CSES Solutions TeamBook

Un económico pecho mixto para la mesa por favor.

Universidad Mayor de San Simón (UMSS)

Contents

1 Additional Problems II

1.0.1 GCD_Subsets

```cpp
// You are given an array of n integers. Your task is to calculate the
↪  number of non-empty subsets whose elements' greatest common divisor is
↪  equal to k for each k = 1,\dots, n.
#include <bits/stdc++.h>
using namespace std;

#define MOD 1000000007
#define ll long long

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

    int n;
    cin >> n;
    vector<int> a(n);
    for (int& x : a) cin >> x;

    // Paso 1: contar cuántos elementos son múltiplos de cada k
    vector<int> freq(n + 1, 0);
    for (int x : a) freq[x]++;

    // cnt[k] = cantidad de elementos que son múltiplos de k
    vector<int> cnt(n + 1, 0);
    for (int k = 1; k <= n; k++) {
        for (int mult = k; mult <= n; mult += k) {
            cnt[k] += freq[mult];
        }
    }

    // Precomputar potencias de 2: pow2[i] = 2^i % MOD
    vector<ll> pow2(n + 1, 1);
    for (int i = 1; i <= n; i++) {
        pow2[i] = (pow2[i - 1] * 2) % MOD;
    }

    // Paso 2 y 3: inclusion-exclusion
    vector<ll> ans(n + 1, 0);
    for (int k = n; k >= 1; k--) {
        ans[k] = (pow2[cnt[k]] - 1 + MOD) % MOD;
        // Restar contribuciones de múltiplos de k mayores
        for (int mult = 2 * k; mult <= n; mult += k) {
            ans[k] = (ans[k] - ans[mult] + MOD) % MOD;
        }
    }

    // Imprimir respuestas para k = 1 hasta n
    for (int k = 1; k <= n; k++) {
        cout << ans[k] << " ";
    }
    cout << "\n";

    return 0;
```

```
}
```

2 Bitwise Operations

2.0.1 All_Subarray_Xors

```cpp
#include <bits/stdc++.h>
using namespace std;
using ull = unsigned long long;
void fwht(vector<ull> &a, bool inv) {
    int n = a.size();
    for (int len = 1; len < n; len <<= 1) {
        for (int i = 0; i < n; i += len << 1) {
            for (int j = 0; j < len; ++j) {
                ull u = a[i + j];
                ull v = a[i + j + len];
                a[i + j] = u + v;
                a[i + j + len] = u - v;
            }
        }
    }
    if (inv) {
        for (ull &x : a) x /= n;
    }
}
signed main() {
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int n;
    cin >> n;
    vector<int> a(n);
    for (int &x : a) cin >> x;
    // prefijos del xor para los rangos
    vector<int> p(n + 1, 0);
    for (int i = 0; i < n; ++i)
        p[i + 1] = p[i] ^ a[i];
    // cosas de la fwht
    int maxVal = *max_element(p.begin(), p.end());
    int B = 0;
    while ((1 << B) <= maxVal) ++B;
    int M = 1 << B;
    // frecuencias
    vector<ull> A(M, 0);
    map<int, int> freq;
    bool hasZero = false;
    for (int x : p) {
        A[x]++;
        if (++freq[x] >= 2) hasZero = 1;
    }
    fwht(A, false);
    for (int i = 0; i < M; ++i)
        A[i] = A[i] * A[i];
    fwht(A, true);
    // los xor posibles
    vector<int> res;
    res.reserve(M);
    for (int i = 0; i < M; ++i) {
        if (A[i] > 0) {
            if (i == 0 && !hasZero) continue;
            //i ya es tu respuesta
            res.push_back(i);
        }
    }
    cout<<(int)res.size()<<'\n';
    for(int x:res)cout<<x<<' ';
}
```

2.0.2 Counting_Bits

```cpp
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define ll long long
const int tam = 1;

int cantbits(int n){
    if(n == 0)return 0;
    else{
        int k = log2(n);
        int ans = k*(1LL<<(k-1)) + (n - (1LL<<k) + 1);
        ans+=cantbits(n - (1LL<<k));
        return ans;
    }
}

void solve(){
    int n;cin>>n;
    cout<<cantbits(n)<<endl;
}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
   // cin>>t;
    while(t--){
        solve();
    }
}
```

2.0.3 K_Subset_Xors

```cpp
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for (int i = (a); i < (b); i++)
#define FOR(i, n) for (int i = 0; i < (n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
```

```cpp
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 31;
int basis[tam] = {0};
void insert(int x) {
    for (int i = tam - 1; i >= 0; i--) {
        if (x & (1 << i)) {
            if (!basis[i]) {
                basis[i] = x;
                return;
            }
            x ^= basis[i];
        }
    }
}

void solve() {
    //freopen("input.txt", "r", stdin);
    int n, k;
    cin >> n >> k;
    vector<int> arr(n);
    fore(i, 0, n) cin >> arr[i], insert(arr[i]);
    vector<int> base;
    fore(i, 0, tam)
        if (basis[i])
            base.push_back(basis[i]);
    int sz = sz(base);
    int mul = 1<<(min(n-sz, 30));
    int comb = 1<<sz;
    int unitam = min({k/mul + (k%mul>0), 200000, comb});
    vector<int> unicos;
    for(int mask = 0; mask<min(comb, 2*unitam); mask++){
        int actual = 0;
        for(int i = 0; i<sz; i++){
            if(mask&(1<<i)){
                actual^=base[i];
            }
        }
        unicos.push_back(actual);
    }
    sort(all(unicos));
    int impresos = 0;
    for(int i = 0; i<sz(unicos) and impresos < k; i++){
        for(int j = 0; j<mul and impresos < k; j++,
        ↪  impresos++)cout<<unicos[i]<<' ';
    }
    cout<<'\n';
}

signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    int t = 1;
    //cin >> t;
    while (t--) { solve(); }
}
```

2.0.4 Maximum_Xor_Subarray

```cpp
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;

struct nodo {
    nodo* hijos[2];
    nodo() {
        hijos[0] = nullptr;
        hijos[1] = nullptr;
    }
};

nodo* raiz;

void insertar(int x) {
    nodo* it = raiz;
    for (int j = 30; j >= 0; j--) {
        int bit = (x & (1 << j)) != 0;
        if (!it->hijos[bit]) { it->hijos[bit] = new nodo(); }
        it = it->hijos[bit];
    }
}

int buscar(int x) {
    nodo* it = raiz;
    int ans = 0;
    for (int j = 30; j >= 0; j--) {
        int bit = (x & (1 << j)) != 0;
        if (it->hijos[1 - bit]) {
            ans |= (1 << j);
            it = it->hijos[1 - bit];
        } else {
            it =
                it->hijos[bit];
        }
    }
    return ans;
}

void solve() {
    int n;
```

```
    cin >> n;
    raiz = new nodo();
    vector<int> prefix(n + 1, 0);
    vector<int> arr(n);
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
        prefix[i + 1] = prefix[i] ^ arr[i];
    }
    insertar(0);
    int ans = 0;
    for (int i = 1; i <= n; i++) {
        ans = max(ans, buscar(prefix[i]));
        insertar(prefix[i]);
    }
    cout << ans << endl;
}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
    //cin>>t;
    while(t--){
        solve();
    }
}
```

2.0.5 Maximum_Xor_Subset

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for (int i = (a); i < (b); i++)
#define FOR(i, n) for (int i = 0; i < (n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 32;
int basis[tam];
void insert(int x) {
    for (int i = tam - 1; i >= 0; i--) {
        if (x & (1 << i)) {
            if (!basis[i]) {
                basis[i] = x;
                return;
            }
            x ^= basis[i];
        }
    }
}
void solve() {
    int n;
    cin >> n;
    for (int i = 0; i < n; i++) {
        int x;
        cin >> x;
        insert(x);
    }
    int ans = 0;
    for(int i = tam-1; i>=0; i--){
        ans = max(ans, ans^basis[i]);
    }
    cout<<ans<<endl;
}

signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    int t = 1;
    //cin >> t;
    while (t--) { solve(); }
}
```

2.0.6 Number_of_Subset_Xors

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for (int i = (a); i < (b); i++)
#define FOR(i, n) for (int i = 0; i < (n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 32;
int basis[tam];
void insert(int x) {
    for (int i = tam - 1; i >= 0; i--) {
        if (x & (1 << i)) {
            if (!basis[i]) {
                basis[i] = x;
                return;
            }
            x ^= basis[i];
        }
    }
}
void solve() {
    int n;
    cin >> n;
    for (int i = 0; i < n; i++) {
        int x;
        cin >> x;
```

```
        insert(x);
    }
    int ans = 0;
    for(int i = tam-1; i>=0; i--){
        ans+=(basis[i] != 0);
    }
    cout<<(1<<ans)<<endl;
}

signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    int t = 1;
    //cin >> t;
    while (t--) { solve(); }
}
```

2.0.7 Xor_Pyramid_Peak

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

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

    int n;
    cin >> n;
    vector<ll> a(n);
    for(int i = 0; i < n; i++){
        cin >> a[i];
    }

    ll ans = 0;
    int N = n - 1;
    for(int i = 0; i < n; i++){
        // C(n-1, i) es impar ⟺ (i & (n-1)) == i
        if ((i & N) == i) {
            ans ^= a[i];
        }
    }

    cout << ans << "\n";
    return 0;
}
```

3 Dynamic Programming

3.0.1 Book_Shop

```
// You are in a book shop which sells n different books. You know the
↪  price and number of pages of each book.
// You have decided that the total price of your purchases will be at most
↪  x. What is the maximum number of pages you can buy? You can buy each
↪  book at most once.
#include <bits/stdc++.h>
using namespace std;
//#define int long long
#define f first
#define sst stringstream
#pragma GCC optimize("Ofast")
#define s second
#define pb push_back
#define sz(x) (int)(x).size()
#define all(a) (a).begin(), (a).end()
#define rall(a) (a).rbegin(), (a).rend()
#define fore(i, a, n) for (int i = (a); i < (n); i++)
#define forb(i, n) for (int i = (n) - 1; i >= 0; i--)
#define popcount(x) __builtin_popcountll(x);
typedef pair<int, int> pii;
typedef vector<int> vi;
const int MOD = 1000000007;
const double EPS = 1e-9;
const double PI = acos(-1);
const int INF = 1e18;
// PLUS ULTRA RECARGADO!!!
void solve() {
    int n, dinero;
    cin >> n >> dinero;
    int precio[n], pag[n];
    fore(i, 0, n) cin >> precio[i];
    fore(i, 0, n) cin >> pag[i];
    vector<vi> dp(n+1, vi(dinero+1, 0));
    for (int i = n - 1; i >= 0; i--) {
        for (int d = 0; d <= dinero; d++) {
            int notomar = dp[i+1][d];
            int tomar = 0;
            if(d - precio[i]>=0){
                tomar = dp[i+1][d - precio[i]] + pag[i];
            }
            dp[i][d] = max(tomar, notomar);
        }
    }
    cout<<dp[0][dinero]<<endl;
}
signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    //   int t;cin>>t;while(t--)solve();
    solve();
}
```

3.0.2 Coin_Combinations

```cpp
// Consider a money system consisting of n coins. Each coin has a positive integer value. Your task is to calculate the number of distinct ways you can produce a money sum x using the available coins.
// For example, if the coins are \{2,3,5\} and the desired sum is 9, there are 8
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define f first
#define sst stringstream
#define s second
#define pb push_back
#define sz(x) (int)(x).size()
#define all(a) (a).begin(), (a).end()
#define rall(a) (a).rbegin(), (a).rend()
#define fore(i, a, n) for(int i = (a); i < (n); i++)
#define forb(i, n) for (int i = (n) - 1; i >= 0; i--)
#define popcount(x) __builtin_popcountll(x);
typedef pair<int, int> pii;
typedef vector<int> vi;
const int MOD = 1000000007;
const double EPS = 1e-9;
const double PI = acos(-1);
const int INF = 1e18;
//PLUS ULTRA RECARGADO!!!
void solve() {
   int n, x;cin>>n>>x;
   vi monedas(n);
   fore(i,0,n)cin>>monedas[i];
   vi memo(x+1, 0);
   memo[0] = 1;
   fore(i, 1, x+1){
        for(int moneda:monedas){
            if(i - moneda>=0){
                memo[i]+=memo[i-moneda];
                memo[i]%=MOD;
            }
        }
   }
   cout<<memo[x]<<endl;
}
signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
 //   int t;cin>>t;while(t--)solve();
    solve();
}
```

3.0.3 Coin_Combinations_II

```cpp
// Consider a money system consisting of n coins. Each coin has a positive integer value. Your task is to calculate the number of distinct ordered ways you can produce a money sum x using the available coins.
// For example, if the coins are \{2,3,5\} and the desired sum is 9, there are 3
#include <bits/stdc++.h>
using namespace std;
//#pragma optimize("Ofast")
//#define int long long
#define f first
#define sst stringstream
#define s second
#define pb push_back
#define sz(x) (int)(x).size()
#define all(a) (a).begin(), (a).end()
#define rall(a) (a).rbegin(), (a).rend()
#define fore(i, a, n) for(int i = (a); i < (n); i++)
#define forb(i, n) for (int i = (n) - 1; i >= 0; i--)
#define popcount(x) __builtin_popcountll(x);
typedef pair<int, int> pii;
typedef vector<int> vi;
const int MOD = 1000000007;
const double EPS = 1e-9;
const double PI = acos(-1);
//PLUS ULTRA RECARGADO!!!
void solve() {
   int n, x;cin>>n>>x;
   vi monedas(n);
   fore(i, 0, n)cin>>monedas[i];
   vector<vi> dp(n+1, vi(x+1));
   for(int i = 0; i<=n; i++){
    dp[i][0] = 1;
   }
   for(int i = n-1; i>=0; i--){
    for(int suma = 1; suma<=x; suma++){
        int notomar = dp[i+1][suma];
        int tomar = 0;
        if(suma - monedas[i]>=0){
            tomar = dp[i][suma - monedas[i]];
        }
        dp[i][suma] = tomar + notomar;
        dp[i][suma]%=MOD;
    }
   }
   cout<<dp[0][x]<<endl;
}
signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    solve();
}
```

3.0.4 Dice_Combinations

```cpp
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1000000+5;

void solve(){
    int x;cin>>x;
    int memo[tam];
    memset(memo, 0, sizeof memo);
    memo[0] = 1;
    for(int i = 1; i<=x; i++){
        ll ans = 0;
        for(int m = 1; m<=6;m++){
            if(i-m>=0){
                ans+=memo[i-m];
                ans%=MOD;
            }
        }
        memo[i]  = ans;
    }
    cout<<memo[x]<<endl;
}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
    //cin>>t;
    while(t--){
        solve();
    }
}
```

3.0.5 Edit_Distance

```cpp
// The edit distance between two strings is the minimum number of operations required to transform one string into the other.
// The allowed operations are:
// Add one character to the string.
// Remove one character from the string.
// Replace one character in the string.
// For example, the edit distance between LOVE and MOVIE is 2, because you can first replace L with M, and then add I.
// Your task is to calculate the edit distance between two strings.
#include <bits/stdc++.h>
using namespace std;
//#define int long long
#define f first
#define sst stringstream
#define s second
#define pb push_back
#define sz(x) (int)(x).size()
#define all(a) (a).begin(), (a).end()
#define rall(a) (a).rbegin(), (a).rend()
#define fore(i, a, n) for(int i = (a); i < (n); i++)
#define forb(i, n) for (int i = (n) - 1; i >= 0; i--)
#define popcount(x) __builtin_popcountll(x);
typedef pair<int, int> pii;
typedef vector<int> vi;
const int MOD = 1000000007;
const double EPS = 1e-9;
const double PI = acos(-1);
const int INF = 1e18;
//PLUS ULTRA RECARGADO!!!
string s, t;
int memo[5011][5011];

int dp(int i, int j) {
    if (i < 0) return j + 1;
    if (j < 0) return i + 1;
    if (memo[i][j] != -1) return memo[i][j];

    if (s[i] == t[j]) {
        memo[i][j] = dp(i - 1, j - 1);
    } else {
        memo[i][j] = min({
            dp(i - 1, j) + 1,
            dp(i, j - 1) + 1,
            dp(i - 1, j - 1) + 1
        });
    }
    return memo[i][j];
}

void solve() {
    cin >> s >> t;
    memset(memo, -1, sizeof(memo));
    int ns = sz(s), nt = sz(t);
    cout << dp(ns - 1, nt - 1) << endl;
}


signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    //int t;cin>>t;while(t--)solve();
    solve();
}
```

3.0.6 Grid_Paths_I

