International Islamic University Chittagong

MAIN TEMPLATE :-

```
#include<bits/stdc++.h>
using namespace std;
///POLICY BASED DATA STRUCTURE..
///order_of_key return number of element which are strictly greater/smaller than x..
///find_by_order return ans iterator corresponding to the xth position of the set..

//#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
#define ordered_set tree<int, null_type,less_equal<int>, rb_tree_tag, tree_order_statistics_node_update>

#define faster
ios_base::sync_with_stdio(false);cin.tie(NULL)

#define pb push_back
#define all(x) (x).begin(),(x).end()
#define allr(x) (x).rbegin(),(x).rend()
#define sz(n) (int)(n).size()
#define eps 1e-10
#define YES cout << "YES" << endl
#define NO cout << "NO" << endl
#define Yes cout << "Yes" << endl;
#define No cout << "No" << endl;
#define yes cout << "yes" << endl;
#define no cout << "no" << endl;
#define min3(a, b, c) min(c, min(a, b))
#define min4(a, b, c, d) min(d, min(c, min(a, b)))
#define max3(a, b, c) max(c, max(a, b))
#define max4(a, b, c, d) max(d, max(c, max(a, b)))
#define pi 2*acos(0)  /// acos(-1.0)
#define deg_to_rad(x) ((x)*((2*acos(0))/180.0))
#define rad_to_deg(x) ((x)*(180.0/(2*acos(0))))
#define fi first
#define sc second
#define mp make_pair

typedef long long ll;
const ll INF = 0x3f3f3f3f3f3f3f3f;
const int M = 1e9 + 7;
const int N = 200020;

//ll n,m,i,k,h;

vector<ll>prime_divisor[N];
int vis[N];
vector<int>edge[N];

bool cmp(pair<ll,ll>p1,pair<ll,ll>p2)
{
   return p1.fi<p2.fi;
}

void solve()
{
    ll
i,j,k,n,m,p,q,x,y,z,u,v,l,r,mod=1e9+7,mx,mn,mx1,mn1,cnt1,cnt;
    cin>>n;
    ll a[n+5];
    for(i=0;i<n;i++)
    {
       cin>>a[i];
    }

}
int main()
{
   faster;
   ll tc=1;
   cin>>tc;
   for(ll t=1;t<=tc;t++)
   {
  /// cout<<"Case "<<t<<": ";
      solve();
   }
   return 0;
}
```

Z - ALGORITHM :-

```
cin>>s;
// An element Z[i] of Z array stores length of the longest substring
// starting from str[i] which is also a prefix of str[0..n-1].
// The first entry of Z array is meaning less as complete string is always prefix of itself.
// Here Z[0]=0.
   n = sz(s);
   vector<ll> z(n);
   for (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;
   }

   for(i=0;i<n;i++)
   {
     cout<<z[i]<<" ";
   }
```

DFS :-

```
bool vis[200010];
vector<int>edge[200010];
void dfs(int x)
{
   vis[x]=1;
   for(int j=0; j<sz(edge[x]); j++)
   {
      int y=edge[x][j];
      if(vis[y]==0)
      {
         dfs(y);
      }
   }
}
```

HASHING :-

```
typedef long long ll;
typedef pair<ll,ll> pi;

ll mod_inverse(ll a, ll p, ll m){ ll
r=1;while(p){if(p%2)r=((r%m)*(a%m))%m;a=((a%m)*(a%
m))%m;p/=2;}return r;}

ll add(ll a, ll b, ll mod){return
(a%mod+b%mod+mod)%mod;}
ll substract(ll a, ll b, ll mod){return (a%mod-
b%mod+mod)%mod;}
ll mult(ll a, ll b, ll mod){return (a%mod*b%mod)%mod;}

const ll N = 1e6+10;
const ll base = 65; // base = 30 ['a', 'z']

pi pw[N + 10];
pi inv[N + 10];
pi h[N + 10];

pi mod = {1e9+7, 1e9+9};
vector<ll>pr = {1000000007, 1000000009, 1000000021,
1000000033, 1000000087, 1000000093, 1000000097,
1000000103, 1000000123, 1000000181, 1000000207,
1000000223, 1000000241, 1000000271, 1000000289};

mt19937
rng(chrono::steady_clock::now().time_since_epoch().co
unt());

ll get_random()
{
   return (ll)(rng());
}

ll get_random_in_range(ll L,ll R)
{
   ll rndm = get_random();
return L + (rndm % (R - L + 1));
}

void randomMod()
{
   ll i = get_random_in_range(0, 14);
   mod.fi = pr[i];
   pr.erase(pr.begin()+i);
   i = get_random_in_range(0, 13);
   mod.sc = pr[i];
}

void preCalc()
{
   pw[0].fi = pw[0].sc = 1;
   for(ll i=1; i<N; i++)
   {
      pw[i].fi = mult(pw[i-1].fi, base, mod.fi);
      pw[i].sc = mult(pw[i-1].sc, base, mod.sc);
   }
   ll pw_inv1 = mod_inverse(base, mod.fi-2, mod.fi);
   ll pw_inv2 = mod_inverse(base, mod.sc-2, mod.sc);
   inv[0].fi = inv[0].sc = 1;
   for(ll i=1; i<N; i++)
   {
      inv[i].fi = mult(inv[i-1].fi, pw_inv1, mod.fi);
      inv[i].sc = mult(inv[i-1].sc, pw_inv2, mod.sc);
   }
}

ll fn(char ch)
{
   if(islower(ch))return ch - 'a' + 1;
   if(isupper(ch))return ch - 'A' + 27;
   return ch - '0' + 53;
}

void build(string s)
{
   ll n = sz(s);
   for(ll i=0; i<n; i++)
{
      ll a1 = 0, b1, a2 = 0, b2;
      if(i)
      {
         a1 = h[i-1].fi;
         a2 = h[i-1].sc;
      }
      b1 = mult(pw[i].fi, fn(s[i]), mod.fi); // b = mult(pw[i],
s[i] - 'a' + 1);
      b2 = mult(pw[i].sc, fn(s[i]), mod.sc);
      h[i].fi = add(a1, b1, mod.fi);
      h[i].sc = add(a2, b2, mod.sc);
   }
}

pi gethash(ll l,ll r)
{
   ll res1 = h[r].fi;
   ll res2 = h[r].sc;
   if(l)
   {
      res1 = substract(res1, h[l-1].fi, mod.fi);
      res2 = substract(res2, h[l-1].sc, mod.sc);
   }
   res1 = mult(res1, inv[l].fi, mod.fi);
   res2 = mult(res2, inv[l].sc, mod.sc);
   return {res1, res2};
}

void solve()
{
   string s1;
   cin>>s1;
    build(s1);
}

int main()
{
   faster;
   randomMod();
   preCalc();
}
```

