```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define vpnt(ans) for(ll i = 0; i <
ans.size(); i++)
            cout << ans[i] << (i + 1 <
ans.size() ? ' ' : '\n');
#define setbit(x, k) (x |= (1LL << k))
#define clearbit(x, k) (x &= ~(1LL << k))
#define checkbit(x, k) (x & (1LL << k))
#define pb push_back
#define ff first
#define ss second
#define pii pair<int, int>
#define pll pair<ll, ll>
#define nn cout << "\n";
#define INF (1 << 30)
#define LL_INF (1LL << 62)
#define pp(n, p) (ll)(pow(n, p) + 0.5)
#define fastio
ios::sync_with_stdio(false);cin.tie(nullptr);
Yes, no
const ll mod = 1e9 + 7;
const int N = 2e6 + 10;
```

Base Conversion:

```
ll to_deci (ll n, int base){
    ll sum = 0, x = 0, p = 1;
    while(n>0)  {
        sum += (n%10 * p);
        N /= 10;  x++;
        p *= base;
    }
    return sum;
}
ll from_deci (ll n, int base) {
    int x=0;
    int num [40] = {0};
    while(n>0) {
        num[x] = n%base;
        n /= base; x++;
    }
    ll res = num[x-1];
    for(int i = x-2; i >= 0; i--)
        res = res*10+num[i];
    return res;
}
```

Big Integer:

```
string add(string x, string y) {
    string res = "";
    int p1 = x.size() - 1, p2 = y.size() - 1;
    int carry = 0;
    while (p1 >= 0 || p2 >= 0) {
        int d1 = (p1 >= 0) ? x[p1] - '0' : 0;
        int d2 = (p2 >= 0) ? y[p2] - '0' : 0;
        int d = (d1 + d2 + carry);
        carry = d / 10;
        d %= 10;
        res += ('0' + d);
        p1--, p2--;
    }
    if (carry)  res += ('0' + carry);
    while (res.size() > 1 && res.back()=='0')
                            res.pop_back();
    reverse(res.begin(), res.end());
    return res;
}
string multiply(string x, string y) {
    string res = "0";
    for (int i = x.size() - 1; i >= 0; i--) {
        string r = "";
        for(int k = x.size()-1; k>i; k--)  r+='0';
        int d1 = x[i] - '0';
        int carry = 0;
        for(int j = y.size()-1; j>=0; j--) {
            int d2 = y[j] - '0';
            int d = d1 * d2 + carry;
            carry = d / 10;
            d %= 10;
            r += ('0' + d);
        }
        if (carry) r += ('0' + carry);
        reverse(r.begin(), r.end());
        res = add(res, r);
    }
    return res;
}
string multplyStrngNum(string s,ll n) {
    //s is a reversed string, returns a
reversed string
    string res = "";
    ll carry = 0;
    for (int i = 0; i < s.size(); i++) {
        carry += n * (s[i] - '0');
        res += '0' + carry % 10;
        carry /= 10;
    }
    while (carry) {
        res += '0' + carry % 10;
        carry /= 10;
    }
    return res;
}
string num_to_string(ll n) {
    string s = "";
    while (n) {
        s += '0' + n % 10;
        n /= 10;
    }
    reverse(s.begin(), s.end());
    return s;
}
string div_by_2(string x) {
    int p = 0, rem = 0;
    string res = "";
    while (p < x.size()) {
        int num = rem * 10 + (x[p] - '0');
        if (num <  2) {
            if (p + 1 < x.size())
                num=num*10+(x[p+1]-'0');
            else {
                res += '0';
                break;
            }
            if (res.size())  res += '0';
            p++;
        }
        res += ('0' + num / 2);
        rem = num % 2;
        p++;
    }
    return res;
}
ll mod_string_num(string s, ll x) {
    ll res = 0, p = 1;
    for(int i=s.size()-1; i>=0; i--) {
        res = (res + (p * (s[i] - '0'))%x)%x;
        p = (p*10)%x;
    }
    return res;
}
```

Binary Search:

```
// for 0 0 0 1 1 1, to get the first 1
ll bins(ll l, ll h)  {
    while (l < h) {
        ll mid = (l + h) / 2;
        if (valid(mid))  h = mid;
        else  l = mid + 1;
    }
    return l;
}
// for 1 1 1 0 0 0, to get the last 1
ll bs(ll l, ll h) {
    while (l != h) {
        ll mid = (l + h) / 2;
        if ((l + h) % 2 != 0)  mid++;
        if (valid(mid))  l = mid;
        else  h = mid - 1;
    }
    return l;
}
```