```cpp
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define f first
#define sst stringstream
#define s second
#define pb push_back
#define sz(x) (int)(x).size()
#define all(a) (a).begin(), (a).end()
#define rall(a) (a).rbegin(), (a).rend()
#define fore(i, a, n) for (int i = (a); i < (n); i++)
#define forb(i, n) for (int i = (n) - 1; i >= 0; i--)
#define popcount(x) __builtin_popcountll(x);
typedef pair<int, int> pii;
typedef vector<int> vi;
const int MOD = 1000000007;
const double EPS = 1e-9;
const double PI = acos(-1);
const int INF = 1e18;
// PLUS ULTRA RECARGADO!!!
void solve() {
    int n;
    cin >> n;
    char mat[n][n];
    fore(i, 0, n) {
        fore(j, 0, n) {
            cin >> mat[i][j];
        }
    }
    int dp[n][n];

    memset(dp, 0, sizeof dp);
    if (mat[n - 1][n - 1] != '*') dp[n - 1][n - 1] = 1;
    for (int i = n - 1; i >= 0; i--) {
        for (int j = n - 1; j >= 0; j--) {
            if (mat[i][j] == '*') {
                dp[i][j] = 0;
                continue;
            }
            if (j + 1 < n) dp[i][j] += dp[i][j + 1];
            if (i + 1 < n) dp[i][j] += dp[i + 1][j];
            dp[i][j] %= MOD;
        }
    }
    cout<<dp[0][0]<<endl;
}
signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    int t;
    // cin >> t;
    // while (t--) solve();
    solve();
}
```

3.0.7 Longest_Common_Subsequence

```cpp
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1005;
// const int n  =1 , m = 1;
// int dp(int i, int j){
//     if(i>=n){
//         return 0;
//     }
//     if(j>=m){
//         return 0;
//     }
//     if(memo[i][j] == -1){
//         int a = 0, b, c;
//         if(s[i] == t[j]){
//             a = dp(i+1, j+1) + 1;
//         }
//         b = dp(i+1, j);
//         c = dp(i, j+1);
//         memo[i][j] = max({a, b, c});
//     }
//     return memo[i][j];

// }

void solve(){
    int n, m;cin>>n>>m;
    int dp[tam][tam];
    memset(dp, 0, sizeof dp);
    int s[n], t[m];
    fore(i, 0, n)cin>>s[i];
    fore(i, 0, m)cin>>t[i];
    for(int i = n-1; i>=0; i--){
        for(int j = m-1; j>=0; j--){
            int a = 0, b, c;
            if(s[i] == t[j]){
                a = dp[i+1][j+1] + 1;
            }
            b = dp[i+1][j];
            c = dp[i][j+1];
            dp[i][j] = max({a, b, c});
        }
```

```
    }
    cout<<dp[0][0]<<endl;
    int i = 0, j = 0;
    vector<int> ans;
    while(i<n and  j<m){
        if(s[i] == t[j]){
            ans.push_back(s[i]);
            ++i, ++j;
        }else if(dp[i+1][j] >= dp[i][j+1]){
            ++i;
        }else{
            ++j;
        }
    }
    for(int x:ans)cout<<x<<' ';
    cout<<'\n';

}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
 //   cin>>t;
    while(t--){
        solve();
    }
}
```

3.0.8 Longest_Common_Subsequences

```
// Given two arrays of integers, find their longest common subsequence.
// A subsequence is a sequence of array elements from left to right that can contain gaps. A common subsequence is a subsequence that appears in both arrays.
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1005;
void solve(){
    int n, m;cin>>n>>m;
    int dp[tam][tam];
    memset(dp, 0, sizeof dp);
    int s[n], t[m];
    fore(i, 0, n)cin>>s[i];
    fore(i, 0, m)cin>>t[i];
    for(int i = n-1; i>=0; i--){
        for(int j = m-1; j>=0; j--){
            int a = 0, b, c;
            if(s[i] == t[j]){
                a = dp[i+1][j+1] + 1;
            }
            b = dp[i+1][j];
            c = dp[i][j+1];
            dp[i][j] = max({a, b, c});
        }
    }
    cout<<dp[0][0]<<endl;
    int i = 0, j = 0;
    vector<int> ans;
    while(i<n and  j<m){
        if(s[i] == t[j]){
            ans.push_back(s[i]);
            ++i, ++j;
        }else if(dp[i+1][j] >= dp[i][j+1]){
            ++i;
        }else{
            ++j;
        }
    }
    for(int x:ans)cout<<x<<' ';
    cout<<'\n';

}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
 //   cin>>t;
    while(t--){
        solve();
    }
}
```

3.0.9 Minimizing_Coins

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define f first
#define sst stringstream
#define s second
#define pb push_back
#define sz(x) (int)(x).size()
#define all(a) (a).begin(), (a).end()
#define rall(a) (a).rbegin(), (a).rend()
#define fore(i, a, n) for(int i = (a); i < (n); i++)
#define forb(i, n) for (int i = (n) - 1; i >= 0; i--)
#define popcount(x) __builtin_popcountll(x);
typedef pair<int, int> pii;
typedef vector<int> vi;
const int MOD = 1000000007;
```

```
const double EPS = 1e-9;
const double PI = acos(-1);
const int INF = 1e18;
// Consider a money system consisting of n coins. Each coin has a positive integer value. Your task is to produce a sum of money x using the available coins in such a way that the number of coins is minimal.
// For example, if the coins are \{1,5,7\} and the desired sum is 11, an optimal solution is 5+5+1 which requires 3 coins.

void solve() {
    int n, x;cin>>n>>x;
    vi monedas(n);
    fore(i, 0, n)cin>>monedas[i];
    vi dp(x+1, INF);
    dp[0] = 0;
    int ans = 0;
    fore(i, 1, x+1){
        for(int moneda:monedas){
            if(i-moneda>=0){
                dp[i] = min(dp[i], dp[i-moneda]+1);
            }
        }
    }
   cout<<(dp[x] == INF?-1:dp[x])<<endl;
}
signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
 //   int t;cin>>t;while(t--)solve();
    solve();
}
```

3.0.10 Money Sums

```
#include <cstring>
#include <iostream>
#include <vector>
using namespace std;
// You have n coins with certain values. Your task is to find all money sums you can create using these coins.
void solve() {
    int n;
    cin >> n;
    vector<int> monedas(n);
    int maxSuma = 0;
    for (int i = 0; i < n; i++) {
        cin >> monedas[i];
        maxSuma += monedas[i];
    }

    vector<bool> dp(maxSuma + 1, false);
    dp[0] = true;

    for (int coin : monedas) {
        for (int s = maxSuma; s >= coin; s--) {
            if (dp[s - coin]) dp[s] = true;
        }
    }

    int count = 0;
    for (int s = 1; s <= maxSuma; s++) {
        if (dp[s]) count++;
    }
    cout << count << "\n";
    for (int s = 1; s <= maxSuma; s++) {
        if (dp[s]) cout << s << " ";
    }
    cout << "\n";
}

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

    int t = 1;
    //cin >> t;
    while (t--) { solve(); }
    return 0;
}
```

3.0.11 Rectangle_Cutting

```
// Given an a \times b rectangle, your task is to cut it into squares. On each move you can select a rectangle and cut it into two rectangles in such a way that all side lengths remain integers. What is the minimum possible number of moves?
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
int memo[501][501];
ll dp(int a, int b) {
    if (a == b) return 0;
    if (memo[a][b] != -1) return memo[a][b];
    ll res = INF;
    for (int i = 1; i < a; ++i) {
        res = min(res, dp(i, b) + dp(a - i, b) + 1);
    }
    for (int i = 1; i < b; ++i) {
        res = min(res, dp(a, i) + dp(a, b - i) + 1);
    }
    memo[a][b] = res;
```

```
    return res;
}
void solve(){
   int a, b;cin>>a>>b;
    memset(memo, -1, sizeof memo);
   cout<<dp(a, b)<<endl;
}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    solve();
}
```

3.0.12 Removing_Digits

```
// You are given an integer n. On each step, you may subtract one of the digits from the number.
// How many steps are required to make the number equal to 0?
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define f first
#define sst stringstream
#define s second
#define pb push_back
#define sz(x) (int)(x).size()
#define all(a) (a).begin(), (a).end()
#define rall(a) (a).rbegin(), (a).rend()
#define fore(i, a, n) for(int i = (a); i < (n); i++)
#define forb(i, n) for (int i = (n) - 1; i >= 0; i--)
#define popcount(x) __builtin_popcountll(x);
typedef pair<int, int> pii;
typedef vector<int> vi;
const int MOD = 1000000007;
const double EPS = 1e-9;
const double PI = acos(-1);
const int INF = 1e18;
//PLUS ULTRA RECARGADO!!!
void solve() {
   int n;cin>>n;
   vi memo(n+1, INF);
   memo[0] = 1;
   for(int i = 1; i<=n; i++){
    int num = i;
    while(num){
        if((i - num%10)>=0){
            memo[i] = min(memo[i], memo[i - num%10]+1);
        }
        num/=10;
    }
   }
   cout<<memo[n]-1<<endl;
}
signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
 //   int t;cin>>t;while(t--)solve();
    solve();
}
```

4 Geometry

4.0.1 All_Manhattan_DIstances

```
// Given a set of points, calculate the sum of all Manhattan distances between two point pairs.
import sys
d=sys.stdin.readlines()
n=int(d[0])
x=[0]*n
y=[0]*n
def f(a):
    a.sort()
    s=0
    for i in range(n):
        s+=a[i]*(2*i-n+1)
    return s
for i in range(1, n+1):
    x[i-1],y[i-1]=map(int,d[i].split())
print(f(x)+f(y))
```

4.0.2 Area_of_Rectangles

```
#include <bits/stdc++.h>
using namespace std;
#define int ll
#define ll long long
#define ull unsigned ll
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;

struct Node {
    unique_ptr<Node> left, right;
    int count = 0;
    ll length = 0;
};

void recalc(Node* node, ll l, ll r) {
    if (node->count > 0) {
        node->length = r - l; // todo cubierto
```

```
    } else {
        ll left_len = node->left ? node->left->length : 0;
        ll right_len = node->right ? node->right->length : 0;
        node->length = left_len + right_len;
    }
}

void update(Node* node, ll l, ll r, ll y_l, ll y_r, int val) {
    if (y_r <= l or r <= y_l) return;
    if (y_l <= l and r <= y_r) {
        node->count += val;
    } else {
        ll m = l + (r - l)/2;
        if (!node->left) node->left = make_unique<Node>();
        if (!node->right) node->right = make_unique<Node>();
        update(node->left.get(), l, m, y_l, y_r, val);
        update(node->right.get(), m, r, y_l, y_r, val);
    }
    recalc(node, l, r);
}

struct event{
    int x, y1, y2, tipo;//apertura es +1, cierre es -1
    event(int a = 0, int b = 0, int c = 0, int d = 1){
        x = a, y1 = b, y2 = c, tipo = d;
    }
    bool operator<(const event& otro) const {
        return x < otro.x or (x == otro.x and tipo > otro.tipo);
    }
};
const int ymin = -1e6-10;
const int ymax = 1e6 + 10;
void solve(){
    auto root = make_unique<Node>();
    int n;cin>>n;
    vector<event> eventos;
    for(int i = 0; i<n; i++){
        int a, b, c, d;cin>>a>>b>>c>>d;
        eventos.push_back(event(a, b, d, 1));
        eventos.push_back(event(c, b, d, -1));
    }
    sort(all(eventos));

    int ans = 0;
    int ulti = eventos[0].x;
    for(auto e:eventos){
        int dx = e.x - ulti;
        ulti = e.x;
        ans+=dx*root->length;
        update(root.get(), ymin, ymax, e.y1, e.y2, e.tipo);
    }
    cout<<ans<<'\n';
}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
   // cin>>t;
    while(t--){
        solve();
    }
}
```

4.0.3 Convex_Hull

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

struct Punto {
    long long x, y;
    bool operator<(const Punto& o) const {
        return x < o.x || (x == o.x && y < o.y);
    }
};

// Producto cruzado (para ver si el giro es izquierda/derecha)
long long cruz(const Punto& O, const Punto& A, const Punto& B) {
    return (A.x - O.x) * (B.y - O.y) -
           (A.y - O.y) * (B.x - O.x);
}

vector<Punto> convexHull(vector<Punto>& pts) {
    int n = pts.size(), k = 0;
    if (n <= 1) return pts;
    sort(pts.begin(), pts.end());

    vector<Punto> hull(2*n);

    // Construir lower hull
    for (int i = 0; i < n; i++) {
        while (k >= 2 && cruz(hull[k-2], hull[k-1], pts[i]) <= 0) k--;
        hull[k++] = pts[i];
    }

    // Construir upper hull
    for (int i = n-2, t = k+1; i >= 0; i--) {
        while (k >= t && cruz(hull[k-2], hull[k-1], pts[i]) <= 0) k--;
        hull[k++] = pts[i];
    }

    hull.resize(k-1); // último punto es el mismo que el primero
    return hull;
}

int main() {
    int n;
    cin >> n;
    vector<Punto> pts(n);
    for (int i = 0; i < n; i++) cin >> pts[i].x >> pts[i].y;

    vector<Punto> hull = convexHull(pts);

    cout << hull.size() << "\n";
```

```
    for (auto& p : hull)
        cout << p.x << " " << p.y << "\n";
}
```

4.0.4 Intersection_Points

```
// Given n horizontal and vertical line segments, your task is to calculate the number of their intersection points.
// You can assume that no parallel line segments intersect, and no endpoint of a line segment is an intersection point.
#include <bits/stdc++.h>
#define int long long
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;
struct segment{
    int inix, iniy, finx, finy;
    segment(){}
    segment(int a, int b, int c, int d){
        pair<int, int> aa = {a, b};
        pair<int, int> bb = {c, d};
        if(aa > bb){
            swap(aa, bb);
        }
        inix = aa.first;
        iniy = aa.second;
        finx = bb.first;
        finy = bb.second;
    }
    // bool operator < (const segment& otro) const{
    //     return make_pair(make_pair(inix, iniy), make_pair(finx, finy)) < make_pair(make_pair(otro.inix, otro.iniy), make_pair(otro.finx, otro.finy));
    // }
};
#define abrir 10
#define consulta 20
#define cerrar 30

struct event{
    int x, y1, y2, tipo;
    event(int a = 0, int b = 0, int c = 0, int d = 0){
        x = a, y1 = b, y2 = c, tipo = d;
    }
    bool operator < (const event& otro)const{
        return x < otro.x or (x == otro.x and tipo < otro.tipo);
    }
};

#define lsb(x) ((x) & (-x))

struct BIT {
    // indexado a 1
    vector<int> bit;
    BIT(int N) : bit(N + 1) {}
    void add(int i, int x) {
        while (i < sz(bit)) {
            bit[i] += x;
            i += lsb(i);
        }
    }
    int sum(int i) {
        int ans = 0;
        while (i > 0) {
            ans += bit[i];
            i -= lsb(i);
        }
        return ans;
    }
    int sum(int l, int r) {
        if (l > r) return 0;
        return sum(r) - sum(l - 1);
    }
};

void solve(){
   int n;cin>>n;
    vector<segment> arr(n);
    vector<int> ys;
    ys.reserve(4*n);
    vector<tuple<int, int, int , int>> we;
    for(int i = 0; i<n ;i++){
        int a, b, c, d;cin>>a>>b>>c>>d;
        we.emplace_back(a, b, c, d);
        ys.push_back(b);
        ys.push_back(b+1);
        ys.push_back(d);
        ys.push_back(d+1);
    }
    sort(all(ys));
    ys.erase(unique(all(ys)), ys.end());
    unordered_map<int, int> f;
    f.reserve(sz(ys)+2);
    for(int i = 1; i<=sz(ys); i++){
        f[ys[i-1]] = i;
    }
    int idx = 0;
    for(auto [a, b,c , d]:we){
        arr[idx++] = segment(a, f[b], c, f[d]);
    }
    vector<event> eventos;
    eventos.reserve(sz(arr));
    idx = 0;
    for(auto x:arr){
```