MANACHER'S :-

```
cin>>s;
    n=sz(s);
   vector<ll> d1(n); // maximum odd length palindrome centered at i
  // here d1[i]=the palindrome has d1[i]-1 right characters from i
  // e.g. for aba, d1[1]=2;
  for (i = 0, l = 0, r = -1; i < n; i++)
  {
    k = (i > r) ? 1 : min(d1[l + r - i], r - i);
    while (0 <= i - k && i + k < n && s[i - k] == s[i + k])
    {
      k++;
    }
    d1[i] = k--;
    if (i + k > r)
    {
      l = i - k;
      r = i + k;
    }
  }

  vector<ll> d2(n);  // maximum even length palindrome centered at i
  // here d2[i]=the palindrome has d2[i]-1 right characters from i
  // e.g. for abba, d2[2]=2;
  for ( i = 0, l = 0, r = -1; i < n; i++)
  {
    k = (i > r) ? 0 : min(d2[l + r - i + 1], r - i + 1);
    while (0 <= i - k - 1 && i + k < n && s[i - k - 1] == s[i + k])
    {
      k++;
    }
    d2[i] = k--;
    if (i + k > r)
    {
      l = i - k - 1;
      r = i + k ;
    }
  }

  for(i=0;i<n;i++)
  {
    cout<<d1[i]<<" ";
  }
```

DSU :-

```
const int N = 200020;
int parent[N];
int siz[N];
int n,i;

void make_set(int n)
{
  for(i=1;i<=n;i++)
  {
    parent[i]=i;
    siz[i]=1;
  }
}
int find_set(int u)
{
  if(parent[u]==u)return u;
  return parent[u]=find_set(parent[u]);
}

void union_set(int u,int v)
{
  int a=find_set(u);
  int b=find_set(v);
  if(siz[a]<siz[b])
  {
    swap(a,b);
  }
  if(a!=b)
  {
    ///parent[small]=big;
    parent[b]=a;
    ///sz[big]+=sz[small];
    siz[a]+=siz[b];
  }
}
int main()
{
  int i,j;
  cin>>n;
  make_set(n);
  for(i=0;i<n-1;i++)
  {
    int u,v;
    cin>>u>>v;
    if(find_set(u)!=find_set(v))
    {
      union_set(u,v);
    }
  }
}
```

EXTENDED EUCLID :-

```
ll extended_euclid(ll a,ll b,ll &x,ll &y)
{
///base case
  if(b==0)
  {
    y=0, x=1;
    return a;
  }
  ll x1,y1;
  ll g = extended_euclid(b,a%b,x1,y1);
  x=y1;
  y=x1-y1*(a/b);
  return g;
}
ll inverse(ll a,ll m)
{
  ll x,y, g=extended_euclid(a,m,x,y);
  if(g!=1)
    return -1;

  return ((x%m)+m)%m;
}
```

GRAPH COLORING :-

```
const int N = 200020;
vector<int>edge[200010];
int col[N];
bool flag=0;
///colour korbo 1 or 2 diye....
void dfs(int u,int c)
{
   if(col[u])///aghe jodi colour kora takhe...
   {
      if(col[u]!=c)/// colour ta jodi kankito colour na hoi...
      {
         flag=1;///odd length er cycle exist kore  && graph colour kora jabe na(!BIPARTITE).
         return;
      }
      return;
   }
   col[u]=c;
   for(int v : edge[u])
   {
      dfs(v,3-c);///(colour jodi 1 hoi amra er adjacent gula (3-1)==2 diye colour korbo)...
   }          ///(colour jodi 2 hoi amra er adjacent gula (3-2)==1 diye colour korbo)...
}
```

GRID DFS :-

```
ll n,r,c;
string s[1010];
bool vis[1000][1000];

/// direction array...
int drx[4]={1,-1,0,0};
int dry[4]={0,0,1,-1};

///check move is valid or not...
bool valid(int x,int y)
{
   return x>=0 && x<r && y>=0 && y<c &&  !vis[x][y] && s[x][y]=='.';
}

void dfs(int x,int y)
{
   vis[x][y]=1;
   for(int k=0; k<4; k++)
   {
      int a=x+drx[k];
      int b=y+dry[k];
      if(valid(a,b))
      {
         dfs(a,b);
      }
   }
}
```

GRID BFS :-

```
void bfs()
{
   vector<vector<bool> >can(r,vector<bool>(c,0));
   vector<vector<bool> >vis(r,vector<bool>(c,0));

   can[1][1]=1;
   vis[1][1]=1;
   ll ans=0;
   queue<pair<int,int> >q;
   q.push({1,1});
   while(!q.empty())
   {
//pair<int,int>p=q.front();
      int x=q.front().fi;
      int y=q.front().sc;
      q.pop();
      for(i=0; i<4; i++)
      {
         int x1=x;
         int y1=y;
         while(s[ x1+drx[i] ][ y1+dry[i] ]=='.')
         {
            x1+=drx[i];
            y1+=dry[i];
            can[x1][y1]=1;
         }
         if(!vis[x1][y1])
         {
            vis[x1][y1]=1;
            q.push({x1,y1});
         }
      }
   }
}
```

LPS :-

```
    n = sz(s1);
   ll lps[n + 5];
   i = 0;
   j = -1;
   lps[0] = -1;
   // lps[n];
   while (i < n)
   {
      while (j != -1 && s1[j] != s1[i])
      {
         j = lps[j];
      }
      i++, j++;
      lps[i] = j;
   }
```

FLOYD WARSHALL :-

```
const int N = 105;
int d[N][N];
int main() {
 int n = 10;
 for (int i = 1; i <= n; i++) {
  for (int j = 1; j <= n; j++) {
   if (i != j) {
    d[i][j] = 1e9;}  } }

 for (int k = 1; k <= n; ++k) {
  for (int i = 1; i <= n; ++i) {
   for (int j = 1; j <= n; ++j) {
    d[i][j] = min(d[i][j], d[i][k] + d[k][j]);
    }
  }
 }
```

PRINT ALL SUBSTRING :-

```
void printAllSubstrings( string& str )
{
   int len = str.length();
   for (int start = 0; start < len; ++start)
   {
      for (int end = start; end < len; ++end)
      {
         cout << str.substr(start, end - start + 1) << endl;
      }
   }
}
```

BINARY EXPONENTIATION[(a^n)%p] :-

```
long long int power( ll a, ll n, ll p  )
{
   ll res = 1;
   while(n)
   {
      if(n%2)
         res = (res*a)%p, n--;
      else
         a = (a*a)%p, n/=2;
   }
   return res;
}
```

EULER TOTIENT PHI:-

phi(n)=count of no. from 1 to n that are co-prime with n.

```
void eulerPhi(int n){
 int phi[n+1];
 for(int i=1;i<=n;i++){
  phi[i] = i;
 }
 for(int i =2;i <= n;i++){
  if(phi[i] == i){
   phi[i] = i-1;
   for(int j=2*i;j <= n;j +=i){
    phi[j] = (phi[j]*(i-1))/i;
   }
  }
 }
}
```

SMALLEST PRIME FACTOR :-

```
int spf[1000010];
void sieve()
{
   int maxn=1000000;
   for(int i=0;i<=maxn;i++)
   {
      spf[i]=-1;
   }
   for(int i=2;i<=maxn;i++)
   {
      if(spf[i]==-1)
      {
         for(int j=i;j<=maxn;j+=i)
         {
            if(spf[j]==-1)
            {
               spf[j]=i;
            }
         }
      }
   }
}
```

how does it works in main function:-

```
while(n!=1)
   {
      cout<<spf[n]<<" ";
      n/=spf[n];
   }
```