Binlift__LCA:

```
int n, LOG = 0;
vector<int>g[NN];
int depth[NN];
int up[NN][Log];
//kono tree er k-th ancestor = binlift
void binlift(int nd) {
    for(int y: g[nd])  {
        if(depth[y] != -1) continue;
        depth[y] = depth[nd] + 1;
        up[y][0] = nd;
        for(int i = 1; i < LOG; i++) {
            up[y][i] = up[up[y][i-1]][i-1];
        }
        binlift(y);
    }
}
int lca(int x, int y) //kono tree er lca = lca,
{
    if(depth[x] < depth[y]) swap(x, y);
    int k = depth[x] - depth[y];
    for(int i=LOG; i>=0; i--) {
        if(checkbit(k, i)) x = up[x][i];
    }
    // depths are now same;
    if(x == y) return x;
    for(int i=LOG; i>=0; i--) {
        int a = up[x][i];
        int b = up[y][i];
        if(a != b){
            x = up[x][i];
            y = up[y][i];
        }
    }
    return up[x][0];
}
```

```
int main()
{
    cin>>n;
    while((1 << LOG) < n) LOG++;
    for(int i=0; i<n-1; i++)  {
        cin>>x>>y;
        g[x].pb(y);
        g[y].pb(x);
    }
    int root = 1; //ba ja hoy
    memset(depth, -1)
    depth[root] = 0;
    for(int i = Log; i>=0; i--) up[root][i] = 0;
    binlift(root); //up[nd][LOG_VAL] = nd er
2^LOG_VAL tomo ancestor janlam.. if(0)
out of tree (1 indexed nodes)
    cin>>q;
    while(q--) {
        cin>>x>>y;
        cout<<lca(x, y);nn;
    }
}
```

BIT: // 1 based

```
ll bit[N + 5];
int a[N];
int n;
ll pre_sum(ll x) {
      ll res = 0;
      while (x > 0) {
            res += bit[x];
            x -= (x & -x);
      }
      return res;
}
void update_pos(ll x, ll add) {
       while (x <= n) {
            bit[x] += add;
            x += (x & -x);
       }
}
void update_range(int l, int r, ll add) {
       update_pos(l, add);
       update_pos(r + 1, -add);
}
int main() {
       cin>>n;
       //memset(bit, 0); [if not 0]
       for(int i=1; i<=n; i++)  {
            cin>>a[i];  update_pos(i, a[i]);
       } cout<<pre_sum(3); nn;
}
```

Bridge finding graph:

```
vector<int>g[N];
int low[N], tin[N];
int n, tim = 1;
vector<pii>br;
void dfs(int nd , int p) {
    if(tin[nd]) return;
    tin[nd] = low[nd] = tim++;
    for(int y: g[nd]) {
        if(y == p) continue;
        dfs(y, nd);
        if(low[y] > tin[nd])
                br.pb({min(nd, y), max(nd, y)});
        low[nd] = min(low[nd], low[y]);
    }
    return;
}
void find_bridges() {
    for(int i=1; i<=n; i++) {
        if(tin[i] == 0) dfs(i, -1);
    }
}
int main() {
   find_bridges();//& store the pairs in vct v
}
```

BFS:

```
int vis[N] = {0}, src = 1;
queue<int> q;
q.push(src);  vis[src] = 1;
while (!q.empty()) {
    int u = q.front();
    q.pop();
    for (int v: g[u]) {
        if (vis[v]) continue;
        vis[v] = 1;
        q.push(v);
    }
}
```

Dijkstra:

```
int src = 1;
int dis[N];
for(int i=0; i<N; i++)  dis[i]=(1 << 30);
priority_queue<pii> pq;
pq.push({0, src});
dis[src] = 0;
while (!pq.empty()) {
     int u = pq.top().second;
     pq.pop();
     for (pii v : g[u]) {
          if(dis[v.ff]<=dis[u]+v.ss) continue;
          dis[v.ff] = dis[u]+v.ss;
          pq.push({-dis[v.ff],v.ff});
     }
}
```

DSU:

```
ll par[N], sz[N];
class DSU
{
    public:
    DSU(int n) {
        for(int i=0; i<=n; i++) {
            par[i] = i;
            sz[i] = 1;
        }
    }
    int find_par(int x) {
        if(par[x] == x) return x;
        return par[x] = find_par(par[x]);
    }
    void union_set(int x, int y) {
        x = find_par(x);
        y = find_par(y);
        if(x == y) return;
        if(sz[x] < sz[y]) swap(x, y);
        par[y] = x;
        sz[x] += sz[y];
    }};
```

ETF:

```
int phi[N + 2];
void ETF() {
        phi[1] = 0;
        for (ll i = 2; i < N; i++)  phi[i] = i;
        for (ll i = 2; i < N; i++) {
            if (phi[i] != i) continue;
            for (ll j=i; j<N; j+=i) phi[j]-=(phi[j]/i);
        }
}
```

ETF for a number n:

```
ll phi(ll n) {
 ll res = n;
 for (ll i=0; p[i]*p[i] <= n; i++) {
     if (n % p[i]== 0) {
          // subtract multiples of p[i] from r
          res -= (res / p[i]);
          while (n % p[i]== 0)
          n /= p[i];
     }
 }
 if (n > 1)  res -= (res / n);
 return res;
}
```