```
        if(x.inix == x.finx){
            //es consulta
            eventos.push_back(event(x.inix, x.iniy, x.finy, consulta));
        }else{
            eventos.push_back(event(x.inix, x.iniy, x.iniy, abrir));
            eventos.push_back(event(x.finx, x.finy, x.finy, cerrar));
        }
    }
    sort(all(eventos));
    BIT bit(sz(ys) + 10);
    int ans = 0;
    for(auto ev:eventos){
        if(ev.tipo == abrir){
            bit.add(ev.y1, 1);
        }else if(ev.tipo == cerrar){
            bit.add(ev.y1, -1);
        }else{
            ans+=bit.sum(ev.y1, ev.y2);
        }
    }
    cout<<ans<<'\n';
}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
 //   cin>>t;
    while(t--){
        solve();
    }
}
```

4.0.5 Lines_And_Queries_I

```
// Your task is to efficiently process the following types of queries:
// Add a line ax+b
// Find the maximum point in any line at position x
#include <bits/stdc++.h>
#define int long long
using namespace std;
#define ll long long
#define ull unsigned ll
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 4e18;
const int tam = 1;

struct Line {
    ll a, b;
    Line(ll _a=0, ll _b=-INF) : a(_a), b(_b) {}
    ll eval(ll x) const { return a*x + b; }
};

struct LiChao {
    ll l, r;
    Line line;
    unique_ptr<LiChao> left, right;

    LiChao(ll _l, ll _r) : l(_l), r(_r), line(Line()), left(nullptr),
    ↪  right(nullptr) {}

    // insertar la línea "nueva" en el intervalo [L_, R_] (ambos
    ↪  inclusivos)
    void insert(Line nueva, ll L_, ll R_) {
        // si no hay intersección entre [l,r] (nodo) y [L_,R_]
        ↪  (inserción), cortar
        if (R_ < l || r < L_) return;

        // Si el nodo está completamente cubierto por la inserción,
        ↪  hacemos el procedimiento Li Chao
        if (L_ <= l && r <= R_) {
            ll m = (l + r) >> 1;
            bool leftBetter = nueva.eval(l) > line.eval(l); // > porque
            ↪  buscas máximo en tu código original
            bool midBetter  = nueva.eval(m) > line.eval(m);

            if (midBetter) swap(line, nueva);

            if (l == r) return;

            if (leftBetter != midBetter) {
                // la intersección está en el lado izquierdo [l,m]
                if (!left) left = make_unique<LiChao>(l, m);
                left->insert(nueva, L_, R_);
            } else {
                // la intersección está en el lado derecho [m+1,r]
                if (!right) right = make_unique<LiChao>(m+1, r);
                right->insert(nueva, L_, R_);
            }
            return;
        }

        // Caso parcial: solo recursar a los hijos cuyos intervalos
        ↪  intersectan [L_,R_]
        ll m = (l + r) >> 1;
        // Si la inserción toca la mitad izquierda
        if (L_ <= m) {
            if (!left) left = make_unique<LiChao>(l, m);
            left->insert(nueva, L_, R_);
        }
        // Si la inserción toca la mitad derecha
        if (R_ > m) {
            if (!right) right = make_unique<LiChao>(m+1, r);
            right->insert(nueva, L_, R_);
        }
    }

```

```
    // consulta punto x (máximo)
    ll query(ll x) const {
        ll res = line.eval(x);
        if (l == r) return res;
        ll m = (l + r) >> 1;
        if (x <= m) {
            if (left) res = max(res, left->query(x));
        } else {
            if (right) res = max(res, right->query(x));
        }
        return res;
    }
};

void solve(){
    int q;cin>>q;
    int maxn = 1e5 + 100;
    LiChao tree(-maxn, maxn);
    while(q--){
        int tipo;cin>>tipo;
        if(tipo == 1){
            int a, b;cin>>a>>b;
            int l = -maxn, r = maxn;
            tree.insert(Line(a, b), l, r);
        }else{
            int x;cin>>x;
            auto ans = tree.query(x);
            cout<<ans<<'\n';
            // cout<<(ans == -INF or ans ==
            ↪  INF?"NO":to_string(ans))<<'\n';
        }
    }
}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
    // cin>>t;
    while(t--){
        solve();
    }
}
```

4.0.6 Lines_And_Queries_II

```
// Your task is to efficiently process the following types of queries:
// Add a line ax+b that is active in range [l,r]
// Find the maximum point in any active line at position x
#include <bits/stdc++.h>
#define int long long
using namespace std;
#define ll long long
#define ull unsigned ll
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 4e18;
const int tam = 1;

struct Line {
    ll a, b;
    Line(ll _a=0, ll _b=-INF) : a(_a), b(_b) {}
    ll eval(ll x) const { return a*x + b; }
};

struct LiChao {
    ll l, r;
    Line line;
    unique_ptr<LiChao> left, right;

    LiChao(ll _l, ll _r) : l(_l), r(_r), line(Line()), left(nullptr),
    ↪  right(nullptr) {}

    // insertar la línea "nueva" en el intervalo [L_, R_] (ambos
    ↪  inclusivos)
    void insert(Line nueva, ll L_, ll R_) {
        // si no hay intersección entre [l,r] (nodo) y [L_,R_]
        ↪  (inserción), cortar
        if (R_ < l || r < L_) return;

        // Si el nodo está completamente cubierto por la inserción,
        ↪  hacemos el procedimiento Li Chao
        if (L_ <= l && r <= R_) {
            ll m = (l + r) >> 1;
            bool leftBetter = nueva.eval(l) > line.eval(l); // > porque
            ↪  buscas máximo en tu código original
            bool midBetter  = nueva.eval(m) > line.eval(m);

            if (midBetter) swap(line, nueva);

            if (l == r) return;

            if (leftBetter != midBetter) {
                // la intersección está en el lado izquierdo [l,m]
                if (!left) left = make_unique<LiChao>(l, m);
                left->insert(nueva, L_, R_);
            } else {
                // la intersección está en el lado derecho [m+1,r]
                if (!right) right = make_unique<LiChao>(m+1, r);
                right->insert(nueva, L_, R_);
            }
            return;
        }

        // Caso parcial: solo recursar a los hijos cuyos intervalos
        ↪  intersectan [L_,R_]
        ll m = (l + r) >> 1;
        // Si la inserción toca la mitad izquierda
```

```
        if (L_ <= m) {
            if (!left) left = make_unique<LiChao>(l, m);
            left->insert(nueva, L_, R_);
        }
        // Si la inserción toca la mitad derecha
        if (R_ > m) {
            if (!right) right = make_unique<LiChao>(m+1, r);
            right->insert(nueva, L_, R_);
        }
    }

    // consulta punto x (máximo)
    ll query(ll x) const {
        ll res = line.eval(x);
        if (l == r) return res;
        ll m = (l + r) >> 1;
        if (x <= m) {
            if (left) res = max(res, left->query(x));
        } else {
            if (right) res = max(res, right->query(x));
        }
        return res;
    }
};

void solve(){
    int q;cin>>q;
    int maxn = 1e5 + 100;
    LiChao tree(-maxn, maxn);
    while(q--){
        int tipo;cin>>tipo;
        if(tipo == 1){
            int a, b;cin>>a>>b;
            int l, r;cin>>l>>r;
            tree.insert(Line(a, b), l, r);
        }else{
            int x;cin>>x;
            auto ans = tree.query(x);
            cout<<(ans == -INF or ans == INF?"NO":to_string(ans))<<'\n';
        }
    }
}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
    // cin>>t;
    while(t--){
        solve();
    }
}
```

4.0.7 Line_Segment_Intersection

```
// There are two line segments: the first goes through the points (x_1,y_1) and (x_2,y_2), and the second goes through the points (x_3,y_3) and (x_4,y_4).
// Your task is to determine if the line segments intersect, i.e., they have at least one common point.
#include <bits/stdc++.h>

using namespace std;
typedef long double ld;
const ld EPS = 1e-9;
#define int long long

bool ge(int x, int y) {
    return x + EPS > y;
}
bool le(int x, int y) {
    return x - EPS < y;
}
bool eq(int x, int y) {
    return ge(x, y) and le(x, y);
}
int sign(int x) {
    return ge(x, 0) - le(x, 0);
}

struct point_i {
    // Construccion
    int x, y;
    point_i() : x(0), y(0) {}
    point_i(int x, int y) : x(x), y(y) {}
    // Operadores
    point_i operator-() { return point_i(-x, -y); }
    point_i operator+(point_i p) { return point_i(x + p.x, y + p.y); }
    point_i operator-(point_i p) { return point_i(x - p.x, y - p.y); }
    point_i operator*(int k) { return point_i(k * x, k * y); }
    // producto punto
    int operator*(point_i p) { return x * p.x + y * p.y; }
    // producto cruz
    int operator%(point_i p) { return x * p.y - y * p.x; }
    // Comparadores
    bool operator==(const point_i &p) const { return x == p.x and y == p.y; }
    bool operator!=(const point_i &p) const { return x != p.x or y != p.y; }
    bool operator<(const point_i &p) const {
        return (x < p.x) or (x == p.x and y < p.y);
    }

    // 0 ⇒ same direction
    // 1 ⇒ p is on the left
    //-1 ⇒ p is on the right
    int dir(point_i o, point_i p) {
        int x = (*this - o) % (p - o);
        return (ge(x, 0)) - le(x, 0);
    }
```

```
    bool on_seg(point_i p, point_i q) {
        if (this->dir(p, q)) return 0;
        return ge(x, min(p.x, q.x)) and le(x, max(p.x, q.x)) and
               ge(y, min(p.y, q.y)) and le(y, max(p.y, q.y));
    }

    // Longitudes y distancias
    ld abs() { return sqrtl(x * x + y * y); }
    int abs2() { return x * x + y * y; }
    ld dist(point_i q) { return (*this - q).abs(); }
    int dist2(point_i q) { return (*this - q).abs2(); }
    // Angulo respecto al eje x (1,1) -> 45 (pero en radianes)
    ld arg() { return atan2l(y, x); }
    // DOUBLE
    point_i project(point_i y) { return y * ((*this * y) / (y * y)); }
    // Proyecta el punto actual sobre la línea definida por los puntos '0' y
    // 'y'

    point_i project(point_i x, point_i y) {
        return x + (*this - x).project(y - x);
    }
    // Calcula la distancia del punto actual a la línea infinita que pasa por x
    // y
    ld dist_line(point_i x, point_i y) { return dist(project(x, y)); }
    // Calcula la distancia del punto actual al segmento [x, y]
    ld dist_seg(point_i x, point_i y) {
        return project(x, y).on_seg(x, y) ? dist_line(x, y)
                                          : min(dist(x), dist(y));
    }
    // rotaciones
    point_i rotate(ld sin, ld cos) {
        return point_i(cos * x - sin * y, sin * x + cos * y);
    }
    point_i rotate(ld a) { return rotate(sinl(a), cosl(a)); }

    // rotar respecto a un punto
    // USAR DOUBLES POR LA PRECICION AQUI
    point_i rotate(point_i p) { return rotate(p.y / p.abs(), p.x / p.abs()); }
};
// Duda
int direction(point_i o, point_i p, point_i q) {
    return p.dir(o, q);
}
// Rotacion horaria y antihoraria
point_i rotate_ccw90(point_i p) {
    return point_i(-p.y, p.x);
}
point_i rotate_cw90(point_i p) {
    return point_i(p.y, -p.x);
}

// for reading purposes avoid using * and % operators, use the functions below:
int dot(point_i p, point_i q) {
    return p.x * q.x + p.y * q.y;
}
int cross(point_i p, point_i q) {
    return p.x * q.y - p.y * q.x;
}

// Duda
int area_2(point_i a, point_i b, point_i c) {
    return cross(a, b) + cross(b, c) + cross(c, a);
}
// Duda
int angle_less(const point_i &a1, const point_i &b1, const point_i &a2,
               const point_i &b2) {
    // angle between (a1 and b1) vs angle between (a2 and b2)
    // 1  : bigger
    //-1 : smaller
    // 0  : equal
    point_i p1(dot(a1, b1), abs(cross(a1, b1)));
    point_i p2(dot(a2, b2), abs(cross(a2, b2)));
    if (cross(p1, p2) < 0) return 1;
    if (cross(p1, p2) > 0) return -1;
    return 0;
}

ostream &operator<<(ostream &os, const point_i &p) {
    os << "(" << p.x << "," << p.y << ")";
    return os;
}
istream &operator>>(istream &in, point_i &p) {
    in >> p.x >> p.y;
    return in;
}

bool dentro(point_i &p, point_i &a, point_i &b) {
    return min(a.x, b.x) <= p.x and p.x <= max(a.x, b.x) and
           min(a.y, b.y) <= p.y and p.y <= max(a.y, b.y);
}

void solve() {
    point_i a, b, c, d;
    cin >> a >> b >> c >> d;
    point_i ab = b - a, ac = c - a, ad = d - a;
    point_i cd = d - c, ca = a - c, cb = b - c;
    if (((ab%ac > 0 and ab%ad < 0) or (ab%ac < 0 and ab%ad > 0)) and
        ((cd%ca > 0 and cd%cb < 0) or (cd%ca < 0 and cd%cb > 0))) {
        cout << "YES\n";
        return;
    }
    bool ok = (ab%ac == 0 and dentro(c, a, b)) or
              (ab%ad == 0 and dentro(d, a, b)) or
              (cd%ca == 0 and dentro(a, c, d)) or
              (cd%cb == 0 and dentro(b, c, d));
    if (ok) {
        cout << "YES\n";
    } else {
        cout << "NO\n";
    }
}
signed main(){
```

```
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    int t = 1;
    cin>>t;
    while(t--){solve();}
}
```

4.0.8 Line__Segments__Trace II

```
// There are n line segments whose endpoints have integer coordinates. Each x-coordinate is between 0 and m. The slope of each segment is an integer.
// For each x-coordinate 0,1,\dots,m, find the maximum point in any line segment. If there is no segment at some point, the maximum is -1.
#include <bits/stdc++.h>
#define int long long
using namespace std;
#define ll long long
#define ull unsigned ll
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 4e18;
const int tam = 1;

struct Line {
    ll a, b;
    Line(ll _a=0, ll _b=-INF) : a(_a), b(_b) {}
    ll eval(ll x) { return a*x + b; }
};

struct LiChao {
    int l, r;
    Line line;
    unique_ptr<LiChao> left, right;
    LiChao(int _l, int _r) : l(_l), r(_r), line(Line()), left(nullptr), right(nullptr) {}
    void insert(Line nueva, int L, int R) {
        if (R < l or r < L) return;
        if (L <= l and r <= R) {
            int m = (l+r)/2;
            bool izq_bet = nueva.eval(l) > line.eval(l);
            bool mid_bet = nueva.eval(m) > line.eval(m);

            if (mid_bet) swap(line, nueva);

            if (l == r) return;

            if (izq_bet != mid_bet) {
                if (!left) left = make_unique<LiChao>(l, m);
                left->insert(nueva, L, R);
            } else {
                if (!right) right = make_unique<LiChao>(m+1, r);
                right->insert(nueva, L, R);
            }
            return;
        }
        if (!left) left = make_unique<LiChao>(l, (l+r)/2);
        if (!right) right = make_unique<LiChao>((l+r)/2+1, r);
        left->insert(nueva, L, R);
        right->insert(nueva, L, R);
    }

    ll query(int x) {
        if (l == r) return line.eval(x);
        int m = (l+r)/2;
        ll res = line.eval(x);
        if (x <= m and left) res = max(res, left->query(x));
        if (x > m and right) res = max(res, right->query(x));
        return res;
    }
};


void solve(){
    int n, m;cin>>n>>m;
    LiChao tree(0, 1e9);
    for(int i = 0; i<n; i++){
        int a, b, c, d;cin>>a>>b>>c>>d;
        int dy = d - b;
        int dx = c - a;
        int pendiente = dy/dx;
        int constante = b - pendiente*a;
        tree.insert(Line(pendiente, constante), a, c);
    }
    for(int i = 0; i<=m; i++){
        int ans = tree.query(i);

        cout<<(ans == -INF?-1:ans)<<'\n';
    }
}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
    // cin>>t;
    while(t--){
        solve();
    }
}
```

4.0.9 Line_Segment_Trace_I

```cpp
// There are n line segments whose endpoints have integer coordinates. The left x-coordinate of each segment is 0 and the right x-coordinate is m. The slope of each segment is an integer.
// For each x-coordinate 0,1,\dots,m, find the maximum point in any line segment.
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for (int i = (a); i < (b); i++)
#define FOR(i, n) for (int i = 0; i < (n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;

/**
 * Author: Simon Lindholm
 * Date: 2017-04-20
 * License: CC0
 * Source: own work
 * Description: Container where you can add lines of the form kx+m, and query
 * maximum values at points x. Useful for dynamic programming. Time: O(\log N)
 * Status: tested
 */
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)
  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()) {
      x->p = inf;
      return false;
    }
    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));
  }
  void add_min(long long k, long long m) { add(-k, -m); }