PRINT ALL PERMUTATION :-

```
do
   {
// Print the current permutation
      for (int i = 0; i < v.size(); ++i)
      {
         cout << v[i] << ' ';
      }
      cout <<endl;
   } while (next_permutation(v.begin(), v.end()));
```

PRINT ALL SUBSET :-

```
void findSubsets(int nums[], int n)
{
 // Loop through all possible subsets using bit manipulation
   for (int mask = 0; mask < (1 << n); mask++) {

 // Loop through all elements of the input array
      for (int i = 0; i < n; i++) {
// Check if the ith bit is set in the current subset
         if ((mask) & (1 << i)) {
// If the ith bit is set, add the jth element to the subset
            cout << nums[i] << " ";
         }
      }
      cout << endl;
   }
}
```

TEST CASE GENERATOR :-

```
srand(time(NULL));
   int t = rand() % 10 + 1;
   while (t--)
   {
      int n = rand() % 100 + 1;
      int a[n + 1];
      for (int i = 0; i < n; i++)
      {
         a[i] = (rand() % 100 + 1);
      }
   }
```

GCD && LCM :-

```
ll gcd(ll a, ll b)
{
 if (a == 0) return b;
 return gcd(b % a, a);
}
ll lcm(ll a, ll b) { return (a * b) / gcd(a, b);
 }
```

SEGMENT TREE :-

```
class SGTree {
 vector<int> seg;
public:
 SGTree(int n) {
  seg.resize(4 * n);
 }
 void build(int ind, int low, int high, int arr[]) {
  if(low == high) {
   seg[ind] = arr[low]; //CHANGE
   return;
  }
  int mid = (low + high) >> 1;
  build(2*ind+1, low, mid, arr);
  build(2*ind+2, mid+1, high, arr);
  seg[ind] = min(seg[2*ind+1] ,seg[2*ind+2]);//CHANGE
 }
 void update(int ind, int low, int high, int i, int val){
  if(low==high)
     {
        seg[ind]=val; //CHANGE
        return;
     }
     int mid=(low+high)>>1;
     if(i<=mid)update(2*ind+1, low, mid, i,val);
     else update(2*ind+2, mid+1, high, i,val);
     seg[ind] = min(seg[2*ind+1] ,seg[2*ind+2]);
//CHANGE
 }

 int query(int ind, int low, int high, int l, int r) {

// no overlap
// l r low high or low high l r
   if(high < l or r < low) {
    return INT_MAX;//CHANGE
   }

// complete overlap
// [l low high r]
   if(low>=l && high <= r) return seg[ind]; //CHANGE
   int mid = (low + high) >> 1;
   int left = query(2*ind+1, low, mid, l, r);
   int right = query(2*ind+2, mid+1, high, l, r);
   return min(left,right); //CHANGE
 }
};
int main()
{
   int n,i;
 cin >> n;
 int arr[n];
 for(int i = 0;i<n;i++) cin >> arr[i];
 SGTree st(n);
 st.build(0,0,n-1, arr);
 int q;
 cin >> q;
 while(q--) {
  int type;
  cin >> type;
  if(type==1) {
   int l, r;
   cin >> l >> r;
   cout << st.query(0,0,n-1,l,r) << endl;
  }
  else {
   int l, r, val;
   cin >> i >> val;
   st.update(0,0,n-1,i,val);
  }
 }
 return 0;
}
```

DIJKSTRA'S :-

```
#define N 2000005
vector<pair<int, int>> adj[N];
priority_queue<pair<int, int>> pq;
int dis[N];
int prev[N];
void dijkthras()
{
while(!pq.empty())
   {
      auto x =pq.top();
      pq.pop();
      int u = x.second;
      int d = -x.first;
      if (dis[u]<d)
         continue;
      for (auto y: adj[u])
      {
         int v=y.first ;
         int w= y.second;
         if (dis[v]>(d+w))
         {
            dis[v]=d+w;
            pq.push({-dis[v], v});
            //prev[v]=u;
         }
      }
   }
}
int main()
{
   int n, m;
   cin >>  n >> m;
   for (int i=0; i<m; i++)
   {
      int u, v, w;
      cin >> u >> v >> w;
      adj[u].pb({v, w});
      adj[v].pb({u, w});
   }
   for (int i=1; i<=n; i++)
      dis[i]=INT_MAX;
   int src=1;
   dis[src]=0;
   pq.push({0, src});
   dijkthras();
   for (int i=1; i<=n; i++)
      cout << "i: " << i << " dis: " << dis[i] << endl;
}
```

LAZY PROPAGATE :-

```
class ST {
 vector<int> seg, lazy;
public:
 ST(int n) {
  seg.resize(4 * n);
  lazy.resize(4 * n);
 }
public:
 void build(int ind, int low, int high, int arr[]) {
  if(low == high) {
   seg[ind] = arr[low];
   return;
  }
  int mid = (low + high) >> 1;
  build(2*ind+1, low, mid, arr);
  build(2*ind+2, mid+1, high, arr);
  seg[ind] = seg[2*ind+1] + seg[2*ind+2];
 }
public:
 void update(int ind, int low, int high, int l, int r,
  int val) {
// update the previous remaining updates
// and propogate downwards
  if(lazy[ind] != 0) {
   seg[ind] += (high - low + 1) * lazy[ind];
// propogate the lazy update downwards
// for the remaining nodes to get updated
   if(low != high) {
    lazy[2*ind+1] += lazy[ind];
    lazy[2*ind+2] += lazy[ind];
   }

   lazy[ind] = 0;
  }
// no overlap
// we don't do anything and return
// low high l r or l r low high
  if(high < l or r < low) {
   return;
  }
// complete overlap
// l low high r
  if(low>=l && high <= r) {
   seg[ind] += (high - low + 1) * val;
// if a leaf node, it will have childrens
   if(low != high) {
    lazy[2*ind+1] += val;
    lazy[2*ind+2] += val;
   }
   return;
  }
// last case has to be no overlap case
  int mid = (low + high) >> 1;
  update(2*ind+1, low, mid, l, r, val);
  update(2*ind+2, mid+1, high, l, r, val);
  seg[ind] = seg[2*ind+1] + seg[2*ind+2];
 }
public:
 int query(int ind, int low, int high, int l, int r) {

// update if any updates are remaining
// as the node will stay fresh and updated
  if(lazy[ind] != 0) {
   seg[ind] += (high - low + 1) * lazy[ind];
// propogate the lazy update downwards
// for the remaining nodes to get updated
   if(low != high) {
    lazy[2*ind+1] += lazy[ind];
    lazy[2*ind+2] += lazy[ind];
   }

   lazy[ind] = 0;
  }

// no overlap return 0;
  if(high < l or r < low) {
   return 0;
  }
// complete overlap
  if(low>=l && high <= r) return seg[ind];

  int mid = (low + high) >> 1;
  int left = query(2*ind+1, low, mid, l, r);
  int right = query(2*ind+2, mid+1, high, l, r);
  return left + right;
 }
};
int main()
{
  int n;
 cin >> n;
 int arr[n];
 for(int i = 0;i<n;i++) cin >> arr[i];
 ST st(n+1);
 st.build(0,0,n-1, arr);

 int q;
 cin >> q;
 while(q--) {
  int type;
  cin >> type;
  if(type==1) {
   int l, r;
   cin >> l >> r;
   cout << st.query(0,0,n-1,l,r) << endl;
  }
  else {
   int l, r, val;
   cin >> l >> r >> val;
   st.update(0,0,n-1,l,r,val);
  }
 }
 return 0;
}
```