EXTENDED_EUCLIDEAN :

```
ll gcd(ll a, ll b, ll &x, ll &y) {
       if(b == 0) {
              x = 1; y = 0;
              return a;
       }
       ll x1, y1;
       ll d = gcd(b, a % b, x1, y1);
       x = y1;
       y = x1 - y1 * (a / b);
       return d;
}
```

HASH:

```
ll p1[N], p2[N], base1 = 1e9 + 21, base2
= 1e9 + 181;
ll h1[N], h2[N];
string s;  // s is 1 based
void init_power_calc() {
       p1[0] = p2[0] = 1;
       for (int i = 1; i < N; i++) {
            p1[i] = (p1[i - 1] * base1) % m1;
            p2[i] = (p2[i - 1] * base2) % m2;
       }
}
class hsh {
public:
  void hash_calc(string &s) {
      h1[0] = h2[0] = 0;
      int sl = s.length() - 1; // 1 based
      for (int i = 1; i <= sl; i++) {
         h1[i]=(h1[i-1]*base1+s[i])%m1;
         h2[i]=(h2[i-1]*base2+s[i])%m2;
      }
  }
  ll get_hash_val(int l, int r) {
    ll hash1=(h1[r] - h1[l-1]* p1[r-l+1])%m1;
    ll hash2=(h2[r] - h2[l-1]* p2[r-l+1])%m2;
    if (hash1 < 0)   hash1 += m1;
    if (hash2 < 0)   hash2 += m2;
    return (hash1 << 32) | hash2;
  }
  ll hash_1(int l, int r) {
    ll hash1=(h1[r] - h1[l-1]* p1[r-l+1])%m1;
    if (hash1 < 0)  hash1 += m1;
    return hash1;
  }
 ll hash_2(int l, int r) {
  ll hash2=(h2[r] - h2[l-1]* p2[r-l+1])%m2;
  if (hash2 < 0)  hash2 += m2;
  return hash2;
 }
} var;
```

KMP:

```
const int MAX_PATTERN_SIZE =
2000000;
int F[MAX_PATTERN_SIZE + 10];
void build_failure_function(string pattern){
      int m = pattern.size();
      F[0] = F[1] = 0;
      for (int i = 2; i <= m; i++) {
            int j = F[i - 1];
            while (true) {
                if (pattern[j] == pattern[i - 1]){
                      F[i] = j + 1; break;
                }
                if (j == 0) {
                      F[i] = 0; break;
                }
                j = F[j];
            }
      }
}
bool KMP(string text, string pattern) {
      int n = text.size(), m = pattern.size();
      build_failure_function(pattern);
      int i = j = 0;
      while (true) {
            if (j == n)  return false;
            if (text[j] == pattern[i])  {
                  i++; j++;
                  if (i == m)  return true;
            }
            else {
                  if (i == 0)  j++;
                  else    i = F[i];
            }
      }
}
int main() {
      cin >> text >> pattern;
      cout << KMP(text, pattern);
      return 0;
}
```

KMP_POS :

```
const int MAX_PATTERN_SIZE = 2e6;
int F[MAX_PATTERN_SIZE + 10];
void build_failure_function(string pattern){
    int m = pattern.size();
    F[0] = F[1] = 0;
    for (int i = 2; i <= m; i++) {
        int j = F[i - 1];
        while (true) {
            if (pattern[j] == pattern[i - 1]) {
                F[i] = j + 1; break;
            }
            if (j == 0) {
                F[i] = 0; break;
            }
            j = F[j];
        }
    }
}
```

LIS:

```
int lis(vector<int> v) {
  vector<int> a(v.size() + 1, INF);
  a[0] = -INF;
  for (int i = 0; i < v.size(); i++) {
     int j = upper_bound(a.begin(), a.end(),
v[i]) - a.begin();
    if (a[j-1] < v[i] && v[i] < a[j])  a[j] = v[i];
  }
  int mx = 0;
  for (int i = 0; i < v.size() + 1; i++)
          if (a[i] < INF)  mx = i;
  return mx;
}
```

LINEAR DIOPHANTINE: (x>0 && y>0)

```
ll a = 11, b = 111, xx, yy;
ll g = gcd(11, 111, xx, yy);
while (t--) {
      cin >> n;
      ll x = xx * n,  y = yy * n;
      if (x < 0) {
            ll k = (abs(x) * g + b - 1) / b;
            x += k * b / g;
            y -= k * a / g;
      }
      else if (y < 0) {
            ll k = (abs(y) * g + a - 1) / a;
            x -= k * b / g;
            y += k * a / g;
      }
      if (x >= 0 && y >= 0) YES;
      else NO;
}
```

Min weight tree:

```
// ekta weighted undrcted graph thakbe,
oitar edge  kete tree banayte hbe so that
maximum weight ta minimized hoy
vector<pii>g[N];
int vis[N];
void dfs(int x, int mx)
{
    if(vis[x]) return;
    vis[x] = 1;
    for(pii y : g[x])    {
        if(y.ss <= mx) dfs(y.ff, mx);
    }
    return;
}
int check(int mx) {
    memset(vis, 0, sizeof(vis[0])*(n+1));
    dfs(1, mx);
    int f = 1;
    for(int i=1; i<=n; i++) f &= vis[i];
    return f;
}
int bins(int lo, int hi) {
    while(lo<hi) {
        int mid = (lo+hi)/2;
        if(check(mid)) hi = mid;
        else lo = mid+1;
    }
    return lo;
}
```