  // consultar mínimo
  long long query_min(long long x) { return -query(x); }
  ll query(ll x) {
    assert(!empty());
    auto l = *lower_bound(x);
    return l.k * x + l.m;
  }
};

void solve() {
    int n, m;cin>>n>>m;
    LineContainer lc;
    for(int i = 0; i<n; i++){
        int a, b;cin>>a>>b;
        int dy = b-a;
        int dx = m - 0;
        int pendiente = dy/dx;
        int bb = a;
        lc.add(pendiente, bb);
    }
    for(int i = 0; i<=m; i++){
        cout<<lc.query(i)<<' ';
    }
}

signed main() {
  ios::sync_with_stdio(0);
  cin.tie(0);
  cout.tie(0);
  int t = 1;
//   cin >> t;
  while (t--) {
    solve();
  }
}
```

4.0.10 Maximum_Manhattan_Distances

```cpp
// A set is initially empty and n points are added to it. Calculate the maximum Manhattan distance of two points after each addition.
void solve(){
    int n;cin>>n;
    multiset<int> sum, diff;
    for(int i = 0; i<n; i++){
        int a, b;cin>>a>>b;
```

```
        sum.insert(a+b);
        diff.insert(a-b);
        auto ini_s = sum.begin();
        auto end_s = sum.end();
        end_s--;
        auto ini_d = diff.begin();
        auto end_d = diff.end();
        end_d--;
        cout<<max(*end_s - *ini_s, *end_d - *ini_d)<<'\n';
    }
}
```

4.0.11 Minimum_Euclidean_Distance

```
// Given a set of points in the two-dimensional plane, your task is to find the minimum Euclidean distance between two distinct points.
// The Euclidean distance of points (x_1,y_1) and (x_2,y_2) is \sqrt{(x_1-x_2)^2+(y_1-y_2)^2}.
#include <bits/stdc++.h>
using namespace std;
#pragma GCC optimize("Ofast")
#pragma GCC target("avx2,bmi,bmi2,lzcnt,popcnt")
#pragma GCC optimize("unroll-loops")

#define int long long
#define all(x) (x).begin(), (x).end()
const int INF_INT = 2000000000;

int dis(pair<int,int> a, pair<int,int> b){
    int dx = a.first - b.first;
    int dy = a.second - b.second;
    return dx*dx + dy*dy;
}

void solve(){
    int n; cin >> n;
    vector<pair<pair<int,int>, int>> ptos(n); // {{x,y}, id}
    for(int i=0;i<n;i++){
        int a,b; cin>>a>>b;
        ptos[i] = {{a,b}, i};
    }
    sort(all(ptos)); // orden por x

    // inicializar best (asegúrate n ≥ 2 según restricciones)
    int best = dis(ptos[0].first, ptos[1].first);
    pair<int,int> mejores = {ptos[0].second, ptos[1].second};

    // set ordenado por y: elemento = {{y,x}, id}
    set<pair<pair<int,int>, int>> ventana;

    int left = 0; // puntero que avanza para cerrar la ventana por X
    for(int i=0;i<n;i++){
        int x = ptos[i].first.first;
        int y = ptos[i].first.second;
        int id = ptos[i].second;

        // 1) eliminar del set los puntos que quedaron demasiado a la izquierda (por X)
        while(left < i){
            int x_left = ptos[left].first.first;
            int dx = x - x_left;
            if(dx*dx > best){
                int y_left = ptos[left].first.second;
                int id_left = ptos[left].second;
                ventana.erase({{y_left, x_left}, id_left});
                left++;
            } else break;
        }

        // 2) limitar por Y usando d = floor(sqrt(best)) para no iterar todo el set
        int d = (int)floor(sqrt((long double)best));
        auto it_low  = ventana.lower_bound({{y - d, -INF_INT}, -INF_INT});
        auto it_high = ventana.upper_bound({{y + d, INF_INT}, INF_INT});

        for(auto it = it_low; it != it_high; ++it){
            int x2 = it->first.second;
            int y2 = it->first.first;
            int id2 = it->second;
            int dx = x - x2;
            int dx2 = dx*dx;
            if(dx2 > best) continue; // filtro extra
            int dy = y - y2;
            int dist = dx2 + dy*dy;
            if(dist < best){
                best = dist;
                mejores = {id, id2};
            }
        }

        // 3) insertar el punto actual para futuros i
        ventana.insert({{y, x}, id});
    }

    cout << best << '\n';
    // cout << mejores.first << ' ' << mejores.second << '\n';
}

signed main(){
    ios::sync_with_stdio(0);
    cin.tie(0); cout.tie(0);
    solve();
}
```

4.0.12 Point_in_Polygon

```
// You are given a polygon of n vertices and a list of m points. Your task is to determine for each point if it is inside, outside or on the boundary of the polygon.
```

```cpp
// The polygon consists of n vertices (x_1,y_1),(x_2,y_2),\dots,(x_n,y_n). The vertices (x_i,y_i) and (x_{i+1},y_{i+1}) are adjacent for i=1,2,\dots,n-1, and the vertices (x_1,y_1) and (x_n,y_n) are also adjacent.
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for (int i = (a); i < (b); i++)
#define FOR(i, n) for (int i = 0; i < (n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;

template <typename T>
struct pto {
    T x, y;
    pto(T _x = 0, T _y = 0) : x(_x), y(_y) {}
    const pto operator+(const pto& otro) const {
        return pto(x + otro.x, y + otro.y);
    }
    const pto operator-(const pto& otro) const {
        return pto(x - otro.x, y - otro.y);
    }
    const pto operator*(const pto& otro) const {
        return pto(x * otro.x, y * otro.y);
    }
    const T operator%(const pto& otro) const { return x * otro.y - y * otro.x; }
    const bool operator==(const pto& otro) const {
        return x == otro.x and y == otro.y;
    }
    const pto operator*(const T k) const { return pto(x * k, y * k); }
};
const double EPS = 1e-9;
bool gr(double a, double b) {  // greater
    return a > b + EPS;
}

bool le(double a, double b) {  // less
    return a + EPS < b;
}

bool eq(double a, double b) { return fabs(a - b) < EPS; }
bool leq(double a, double b) { return le(a, b) or eq(a, b); }
bool dentro(const pto<double>& p, const pto<int>& a, const pto<int>& b) {
    return leq(min(a.x, b.x), p.x) and leq(p.x, max(a.x, b.x)) and
           leq(min(a.y, b.y), p.y) and leq(p.y, max(a.y, b.y));
}
bool dentro(const pto<int>& p, const pto<int>& a, const pto<int>& b) {
    return min(a.x, b.x) <= p.x and p.x <= max(a.x, b.x) and
           min(a.y, b.y) <= p.y and p.y <= max(a.y, b.y);
}

struct line {
    double a, b, c;
    line(pto<int> p, pto<int> q) {
        a = p.y - q.y;
        b = q.x - p.x;
        c = -a * p.x - b * p.y;
    };
};
double det(double a, double b, double c, double d) { return a * d - b * c; }
pto<double> intersec(line a,
                     line b) {    // primero estar seguro si no son paralelas
    double d = -det(a.a, a.b, b.a, b.b);
    return pto<double>(det(a.c, a.b, b.c, b.b) / d,
                       det(a.a, a.c, b.a, b.c) / d);
}

bool derecha(pto<int> x, pto<int> a, pto<int> b) {
    pto<int> uv = b - a, ux = x - a;
    return uv % ux > 0;
}
void solve() {
    int n, m;
    cin >> n >> m;
    vector<pto<int>> poly(n);
    for (int i = 0; i < n; i++) {
        int a, b;
        cin >> a >> b;
        poly[i] = {a, b};
    }
    while (m--) {
        int a, b;
        cin >> a >> b;
        pto x(a, b);
        int flag = -1;
        line recta(x, pto(a + 1, b));
        int veces = 0;
        for (int i = 0; i < n; i++) {
            int sig = (i + 1) % n;
            pto u = poly[i];
            pto v = poly[sig];
            pto uv = v - u;
            pto ux = x - u;
            if (uv % ux == 0 and dentro(x, u, v)) {
                flag = 1;
                break;
            } else {
                if ((u.y > x.y) != (v.y > x.y)) {
                    // calcular X de la intersección de la horizontal y = x.y
                    // con el segmento u-v v.y ≠= u.y por la condición anterior,
                    // así evitamos división por cero
                    double x_int = u.x + (double)(v.x - u.x) *
                                         (double)(x.y - u.y) /
                                         (double)(v.y - u.y);
```

```
                    // contar solo si la intersección está a la derecha de
                    ↪  x
                    // (rayo hacia +inf)
                    if (x_int > (double)x.x) veces++;
                }
            }
        }
        if (flag == 1) {
            cout << "BOUNDARY\n";
        } else {
            cout << (veces % 2 ? "INSIDE" : "OUTSIDE") << '\n';
        }
    }
}

signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    int t = 1;
    //   cin>>t;
    while (t--) { solve(); }
}
```

4.0.13 Point_Location_Test

```
// There is a line that goes through the points p_1=(x_1,y_1) and
↪  p_2=(x_2,y_2). There is also a point p_3=(x_3,y_3).
// Your task is to determine whether p_3 is located on the left or right
↪  side of the line or if it touches the line when we are looking from
↪  p_1 to p_2.
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define f first
#define sst stringstream
#define s second
#define pb push_back
#define sz(x) (int)(x).size()
#define all(a) (a).begin(), (a).end()
#define rall(a) (a).rbegin(), (a).rend()
#define fore(i, a, n) for (int i = (a); i < (n); i++)
#define forb(i, n) for (int i = (n) - 1; i >= 0; i--)
#define popcount(x) __builtin_popcountll(x);
typedef pair<int, int> pii;
typedef vector<int> vi;
const int MOD = 1000000007;
const double EPS = 1e-9;
const double PI = acos(-1);
const int INF = 1e18;
// PLUS ULTRA RECARGADO!!!

struct point {
    int x, y;
    point() {}
    point(int x, int y) : x(x), y(y) {}
    point& operator+=(const point& t) {
        x += t.x;
        y += t.y;
        return *this;
    }
    point& operator-=(const point& t) {
        x -= t.x;
        y -= t.y;
        return *this;
    }
    point& operator*=(int t) {
        x *= t;
        y *= t;
        return *this;
    }

    point& operator/=(int t) {
        x /= t;
        y /= t;
        return *this;
    }
    point operator+(const point& t) const { return point(*this) += t; }
    point operator-(const point& t) const { return point(*this) -= t; }
    point operator*(int t) const { return point(*this) *= t; }
    point operator/(int t) const { return point(*this) /= t; }

    int norm() const { return x * x + y * y; }
    double length() const { return sqrt(norm()); }
    bool operator<(const point& t) const { return tie(x, y) < tie(t.x,
    ↪  t.y); }
    bool operator==(const point& t) const { return x == t.x && y == t.y; }

    // point projection(const point& onto) const {
    //     int scalar = dot(*this, onto) / onto.norm();
    //     return onto * scalar;
    // }
    friend istream& operator>>(istream& is, point& p) {
        return is >> p.x >> p.y;
    }
    friend ostream& operator<<(ostream& os, const point& p) {
        return os << "(" << p.x << ", " << p.y << ")";
    }

    point rotate(double angle) const {
        double cs = cos(angle), sn = sin(angle);
        return point(round(x * cs - y * sn), round(x * sn + y * cs));
    }
};

int dot(point a, point b) {
    return a.x * b.x + a.y * b.y;
}
point operator*(int a, point b) {
    return b * a;
}
int cross(point& a, point& b) {
    return a.x * b.y - a.y * b.x;
```

```
}

void solve() {
    point a, b, c;
    cin >> a >> b >> c;
    point ab = b - a;
    point ac = c - a;
    int cros = cross(ab, ac);
    if (cros > 0) {
        cout << "LEFT\n";
    } else if (cros < 0) {
        cout << "RIGHT\n";
    } else {
        cout << "TOUCH\n";
    }
}
signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    int t;
    cin >> t;
    while (t--) solve();
    // solve();
}
```

4.0.14 Polygon_Area

```
// Your task is to calculate the area of a given polygon.
// The polygon consists of n vertices (x_1,y_1),(x_2,y_2),\dots,(x_n,y_n).
↪ The vertices (x_i,y_i) and (x_{i+1},y_{i+1}) are adjacent for
↪ i=1,2,\dots,n-1, and the vertices (x_1,y_1) and (x_n,y_n) are also
↪ adjacent.
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define f first
#define sst stringstream
#define s second
#define pb push_back
#define sz(x) (int)(x).size()
#define all(a) (a).begin(), (a).end()
#define rall(a) (a).rbegin(), (a).rend()
#define fore(i, a, n) for (int i = (a); i < (n); i++)
#define forb(i, n) for (int i = (n) - 1; i >= 0; i--)
#define popcount(x) __builtin_popcountll(x);
typedef long double ld;
typedef pair<int, int> pii;
typedef vector<int> vi;
const int MOD = 1000000007;
const double EPS = 1e-9;
const double PI = acos(-1);
const int INF = 1e18;
// PLUS ULTRA RECARGADO!!!
struct point_i {
    // Construccion
    int x, y;
    point_i() : x(0), y(0) {}
    point_i(int x, int y) : x(x), y(y) {}
    // Operadores
    point_i operator-() { return point_i(-x, -y); }
    point_i operator+(point_i p) { return point_i(x + p.x, y + p.y); }
    point_i operator-(point_i p) { return point_i(x - p.x, y - p.y); }
    point_i operator*(int k) { return point_i(k * x, k * y); }
    // producto punto
    int operator*(point_i p) { return x * p.x + y * p.y; }
    // producto cruz
    int operator%(point_i p) { return x * p.y - y * p.x; }
    // Comparadores
    bool operator==(const point_i &p) const { return x == p.x and y ==
    ↪ p.y; }
    bool operator!=(const point_i &p) const { return x != p.x or y !=
    ↪ p.y; }
    bool operator<(const point_i &p) const {
        return (x < p.x) or (x == p.x and y < p.y);
    }
    bool dentro(point_i &a, point_i &b) {
        return min(a.x, b.x) <= (*this).x and (*this).x <= max(a.x, b.x)
        ↪ and
               min(a.y, b.y) <= (*this).y and (*this).y <= max(a.y, b.y);
    }

    // Longitudes y distancias
    ld abs() { return sqrtl(x * x + y * y); }
    int abs2() { return x * x + y * y; }
    ld dist(point_i q) { return (*this - q).abs(); }
    int dist2(point_i q) { return (*this - q).abs2(); }
    // Angulo respecto al eje x (1,1) -> 45 (pero en radianes)
    ld arg() { return atan2l(y, x); }
};

// Rotacion horaria y antihoraria
point_i rotate_ccw90(point_i p) {
    return point_i(-p.y, p.x);
}
point_i rotate_cw90(point_i p) {
    return point_i(p.y, -p.x);
}

int area_2(point_i a, point_i b, point_i c) {
    return (a % b) + (b % c) + (c % a);
}
ostream &operator<<(ostream &os, const point_i &p) {
    os << "(" << p.x << "," << p.y << ")";
    return os;
}
istream &operator>>(istream &in, point_i &p) {
    in >> p.x >> p.y;
    return in;
}

void solve() {
    int n;
```

```
    cin >> n;
    vector<point_i> pol(n);
    fore(i, 0, n) {
        cin >> pol[i];
    }
    int ans = 0;
    fore(i, 0, n) {
        ans += (pol[i].x * pol[(i + 1) % n].y);
        ans -= (pol[i].y * pol[(i + 1) % n].x);
    }
    cout << abs(ans) << endl;
}
signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    int t = 1;
    // cin>>t;
    while (t--) solve();
}
```

4.0.15 Polygon_Lattice_Points

```
// Given a polygon, your task is to calculate the number of lattice points inside the polygon and on its boundary. A lattice point is a point whose coordinates are integers.
// The polygon consists of n vertices (x_1,y_1),(x_2,y_2),\dots,(x_n,y_n). The vertices (x_i,y_i) and (x_{i+1},y_{i+1}) are adjacent for i=1,2,\dots,n-1, and the vertices (x_1,y_1) and (x_n,y_n) are also adjacent.
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;