FIND CYCLE :-

```
const int N = 200020;
int vis[N];
vector<int>edge[N];
int flag=0;
int nd;
///store traversal path cycle...
stack<int>st;
void dfs(int u,int p)
{
  if(vis[u])
  {
///double visited than there is a cycle..
    flag=1;
    nd=u;
    return;
  }
///store path node...
  st.push(u);
  vis[u]=1;
  for(int v : edge[u])
  {
    if(v!=p)
    {
      dfs(v,u);
      if(flag==1)
      {
        return;
      }
    }
  }
///when backtracking a node pop this node..
  st.pop();
}
int main()
{
  int i,n,m;
  cin>>n>>m;
  for(i=0; i<m; i++)
  {
    int u,v;
    cin>>u>>v;
    edge[u].pb(v);
    edge[v].pb(u);
  }
  for(i=1; i<=n; i++)
  {
    if(flag==1)
      break;
    stack<int>st;
    if(!vis[i])
      dfs(i,0);
  }
//dfs(1,0);
  if(flag==0)
  {
    cout<<"IMPOSSIBLE"<<endl;
  }
  else
  {
    vector<int>ans;
    while(1)
    {
      ans.pb(st.top());
      if(st.top()==nd)
      {
        break;
      }
      st.pop();
    }
    cout<<sz(ans)+1<<endl;
    cout<<nd<<" ";
    for(i=0; i<sz(ans); i++)
    {
      cout<<ans[i]<<" ";
    }
    cout<<endl;
  }

}
```

STRUCTURE :-

```
/// structure declaration;
struct prb
{
  int val1, val2, idx, ;
};
///structure sort;
bool cp( prb a, prb b )
{
  if( a.val == b.val )
  {
    return a.idx < b.idx;
  }

  return a.val> b.val;
}
///structure definition
vector<prb>v;
cin>> a>>b>>c;
v.pb({a,b,c});
sort( all(v), cp );
for( auto x : v )
{
  cout<< x.val1 << " " << x.val2 << " " << x.idx << endl;
}
```

2D PREFIX SUM :-

To Build 2D Prefix Sum Equation:

prefix[i][j] = prefix[i 1][j] + prefix[i][j-1] - prefix[i-1][j-1]+arr[i][j]

How to be calculated?

(U,L).....
.....
.....(D,R)
-->p[D][R]-p[D][L-1]-p[U-1][R]+p[U-1][L-1]

FASTER :-

```
#pragma GCC optimize("Ofast")
#pragma GCC target("avx,avx2,fma")
#pragma GCC optimization("unroll-loops")
```

TREE DIAMETER :-

```
const int N = 200020;
vector<int>edge[200010];
/// root teke distance store korar jonno...
int dis[N];
///(node,parent,distance from node)
void dfs(int u,int p,int d)
{
   dis[u]=d;
   for(int v : edge[u])
   {
///adjancy matrix value jodi parent er equal na hoiii..

      if(v!=p)
      {
         dfs(v,u,d+1);
      }
   }
}
int main()
{
   faster;
   int n,i,x,y;
   cin>>n;
   for(i=0; i<n-1; i++)
   {
      cin>>x>>y;
      edge[x].pb(y);
      edge[y].pb(x);
   }
   dfs(1,0,0);
   int mx=0;
   int mx_node=0;
   for(i=1; i<=n; i++)
   {
      if(dis[i]>mx)
      {
         mx=dis[i];
         mx_node=i;
      }
   }
   dfs(mx_node,0,0);
   int ans=0;
   for(i=1; i<=n; i++)
   {
      ans=max(ans,dis[i]);
   }
   cout<<ans<<endl;
   return 0;
}
```

BFS :-

```
const int N = 200020;
bool vis[N];
vector<int>edge[N];
int level[N];
int main()
{
   faster;
   ll t,i,j,u,v,n,m;
/// get input...
   cin>>n>>m;
   for(i=0; i<m; i++)
   {
      cin>>u>>v;
      edge[u].pb(v);
      edge[v].pb(u);
   }
///Q.find shortest path node 1 to n...
   queue<int>pq;
/// here source node 1...
   int src=1;
   pq.push(src);
   vis[src]=1;
   int par[N];
   for(i=1; i<=n; i++)
   {
      par[i]=-1;
   }
   while(!pq.empty())
   {
      int x=pq.front();
      pq.pop();
   for(j=0; j<sz(edge[x]); j++)
      {
         int y=edge[x][j];
         if(vis[y]==0)
         {
            vis[y]=1;
            par[y]=x;
            pq.push(y);
         }
      }
   }
/// it means there is no path between 1 to n.
   if(par[n]==-1)
   {
      cout<<"IMPOSSIBLE"<<endl;
   }
   else
   {
      int last=n;
      stack<int>st;
/// print path by using parent array && store in stack..

      while(last!=-1)
      {
         st.push(last);
         last=par[last];
      }
      cout<<sz(st)<<endl;
      while(!st.empty())
      {
         cout<<st.top()<<" ";
         st.pop();
      }
   }
   return 0;
}
```

SIEVE :-

```
const int M = 1000000;
bool marked[M];
bool isprime(int n)
{
   if (n == 2)
      return true;
   if (n < 2 || n % 2 == 0)
      return false;
   return marked[n]==false;
}
void sieve(ll n)
{
   for (i = 3; i * i <= n; i += 2)
   {
      if (marked[i]==false) // i is a prime
      {
         for (int j = i * i; j <= n; j += i)
         {
            marked[j] = true;
         }
      }
   }
}
int main()
{
   sieve(1000000);
   cin >> n;
   for(i=0;i<n;i++)
   {
      if(isprime(i))
      {
       cout<<i<<endl;
      }
   }
}
```

SEGMENTED SIEVE :-

```
vector<bool> segmentedSieve(long long L, long long R) {
/// generate all primes up to sqrt(R)
   long long lim = sqrt(R);
   vector<bool> mark(lim + 1, false);
   vector<long long> primes;
   for (long long i = 2; i <= lim; ++i) {
      if (!mark[i]) {
         primes.emplace_back(i);
         for (long long j = i * i; j <= lim; j += i)
            mark[j] = true;
      }
   }

   vector<bool> isPrime(R - L + 1, true);
   for (long long i : primes)
      for (long long j = max(i * i, (L + i - 1) / i * i); j <= R; j
+= i)
         isPrime[j - L] = false;
   if (L == 1)
      isPrime[0] = false;
   return isPrime;
}

int main()
{
   ll L, R;
   cin >> L >> R;
   vector<bool> isPrime = segmentedSieve(L, R);
   set<ll> primes;
   for(int i = 0; i < (int)isPrime.size(); i++) {
      if(isPrime[i]) {
         primes.insert(i + L);
      }
   }
   cout << (int)primes.size() << "\n";
   for(ll u : primes) {
      cout << u << " ";
   }
   cout << "\n";
}
```