MO's Algorithm:

```
ll n, a[N], bsz, sum = 0, res[N];
struct que{ int l, r, ind; };
bool cmp(que a, que b) {
 if(a.l/bsz != b.l/bsz) return a.l/bsz<b.l/bsz;
 return a.r < b.r;
}
void add(ll x) { sum += a[x]; }
void del(ll x) { sum -= a[x]; }
int main()
{
    cin>>n;
    bsz = sqrt(n) + 3;
    for(int i=1; i<=n; i++) cin>>a[i];
    cin>>nq;
    vector<que>q;
    for(int i=1; i<=nq; i++) {
        cin>>l>>r;
        q.pb({l, r, i});
    }
    sort(q.begin(), q.end(), cmp);
    int l = 1, r = 0; // for 1 based
    sum = 0;
    for(int i=0; i<nq; i++)
    {
        while(l > q[i].l) add(--l);
        while(r < q[i].r) add(++r);
        while(l < q[i].l) del(l++);
        while(r > q[i].r) del(r–);
        res[q[i].ind] = sum;
    }
    for(int i=1; i<=nq; i++) cout<<res[i];
}
```

Mod:

```
ll fact[N];
void factmod(int md) {
      fact[0] = 1 % md;
      for (int i = 1; i < N; i++)
            fact[i] = (fact[i - 1] * i) % md;
}
ll bp(ll n, ll p, int md)  {
      n = n % md;
      ll res = 1 % md;
      while (p > 0) {
            if (p & 1)  res = (res * n) % md;
            p >>= 1;
            n = (n * n) % md;
       }
       return res;
}
//md must be prime
ll invmod(ll ja_diye_vag, ll md){
      return bp(ja_diye_vag, md - 2, md);
}
//md must be prime
ll ncr(ll n, ll r, ll md) {
 return ((fact[n] * invmod(fact[r], md))%md
            * invmod(fact[n - r], md)) % md;
}
```

nCr time & space optimized:

```
int nCr(int n, int k) {
  int res = 1;
  if ( k > n - k ) k = n - k;
  for (int i = 0; i < k; ++i) {
     res *= (n - i);
     res /= (i + 1);
  }
  return res;
}
```

MST:

```
// KRUSKAL:
// vector a {w, {u, v}} rekhe, w er choto
theke boro sort korbo.
// erpor u, v same set a na hole (dsu diye)
oi edge nibo & u,v union kore dibo
// DONE! O(ElogE + ElogV) (sorting + v
bar union-find)
// alada component thakle sobgular alada
mst hoy
int n, m;
vector<pii> g[N];
vector<pair<int, pii>> v;
ll par[N], sz[N];
//DSU
```

```
void mst() {
    DSU dsu(n);
    ll res = 0;
    for (int i = 0; i < v.size(); i++) {
        int x = dsu.find_par(v[i].ss.ff);
        int y = dsu.find_par(v[i].ss.ss);
        if (x == y) continue;
        res += v[i].ff;
        dsu.union_set(x, y);
    }
    cout << res; nn;
}
int main() {
    cin >> n >> m;
    for (int i = 0; i < m; i++) {
        cin >> x >> y >> w;
        g[x].pb({y, w});
        g[y].pb({x, w});
        v.pb({w, {x, y}});
    }
    sort(v.begin(), v.end());
    mst();
}
// PRIM's
sobtheke kom weight edge theke shuru
kore, vis node gular sathe cnncted emon
sob edge er moddhe abar min ta nibo
// O(ElogV)
// 1+ component process kore na
vector<pll> g[N];
bool vis[N];
void prim_mst(ll mnw, ll mnx, ll mny) {
    priority_queue<pair<ll, pll>> pq;
    pq.push({-mnw, {mnx, mny}});
    ll res = 0;
    while (!pq.empty()) {
        ll w = -pq.top().ff;
        ll x = pq.top().ss.ff;
        ll y = pq.top().ss.ss;
        pq.pop();
        if (vis[x] && vis[y]) continue;
        vis[x] = vis[y] = 1;
        for (auto v : g[x]) {
            if (vis[v.ff])  continue;
            pq.push({-v.ss, {x, v.ff}});
        }
        for (auto v : g[y]) {
            if (vis[v.ff]) continue;
            pq.push({-v.ss, {y, v.ff}});
        }
        res += w;
    }
    cout << res; nn;
}
int main()
{
    int n, m;
    cin >> n >> m;
    ll mnw = LL_INF, mnx, mny;
    if (m == 0) {
        cout << 0 << endl;
        return 0;
    }
    while (m--)
    {
        cin >> x >> y >> w;
        g[x].pb({y, w});
        g[y].pb({x, w});
        if (w < mnw) {
            mnw = w; mnx = x;  mny = y;
        }
    }
    prim_mst(mnw, mnx, mny);
}
```