#define tipo long long
struct pto
{
    tipo x, y;
    pto(tipo _x = 0, tipo _y = 0): x(_x), y(_y){}
    const pto operator + (const pto& otro) const {
        return pto(x + otro.x, y + otro.y);
    }
    const pto operator - (const pto& otro) const {
        return pto(x  - otro.x, y - otro.y);
    }
    const pto operator*(const pto& otro) const {
        return pto(x*otro.x, y*otro.y);
    }
    const tipo operator%(const pto& otro) const {
        return x*otro.y - y*otro.x;
    }
    const bool operator ==(const pto& otro) const{
        return x == otro.x and y == otro.y;
    }
    const pto operator*(const tipo k)const{
        return pto(x*k, y*k);
    }

};

ll boundary_points(const vector<pto>& poly, const int n){
    ll ans = 0;
    for(int i = 0; i<n;i++){
        int sig = (i+1)%n;
        ll dx = abs(poly[i].x - poly[sig].x);
        ll dy = abs(poly[i].y - poly[sig].y);
        ans+=gcd(dx, dy);
    }
    return ans;
}

void solve(){
   int n;cin>>n;
   vector<pto> poly(n);
   for(int i = 0; i<n; i++){
    tipo a, b;cin>>a>>b;
    poly[i] = {a, b};
   }
   ll area2 = 0;
   for(int i = 0; i<n; i++){
    int sig = (i+1)%n;
    auto[x1, y1] = poly[i];
    auto[x2, y2] = poly[sig];
    area2+=((x1*y2)-(y1*x2));
   }
   area2 = abs(area2);
   ll b = boundary_points(poly, n);
   ll ans = area2 - b + 2;
   cout<<ans/2 << ' '<<b<<'\n';
}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
    //cin>>t;
    while(t--){
        solve();
    }
}
```

5 Graph Algorithms

5.0.1 Cycle_Finding

```cpp
#include <bits/stdc++.h>
#define ll long long int
#define INF 1000000000000000LL
//You are given a directed graph, and your task is to find out if it contains a negative cycle, and also give an example of such a cycle.


using namespace std;

// Lista de adyacencia para almacenar el grafo ponderado
vector<pair<ll, ll>> grafo[2501];

// Arreglos para guardar las distancias, padres y el número de relajaciones
ll distancias[2501];
ll padres[2501];
ll contadorRelajaciones[2501];

ll numNodos, numAristas, nodoInicio, nodoDestino, pesoArista;
bool enCola[2501]; // Arreglo para saber si un nodo está en la cola
bool visitado[2501]; // Arreglo para saber si un nodo ya fue visitado
ll nodoConCiclo; // Nodo involucrado en el ciclo negativo

// Función que implementa el algoritmo SPFA (Shortest Path Faster Algorithm)
bool spfa(ll inicio) {
    // Inicializamos la distancia del nodo de inicio a 0
    distancias[inicio] = 0;
    padres[inicio] = -1; // El nodo de inicio no tiene un padre

    queue<ll> cola; // Cola para procesar los nodos
    cola.push(inicio);
    enCola[inicio] = true; // Marcamos como que está en la cola

    while (!cola.empty()) {
        ll nodoActual = cola.front(); // Sacamos el primer nodo de la cola
        visitado[nodoActual] = true;
        enCola[nodoActual] = false;
        cola.pop();

        // Iteramos sobre los vecinos del nodo actual
        for (auto vecino : grafo[nodoActual]) {
            // Si la distancia al vecino es mayor que la distancia al nodo actual más el peso de la arista
            if (distancias[vecino.first] > distancias[nodoActual] + vecino.second) {
                contadorRelajaciones[vecino.first]++; // Aumentamos el contador de relajaciones
                // Si el nodo ha sido relajado más de "numNodos" veces, hay un ciclo negativo
                if (contadorRelajaciones[vecino.first] > numNodos) {
                    nodoConCiclo = vecino.first;
                    padres[vecino.first] = nodoActual;
                    return false; // Se encuentra un ciclo negativo, retornamos falso
                }
                // Actualizamos la distancia al vecino
                distancias[vecino.first] = distancias[nodoActual] + vecino.second;
                // Si el vecino no está en la cola, lo agregamos
                if (!enCola[vecino.first]) {
                    cola.push(vecino.first);
                    enCola[vecino.first] = true;
                }
                // Actualizamos el nodo padre del vecino
                padres[vecino.first] = nodoActual;
            }
        }
    }
    return true; // Si no encontramos ningún ciclo negativo, retornamos true
}

int main() {
    // Leemos el número de nodos y aristas
    cin >> numNodos >> numAristas;

    // Inicializamos todas las distancias a infinito (INF)
    for (int i = 1; i <= numNodos; i++) {
        distancias[i] = INF;
    }

    // Leemos las aristas (nodo de inicio, nodo de destino, peso)
    for (ll i = 0; i < numAristas; i++) {
        cin >> nodoInicio >> nodoDestino >> pesoArista;
        // Añadimos la arista al grafo
        grafo[nodoInicio].push_back({nodoDestino, pesoArista});
    }

    // Intentamos ejecutar SPFA desde cada nodo
    for (ll i = 1; i <= numNodos; i++) {
        // Si encontramos un ciclo negativo desde el nodo "i"
        if (!spfa(i)) {
            cout << "YES" << endl; // Imprimimos que hay un ciclo negativo
            ll nodo = nodoConCiclo;
            stack<ll> pila;
            bool enPila[2501] = {}; // Arreglo para verificar si el nodo está en la pila

            // Reconstruimos el ciclo negativo utilizando el arreglo de padres
            while (!enPila[nodo]) {
                enPila[nodo] = true;
                pila.push(nodo);
                nodo = padres[nodo];
            }

            // Imprimimos el ciclo negativo
            cout << nodo << " "; // Imprimimos el primer nodo del ciclo
            while (pila.top() != nodo) { // Imprimimos el resto de los nodos en el ciclo
```

```
                cout << pila.top() << " ";
                pila.pop();
            }
            cout << nodo << endl; // Imprimimos el último nodo del ciclo
            return 0;
        }
    }

    cout << "NO" << endl; // Si no encontramos ciclo negativo, imprimimos "NO"
}
```

5.0.2 Flight_Routes

```
// Your task is to find the k shortest flight routes from Syrjälä to Metsälä. A route can visit the same city several times.
// Note that there can be several routes with the same price and each of them should be considered (see the example).
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 4e18;
const int tam = 1e5+10;
int n, m, k;
vector<pair<int, int>> g[tam];

vector<vector<int>> dijkstra(){
    vector<vector<int>> dist(n+1, vector<int>(k, INF));
    priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>> pq;
    dist[1][0] = 0;
    pq.push({0, 1});
    while(!pq.empty()){
        auto[d, u] = pq.top();
        pq.pop();
        if(d > dist[u][k-1])continue;
        for(auto[v, w]:g[u]){
            if(d + w < dist[v][k-1]){
                dist[v][k-1] = d + w;
                pq.push({dist[v][k-1], v});
                sort(all(dist[v]));
            }
        }
    }
    return dist;
}

void solve(){
    cin>>n>>m>>k;
    for(int i =0; i<m; i++){
        int a, b, c;cin>>a>>b>>c;
        g[a].push_back({b, c});
    }
    auto d = dijkstra();
    for(int i = 0; i<k; i++){
        cout<<d[n][i]<<' ';
    }
    cout<<'\n';
}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
 //   cin>>t;
    while(t--){
        solve();
    }
}
```

5.0.3 Game_Levels

```
// A game has n levels, connected by m teleporters, and your task is to get from level 1 to level n. The game has been designed so that there are no directed cycles in the underlying graph. In how many ways can you complete the game?
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define ll long long
const int tam = 100000+10;
const int MOD = 1000000000+7;
#define sz(x) (int)(x).size()
int n, m;
vector<vector<int>> adj(tam);
vector<int> dist(tam, -1);
vector<int> topsort(tam);
int dfs(int nodo){
    if(nodo == n){
        return 1;
    }else if(sz(adj[nodo]) == 0){
        return 0;
    }else if(dist[nodo] == -1){
        dist[nodo] = 0;
        for(int vecino:adj[nodo]){
            int tam = dfs(vecino);
            dist[nodo]+=tam;
            dist[nodo]%=MOD;
        }
    }
    return dist[nodo];
}
```

```
void solve(){
    cin>>n>>m;
    for(int i = 0; i<m; i++){
        int a, b;cin>>a>>b;
        adj[a].push_back(b);
    }
    cout<<dfs(1)<<endl;
}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
 //   cin>>t;
    while(t--){
        solve();
    }
}
```

5.0.4 HIgh_Score

```
// You play a game consisting of n rooms and m tunnels. Your initial score is 0, and each tunnel increases your score by x where x may be both positive or negative. You may go through a tunnel several times.
// Your task is to walk from room 1 to room n. What is the maximum score you can get?
#include <bits/stdc++.h>
using namespace std;
#define int long long
const int INF = 1e18;

// Función para determinar accesibilidad desde un nodo
void bfs(int start, vector<vector<int>>& graph, vector<bool>& visited) {
    queue<int> q;
    q.push(start);
    visited[start] = true;
    while (!q.empty()) {
        int u = q.front(); q.pop();
        for (int v : graph[u]) {
            if (!visited[v]) {
                visited[v] = true;
                q.push(v);
            }
        }
    }
}

int bellman_ford(int n, int start, const vector<tuple<int, int, int>>& edges) {
    vector<int> distancia(n, INF);
    distancia[start] = 0;

    // Relajar las aristas n-1 veces
    for (int i = 0; i < n - 1; ++i) {
        for (const auto& e : edges) {
            auto [a, b, cost] = e;
            if (distancia[a] < INF && distancia[a] + cost < distancia[b]) {
                distancia[b] = max(-INF, distancia[a] + cost);
            }
        }
    }

    // Detectar ciclos negativos
    vector<bool> in_cycle(n, false);
    for (int i = 0; i < n; ++i) {
        for (const auto& e : edges) {
            auto [a, b, cost] = e;
            if (distancia[a] < INF && distancia[a] + cost < distancia[b]) {
                distancia[b] = -INF; // Marcar nodo afectado por ciclo negativo
                in_cycle[b] = true;
            }
        }
    }
    return distancia[n - 1]; // Distancia al último nodo
}

void solve() {
    int n, m;
    cin >> n >> m;

    vector<tuple<int, int, int>> edges(m);
    vector<vector<int>> graph(n), reverse_graph(n);

    for (int i = 0; i < m; ++i) {
        int a, b, c;
        cin >> a >> b >> c;
        --a, --b;
        edges[i] = {a, b, -c}; // Invertimos el signo del costo para maximizar
        graph[a].push_back(b);
        reverse_graph[b].push_back(a);
    }

    // Verificar accesibilidad desde el inicio y al final
    vector<bool> reachable_from_start(n, false), reachable_from_end(n, false);
    bfs(0, graph, reachable_from_start);
    bfs(n - 1, reverse_graph, reachable_from_end);

    // Ejecutar Bellman-Ford
    int result = bellman_ford(n, 0, edges);

    // Verificar si un ciclo negativo afecta la solución
    for (int i = 0; i < n; ++i) {
        if (reachable_from_start[i] && reachable_from_end[i] && result == -INF) {
            cout << -1 << endl;
            return;
        }
```

```
    }

    // Si no hay ciclos negativos relevantes, devolver la distancia
    cout << -result << endl;
}

signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    solve();
}
```

5.0.5 Longest_Flight_Route

```
// Uolevi has won a contest, and the prize is a free flight trip that can consist of one or more flights through cities. Of course, Uolevi wants to choose a trip that has as many cities as possible.
// Uolevi wants to fly from Syrjälä to Lehmälä so that he visits the maximum number of cities. You are given the list of possible flights, and you know that there are no directed cycles in the flight network.
#include <bits/stdc++.h>
using namespace std;
#define int long long
const int INF = 1000000000000;
const int tam = 100000 + 1;
#define sz(x) (int)(x).size()
int n;
vector<vector<pair<int, int>>> grafo;
vector<bool> visi;
vector<int> topsort;
vector<int> dis;
vector<int> padre;
void dfs(int nodo) {
    visi[nodo] = true;
    for (auto [vecino, peso] : grafo[nodo]) {
        if (!visi[vecino]) { dfs(vecino); }
    }
    topsort.push_back(nodo);
}

void encontrarCaminoMasLargo(int origen) {
    for (int i = 1; i <= n; i++) {
        if (!visi[i]) { dfs(i); }
    }
    reverse(topsort.begin(), topsort.end());
    dis.resize(n + 1, -INF);
    dis[origen] = 0;
    for (int nodo : topsort) {
        if (dis[nodo] == -INF) continue;
        for (auto [vecino, peso] : grafo[nodo]) {
            if (dis[vecino] < dis[nodo] + peso) {
                dis[vecino] = dis[nodo] + peso;
                padre[vecino] = nodo;
            }
        }
    }
}
void solve() {
    int m;
    cin >> n >> m;
    padre.resize(n + 1, -1);
    grafo.resize(n + 1);
    visi.resize(n + 1, false);
    for (int i = 0; i < m; i++) {
        int a, b;
        cin >> a >> b;
        grafo[a].push_back({b, 1});
    }
    encontrarCaminoMasLargo(1);
    if (dis[n] == -INF) {
        cout << "IMPOSSIBLE\n";
        return;
    }
    cout << dis[n]+1 << endl;
    int it = n;
    deque<int> path;
    path.push_front(n);
    while (padre[it] != -1) {
        path.push_front(padre[it]);
        it = padre[it];
    }
    for(int i:path)cout<<i<<' ';
    cout<<"\n";
}

signed main() {
    // ios::sync_with_stdio(0);
    // cin.tie(0);
    // cout.tie(0);
    int t = 1;
    //   cin>>t;
    while (t--) { solve(); }
}
```

5.0.6 Monsters

```
// You and some monsters are in a labyrinth. When taking a step to some direction in the labyrinth, each monster may simultaneously take one as well. Your goal is to reach one of the boundary squares without ever sharing a square with a monster.

// Your task is to find out if your goal is possible, and if it is, print a path that you can follow. Your plan has to work in any situation; even if the monsters know your path beforehand.
#include <bits/stdc++.h>
using namespace std;
#define int long long


```

```cpp
int n, m;
vector<pair<int, int>> monstruos;
vector<vector<int>> lava(1000 + 10, vector<int>(1000 + 10, INT_MAX));
pair<int, int> inicio, fin;
                            // 0 = U   1 = D   2 = R    3 = L
                            //Estos indices se usan para calc
vector<pair<int, int>> mov = {{-1, 0}, {1, 0}, {0, 1}, {0, -1}};
char calc(int x){
    char res;
    switch (x)
    {
    case 0:
        res = 'U';
        break;
    case 1:
        res = 'D';
        break;
    case 2:
        res = 'R';
        break;
    case 3:
        res = 'L';
        break;
    }
    return res;
}
vector<vector<char>> direcciones(1000 + 10, vector<char>(1000 + 10, '#'));
void reconstruircamino(){
    vector<char> res;
    auto[x, y] = fin;
    while(direcciones[x][y] != '$'){
        char aux = direcciones[x][y];
        res.push_back(aux);
        if(aux == 'U'){
            x++;
        }else if(aux == 'D'){
            x--;
        }else if(aux == 'L'){
            y++;
        }else{
            y--;
        }
    }
    cout<<(int)res.size()<<'\n';
    reverse(res.begin(), res.end());
    for(char& it:res){
        cout<<it;
    }
    cout<<'\n';
}
bool esvalida(int x, int y, int tiempo)
{
    if (x < 0 or y < 0 or x >= n or y >= m)
    {
        return false;
    }
    if (lava[x][y] <= tiempo)
    {
        return false;
    }
    return true;
}

bool esSalida(int x, int y, int tiempo)
{
    if (!esvalida(x, y, tiempo))
        return false;
    if (x == 0 or y == 0 or
        x == n - 1 or y == m - 1)
        return true;
    return false;
}

bool bfsDeSalida()
{
    queue<pair<pair<int, int>, int>> q;
    q.push(make_pair(inicio, 0));
    direcciones[inicio.first][inicio.second] = '$';
    while (!q.empty())
    {
        int cx = q.front().first.first;
        int cy = q.front().first.second;
        int tiempo = q.front().second;
        tiempo++;
        q.pop();
        for (int i = 0; i<4; i++)
        {
            auto mv = mov[i];
            int tx = cx + mv.first;
            int ty = cy + mv.second;
            if (esSalida(tx, ty, tiempo))
            {
                direcciones[tx][ty] = calc(i);
                fin = {tx, ty};
                return true;
            }
            if (esvalida(tx, ty, tiempo))
            {
                direcciones[tx][ty] = calc(i);
                lava[tx][ty] = tiempo;
                q.push({{tx, ty}, tiempo});
            }
        }
    }
    return false;
}

void precalculoLava()
{
    queue<pair<pair<int, int>, int>> q;
    for (auto m : monstruos)
    {
        q.push(make_pair(m, 0));
    }
    while (!q.empty())
    {
```

