
            }

        ll cur = lim / mult;
        if (bits % 2 == 1)
            sum += cur;
        else
            sum -= cur;
    }

    return lim-sum;
}
```

4.4 Matrix

```
#define maxn 2
struct Mat{
    int mat[maxn][maxn];
    int row,col;
    Mat(int _row=2,int _col=2){
        row=_row;col=_col;
        mat[0][0]=1;mat[0][1]=0;
        mat[1][0]=0;mat[1][1]=1;
    }
    bool identity(){
        if(mat[0][0]==1&&mat[0][1]==0&&mat[1][0]==0&&mat
            [1][1]==1)return 1;
        else return 0;
    }
};

Mat mod_add(Mat a,Mat b,int p=MOD){
    Mat ans(a.row,b.col);
    memset(ans.mat,0,sizeof(ans.mat));
    for(int i=0;i<a.row;i++)
        for(int j=0;j<a.col;j++){
            ans.mat[i][j]=a.mat[i][j]+b.mat[i][j];
            ans.mat[i][j]%=p;
        }
    return ans;
}

Mat mod_mul(Mat a,Mat b,int p=MOD){
    Mat ans(a.row,b.col);
    memset(ans.mat,0,sizeof(ans.mat));
    for(int i=0;i<ans.row;i++)
        for(int k=0;k<a.col;k++)
            if(a.mat[i][k])
                for(int j=0;j<ans.col;j++)
                {
                    ans.mat[i][j]=(ans.mat[i][j]+1LL*a.mat[i][
                        k]*b.mat[k][j])%p;
                }
    return ans;
}
Mat mod_pow(Mat a,int k,int p=MOD) {
    Mat ans(a.row,a.col);
    for(int i=0;i<a.row;i++)for(int j=0;j<a.col;j++)ans.mat[i
        ][j]=(i==j);
    while(k){
        if(k&1)ans=mod_mul(ans,a,p);
        a=mod_mul(a,a,p);
        k>>=1;
    }
    return ans;
}

Mat fib(int n){
    Mat ans(2,2);
    ans.mat[0][1]=1;
    ans.mat[1][0]=1;
    ans.mat[1][1]=0;
    return mod_pow(ans,n,MOD);
}
```

4.5 mint

```
ll myRand(ll B) {
 return (ull)rng() % B;
}

const ll MOD = 998244353;
template<ll mod = MOD> struct mint { // 1000000007
    1000000009
 ll x;

 mint() : x(0) {}
 mint(ll _x) {
  _x %= mod;
  if (_x < 0) _x += mod;
  x = _x;
 }

 mint& operator += (const mint &a) {
  x += a.x;
  if (x >= mod) x -= mod;
  return *this;
 }
 mint& operator -= (const mint &a) {
  x += mod - a.x;
  if (x >= mod) x -= mod;
  return *this;
 }
 mint& operator *= (const mint &a) {
  x = (ull)x * a.x % mod;
  return *this;
 }
 mint pow(ll pw) const {
  mint res = 1;
  mint cur = *this;
  while(pw) {
   if (pw & 1) res *= cur;
   cur *= cur;
   pw >>= 1;
```

```
 }
 return res;
}
mint inv() const {
 assert(x != 0);
 ll t = x;
 ll res = 1;
 while(t != 1) {
  ll z = mod / t;
  res = (ull)res * (mod - z) % mod;
  t = mod - t * z;
 }
 return res;
}
mint& operator /= (const mint &a) {
 return *this *= a.inv();
}
mint operator + (const mint &a) const {
 return mint(*this) += a;
}
mint operator - (const mint &a) const {
 return mint(*this) -= a;
}
mint operator * (const mint &a) const {
 return mint(*this) *= a;
}
mint operator / (const mint &a) const {
 return mint(*this) /= a;
}

bool sqrt(mint &res) const {
 if (mod == 2 || x == 0) {
  res = *this;
  return true;
 }
 if (pow((mod - 1) / 2) != 1) return false;
 if (mod % 4 == 3) {
  res = pow((mod + 1) / 4);
  return true;
 }
 int pw = (mod - 1) / 2;
 int K = 30;
 while((1 << K) > pw) K--;
 while(true) {
  mint t = myRand(mod);
  mint a = 0, b = 0, c = 1;
  for (int k = K; k >= 0; k--) {
   a = b * b;
   b = b * c * 2;
   c = c * c + a * *this;
   if (((pw >> k) & 1) == 0) continue;
   a = b;
   b = b * t + c;
   c = c * t + a * *this;
  }
  if (b == 0) continue;
  c -= 1;
  c *= mint() - b.inv();
  if (c * c == *this) {
   res = c;
   return true;
  }
 }
 assert(false);
}

bool operator == (const mint &a) const {
 return x == a.x;
}
bool operator != (const mint &a) const {
 return x != a.x;
}
bool operator < (const mint &a) const {
 return x < a.x;
}
};
template<ll mod = MOD> struct Factorials {
 using Mint = mint<mod>;
 vector<Mint> f, fi;