MAXIMUM_BIPARTITE_MATCHING :

```
int n, k, a[N], vis[N], occupied[N];
ll kar_kase[N];
vector<ll> g[N];  vector<pair<ll, pll>> v;
bool kuhn(ll cus) {
      for (auto bag : g[cus]) {
            if(occupied[bag] == 1) continue;
            occupied[bag] = 1;
            if (kar_kase[bag] == -1 ||
                        kuhn(kar_kase[bag]))
            {
                kar_kase[bag] = cus;
                return true;
            }
      }
      return false;
}
ll res = 0;  //memset(kar_kase,-1);
for (int i = 0; i < q; i++){
  memset(occupied,-1,sizeof(occupied));
  if (kuhn(i))   res += v[i].ss.ff;
}
```

PRIME_Sieve:

```
bitset<1000005> pr;
vector<int> p;
void siv() {
    ull i, j;
    pr[1] = 1; p.pb(2);
    for (i = 3; i < N; i += 2) {
        if (pr[i]) continue;
        for(j=i*i; j<N; j+=2*i) pr[j] = 1;
        p.pb(i);
    }
}
```

PRIME_Divisor:

```
int p[N];
vector<ll> p;
//sieve
vector<int>divs(int n) {
    vector<int> res;
    for (int i = 0; i < p.size(); i++) {
        if (n % p[i] == 0) {
            res.pb(p[i]);
            while (n % p[i] == 0) n /= p[i];
        }
        if(p[i] * p[i] > n) break;
    }
    if (n != 1) res.pb(n);
    return res;
}
```

PRIME_Segmented Sieve:

```
bitset<1000005> pr, segpr;
vector<int> p;
//siv code
void segsiv(ll a, ll b) {
    ull i, j;
    ll r = sqrt(b) + 1;
    if(a <= 1)  segpr[1 - a] = 1;
    for (i = 0; p[i] <= r; i++) {
        j = floor(a / p[i]) * p[i];
        if (j < a) j += p[i];
        if (p[i] == j) j += p[i];
        for ( ; j<=b; j+=p[i]) {
            if (j < a) continue;
            segpr[j - a] = 1;
        }
    }
}
```

SCC_Condensation_Graph:_

```
int n, m;
vector<int>g[N];
vector<int>gRev[N];
vector<int>condensation[N];
vector<int>ord;
vector<int>comp;
int root[N], vis[N];
// kon order a visit korsi tar ultata rakha
void dfs1(int nd) {
    vis[nd] = 1;
    for(int y: g[nd]){
        if(vis[y]) continue;
        dfs1(y);
    }
    ord.pb(nd);
}
//age je order a visit korsi, same order a
rev graph a visit //kora & jotogula comp =
totogula SCC
void dfs2(int nd) {
    vis[nd] = 1;
    comp.pb(nd);
    for(int y: gRev[nd]){
        if(vis[y]) continue;
        dfs2(y);
    }
}
int main() {
    cin>>n>>m;
    while(m--){
        cin>>x>>y;
        g[x].pb(y);
        gRev[y].pb(x);
    }
    memset(vis, 0, sizeof(vis[0])*(n+2));
    for(int i=1; i<=n; i++){
        if(vis[i]) continue;
        dfs1(i);
    }
    memset(vis, 0, sizeof(vis[0])*(n+2));
    reverse(ord.begin(), ord.end());
    for(int y : ord){
        if(vis[y]) continue;
        dfs2(y);
        /*
            //processings: (here,
                  condensation nodes create)
            // full comp = 1 node(root)
            int now_root = comp.front();
            for(int y: comp){
                root[y] = now_root;
            }
            roots.pb(root);
            //lagle (roots vector declare kore)
        */
        comp.clear();
    }
    //condnstn graph create-SCC gular
moddhe edge dewa
    for(int nd=1; nd<=n; nd++) {
        for(int y: g[nd]){
            if(root[nd] != root[y]) {
            condensation[root[nd]].pb(root[y]);
```

```
        }
      }
   }
}
```