NPR & NCR WITH MOD :-

```
const int N = 500010;
int mod = 1e9 + 7;
int power(int a, int b, int M) {
   if(!b) return 1;
   int temp = power(a, b / 2, M);
   temp = 1LL * temp * temp % M;
   if(b % 2) temp = 1LL * temp * a % M;
   return temp;
}
int fact[N];
void pre() {
   fact[0] = 1;
   for(int i = 1; i < N; i++) {
      fact[i] = 1LL * fact[i - 1] * i % mod;
   }
}
ll npr(ll n, ll r)
{
 if(r > n) return 0;
 ll res = fact[n];
 res = (res * power(fact[n-r],mod-2,mod))%mod;
 return res;
}
int ncr(int n, int r) {
   if(n < r) return 0;
   assert(n >= 0 && n < N && r >= 0 && r < N);
   int ans = fact[n];
   ans = 1LL * ans * power(fact[n - r], mod - 2, mod) %
mod;
   ans = 1LL * ans * power(fact[r], mod - 2, mod) % mod;
   if(ans < 0) ans += mod;
   assert(ans >= 0 && ans < mod);
   return ans;
}
int main()
{
   pre();
   int n,m;
   cin>>n>>m;
   cout<<ncr(n,m)<<endl;
   cout<<npr(n,m)<<endl;   }
```

NPR & NCR WITHOUT MOD :-

```
typedef long long ll;
long long nCr(ll n, ll r)
{
   long long p, k;
   p = k = 1;
   // C(n,r) == C(n,n-r)
   if (n - r < r)
      r = n - r;
   if (r != 0)
   {
      while (r)
      {
         p *= n;
         k *= r;
         long long m = __gcd(p, k);
         p /= m;
         k /= m;
         n--;
         r--;
      }
   }
   else
      p = 1;
   return p;
}
int main()
{
   int n,m;
   cin>>n>>m;
   cout<<nCr(n,m)<<endl;
}
```

POLICY BASED DATA STRUCTURE :-

```
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
/*template<class T> using oset = tree<T, null_type,
less<T>, rb_tree_tag,
tree_order_statistics_node_update>; */
typedef tree<int, null_type, less<int>, rb_tree_tag,
tree_order_statistics_node_update> ordered_set1;
// os1 less => Small to Big
typedef tree<int, null_type, greater<int>, rb_tree_tag,
tree_order_statistics_node_update> ordered_set2;
// For Pair
typedef tree<pair<int, int>, null_type, less<pair<int,
int>>, rb_tree_tag, tree_order_statistics_node_update>
ordered_set3;
// for contest
typedef tree<pair<ll, int>, null_type, less<pair<ll, int>>,
rb_tree_tag, tree_order_statistics_node_update> os01;
// Less -> Small To Big
typedef tree<pair<ll, int>, null_type, greater<pair<ll,
int>>, rb_tree_tag, tree_order_statistics_node_update>
os02;
/* Time Complexity: O(log n)
 - Order_of_key(k): Number of items strictly smaller
then k
 => name.order_of_key(100);
- find_by_order(k): K-th element in a set ( Counting
from zero)
 => *name.find_by_order(5);
*/
```

SUM OF DIVISOR'S(SOD) :-

```
ll SOD(ll n)
{
 sod=1;
 for(ll i=2; i*i<=n; i++)
 {
 if(n%i==0)
 {
 cur=i;
 while(n%i==0)
 {  cur*=i,  n/=i;  }
 cur=(cur-1)/(i-1);
 sod*=cur;
 }
 }
 if(n!=1)sod*=(n*n)/(n-1);
 return sod;
}
```

NUMBER OF DIVISOR'S(NOD) :-

```
ll NOD(ll n)
{
   nod = 1;
   for (ll i = 2; i * i <= n; i++)
   {
      if (n % i == 0)
      {
         p = 0;
         while (n % i = 0)
         {
            n /= i;
            p++;
         }
         nod *= (p + 1);
      }
   }
   if (n != 1)
      nod *= 2;
   return nod;
}
```

NUMBER THEORY(BASIC) :-

```
a = (p1)^a1  *  (p2)^a2  *  (p3)^a3.....;
b = (p1)^b1  *  (p2)^b2  *  (p3)^b3.....;

GCD(a,b)   = (p1)^min(a1, b1 )  *  (p2)^min(a2, b2 )......;

LCM(a, b ) = (p1)^max(a1, b1 )  *  (p2)^max(a2, b2 )......;

Number_of_divisors(a) = (a1+1) * (a2+1) * (a3+1).........;

Sum_of_divisors(a)  =  [ (((p1)^a1+1) - 1 ) / (p1 - 1 ) ] *
                       [ (((p2)^a2+1) - 1 ) / (p2 - 1 ) ] *
                       [ (((p3)^a3+1) - 1 ) / (p3 - 1 ) ] *....
```

PRIORITY QUEUE(BASIC) :-

```
priority_queue <int> pq; // Boro theke choto
priority_queue<int, vector<int>, greater<int> > q;
//choto theke boro
```

NUMBER THEORY(BASIC) :-

1. Diophantine equation : ax + by = c
solution exist iff c | gcd(a, b)
2. 3 ways to do modulo inverse. 1/a % m
which is equivalent to ax = 1 mod m
a. m is prime - a^(m-2) = 1/a mod m
b. m is not prime - ax+my = 1 find x by egcd
c. m is not prime - a^phi(m)-1 = 1/a mod m.
3. Egcd solve the equation ax + by = gcd(a, b)
a. X += k * b/gcd(a,b) ;
Y -= k *a/gcd(a,b);

BIT :-

1. Check kth bit of a number by : x & (1<< k) if greater then 0 then kth bit is 1 otherwise 0.
2. Set kth bit of a number by : x = x | (1 << k) .
3. Reset kth bit of a number by : x = x & ~ (1<<k)
4. Toggle kth bit of a number by : x = x ^ (1<<k).
5. Check x is a power of 2 by : x & (x-1) == 0
then it is power of 2 otherwise not.
6. All bit of x are on if x & (x + 1) == 0 .
7. IF(a|b|c|...)=p,then->(p|a)=p,(p|b)=p,(p|c)=p...
8. (a<<b) = a*2^b
9. count the number of leading zero's:-
__builtin_clz(a);
10. Count the number of trailing zero's
__builtin_ctz(a);
11. number of 1's in the representation
__bulitin_popcount(a);
12. Testing i'th bit on or off -> if ((n>>i)&1).
13. Removing the last set bit
(a&a-1) -> (110100)to(110000).
14. a+b = (a&b + a|b) = 2*(a&b)+(a^b)

TRIE-01 :-

```
struct Node {
 Node* arr[26];
 bool flag = false;

 bool contains(char c) {
  return (arr[c - 'a'] != NULL);
 }

 void put(char c, Node* newNode) {
  arr[c - 'a'] = newNode;
 }

 Node* getNext(char ch) {
  return arr[ch - 'a'];
 }

 void setFlag() {
  flag = true;
 }

 bool isFlagSet() {
  return flag;
 }
};

Node* root = new Node();

void insert(string word) {
 Node* temp = root;
 for(int i = 0; i < word.size(); i++) {
// there might be a previous instance of s[i]
// previously there was word, now i am inserting
//worse
  if(!temp->contains(temp, word[i])) {
   Node* newNode = new Node();
   temp->put(word[i], newNode);
  }
  temp = temp->getNext(word[i]);
 }
 temp->setFlag();
}

bool search(string word) {
 Node* temp = root;
 for(int i = 0; i< word.size(); i++) {
  if(temp->contains(word[i])) {
   temp = temp->getNext();
  }
  else {
   return false;
  }
 }
 return temp->isFlagSet();
}

bool startsWith(string word) {
 Node* temp = root;
 for(int i = 0; i < word.size(); i++) {
  if(temp->contains(word[i])) {
   temp = temp->getNext();
  }
  else {
   return false;
  }
 }
 return true;
}
```