```
        int cx = q.front().first.first;
        int cy = q.front().first.second;
        int tiempo = q.front().second;
        tiempo++;
        q.pop();

        for (auto mv : mov)
        {
            int tx = cx + mv.first;
            int ty = cy + mv.second;
            if (esvalida(tx, ty, tiempo))
            {
                lava[tx][ty] = tiempo;
                q.push({{tx, ty}, tiempo});
            }
        }
    }
}

signed main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    cin >> n >> m;
    for (int i = 0; i < n; ++i)
    {
        for (int j = 0; j < m; ++j)
        {
            char c;
            cin >> c;
            if (c == '#')
            {
                lava[i][j] = 0;
            }
            else if (c == 'M')
            {
                lava[i][j] = 0;

                monstruos.push_back({i, j});
            }
            else if (c == 'A')
            {
                lava[i][j] = 0;
                inicio = {i, j};
            }
            else
            {
                lava[i][j] = INT_MAX;
            }
        }
    }
    if (inicio.first == 0 or inicio.second == 0 or inicio.first == n - 1
    ↪  or inicio.second == m - 1)
    {
        cout << "YES" << endl;
        cout << 0;
        return 0;
    }
    precalculoLava();

    if (!bfsDeSalida())
    {
        cout << "NO";
        return 0;
    }
    cout << "YES" << endl;
    reconstruircamino();
}
```

5.0.7 Round_Trip

```
// Byteland has n cities and m roads between them. Your task is to design
↪  a round trip that begins in a city, goes through two or more other
↪  cities, and finally returns to the starting city. Every intermediate
↪  city on the route has to be distinct.
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define uset unordered_set
#define umap unordered_map
#define mp make_pair
#define pb push_back
#define all(a) (a).begin(), (a).end()
#define rall(a) (a).rbegin(), (a).rend()
#define floatigual(a, b) (fabs(a - b) < EPS)
#define mod(a) md(a, MOD)
#define FOR(i, n) for (int i = 0; i < (n); ++i)
#define FOR3(i, a, b) for (int i = (a); i < (b); ++i)
#define FORD(i, n) for (int i = (n) - 1; i >= 0; --i)
#define FORDD(i, a, b) for (int i = (b) - 1; i >= (a); --i)
#define techo(a, b) (a / b + (a % b =!= 0))
#define popcount(x) __builtin_popcountll(x);
using namespace std;
typedef long double ld;
typedef unsigned long long ull;
typedef pair<int, int> pii;
typedef vector<int> vi;
typedef vector<bool> vbol;

void solve() {
    int n, m;
    cin >> n >> m;
    vector<vi> g(n + 1);
    vbol visi(n + 1);
    vi losamopadres(n + 1);

    while (m--) {
        int a, b;
        cin >> a >> b;
        g[a].pb(b);
        g[b].pb(a);
    }

```

```
    bool hay = false;
    int com = -1, fin = -1;

    FOR3(i, 1, n + 1) {
        if (visi[i]) continue;
        stack<pii> pila;

        // Nodo actual, nodo padre
        pila.push({i, -1});
        losamopadres[i] = -1;

        while (!pila.empty()) {
            auto [nodo, padre] = pila.top();
            pila.pop();

            visi[nodo] = true;
            losamopadres[nodo] = padre;

            for (int vecino : g[nodo]) {
                if (vecino == padre) continue;

                if (visi[vecino]) {
                    // Si ya fue visitado, detectamos el ciclo
                    com = vecino;
                    fin = nodo;
                    hay = true;
                    break;
                }

                if (!visi[vecino]) {
                    pila.push({vecino, nodo});
                }
            }
            if (hay) break;
        }
        if (hay) {
            // Reconstrucción del ciclo
            vi res;
            res.pb(fin); // Nodo donde se detectó el ciclo
            while (fin != com) {
                res.pb(losamopadres[fin]);
                fin = losamopadres[fin];
            }
            //res.pb(com); // Anadimos el nodo donde comenzó el ciclo
            res.pb(res[0]); // Para cerrar el ciclo

            cout << res.size() << endl;
            for (int i : res) {
                cout << i << ' ';
            }
            return;
        }
    }
    cout << "IMPOSSIBLE\n";
}

signed main() {
    //int t;cin>>t;while(t--)solve();
    solve();
}
```

5.0.8 Round_Trip_II

```
// Byteland has n cities and m flight connections. Your task is to design a round trip that begins in a city, goes through one or more other cities, and finally returns to the starting city. Every intermediate city on the route has to be distinct.
#include <bits/stdc++.h>
using namespace std;

const int tam = 1e5 + 10;
int n, m;
vector<int> g[tam];
int estado[tam];
vector<int> path;
pair<int,int> ese = {-1,-1};

bool dfs(int u) {
    estado[u] = 1;
    path.push_back(u);

    for(int v : g[u]) {
        if(estado[v] == 0) {
            if(dfs(v)) return true;
        } else if(estado[v] == 1) {
            ese = {u, v};
            return true;
        }
    }

    path.pop_back();
    estado[u] = 2;
    return false;
}

void solve() {
    cin >> n >> m;
    for(int i = 1; i <= n; i++) g[i].clear();
    fill(estado, estado+tam, 0);
    path.clear();
    ese = {-1,-1};

    for(int i = 0; i < m; i++) {
        int a, b;
        cin >> a >> b;
        g[a].push_back(b);
    }

    for(int i = 1; i <= n; i++)
        if(estado[i] == 0)
            if(dfs(i)) break;

    if(ese.first == -1) {
```

```
        cout << "IMPOSSIBLE\n";
        return;
    }

    vector<int> ans;
    bool started = false;
    for(int node : path) {
        if(node == ese.second) started = true;
        if(started) ans.push_back(node);
        if(node == ese.first and started) break;
    }
    ans.push_back(ese.second);

    cout << ans.size() << "\n";
    for(int x : ans) cout << x << " ";
    cout << "\n";
}

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

    int t = 1;
    while(t--) solve();
}
```

5.0.9 Shortest_Routes

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

const int INF = LONG_MAX/100;


void dijkstra(vector<vector<pii>>& grafo, int cant, int ini){
    vi dis(cant+1, INF);
    dis[ini] = 0;
    priority_queue<pii, vector<pii>, greater<pii>> pq;
    pq.push({0, ini});
    while(!pq.empty()){
        pii menor = pq.top();pq.pop();
        int w = menor.first;
        int nodo = menor.second;
        if(dis[nodo]<w)continue;
        for(auto a:grafo[nodo]){
            int v = a.first;
            int costo = a.second;
            if(dis[nodo] + costo <dis[v]){
                dis[v] = dis[nodo]+costo;
                pq.push({dis[v], v});
            }
        }
    }
    for(int i = 1;i<=cant; i++){
        cout<<dis[i]<<' ';
    }
}


void solve()
{
    int n, m;cin>>n>>m;
    vector<vector<pii>> grafo(n+1);
    while(m--){
        int a, b, c;cin>>a>>b>>c;
        grafo[a].pb({b,c});
    }
    dijkstra(grafo, n, 1);
    cout<<'\n';
}

signed main()
{

    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    int t;
    //cin >> t;
    //while (t--)
      //  solve();
    solve();
}
```

5.0.10 Shortest_Routes_II

```
// There are n cities and m roads between them. Your task is to process q
↪  queries where you have to determine the length of the shortest route
↪  between two given cities.
#include <bits/stdc++.h>
using namespace std;
#define int ll
#define uset unordered_set
#define umap unordered_map
#define mp make_pair
#define pb push_back
#define all(a) (a).begin(), (a).end()
#define rall(a) (a).rbegin(), (a).rend()
#define floatigual(a, b) (fabs(a - b) < EPS)
#define mod(a) md(a, MOD)
#define FOR(i, n) for (int i = 0; i < (n); ++i)
#define FOR3(i, a, b) for (int i = (a); i < (b); ++i)
#define FORD(i, n) for (int i = (n) - 1; i >= 0; --i)
#define FORDD(i, a, b) for (int i = (b) - 1; i >= (a); --i)
#define si cout << "YES" << endl
#define no cout << "NO" << endl

vector<vi> floyd(vector<vector<pii>>& grafo, int n){
    vector<vector<int>> dis(n+1, vi(n+1, INF));
    for(int i = 1;i<=n; i++){
```

```
        dis[i][i] = 0;
    }
    FOR3(u, 1, n+1){
        for(auto donde:grafo[u]){
            int v = donde.first;int w = donde.second;
            dis[u][v] = min(dis[u][v], w);
            dis[v][u] = min(dis[v][u], w);
        }
    }
    FOR3(k, 1, n+1){
        FOR3(u, 1, n+1){
            FOR3(v, 1, n+1){
                dis[u][v] = min(dis[u][v], dis[u][k] + dis[k][v]);
            }
        }
    }
    return dis;
}

void solve()
{
    //Floi warshal pe
    int n, m, q;cin>>n>>m>>q;
    unionFind dsu(n+1);
    vector<vector<pii>>grafo(n+1);
    while(m--){
        int a, b, c;cin>>a>>b>>c;
        grafo[a].pb({b, c});
        grafo[b].pb({a, c});
        dsu.join(a, b);
    }
    auto var = floyd(grafo, n);
    while(q--){
        int a, b;cin>>a>>b;
        if(dsu.find(a)!=dsu.find(b)){
            cout<<"-1\n";
        }else{
            cout<<var[a][b]<<'\n';
        }
    }
}

signed main()
{

    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    int t;
    //cin >> t;
    //while (t--)
      //  solve();
    solve();
}
```

6 Introductory Problems

6.0.1 Apple_Division

```
#include <bits/stdc++.h>
#define int long long
using namespace std;
int arr[21];
void dp(pair<int, int> par, int &mini, int i, int n) {
    if (i == n) {
        mini = min(mini, abs(par.first - par.second));
    } else {
        par.first += arr[i];
        dp(par, mini, i + 1, n);
        par.first -= arr[i];
        par.second += arr[i];
        dp(par, mini, i + 1, n);
    }
}
const int INF = 1e18;
signed main() {
    int n;
    cin >> n;
    for (int i = 0; i < n; i++) cin >> arr[i];
    pair<int, int> ini = {0, 0};
    int mini = INF;
    dp(ini, mini, 0, n);
    cout << mini << endl;
}
```

6.0.2 Bit_Strings

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;

int expMod(int base, int exponente, int mod) {
    int res = 1;
    base %= mod;
    while (exponente > 0) {
        if (exponente %2 == 1)
            res = (res * base) % mod;
        exponente >>= 1;
        base = (base * base) % mod;
    }
```

```
    return res;
}

void solve(){
    int n;cin>>n;cout<<expMod(2,n,MOD);
}

signed main(){

    ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
    //int t;cin>>t;while(t--)solve();
    solve();
}
```

6.0.3 Coin_Piles

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for (int i = (a); i < (b); i++)
#define FOR(i, n) for (int i = 0; i < (n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;
//You have two coin piles containing a and b coins.
//On each move, you can either remove one coin from the left pile
//and two coins from the right pile, or two coins from
// the left pile and one coin from the right pile.
//Your task is to efficiently find out if you can empty both the piles.
void solve() {
    int a, b;
    cin >> a >> b;
    if (a > b) { swap(a, b); }
    a = a * 2 - b;
    if (a % 3 == 0 && a >= 0) {
        cout << "YES\n";
    } else {
        cout << "NO\n";
    }
}

signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    int t;
    cin >> t;
    while (t--) solve();
    // solve();
}
```

6.0.4 Creating_Strings

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for (int i = (a); i < (b); i++)
#define FOR(i, n) for (int i = 0; i < (n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;

void solve() {
    string cad;
    cin >> cad;
    sort(all(cad));
    vector<string> vec;
    do { vec.pb(cad); } while (next_permutation(all(cad)));
    cout << vec.size() << endl;
    sort(all(vec));
    for (auto &c : vec) { cout << c << endl; }
}

signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    // int t;cin>>t;while(t--)solve();
    solve();
}
```

6.0.5 Gray_Code

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for (int i = (a); i < (b); i++)
#define FOR(i, n) for (int i = 0; i < (n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;

```

```cpp
int binpow(int a, int b) {
    int res = 1;
    while (b > 0) {
        if (b & 1)
            res = res * a;
        a = a * a;
        b >>= 1;
    }
    return res;
}

void solve() {
    int n;
    cin >> n;
    int l = binpow(2, n);
    for (int i = 0; i < l; i++) {
        int x = i ^ (i >> 1);
        bitset<32> b(x);
        string s = b.to_string();
        cout << s.substr(32 - n) << endl;
    }
}

signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    solve();
    // int t;cin>>t;while(t--)solve();
}
```

6.0.6 Increasing_Array

```cpp
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;

void solve(){
    int n;cin>>n;
    vi vec(n);FOR(i, n)cin>>vec[i];
    int cont = 0;
    for(int i = 1;i<n; i++){
        if(vec[i]<vec[i-1]){
            int mov = vec[i-1]-vec[i];
            cont+=mov;
            vec[i]+=mov;
        }
    }
    cout<<cont<<'\n';
}
signed main(){

    ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
    //int t;cin>>t;while(t--)solve();
    solve();
}
```

6.0.7 Missing_Number

```cpp
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;

void solve(){
    int n;cin>>n;
    vi vec(n-1);
    FOR(i, n)cin>>vec[i];
    sort(all(vec));
    int mex = 1;
    for(int a:vec){
        if(mex == a){
            mex++;
        }else{
            cout<<mex<<endl;
            return;
        }
    }
    cout<<mex<<endl;
}

signed main(){

    ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
    //int t;cin>>t;while(t--)solve();
}
```

6.0.8 Palindrome_Reorder

```cpp
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for (int i = (a); i < (b); i++)
#define FOR(i, n) for (int i = 0; i < (n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;

string stringuear(int a) {
    char c = (char)a;
    string res = "";
    res += c;
    return res;
}

void solve() {
    string cad;
    cin >> cad;
    int n = (int)cad.size();
    vector<int> letras(26);
    for (char &c : cad) { letras[c - 'A']++; }
    int imp = 0;
    char algo;
    for (int i = 0; i < 26; i++) {
        int rep = letras[i];
        if (rep % 2 != 0) {
            imp++;
            algo = char(i + 'A');
        }
    }
    if (imp > 1) {
        cout << "NO SOLUTION\n";
        return;
    }
    string res = "";
    for (int i = 0; i < 26; i++) {
        if (letras[i] % 2 != 0) { continue; }
        for (int j = 0; j < letras[i] / 2; j++) {
            res.append(stringuear(i + 'A'));
        }
    }
    if (imp != 0) {
        cout << res;
        for (int i = 0; i < letras[algo - 'A']; i++) { cout << algo; }
    } else {
        cout << res;
    }
    for (int i = (int)res.size() - 1; i >= 0; i--) { cout << res[i]; }
}

signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    // int t;cin>>t;while(t--)solve();
    solve();
}
```

6.0.9 Permutation

```cpp
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for (int i = (a); i < (b); i++)
#define FOR(i, n) for (int i = 0; i < (n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;

void solve() {
    int n;
    cin >> n;
    if (n == 2 || n == 3) {
        cout << "NO SOLUTION\n";
    } else if (n == 4) {
        cout << "2 4 1 3";
    } else {
        vi vec(n);
        int cont = n;
        for (int i = 0; i < n; i += 2) { vec[i] = cont--; }
        for (int i = 1; i < n; i += 2) { vec[i] = cont--; }
        for (auto a : vec) { cout << a << ' '; }
        cout << '\n';
    }
}
signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    // int t;cin>>t;while(t--)solve();
    solve();
}
```

6.0.10 Repeitions

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

```cpp
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;

void solve(){
    string cad;cin>>cad;
    int n = (int)(cad.size());
    int cont = 1;
    int m = 1;
    for(int i = 1; i < n; i++){
        if(cad[i] == cad[i - 1]){
            cont++;
        }else{
            m = max(m, cont);
            cont = 1;
        }
    }
    m = max(m, cont);
    cout<<m<<endl;
    }
signed main(){

    ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
    //int t;cin>>t;while(t--)solve();
    solve();
}
```