Segment Tree:

```
ll n, a[N], sg[4*N], lz[4*N];
// building the tree (4N)
void build(int nd, int st, int en) {
      if (st == en) {
         sg[nd] = a[st]; return;
      }
      int mid = (st + en) / 2;
      build(nd * 2, st, mid);
      build(nd * 2 + 1, mid + 1, en);
      sg[nd]=max(sg[nd*2],sg[nd*2+1]);
}
// updating range of lazy prop (logN)
void update_r(int nd, int st, int en,  int l, int
r, ll add)
{
      if(lz[nd] != 0) {   //nd needs update
            sg[nd] += lz[nd] * (en - st + 1);
            if (st != en) {
                  lz[nd * 2] += lz[nd];
                  lz[nd * 2 + 1] += lz[nd];
            }
          lz[nd] = 0;
      }
      if (st > r || en < l) return;
      if (st >= l && en <= r) {
            sg[nd] += add * (en - st + 1);
            if (st != en) {
                  lz[nd * 2] += add;
                  lz[nd * 2 + 1] += add;
            }
            return;
      }
      int mid = (st + en) / 2;
      update_r(nd*2,st,mid,l,r,add);
      update_r(nd*2+1,mid+1,en,l,r,add);
      sg[nd] = max(sg[nd*2],sg[nd*2+1]);
}
// updating a single pos (logN)
void update(int nd, int st, int en, int pos,
int val)
{
      if (st == en) {   // == pos
            a[pos] = val;  sg[nd] = val;
            return;
      }
      int mid = (st + en) / 2;
if(pos<=mid) update(nd*2,st,mid,pos,val);
else    update(nd*2+1,mid+1,en,pos,val);
sg[nd] = max(sg[nd*2], sg[nd*2+1]);
}// range query (logN)
ll query(int nd, int st, int en, int l, int r)
{
    if (st > r || en < l) return -INF;
    if (lz[nd] != 0) {
        sg[nd] += lz[nd] * (en - st + 1);
        if (st != en) {
            lz[nd * 2] += lz[nd];
            lz[nd * 2 + 1] += lz[nd];
        }
        lz[nd] = 0;
     }
    if(st>=l && en<=r) return sg[nd];
    int mid = (st + en) / 2;
    return max(query(nd * 2, st, mid, l, r),
          query(nd * 2 + 1, mid + 1, en, l, r));
}
```

SEGMENT TREE_Persistent:

```
ll n, a[N], avl = 1;
ll sg[msz], lt[msz], rt[msz], vrsn[N];
void build(ll nd, ll st, ll en) {
    if(st == en) {
        sg[nd] = 0;
        return;
    }
    ll mid = (st + en)/2;
    lt[nd] = ++avl;
    rt[nd] = ++avl;
    build(lt[nd], st, mid);
    build(rt[nd], mid+1, en);
    sg[nd] = sg[lt[nd]] + sg[rt[nd]];
}
ll upd(ll nd, ll st, ll en, ll pos, ll val)
{
    if(st > pos || en < pos) return nd;
    ll newnd = ++avl;
    if(st == en) {
        sg[newnd] = sg[nd] + val;
        return newnd;
    }
    ll mid = (st + en)/2;
  lt[newnd] = upd(lt[nd], st,mid, pos, val);
  rt[newnd]= upd(rt[nd],mid+1,en,pos,val);
  sg[newnd] = sg[lt[newnd]]+sg[rt[newnd]];
  return newnd;
}
ll query(ll nd, ll st, ll en, ll l, ll r) {
    if(st > r || en < l) return 0;
    if(st >= l && en <= r) return sg[nd];
    ll mid = (st + en)/2;
    return query(lt[nd], st, mid, l, r) +
              query(rt[nd], mid+1, en, l, r);
}
int main() {
    cin>>n;
    for(int i=1; i<=n; i++) cin>>a[i];
    vrsn[0] = 1;
    build(1, 1, n);
    int cur_vr = 0;
    cin>>q;
    while(q--)
    {
        cin>>type;
        if(type == 1) {
            int vr, pos, val;
            cin>>vr>>pos>>val;
            vrsn[++cur_vr] = upd(vrsn[vr], 1,
                                  n, pos, val);
        }
        else {
            cin>>vr>>l>>r;
            cout<<query(vrsn[vr], 1, n, l, r);nn;
        }
    }
}
```

Square Root Decomposition:

```
vector<vector<int>> v;
vector<pii> range;
vector<int> mx;
void init(vector<int> x) {
  int n = x.size();
  int m = sqrt(n), l = 1;
    while (l <= n) {
      int r = min(n, l + m - 1);
      vector<int> temp;
      int mxx = -INF;
      for (int i = l; i <= r; i++) {
          temp.pb(x[i - 1]);
          mxx = max(mxx, x[i - 1]);
      }
      v.pb(temp);
      range.pb({l, r});
      mx.pb(mxx);
      l = r + 1;
    }
}
void update(int pos, int val) {
    int n = v.size();
    for (int i = 0; i < n; i++) {
        if (pos > range[i].ss) continue;
        if (pos < range[i].ff) break;
        int mxx = -INF;
     for(int j=range[i].ff; j<=range[i].ss; j++){
            int ind = j - range[i].ff;
            if (j == pos) v[i][ind] = val;
            mxx = max(mxx, v[i][ind]);
       }
       mx[i] = mxx;
    }
}
int query(int l, int r) {
    int n = v.size(), mxx = -INF;
    for (int i = 0; i < n; i++)
    {
        int L = range[i].ff, R = range[i].ss;
        if (l > R || r < L) continue;
        if (L >= l && R <= r)
                    mxx = max(mxx, mx[i]);
        else {
            for (int j = L; j <= R; j++) {
                if (j > r || j < l) continue;
                mxx = max(mxx, v[i][j - L]);
            }
        }
    }
    return mxx;
}
int main() {
    vector<int> v;
    for (int i = 0; i < n; i++) {
        cin >> x;
        v.pb(x);
    }
    init(v);
    while (1) {
        cin >> x;
        if (x == 0) break;
        if (x == 1){
            cin >> l >> r;
            cout << query(l, r) << "\n";
        }
        else {
            cin >> pos >> val;
            update(pos, val);
        }
    }
}
```