TRIE-02 :-

```
struct Node {
 Node* arr[26];
 int cntWordsEnd = 0;
 int cntPrefixs = 0;

 bool contains(char c) {
  return (arr[c - 'a'] != NULL);
 }
 void put(char c, Node* newNode) {
  arr[c - 'a'] = newNode;
 }

 Node* getNext(char ch) {
  return arr[ch - 'a'];
 }

 void increaseWordsEndsWith() {
  cntWordsEnd++;
 }

 void increaseCntPrefixs() {
  cntPrefixs++;
 }

 void decreaseWordsEndsWith() {
  cntWordsEnd--;
 }

 void decreaseCntPrefixs() {
  cntPrefixs--;
 }

 int getCountWordsEnd() {
  return cntWordsEnd;
 }

 int getCntPrefixs() {
  return cntPrefixs;
}
};
```

```
Node* root = new Node();

void insert(string word) {
 Node* temp = root;
 for(int i = 0; i < word.size(); i++) {
// there might be a previous instance of s[i]
// previously there was word, now i am inserting worse
  if(!temp->contains(temp, word[i])) {
   Node* newNode = new Node();
   temp->put(word[i], newNode);
  }
  temp->increaseCntPrefixs();
  temp = temp->getNext(word[i]);
 }
 temp->increaseWordsEndsWith();
}

int countWordsEqualTo(string word) {
 Node* temp = root;
 for(int i = 0; i< word.size(); i++) {
  if(temp->contains(word[i])) {
   temp = temp->getNext();
  }
  else {
   return false;
  }
 }
 return temp->getCountWordsEnd();
}

int startsWith(string word) {
 Node* temp = root;
 for(int i = 0; i < word.size(); i++) {
  if(temp->contains(word[i])) {
   temp = temp->getNext();
  }
  else {
   return 0;
  }
 }
 return temp->getCntPrefixs();
}

void erase(string word) {
 Node* temp = root;
// If does not exists, no need to erase anything
 if(search(word) == 0) return;

 for(int i = 0;i<word.size(); i++) {
  temp->decreaseCntPrefixs();
  temp->getNext();
 }
 temp->decreaseWordsEndsWith();
}
```

Co-Ordinate Compression:

```
void compress(int n)
{
   set<int>ss;
   map<int,int>mm;
   for(int i=1;i<=n;i++)
   {
      ss.insert(a[i]);
   }
   int now=1;
   for(auto it: ss)
   {
      mm[it]=now;
      now++;
   }
   for(int i=1;i<=n;i++)
   {
      a[i]=mm[a[i]];
   }
}

void compress2(int n)
{
   vector<pair<int,int> >pp(n);
   for(int i=0;i<n;i++)
   {
      pp[i]={a[i],i};
   }
   sort(all(pp));
   int nxt=0;
   for(int i=0;i<n;i++)
   {
      if(i>0 && pp[i-1].fi!=pp[i].fi)
      {
         nxt++;
      }
      a[pp[i].sc]=nxt;
   }
}
```

Inversion Count:

Build(base case -> seg[ind]= 0)
Update(base case -> seg[ind] += val)
Query(range sum)

```
ll ans = 0;
    for (i = 1; i <= n; i++)
   {
      ans+=st.query(1, 1, n, a[i]+1,n);
      st.update(1, 1, n, a[i], 1);
   }
   cout<<ans<<endl;
```

Closest Pair Distance:

```
using i128 = __int128;
const i128 INF = 1e30;
struct p{ int x,y; };
ll sq(ll a){return 1LL*a*a;}
bool cmpx(p a,p b){return a.x<b.x;}
bool cmpy(p a,p b){return a.y<b.y;}
ll dis(p a,p b){return sq(a.x-b.x)+sq(a.y-b.y);}
//Merge sort is O(n*lgn) so this is O(n*lg^2(n))
ll small_dis(vector<p>points)// Return square of
smallest distance
{
   ll n=sz(points);
   if(n<=1){ return LONG_LONG_MAX; }

vector<p>lft=vector<p>(points.begin(),points.begin()+n/
2);

vector<p>rgt=vector<p>(points.begin()+n/2,points.end()
);
   ll d1=small_dis(lft);
   ll d2=small_dis(rgt);
   ll d= min(d1,d2);
   ll mid_x=lft.back().x;// can be right[0].x or anything in
between
   vector<p>stripe;
   for(p a: lft){
      if(sq(a.x-mid_x)<d)
      {
         stripe.pb(a);
      }}
    for(p a: rgt){
      if(sq(mid_x-a.x)<d)
      {
         stripe.pb(a);
      }}
   sort(all(stripe),cmpy);
   for(int i=0;i<sz(stripe);i++)
   {
      // every stripe[i] pair up with points above it by at
most d
      // lemma says that the following for loop is O(1);
      for(int j=i+1;j<sz(stripe) && sq(stripe[j].y-
stripe[i].y)<d;j++)
      {
         d=min(d,dis(stripe[i],stripe[j]));
      }
   }
   return d;
}
main(){
    cin>>n;
    vector<p>points(n);
    for(i=0;i<n;i++)
    {
      cin>>points[i].x>>points[i].y;
    }
    sort(all(points),cmpx);
    cout<<small_dis(points)<<endl;
}
```

Interactive Problems:

```
ll query(int x, int y)//pos1,pos2
{
  cout<<"? "<<x<<" "<<y<<endl;
  ll z;
  cin>>z;
  return z;
```

Marge Sort Tree:

```
vector<int> seg[4*M];
int a[M];
class SGTree
{
public:
   SGTree(int n) {}
   void build(int ind, int low, int high)
   {
      if (low == high)
      {
         seg[ind].pb(a[low]);
         return;
      }
      int mid = (low + high) >> 1;
      build(2 * ind, low, mid);
      build(2 * ind + 1, mid + 1, high);

merge(all(seg[2*ind]),all(seg[2*ind+1]),back_inserter(se
g[ind]));
   }
   int query(int ind, int low, int high, int l, int r,int val)
   {
      if (r < low || high < l)
         return 0;
      if (l <= low && high <= r)// <=val
      {
         return (upper_bound(all(seg[ind]),val)-
seg[ind].begin());
      }
      int mid = (low + high) >> 1;
      int left = query(2 * ind, low, mid, l, r,val);
      int right = query(2 * ind + 1, mid + 1, high, l, r,val);
      return left + right;
   }
};
```