6.0.11 String_Reorder

```cpp
#include <bits/stdc++.h>
using namespace std;
#define sz(x) (int)(x).size()
void solve() {
    string s;
    cin >> s;
    vector<char> ans(sz(s));
    map<char, int> mp;
    int n = sz(s);
    int maxi = 0;
    for (char& c : s) {
        mp[c]++;
        maxi = max(maxi, mp[c]);
    }
    if (maxi > (n + 1) / 2) {
        cout << "-1\n";
    } else {
        char last = '(';
        priority_queue<pair<char, int>, vector<pair<char, int>>,
                       greater<pair<char, int>>>
            pq;
        for (auto x : mp) pq.push(x);
        for (int i = 0; i < n; i++) {
            deque<pair<char, int>> otro;
            bool found = false;
            pair<char, int> elegido;

            while (!pq.empty()) {
                auto top = pq.top();
                pq.pop();
                if (!found && top.first != last) {
                    elegido = top;
                    found = true;
                } else {
                    otro.push_back(top);
                }
            }

            if (!found) {
                if (otro.empty()) {
                    elegido = {last, 0};
                    found = true;
                } else {
                    cout << "-1\n";
                    return;
                }
            }
            ans[i] = elegido.first;
            last = elegido.first;
            elegido.second--;
            if (elegido.second > 0) pq.push(elegido);
            while (!otro.empty()) pq.push(otro.front()), otro.pop_front();
        }
        for (char c : ans) cout << c;
        cout << '\n';
    }
}
signed main() {
    int t = 1;
    // cin>>t;
    while (t--) solve();
}
```

6.0.12 Trailing_Zeros

```cpp
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for (int i = (a); i < (b); i++)
#define FOR(i, n) for (int i = 0; i < (n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
```

```cpp
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;
//Your task is to calculate the number of trailing zeros in the factorial n!.
//For example, 20! = 2432902008176640000 and it has 4 trailing zeros.
void solve() {
    int n;
    cin >> n;
    int cont = 0;
    for (int i = 5; n / i >= 1; i *= 5) { cont += n / i; }
    cout << cont << '\n';
}

signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    // int t;cin>>t;while(t--)solve();
    solve();
}
```

6.0.13 Two_Knights

```cpp
#include<bits/stdc++.h>
using namespace std;
#define int long long
signed main(){
    int n;
    cin>>n;
    for(int i=1;i<=n;i++){
       int totManeras = (i*i)*(i*i-1) / 2;
       int totAtaques = 4*(i-1)*(i-2);
       cout<<totManeras-totAtaques<<endl;
    }
    return 0;
}
```

6.0.14 Two_Sets

```cpp
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for (int i = (a); i < (b); i++)
#define FOR(i, n) for (int i = 0; i < (n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;

void solve() {
    ll n;
    cin >> n;
    ll suma = n * (n + 1) / 4;
    if (n * (n + 1) % 4) {
        cout << "NO\n";
    } else {
        cout << "YES\n";
        ll target = suma;
        set<int> uno, dos;
        ll sumUno = 0;
        for (int i = n; i >= 1; i--) {
            if (sumUno + i <= target) {
                uno.insert(i);
                sumUno += i;
            } else {
                dos.insert(i);
            }
        }
        cout << sz(uno) << "\n";
        for (int x : uno) cout << x << " ";
        cout << "\n" << sz(dos) << "\n";
        for (int x : dos) cout << x << " ";
        cout << "\n";
    }
}

signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    int t = 1;
    // cin>>t;
    while (t--) { solve(); }
}
```

6.0.15 Weird_Algorithm

```cpp
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;

```

```cpp
void collatz(int n){
    if(n == 1){
        return;
    }else if(!(n%2)){
        cout<<n/2<<' ';
        collatz(n/2);
    }else{
        cout<<3*n + 1<<' ';
        collatz(3*n+1);
    }
}

void solve(){
    int n;cin>>n;
    cout<<n<<' ';
    collatz(n);
}

signed main(){

    ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
    //int t;cin>>t;while(t--)solve();
    solve();
}
```

7 Mathematics

7.0.1 Bracket_Sequences_I

```cpp
// Your task is to calculate the number of valid bracket sequences of length n. For example, when n=6, there are 5 sequences:
// ()()()
// ()(())
// (())()
// ((()))
// (()())
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1000000 + 20;
ll fact[tam], invFact[tam];
ll expmod(ll a, ll b, ll m) {
    a %= m;
    int res = 1;
    while (b > 0) {
        if (b & 1)
            res = res * a % m;
        a = a * a % m;
        b >>= 1;
    }
    return res;
}
void prec() {
    fact[0] = 1;
    for (int i = 1; i < tam; i++) {
        fact[i] = fact[i - 1] * i % MOD;
    }
    invFact[tam - 1] = expmod(fact[tam - 1], MOD - 2, MOD);
    for (int i = tam - 2; i >= 0; i--) {
        invFact[i] = invFact[i + 1] * (i + 1) % MOD;
    }
}
ll catalan(int n){
    return fact[2*n] * invFact[n+1] %MOD * invFact[n]%MOD;
}
void solve(){
    int n;cin>>n;
    if(n&1){
        cout<<"0\n";
        return;
    }
    cout<<catalan(n/2)<<endl;
}

signed main(){
    prec();
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
    //cin>>t;
    while(t--){
        solve();
    }
}
```

7.0.2 Christmas_Party

```cpp
// There are n children at a Christmas party, and each of them has brought a gift. The idea is that everybody will get a gift brought by someone else.
// In how many ways can the gifts be distributed?
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for (int i = (a); i < (b); i++)
#define FOR(i, n) for (int i = 0; i < (n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
```

```cpp
const int tam = 1000000 + 10;

ll des[tam];

void precdes() {
    des[0] = 1;
    des[1] = 0;
    for (int i = 2; i < tam; ++i) {
        des[i] = (i - 1) * ((des[i - 1] + des[i - 2]) % MOD) % MOD;
    }
}

void solve() {
    int n;
    cin >> n;
    cout << des[n] << endl;
}

signed main() {
    precdes();
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);

    int t = 1;
    // cin >> t;
    while (t--) { solve(); }
}
```

7.0.3 Common_Divisors

```cpp
#include <bits/stdc++.h>
// sin doubles
#pragma GCC optimize("Ofast")
#pragma GCC target("avx2,bmi,bmi2,lzcnt,popcnt")
#pragma GCC optimize("unroll-loops")
// con doubles
using namespace std;
#define int long long
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for (int i = (a); i < (b); i++)
#define FOR(i, n) for (int i = 0; i < (n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1e6 + 100;
int frec[tam];
int arr[tam];
int cnt[tam];

// You are given an array of n positive integers. Your task is to find two
// integers such that their greatest common divisor is as large as
// possible.

void solve() {
    int n;
    cin >> n;
    memset(frec, 0, sizeof frec);
    for (int& i : arr) cin >> i, frec[i]++;
    memset(cnt, 0, sizeof cnt);
    for (int i = 1; i < tam; i++) {
        for (int j = i; j < tam; j += i) { cnt[i] += frec[j]; }
    }
    for (int i = tam - 1; i >= 1; i--) {
        if (cnt[i] >= 2) {
            cout << i << '\n';
            return;
        }
    }
}

signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    int t = 1;
    //  cin>>t;
    while (t--) { solve(); }
```

7.0.4 Distribuiting_Apples

```cpp
// There are n children and m apples that will be distributed to them.
// Your task is to count the number of ways this can be done.
// For example, if n=3 and m=2, there are 6 ways: [0,0,2], [0,1,1],
// [0,2,0], [1,0,1], [1,1,0] and [2,0,0].
#include <bits/stdc++.h>
using namespace std;

#define ll long long
const int MOD = 1000000007;
const int tam = 2 * 1000000 + 10; // Límite para los factoriales

ll fact[tam], invFact[tam];

ll expmod(ll a, ll b, ll m) {
    a %= m;
    ll res = 1;
    while (b > 0) {
        if (b & 1) res = res * a % m;
        a = a * a % m;
        b >>= 1;
    }
    return res;
}

void prec() {
```

```
    fact[0] = 1;
    for (int i = 1; i < tam; i++) {
        fact[i] = fact[i - 1] * i % MOD;
    }
    invFact[tam - 1] = expmod(fact[tam - 1], MOD - 2, MOD); // Inverso del factorial más grande
    for (int i = tam - 2; i >= 0; i--) {
        invFact[i] = invFact[i + 1] * (i + 1) % MOD; // Calculamos los inversos de forma descendente
    }
}

ll nck(int n, int k) {
    if (k > n) return 0;
    return fact[n] * invFact[k] % MOD * invFact[n - k] % MOD;
}

int main() {
    prec(); // Precomputamos los factoriales e inversos
    int n, m;
    cin >> n >> m;
    cout << nck(n + m - 1, m) << endl; // Combinaciones con repetición
    return 0;
}
```

7.0.5 Divisor_Analysis

```
// Given an integer, your task is to find the number, sum and product of its divisors. As an example, let us consider the number 12:
// the number of divisors is 6 (they are 1, 2, 3, 4, 6, 12)
// the sum of divisors is 1+2+3+4+6+12=28
// the product of divisors is 1 \cdot 2 \cdot 3 \cdot 4 \cdot 6 \cdot 12 = 1728
// Since the input number may be large, it is given as a prime factorization.
#include <bits/stdc++.h>
#define int long long
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;
#define bint __int128_t
bint expmod(bint a, bint b, bint m) {
    a %= m;
    bint res = 1;
    while (b > 0) {
        if (b & 1)
            res = res * a % m;
        a = a * a % m;
        b >>= 1;
    }
    return res;
}
bint prodiv(map<bint, bint>& factores) {
    const bint MOD1 = (bint)(MOD - 1);
    const bint MOD2 = MOD1 * 2;

    bint cant_mod2 = 1;
    for (auto [p, k] : factores) {
        cant_mod2 = (cant_mod2 * (k + 1)) % MOD2;
    }

    bint cant_div2_mod1 = (cant_mod2 / 2) % MOD1;

    bint ans = 1;
    for (auto [p, k] : factores) {
        bint nuevo = ( (k % MOD1) * cant_div2_mod1 ) % MOD1;
        ans = ans * expmod(p, nuevo, (bint)MOD) % (bint)MOD;
    }
    if (cant_mod2 % 2 == 1) {
        bint raiz = 1;
        for (auto [p, k] : factores) {
            raiz = raiz * expmod(p, k / 2, (bint)MOD) % (bint)MOD;
        }
        ans = ans * raiz % (bint)MOD;
    }

    return ans;
}

bint invmod(bint num){
    return expmod(num, MOD-2, MOD);
}
bint sumdiv(map<bint, bint>& factores){
    bint ans = 1;
    for(auto[factor, exp]:factores){
        ans*=([&](){
            bint inv = invmod((factor-1+MOD)%MOD);
            bint numer = (expmod(factor, exp+1, MOD) - 1 + MOD)%MOD;
            return numer*inv%MOD
            ;
        }());
        ans%=MOD;
    }
    return ans;
}
bint numdiv(map<bint, bint>& factores){
    bint ans = 1;
    for(auto [a,b]:factores){
        ans*=(b+1);
        ans%=MOD;
    }
    return ans;
}
void solve(){
```

```
    int n;cin>>n;
    map<bint, bint> factores;
    // factores.reserve(n);
    for(int i = 0; i<n; i++){
        int a, b;cin>>a>>b;
        factores[a]+=b;
    }
    bint cant = numdiv(factores);
    cout<<(int)cant<<' '<<(int)sumdiv(factores)<<'
    ↪  '<<(int)prodiv(factores)<<'\n';
}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
 //   cin>>t;
    while(t--){
        solve();
    }
}
```

7.0.6 Exponentiation_II

```
void solve(){
    int a, b, c;
    cin >> a >> b >> c;
    cout << expMod(a,expMod(b, c, MOD-1), MOD) << endl;
}
```

7.0.7 Next_Prime

```
// Given a positive integer n, find the next prime number after it.
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;
bool esPrimo(ll n) {
    if (n < 2) return 0;
    if (n == 2 or n == 3) return 1;
    if (n % 2 == 0 or n % 3 == 0) return 0;
    for (ll i = 5; i * i <= n; i += 6) {
        if (n % i == 0 or n % (i + 2) == 0)
            return 0;
    }
    return 1;
}
void solve(){
    int x;cin>>x;
    int num = x+1;
    while(!esPrimo(num))++num;
    cout<<num<<'\n';
}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
    cin>>t;
    while(t--){
        solve();
    }
}
```

7.0.8 Sum_Of_Divisors

```
// Let \sigma(n) denote the sum of divisors of an integer n. For example,
↪  \sigma(12)=1+2+3+4+6+12=28.
// Your task is to calculate the sum \sum_{i=1}^n \sigma(i) modulo 10^9+7.
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for (int i = (a); i < (b); i++)
#define FOR(i, n) for (int i = 0; i < (n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const ll MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;
ll expmod(ll a, ll b, ll m) {
  a %= m;
  ll res = 1;
  while (b > 0) {
    if (b & 1)
      res = res * a % m;
    a = a * a % m;
    b >>= 1;
  }
  return res;
}
ll invmod(int x, int m) { return expmod(x, MOD - 2, MOD); }
void solve() {
  ll n;
  cin >> n;

  ll ans = 0;
  ll d = 1;
```

```cpp
  while (d <= n) {
    ll t = n / d;
    ll r = n / t;       // último d con mismo valor de floor(n/d)
    ll cnt = r - d + 1; // tamaño del bloque [d..r]
    // suma aritmética de d..r = (d + r) * cnt / 2
    ll sum = ((__int128)(d + r) * cnt / 2) % MOD;
    ans = (ans + sum * (t % MOD)) % MOD;
    d = r + 1; // saltamos al siguiente bloque
  }

  cout << ans << "\n";
}

signed main() {
  ios::sync_with_stdio(0);
  cin.tie(0);
  cout.tie(0);
  int t = 1;
  // cin >> t;
  while (t--) {
    solve();
  }
}
```

8 Range Queries

8.0.1 Distinct_Values_Queries

```cpp
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;
// You are given an array of n integers and q queries of the form: how many distinct values are there in a range [a,b]?
vector<ll> bit;
int len;

void update(int pos, ll delta){
    while (pos < len) {
        bit[pos] += delta;
        pos += pos & -pos;
    }
}

ll query(int pos){
    ll sum = 0;
    while (pos > 0) {
        sum += bit[pos];
        pos -= pos & -pos;
    }
    return sum;
}

void solve(){
    int n, q;cin>>n>>q;
    len = n+1;
    bit.resize(n+1);
    //for(int i = 1; i<=n; i++)update(i, 1);
    vector<int> elementos(n+1);
    map<int, int> ult;
    fore(i, 1, n+1)cin>>elementos[i];
    map<int, vector<pair<int, int>>> querys;
    fore(i, 0, q){
        int a, b;cin>>a>>b;
        querys[b].push_back({a, i});
    }
    vector<int> ans(q);
    for(int i = 1; i<=n; i++){
        if(ult.find(elementos[i]) == ult.end()){
            update(i, 1);
            ult[elementos[i]] = i;
        }else{
            update(ult[elementos[i]], -1);
            ult[elementos[i]] = i;
            update(i, 1);
        }
        for(auto[l, pos]:querys[i]){
            ans[pos] = query(i) - query(l-1);
        }
    }
    fore(i,0, q)cout<<ans[i]<<endl;
}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
    //cin>>t;
    while(t--){
        solve();
    }
}
```