SPARSE TABLE:

```
// l -> r range er min value return kore
constant time a, preprocessing a N*logN
```

```
ll n, a[N], sp[N][20], ager_pow[N];
void init() {
    ll x = 1, p = 0;
    ager_pow[1] = 0;
    for (int i = 2; i < N; i++) {
        if (i > x * 2) {
            x *= 2;
            p++;
        }
        ager_pow[i] = p;
    }
    for (int i = 1; i <= n; i++)  sp[i][0] = a[i];
    p = 1;
    for (ll x = 2; x <= n;) {
        for (int i = 1; i + x - 1 <= n; i++) {
            sp[i][p] = min(sp[i][p - 1], sp[i +
                                x / 2][p - 1]);
        }
        x *= 2; p++;
    }
}
ll query(ll l, ll r) {
    ll rng = r - l + 1, p = ager_pow[rng];
    return min(sp[l][p], sp[r-(1<<p) + 1][p]);
}
int main()
{
    cin >> n;
    for (int i = 1; i <= n; i++)  cin >> a[i];
    init();
    cin >> q;
    while (q--) {
        cin >> l >> r;
        l++, r++;
        cout << query(l, r); nn;
    }
}
```

Ordered 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<int>,
rb_tree_tag,tree_order_statist
ics_node_update>
ordered_set st;
st[0] =*(st.find_by_order(0));
<x koyta= set.order_of_key(x);
```

RandomNumberGenerator:

```
mt19937_64
rng(chrono::steady_clock::now().time_sin
ce_epoch().count());
inline int gen_random(int l, int r) {
    return uniform_int_distribution<int>(l,
r) (rng);
}
```

Generate_Input:

```
//random_num_generator
main: freopen("in.txt", "w", stdout); // input
   srand(999889999);
```

Brute_sol:

```
   freopen("in.txt", "r", stdout);
   freopen("outb.txt", "w", stdout);
// outb = brute out, out = test out
```

e.cpp: out.txt

Comparator:

```
void compareFiles(FILE *fp1, FILE *fp2)
{
    char ch1 = getc(fp1);
    char ch2 = getc(fp2);
    int error = 0, pos = 0, line = 1;
    while (ch1 != EOF && ch2 != EOF) {
        pos++;
        if (ch1 == '\n' && ch2 == '\n') {
            line++;
            pos = 0;
        }
        if (ch1 != ch2) {
            error++;
            printf("Line Number : %d \tError"
                   " Position : %d \n", line, pos);
        }
        ch1 = getc(fp1);
        ch2 = getc(fp2);
    }
    printf("Total Errors : %d\t", error);
}
int main() {
    FILE *fp1 = fopen("out.txt", "r");
    FILE *fp2 = fopen("outb.txt", "r");
    if (fp1 == NULL || fp2 == NULL) {
        printf("Error : Files not open");
        exit(0);
    }
     compareFiles(fp1, fp2);
     fclose(fp1);
     fclose(fp2);
}
```

FORMULA:

num -> p1^e1 * p2^e2 * p3^e3…
nod = (e1 + 1) (e2+1)... (en+1)
sod = [(p1^(e1+1) - 1) / (p1-1)] * [(p2^(e2+1) - 1) / (p2-1)]..

1. Catalan triangle : Total number of permutation having n X and k Y so that Count(X)-Count(Y)>=0 in any prefix (Non-negative Partial Sum) :
ans = C(n+k,k) - C(n+k, k-1)
2. ncr = n-1cr + n-1cr-1
3. Arithmetic progression: n-th term = a + (n-1)d,
Sn = n/2[2a + (n-1)*d]

Geometric Series Formulas

cuemath THE MATH EXPERT

n^{th} term of a geometric sequence a , ar , ar^2,..... is,

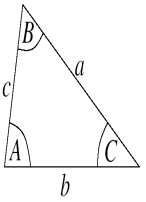
$$a_n = ar^{n-1}$$

Sum of n term of a finite geometric sequence is,

$$a + ar^2 + ar^3 + ... + ar^{n-1} = \frac{a(1 - r^n)}{1 - r} \text{ (OR) } \frac{a(r^n - 1)}{r - 1}, \text{ when } r \neq 1$$

Sum of infinite geometric sequence is,

$$a + ar^2 + ar^3 + ... = \frac{a}{1 - r}, \text{ when } r < 1$$



Sine Rule

$$\frac{a}{\sin(A)} = \frac{b}{\sin(B)} = \frac{c}{\sin(C)} \quad \text{or} \quad \frac{\sin(A)}{a} = \frac{\sin(B)}{b} = \frac{\sin(C)}{c}$$

(for finding sides) (for finding angles)

Cosine Rule

$$a^2 = b^2 + c^2 - 2bc\cos(A) \quad \text{or} \quad \cos(A) = \frac{b^2 + c^2 - a^2}{2bc}$$

(for finding sides) (for finding angles)

$$\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$$

$$\sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}$$

$$\sum_{k=1}^{n} k^3 = \frac{n^2(n+1)^2}{4}$$

$$\sum_{k=1}^{n} k(k+1) = \frac{n(n+1)(n+2)}{3}$$

$$\sum_{k=1}^{n} \frac{1}{k(k+1)} = \frac{n}{n+1}$$

$$\sum_{k=1}^{n} k(k+1)(k+2) = \frac{n(n+1)(n+2)(n+3)}{4}$$

$$\sum_{k=1}^{n} \frac{1}{k(k+1)(k+2)} = \frac{n(n+3)}{4(n+1)(n+2)}$$

$$\sum_{k=1}^{n} (2k-1) = n^2$$

Sum of the first n positive integers:	$\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$
Sum of the first n squares:	$\sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}$
Sum of the first n cubes:	$\sum_{k=1}^{n} k^3 = \frac{n^2(n+1)^2}{4}$
Sum of the first n terms of an arithmetic sequence:	$S_n = \frac{n}{2}[2a_1 + (n-1)d] = \frac{n}{2}(a_1 + a_n)$
Sum of the first n terms of a geometric sequence:	$S_n = \frac{g_1(1-r^n)}{1-r}$
Sum of all of the terms of a geometric sequence with $\lvert r \rvert < 1$:	$S_n \to \frac{g_1}{1-r}$

1. 2D shape

area $= \frac{1}{2}bh$

perimeter $= a+b+c$

Area $= \sqrt{s(s-a)(s-b)(s-c)}$

$s = \frac{a+b+c}{2} = \frac{P}{2}$

(2) Median

$m = \frac{1}{2}\sqrt{2a^2+2c^2-b^2}$

$q = \frac{2}{a+c}\sqrt{ac\,(s-b)}$

$\frac{CI}{CB} = \frac{AI}{AB}$

(3) Trapezoidal

$q = \frac{a+b}{2}$

$A = h \times \frac{(a+b)}{2} = hq$

(4) Right Triangle

$a^2+b^2=c^2$

$A = \frac{1}{2}ab = \frac{1}{2}ch$

$\frac{1}{h^2} = \frac{1}{a^2} + \frac{1}{b^2}$

$a^2 = BH \cdot c$

$b^2 = AH \cdot c$

$h^2 = AH \cdot BH$

(7) Parallelogram

$P = (a+b)\times 2$

$\alpha+\beta = 180°$

$h = a\sin\alpha = a\sin\beta$

$m^2+n^2 = 2(a^2+b^2)$

$A = bh = ab\sin\alpha$

(8) Lozenge (diamond)

$P = 4a$

area $= \frac{m\times n}{2} = a\times h$

$\alpha+\beta = 180°$

$m^2+n^2 = 4a^2$

$h = \frac{mn}{2a} = a\sin\alpha$

(10) convex quadrilateral

$\frac{1}{2}ab\sin\alpha$

(11) Circle

$P = 2\pi r = \pi d$

$A = \pi r^2 = \pi\frac{d^2}{4}$

(12) Segment of a circle

$A = \frac{r^2}{2}\left(\frac{\pi\alpha°}{180°} - \sin\alpha°\right)$

(13) Sector of a circle

$A = \pi r^2 \frac{\alpha}{360} = \frac{br}{2}$

$b = 2\pi r \frac{\alpha°}{360}$

(14) Regular polygon of N sides

$P = a\times N$

$A = \frac{a^2N}{4\tan\left(\frac{180}{N}\right)°} = \frac{a^2N}{4\tan\left(\frac{\pi}{N}\right)}$

$A = \frac{R^2N\sin\left(\frac{360}{N}\right)}{2} = \frac{R^2N\sin\left(\frac{2\pi}{N}\right)}{2}$

$r = \frac{a}{2}\cot\frac{180°}{N}$

$R = \frac{a}{2\sin\frac{180°}{N}}$

(15) Hexagon

$A = \frac{3\sqrt{3}}{2}a^2$

(16) 3D SPHERE

$V = \frac{4}{3}\pi r^3$

$A = 4\pi r^2$

(17) Spherical cap

$V = \pi h^2\left(R - \frac{h}{3}\right) = \frac{1}{6}\pi h\,(h^2+3R^2)$

$A = 2\pi Rh = \pi(r^2+h^2)$

(18) Spherical segment

$V = \frac{1}{6}\pi h^3 + \frac{1}{2}\pi(R_1^2+R_2^2)h$

$A = 2\pi Rh$

(19) spherical sector

$V = \frac{2}{3}\pi R^2 h$

$A = \pi R(r+2h)$

(21) cylinder

$V = \pi r^2 h$

$A = 2\pi rh$

(22) cone

$V = \frac{\pi r^2 h}{3}$

$A = \pi r s$