Mo's Algorithm:

```
int a[100005];
int freq[1000004];
int block = 175;
struct query
{
   int l, r, id;
   query() { l = r = id = -1; }
   query(int x, int y, int z)
   {
      l = x;
      r = y;
      id = z;
   }
   bool operator<(query &other) const
   {
      int cur_block = l / block;
      int other_block = other.l / block;
      if (cur_block != other_block)
      {
         return l < other.l;
      }
      return (cur_block & 1) ? r > other.r : r < other.r;
   }
};
vector<query> Q;
vector<int> ans;
ll sum;
void add(int i)
{
   freq[a[i]]++;
   if (freq[a[i]] == 1)
   {
      sum++;
   }
}
void del(int i)
{
   freq[a[i]]--;
   if (freq[a[i]] == 0)
   {
      sum--;
   }
}
main()
{
   cin >> n;
   for (i = 0; i < n; i++)
      cin >> a[i];
   cin >> q;
   for (i = 0; i < q; i++)
   {
      cin >> x >> y;
      x--;
      y--;
      Q.pb(query(x, y, i));
   }
   sort(all(Q));
   ans.resize(q);
   int curl = 0, curr = -1;
   for (auto it : Q)
   {
      int L = it.l;
      int R = it.r;
      int idx = it.id;
      while (curl < L){del(curl++); }
      while (curl > L){add(--curl);}
      while (curr < R){ add(++curr);}
      while (curr > R){del(curr--);}
      ans[idx] =sum;
   }
   for(i=0;i<q;i++)
   {
      cout<<ans[i]<<endl;
   }
}
```

Fenwick Tree:

```
const int lim=1e5+5;
ll bit_tree[lim];
int n;
//1- based indexing
void update(int idx,int val)
{
   while(idx<=n)
   {
      bit_tree[idx]+=val;
      idx+=(idx & (-idx));
   }
}
ll query(int idx)
{
   ll sum=0;
   while(idx>0)
   {
      sum+=bit_tree[idx];
      idx^=(idx&(-idx));//-
   }
   return sum;
}
main()
{
    cin>>n>>q;
    for(i=0;i<=n;i++){bit_tree[i]=0;}
    for(i=1;i<=n;i++){cin>>x;
       update(i,x);
    }
    while(q--)
    {
      int type;
      cin>>type;
      if(type==1)
      {
         cin>>x;
         x++;
         y=query(x)-query(x-1);
         cout<<y<<endl;
         update(x,-y);
      }
      else if(type==2)
      {
         cin>>x>>y;
         x++;
         update(x,y);
      }
      else
      {
         cin>>x>>y;
         y++;
         cout<<query(y)-query(x)<<endl;
      }
    }
}
```

Segment Tree(Offline Query):

```
SGTree st(n);
   st.build(1, 1, n);
   vector<pair<int,int>
>queries[n+5];//queries[r].push_back({l, id})
   for(i=1;i<=q;i++)
   {
      cin>>x>>y;
      queries[y].pb({x,i});
   }
   map<int,int>last;//last[x] = last occurrence of x kothai
chilo
   int ans[q+5];
   for(r=1;r<=n;r++)
   {
      st.update(1,1,n,r,1);
      if(last[a[r]]!=0)
      {
         st.update(1,1,n,last[a[r]],0);
      }
      last[a[r]]=r;
      for(auto it : queries[r])
      {
         l=it.fi;
         int idx=it.sc;
         ll sum=st.query(1,1,n,l,r);
         ans[idx]=sum;
      }
   }
   for(i=1;i<=q;i++)
   {
      cout<<ans[i]<<endl;
   }
```

Randomization:

```
mt19937_64
rng(chrono::steady_clock::now().time_since_epoch().co
unt());

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

// gen_random(l, r); // generates a random integer
between l and r
//shuffle korar function
shuffle(all(v),rng);
```

Euler tour(Tree to Arr/Query on Tree):

```
vector<int>dfsarr;

int st[N],en[N];
ll a[N];
// node i subtree  { dfsarr[st[i],en[i]] }
void dfs(int u,int par)// 1 based
{
   dfsarr.pb(u);
   st[u]=sz(dfsarr)-1;//starting idx of node u in dfs array
   for(int v :edge[u])
   {
      if(v==par) continue;
      dfs(v,u);
   }
   en[u]=sz(dfsarr)-1;//ending idx of node u in dfs array
}
```

Sqrt Decomposition:

```
int a[500001];
int block = 710;
vector<int> b[710];
void solve()
{
   int i, j, n, q, x, y, z, cnt, end, lb;
   scanf("%d", &n);
   for (i = 0; i < n; i++)
   {
      cin >> a[i];
      b[i / block].pb(a[i]);
   }
   for (i = 0; i < block; i++)
   {
      sort(all(b[i]));
   }
   scanf("%d", &q);
   while (q--)
   {
      int type;
      scanf("%d", &type);
      if (type == 0)
      {
         scanf("%d%d%d", &x, &y, &z);
         x--;
         y--;
         cnt = 0;
         int lf_block = x / block;
         int rg_block = y / block;
         if (lf_block == rg_block)
         {
            for (i = x; i <= y; i++)
            {
               cnt += (a[i] > z);
            }
         }
         else
         {
            end = ((lf_block + 1) * block);
            for (i = x; i < end; i++)
            {
               cnt += (a[i] > z);
            }
            for (i = (lf_block + 1); i < rg_block; i++)
            {
               lb = lower_bound(all(b[i]), (z + 1)) -
b[i].begin();
               cnt += (sz(b[i]) - lb);
            }
            for (i = rg_block * block; i <= y; i++)
            {
               cnt += (a[i] > z);
            }
         }
         printf("%d\n", cnt);
      }
      else
      {
         scanf("%d%d", &x, &y);
         x--;
         z=(x/block);
         for(i=0;i<sz(b[z]);i++)
         {
            if(b[z][i]==a[x])
            {
               b[z][i]=y;
               break;
            }
         }
         a[x]=y;
         sort(all(b[z]));
      }

   }
}
```

Tree Re-rooting:

```
void dfs(int u,int p)
{
siz[u]=1;
   for(int v : edge[u])
   {
      if(v!=p)
      {
         dfs(v,u);
         siz[u]+=siz[v];
         score[u]+=score[v];
      }
   }
   score[u]+=siz[u];
}
ll ans=0;
void change_root(int u,int p)
{
   ans=max(ans,score[u]);
    for(int v : edge[u])
   {
      if(v!=p)
      {
         ll xx,yy,zz,qq;// temporary store
         xx=score[u];
         yy=siz[u];
         zz=score[v];
         qq=siz[v];
         score[u]-=score[v]; // child er score badh
         score[u]-=siz[u];  // prv size badh dibo
         siz[u]-=siz[v];   //new size ber korbo
         score[u]+=siz[u]; // new size add korbo
         score[v]-=siz[v];// prv size badh dibo
         siz[v]+=siz[u];  // new size ber korbo
         score[v]+=siz[v]; // new size add korbo
         score[v]+=score[u]; // child er score ber korbo
         change_root(v,u);
        // previous value dhara replace kore dichi
         score[u]=xx;
         siz[u]=yy;
         score[v]=zz;
         siz[v]=qq;
      }
   }
}
```