 Factorials() : f(), fi() {}
 Factorials(int n) {
  n += 10;
  f = vector<Mint>(n);
  fi = vector<Mint>(n);
  f[0] = 1;
  for (int i = 1; i < n; i++)
   f[i] = f[i - 1] * i;
  fi[n - 1] = f[n - 1].inv();
  for (int i = n - 1; i > 0; i--)
   fi[i - 1] = fi[i] * i;
 }

 Mint C(int n, int k) {
  if (k < 0 || k > n) return 0;
  return f[n] * fi[k] * fi[n - k];
 }
};
template<ll mod = MOD> struct Powers {
 using Mint = mint<mod>;
 vector<Mint> p, pi;

 Powers() : p(), pi() {}
 Powers(int n, Mint x) {
  n += 10;
  if (x == 0) {
   p = vector<Mint>(n);
   p[0] = 1;
  } else {
   p = vector<Mint>(n);
   pi = vector<Mint>(n);
   p[0] = pi[0] = 1;
   Mint xi = x.inv();
   for (int i = 1; i < n; i++) {
    p[i] = p[i - 1] * x;
    pi[i] = pi[i - 1] * xi;
   }
  }
 }

 Mint pow(int n) {
  if (n >= 0)
   return p[n];
  else
   return pi[-n];
 }
};
template<ll mod = MOD> struct Inverses {
 using Mint = mint<mod>;
 vector<Mint> ii;

 Inverses() : ii() {}
 Inverses(int n) {
  n += 10;
  ii = vector<Mint>(n);
  ii[1] = 1;
  for (int x = 2; x < n; x++)
   ii[x] = Mint() - ii[mod % x] * (mod / x);
 }

 Mint inv(Mint x) {
  assert(x != 0);
  ll t = x.x;
  ll res = 1;
  while(t >= (int)ii.size()) {
   ll z = mod / t;
   res = (ull)res * (mod - z) % mod;
   t = mod - t * z;
  }
  return ii[t] * res;
```

```
 }
};
using Mint = mint<>;
```

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;
}
```

5 Strings

5.1 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]);
            }
        }
    }
}
```

5.2 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;
```

5.3 Trie

```
typedef struct trie{

    typedef struct node{
        node* nxt[2];
        int cnt = 0;
        node(){
            nxt[0] = nxt[1] = NULL;
            cnt = 0;
        }
    }Node;

    Node* head;
    trie(){
        head = new Node();
    }
    void insert(int x){
        Node* curr = head;
        for(int i=30;i>=0;i--){
            int b = (x>>i)&1;

            if(curr->nxt[b] == NULL){
                curr->nxt[b] = new Node();
```

```
        }
        curr = curr->nxt[b];
        curr->cnt ++;
    }
}
void remove(int x){

    Node* curr = head;
    for(int i=30;i>=0;i--){
        int b = (x>>i)&1;
        curr = curr->nxt[b];
        curr->cnt --;

    }
}

}Trie;
```

6 template

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

#define INF 1e18
#define endl "\n"
#define pb push_back
#define ppb pop_back
#define mp make_pair
#define PI atan(1)*4
#define set_bits __builtin_popcountll0
#define all(x) (x).begin(), (x).end()
#define vi vector<int>
#define vll vector<ll>
#define pll pair<ll,ll>
#define rvsort(a) sort(all(a),greater<int>())
#define read(a,n) for(int i = 0 ; i < n ; i ++){ cin >> a[i
    ];}
#define printv(a) for(auto it: a){cout << it << " ";} cout
    << endl;
#define ms(arr, v) memset(arr, v, sizeof(arr))

typedef long long ll;
typedef unsigned long long ull;
typedef long double lld;

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

ll uid(ll l, ll r) {return uniform_int_distribution<ll>(l, r
    )(rng);}

void solve(){

}

int main() {

ios::sync_with_stdio(0);
cin.tie(0);
cout.tie(0);


ll t = 1;
cin >> t;
for(int i = 1 ; i <= t ; i++){
//google(i);
  solve();
}
return 0;
}
```