Tree Diameter(Farthest Node Method):

```
int f1[100005]; // max farthest
int f2[100005]; // 2nd max farthest
void dfs1(int u, int par)
{
   f1[u] = 0;
   f2[u] = 0;
   for (auto v : edge[u])
   {
      if (v == par)
         continue;
      dfs1(v, u);
      if (f1[u] < f1[v] + 1)
      {
         f2[u] = f1[u];
         f1[u] = f1[v] + 1;
      }
      else if (f2[u] < f1[v] + 1)
      {
         f2[u] = f1[v] + 1;
      }
   }
   return;
}
void dfs2(int u, int par)
{
   if (par != -1)
   {
      int tmp;
      if (f1[par] == f1[u] + 1)
         tmp = f2[par];
      else
         tmp = f1[par];
      if (f1[u] < tmp + 1)
      {
         f2[u] = f1[u];
         f1[u] = tmp + 1;
      }
      else if (f2[u] < tmp + 1)
      {
         f2[u] = tmp + 1;
      }
   }
   for (auto v : edge[u])
   {
      if (v == par)
         continue;
      dfs2(v, u);
   }
}
void solve()
{
  // ll i, j, k, n, m, p, q, x, y, z, u, v, l, r, mod = 1e9 + 7, mx,
mn, mx1, mn1, cnt1, cnt;
   cin >> n;
   ll a[n + 5];
   for (i = 0; i < n - 1; i++)
   {
      cin >> u >> v;
      edge[u].pb(v);
      edge[v].pb(u);
   }
   dfs1(1, -1);
   dfs2(1, -1);
   int ans = 0;
   for (i = 1; i <= n; i++)
   {
      ans = max(ans, f1[i]);
   }
   cout << ans << endl;
}
```

Sparse Table:

```
ll sp[20][100005];
int LOG[1000005];
int a[100010];
void prelog()
{
   LOG[1] = 0;
   for (ll i = 2; i <= 100005; i++)
   {
      LOG[i] = LOG[i >> 1] + 1;
   }
}
void build(int n)
{
   for (int i = 1; i <= n; i++)
   {
      sp[0][i] = a[i];
   }
   for (ll k = 1; k <= LOG[n]; k++)
   {
      for (ll i = 1; i + (1 << k) - 1 <= n; i++)
      {
         sp[k][i] = __gcd(sp[k - 1][i], sp[k - 1][i + (1 << (k -
1))]);
      }
   }
}
ll query(int left, int right)
{
   int len = (right - left) + 1;
   int k = LOG[len];
   return __gcd(sp[k][left], sp[k][right - (1 << k) + 1]); //
change
}
```

Tree Ancestor + Re-rooted LCA:

```
int dep[N];
int anc[N][20];
// anc[x][i] = means 2^i th ancestor of x

void dfs(int u,int par)
{
   for(int i=0;i<20;i++)
   {
      anc[u][i] = -1;
   }
   if(par!=-1)
   {
      dep[u]=dep[par]+1;
      anc[u][0] = par;
      for(int i=1;i<20;i++)
      {
         int v=anc[u][i-1];
         if(v==-1)   break;
         anc[u][i] = anc[v][i-1];
      }
   }
   for(int v: edge[u])
   {
      if(v!=par)
      {
         dfs(v,u);
      }
   }
}

int getanc(int u,int k)/// find u node --> kth ancestor
{
   for(int i=0;i<20;i++)
   {
      if(k&(1<<i))
      {
         u = anc[u][i];
         if(u==-1)return -1;
      }
   }
 return u;
}
int lca(int u,int v)/// find lowest common ancestor of
node u and v
{
   if(dep[u]>dep[v])
   {
      u= getanc(u,dep[u]-dep[v]);
   }
   else if(dep[v]>dep[u])
   {
      v = getanc(v,dep[v]-dep[u]);
   }
   if(u==v) return u;
   for(int i=19;i>=0;i--)
   {
      if(anc[u][i]!=anc[v][i])
      {
         u = anc[u][i];
         v = anc[v][i];
      }
   }
   return anc[u][0];
}
 int rottedlca(int u,int v,int r)/// find lca of u and v in the
subtree rooted at r
 {
    int x = lca(u,r);
    int y = lca(v,r);
    int z= lca(u,v);
    if(x==y) return z;
    if(y==z) return x;
    return y;
 }
void solve()
{
    ll
i,j,k,n,m,p,q,x,y,z,u,v,l,r,mod=1e9+7,mx,mn,mx1,mn1,cn
t1,cnt;
    cin>>n;
    for(i=2;i<=n;i++)
{
cin>>u>>v;
      edge[u].pb(v);
      edge[v].pb(u);
    }
    dfs(1,-1);
    cin>>q;
    while(q--)
    {
      cin>>k;
      cin>>u>>v;
      bool f=0;
      for(i=2;i<=k;i++)
      {
        cin>>l>>r;
        if(u==-1) continue;
        int xx=rottedlca(u,r,l);
        int yy=rottedlca(v,r,l);
       // int zz=rottedlca(l,v,u);
        int zz=rottedlca(u,v,l);
       if(xx==yy && zz!=xx)
       {
         xx=yy=-1;
         f=1;
       }
       u=xx;
       v=yy;
      }
      if(f)
      {
        cout<<0<<endl;
      }
      else
      {
        cout<<dep[u]+dep[v]-2*dep[lca(u,v)]+1<<endl;
      }
    }
    for(i=1;i<=n;i++)
    {
      edge[i].clear();
    }

}
```

Topological-sort:

```
queue<ll> q;
   for (i = 1; i <= n; i++)
   {
      if (indeg[i] == 0)
      {
         q.push(i);
      }
   }
   x = 0;
   while (!q.empty())
   {
      ll node = q.front();
      q.pop();
      x++;

      for (auto it : edge[node])
      {
         indeg[it]--;
         if (indeg[it] == 0)
         {
            q.push(it);
         }
      }
   }
```

Custom Hash:

```
namespace std {
   template <>
   struct hash<std::pair<long long, long long>> {
      size_t operator()(const std::pair<long long, long
long>& p) const {
         // Use a small prime number as a multiplier for
better distribution
         const size_t prime = 31; // Small prime multiplier
         std::hash<long long> hasher;
         return hasher(p.first) ^ (hasher(p.second) +
prime);
      }
   };
}
```

Extra:

```
ll ans[500005];
ll pre[500005][32];
void solve()
{
   ll i, j, k, n, m, p, q, x, y, z, u, v, l, r, mod = 1e9 + 7, mx,
mn, mx1, mn1, cnt1, cnt;
   cin >> n >> q;
   ll a[n + 5];
   for (i = 1; i <= n; i++)
   {
      cin >> a[i];
   }
   for (j = 0; j < 31; j++)
   {
      ll mask = (1LL << j);
      for (i = 1; i <= n; i++)
      {
         x = mask & a[i];
         pre[i][j] = 0;
         if (x > 0)
         {
            pre[i][j] += pre[i - 1][j] + 1;
         }
      }
   }
   for (i = 1; i <= n; i++)
   {
      vector<pair<ll, ll>> pp;
      for (j = 0; j < 31; j++)
      {
         if (pre[i][j])
         {
            pp.pb({pre[i][j], (1LL << j)});
         }
      }
      sort(all(pp), cmp);
      ll now = 0;
      for (j = 0; j < sz(pp); j++)
      {
         ll len = pp[j].fi;
         ll val = pp[j].sc;
         now |= val;
         ans[len] = max(ans[len], now);
      }
   }
   mx = 0;
   for (i = n; i >= 1; i--)
   {
      mx = max(mx, ans[i]);
      ans[i] = mx;
   }
   while (q--)
   {
      cin >> x;
      cout << ans[x] << endl;
   }
}
```