8.0.2 Distinct_Values_Queries_II

```cpp
// Given an array of n integers, your task is to process q queries of the following types:

// update the value at position k to u
// check if every value in range [a, b] is distinct
#include <bits/stdc++.h>
#define ll long long
using namespace std;
```

```cpp
using vi = vector<int>;
const int INF = 1e9;
#define sz(x) (int)(x).size()

struct BIT {
    int n;
    vi bit;
    BIT() { n = 0; }
    BIT(int _n) { init(_n); }
    void init(int _n) { n = _n; bit.assign(n+1, 0); }
    void add(int i, int delta) { // 1-indexed
        for (; i <= n; i += i & -i) bit[i] += delta;
    }
    int sum(int i) {
        int s = 0;
        for (; i > 0; i -= i & -i) s += bit[i];
        return s;
    }
    int rangeSum(int l, int r) { if (r < l) return 0; return sum(r) - sum(l-1); }
};

int n, q;
vector<ll> a; // values
struct consulta{
    int tipo, x;
    ll y;
};
vector<consulta> queries; // store queries

// segment tree nodes
vector<vi> nodeVals;   // sorted unique next-values relevant to this node
vector<BIT> nodeBIT;   // BIT per node

// build node values by merging children's vectors
void build_nodes(int p, int l, int r, const vector<vi>& valsPerPos) {
    if (l == r) {
        nodeVals[p] = valsPerPos[l];
        sort(nodeVals[p].begin(), nodeVals[p].end());
        nodeVals[p].erase(unique(nodeVals[p].begin(), nodeVals[p].end()), nodeVals[p].end());
    } else {
        int m = (l + r) >> 1;
        build_nodes(p<<1, l, m, valsPerPos);
        build_nodes(p<<1|1, m+1, r, valsPerPos);
        vi &L = nodeVals[p<<1], &R = nodeVals[p<<1|1];
        nodeVals[p].resize(sz(L) + sz(R));
        merge(L.begin(), L.end(), R.begin(), R.end(), nodeVals[p].begin());
        nodeVals[p].erase(unique(nodeVals[p].begin(), nodeVals[p].end()), nodeVals[p].end());
    }
    nodeBIT[p].init(sz(nodeVals[p]));
}

// add delta at position posIdx for value val (val must exist in nodeVals[p])
void add_val(int p, int l, int r, int posIdx, int val, int delta) {
    // find index of val in nodeVals[p]
    int idx = int(upper_bound(nodeVals[p].begin(), nodeVals[p].end(), val) - nodeVals[p].begin());
    // we want count of values ⩽ val in queries, but for add we need exact index of val
    // since nodeVals[p] is unique sorted, find exact position via lower_bound
    int idx_exact = int(lower_bound(nodeVals[p].begin(), nodeVals[p].end(), val) - nodeVals[p].begin());
    if (idx_exact < sz(nodeVals[p]) and nodeVals[p][idx_exact] == val) {
        nodeBIT[p].add(idx_exact + 1, delta); // BIT is 1-indexed
    }
    if (l == r) return;
    int m = (l + r) >> 1;
    if (posIdx <= m) add_val(p<<1, l, m, posIdx, val, delta);
    else add_val(p<<1|1, m+1, r, posIdx, val, delta);
}

// query count of values ⩽ bound in positions [i,j]
int query_countLE(int p, int l, int r, int i, int j, int bound) {
    if (r < i or l > j) return 0;
    if (i <= l and r <= j) {
        // count how many values ⩽ bound in this node: upper_bound index
        int pos = int(upper_bound(nodeVals[p].begin(), nodeVals[p].end(), bound) - nodeVals[p].begin());
        return nodeBIT[p].sum(pos); // sum up to pos
    }
    int m = (l + r) >> 1;
    return query_countLE(p<<1, l, m, i, j, bound) + query_countLE(p<<1|1, m+1, r, i, j, bound);
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cin >> n >> q;
    a.assign(n+1, 0);
    for (int i = 1; i <= n; ++i) cin >> a[i];
    queries.reserve(q);
    // read queries
    for (int t = 0; t < q; ++t) {
        int tp; cin >> tp;
        if (tp == 1) {
            int k; ll u; cin >> k >> u;
            queries.push_back({tp, k, u});
        } else {
            int l, r; cin >> l >> r;
            queries.push_back({tp, l, (ll)r});
        }
    }
    const int INF_IDX = n + 1;

    // inicial: pos map
    unordered_map<ll, set<int>> pos;
    pos.reserve((size_t)(n*2));

```

```cpp
    for (int i = 1; i <= n; ++i) pos[a[i]].insert(i);

    // nextPos current state for simulation
    vector<int> nextPos(n+1, INF_IDX);
    for (auto &kv : pos) {
        auto &s = kv.second;
        int prev = -1;
        for (int idx : s) {
            if (prev != -1) nextPos[prev] = idx;
            prev = idx;
        }
        if (prev != -1) nextPos[prev] = INF_IDX;
    }

    // valsPerPos: collect all next-values that will ever appear at position i
    vector<vi> valsPerPos(n+1);
    for (int i = 1; i <= n; ++i) {
        valsPerPos[i].push_back(nextPos[i]); // initial
    }

    // --- SIMULATE ALL QUERIES to collect all next-values per position ---
    // clone structures for simulation
    auto posSim = pos;
    auto aSim = a;
    auto nextSim = nextPos;

    for (auto &qq : queries) {
        int tp = qq.tipo;
        if (tp == 1) {
            int k = qq.x;
            ll u = qq.y;
            if (aSim[k] == u) continue;
            ll old = aSim[k];

            // remove k from old set
            auto &sOld = posSim[old];
            auto it = sOld.find(k);
            int oldPred = -1, oldNext = INF_IDX;
            if (it != sOld.end()) {
                auto itNext = next(it);
                if (itNext != sOld.end()) oldNext = *itNext;
                if (it != sOld.begin()) {
                    auto itPred = prev(it);
                    oldPred = *itPred;
                }
                // after removing k, pred_old's next will become oldNext
                sOld.erase(it);
            }
            if (oldPred != -1) {
                if (nextSim[oldPred] != oldNext) {
                    nextSim[oldPred] = oldNext;
                    valsPerPos[oldPred].push_back(oldNext);
                }
            }
            // k's next will be set according to insertion in new set
            // prepare new set
            auto &sNew = posSim[u];
            sNew.insert(k);
            auto itNew = sNew.find(k);
            int newPred = -1, newNext = INF_IDX;
            if (itNew != sNew.begin()) {
                auto itPred = prev(itNew);
                newPred = *itPred;
            }
            {
                auto itN = next(itNew);
                if (itN != sNew.end()) newNext = *itN;
            }
            if (nextSim[k] != newNext) {
                nextSim[k] = newNext;
                valsPerPos[k].push_back(newNext);
            }
            if (newPred != -1) {
                if (nextSim[newPred] != k) {
                    nextSim[newPred] = k;
                    valsPerPos[newPred].push_back(k);
                }
            }
            aSim[k] = u;
            if (sOld.empty()) posSim.erase(old);
        } // else type 2: nothing to simulate for structure
    }

    // now build segment tree nodes
    int SZ = 4 * (n + 5);
    nodeVals.assign(SZ, vi());
    nodeBIT.assign(SZ, BIT());
    build_nodes(1, 1, n, valsPerPos);

    // now initialize current real structures (pos, nextPos, a)
    // we already had pos, nextPos, a earlier (from initial)
    // insert initial nextPos into BITs
    for (int i = 1; i <= n; ++i) {
        add_val(1, 1, n, i, nextPos[i], +1);
    }

    // process queries online, updating pos, nextPos, a, and the BITs
    for (auto &qq : queries) {
        int tp = qq.tipo;
        if (tp == 1) {
            int k = qq.x;
            ll u = qq.y;
            if (a[k] == u) continue;
            ll old = a[k];

            auto &sOld = pos[old];
            auto it = sOld.find(k);
            int oldPred = -1, oldNext = INF_IDX;
            if (it != sOld.end()) {
                auto itNext = next(it);
                if (itNext != sOld.end()) oldNext = *itNext;
                if (it != sOld.begin()) {
                    auto itPred = prev(it);
                    oldPred = *itPred;
```

```
                }
                // remove k's current next from BITs
                add_val(1, 1, n, k, nextPos[k], -1);
                sOld.erase(it);
            }
            if (oldPred != -1) {
                // pred_old: next changes from nextPos[oldPred] to oldNext
                int before = nextPos[oldPred];
                if (before != oldNext) {
                    add_val(1, 1, n, oldPred, before, -1);
                    nextPos[oldPred] = oldNext;
                    add_val(1, 1, n, oldPred, nextPos[oldPred], +1);
                }
            }
            // insert k into new set
            auto &sNew = pos[u];
            sNew.insert(k);
            auto itNew = sNew.find(k);
            int newPred = -1, newNext = INF_IDX;
            if (itNew != sNew.begin()) {
                auto itPred = prev(itNew);
                newPred = *itPred;
            }
            {
                auto itN = next(itNew);
                if (itN != sNew.end()) newNext = *itN;
            }
            // set next[k] = newNext
            nextPos[k] = newNext;
            add_val(1, 1, n, k, nextPos[k], +1);

            if (newPred != -1) {
                int before = nextPos[newPred];
                if (before != k) {
                    add_val(1, 1, n, newPred, before, -1);
                    nextPos[newPred] = k;
                    add_val(1, 1, n, newPred, nextPos[newPred], +1);
                }
            }

            a[k] = u;
            if (sOld.empty()) pos.erase(old);

        } else {
            int l = qq.x;
            int r = (int)qq.y;
            int countNextLEr = query_countLE(1, 1, n, l, r, r);
            int distinct = (r - l + 1) - countNextLEr;
            cout << (distinct == (r-l+1)?"YES":"NO") << '\n';
        }
    }

    return 0;
}
//------------------------------------------------------------
#include <bits/stdc++.h>
using namespace std;

struct SegTree {
    int n;
    vector<int> st;
    SegTree(int _n = 0) { init(_n); }
    void init(int _n) {
        n = _n;
        st.assign(4*n + 5, INT_MAX);
    }
    void build(int p, int l, int r, const vector<int>& a) {
        if (l == r) { st[p] = a[l]; return; }
        int m = (l + r) >> 1;
        build(p<<1, l, m, a);
        build(p<<1|1, m+1, r, a);
        st[p] = min(st[p<<1], st[p<<1|1]);
    }
    int queryMin(int p, int l, int r, int i, int j) {
        if (r < i || l > j) return INT_MAX;
        if (i <= l && r <= j) return st[p];
        int m = (l + r) >> 1;
        return min(queryMin(p<<1, l, m, i, j), queryMin(p<<1|1, m+1, r,
        ↪  i, j));
    }
    void update(int p, int l, int r, int idx, int val) {
        if (l == r) { st[p] = val; return; }
        int m = (l + r) >> 1;
        if (idx <= m) update(p<<1, l, m, idx, val);
        else update(p<<1|1, m+1, r, idx, val);
        st[p] = min(st[p<<1], st[p<<1|1]);
    }
};

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    int n, q;
    if (!(cin >> n >> q)) return 0;
    vector<long long> a(n+1);
    for (int i = 1; i <= n; ++i) cin >> a[i];

    const int INF = n + 1;
    unordered_map<long long, set<int>> pos;
    pos.reserve(n * 2);

    for (int i = 1; i <= n; ++i) pos[a[i]].insert(i);

    vector<int> nextPos(n+1, INF);
    for (auto &kv : pos) {
        auto &s = kv.second;
        int prev = -1;
        for (int idx : s) {
            if (prev != -1) nextPos[prev] = idx;
            prev = idx;
        }
        if (prev != -1) nextPos[prev] = INF;
    }

    SegTree seg(n);
    seg.build(1, 1, n, nextPos);
```

```
    while (q--) {
        int type;
        cin >> type;
        if (type == 1) {
            int k; long long u;
            cin >> k >> u;
            if (a[k] == u) continue; // nada que hacer
            long long old = a[k];

            // --- quitar k del conjunto antiguo ---
            auto &sOld = pos[old];
            auto it = sOld.find(k);
            int oldPred = -1, oldNext = INF;
            if (it != sOld.end()) {
                auto itNext = next(it);
                if (itNext != sOld.end()) oldNext = *itNext;
                if (it != sOld.begin()) {
                    auto itPred = prev(it);
                    oldPred = *itPred;
                }
                sOld.erase(it);
            }
            // actualizar next de oldPred (si existía)
            if (oldPred != -1) {
                nextPos[oldPred] = (oldNext == INF ? INF : oldNext);
                seg.update(1, 1, n, oldPred, nextPos[oldPred]);
            }
            // temporalmente el next de k será INF (se va a insertar en
            ↪  nuevo conjunto)
            nextPos[k] = INF;
            seg.update(1, 1, n, k, nextPos[k]);

            // --- insertar k en el conjunto nuevo ---
            auto &sNew = pos[u];
            auto resItPair = sNew.insert(k);
            auto itNew = resItPair.first;
            int newPred = -1, newNext = INF;
            if (itNew != sNew.begin()) {
                auto itPred = prev(itNew);
                newPred = *itPred;
            }
            {
                auto itN = next(itNew);
                if (itN != sNew.end()) newNext = *itN;
            }
            if (newPred != -1) {
                // el next del predecesor ahora apunta a k
                nextPos[newPred] = k;
                seg.update(1, 1, n, newPred, nextPos[newPred]);
            }
            // next de k es newNext (o INF)
            nextPos[k] = (newNext == INF ? INF : newNext);
            seg.update(1, 1, n, k, nextPos[k]);

            a[k] = u;

            // (opcional) borrar entrada de pos[old] si quedó vacía
            if (sOld.empty()) pos.erase(old);

        } else if (type == 2) {
            int l, r;
            cin >> l >> r;
            int mn = seg.queryMin(1, 1, n, l, r);
            if (mn > r) cout << "YES\n";
            else cout << "NO\n";
        }
    }
    return 0;
}
```

8.0.3 Forest_Quieries

```
#include "bits/stdc++.h"
using namespace std;
#define int long long
const int tam = 1009;

int BIT[tam][tam];
int A[tam][tam]; // Matriz original
int n, m;

void update(int row, int col, int val)
{
    row++; col++;
    for (int i = row; i <= n; i += (i & -i))
    {
        for (int j = col; j <= m; j += (j & -j))
        {
            BIT[i][j] += val;
        }
    }
}

// Consulta acumulativa desde (0,0) hasta (row, col)
int query(int row, int col)
{
    int res = 0;
    row++; col++;
    for (int i = row; i > 0; i -= (i & -i))
    {
        for (int j = col; j > 0; j -= (j & -j))
        {
            res += BIT[i][j];
        }
    }
    return res;
}

int queryRange(int x1, int y1, int x2, int y2) {
    return query(x2, y2) - query(x1 - 1, y2) - query(x2, y1 - 1) +
    ↪  query(x1 - 1, y1 - 1);
}
```

```
// Actualiza completamente un valor en la posición (i, j)
void setValue(int i, int j, int nuevo) {
    int diff = nuevo - A[i][j]; // Calcula la diferencia
    update(i, j, diff);         // Aplica la diferencia al BIT
    A[i][j] = nuevo;            // Actualiza el valor en la matriz original
}

signed main() {
    memset(BIT, 0, sizeof BIT);
    int q;
    cin >> n >> q;
    m = n;
    for (int i = 0; i < n; i++)
        for (int j = 0; j < m; j++) {
            char val;
            cin >> val;
            A[i][j] = (val == '*');
            update(i, j, val == '*'); // Construcción inicial
        }
    while(q--){
        int a, b, c, d;cin>>a>>b>>c>>d;
        cout<<queryRange(a-1, b-1, c-1, d-1)<<endl;
    }


    return 0;
}
```

8.0.4 Hotel_Queries

```
// There are n hotels on a street. For each hotel you know the number of free rooms. Your task is to assign hotel rooms for groups of tourists. All members of a group want to stay in the same hotel.
// The groups will come to you one after another, and you know for each group the number of rooms it requires. You always assign a group to the first hotel having enough rooms. After this, the number of free rooms in the hotel decreases.
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 200010;
struct Node{
    int x;
    //algo
    Node(int _x = 0/*para poner por default y no necesitar constructor por defecto*/){x = _x;}
    static inline Node merge(const Node&a, const Node& b){
        return Node(max(a.x, b.x));
    }
};

Node t[4*tam];
int arr[tam];

void init(int b, int e, int node){
    if(b == e){
        t[node] = Node(arr[b]);
        return;
    }
    int mid = (b+e)>>1, l = (node << 1) | 1, r = l+1;
    init(b, mid, l);
    init(mid+1, e, r);
    t[node] = Node::merge(t[l], t[r]);
}

void query(int b, int e, int node, int x){
    if(b == e){
        t[node].x-=x;
        printf("%d ", b+1);
        return;
    }
    int mid = (b+e)>>1, l = (node << 1) | 1, r = l+1;
    if(t[l].x >=x){
        query(b, mid, l, x);
    }else{
        query(mid+1, e, r, x);
    }
    t[node] = Node::merge(t[l], t[r]);
}

void update(int b, int e, int node, int pos, Node& val){
    if(b == e){
        //reemplazar, sumar, xor, lo que sea
        t[node] = val;
        return;
    }
    int mid = (b+e)>>1, l = (node << 1) | 1, r = l+1;
    if(pos <= mid){
        update(b, mid, l, pos, val);
    }else{
        update(mid+1, e, r, pos, val);
    }
    t[node] = Node::merge(t[l], t[r]);
}
void solve(){
   int n, q;scanf("%d %d", &n, &q);
   for(int i = 0; i<n; i++)scanf("%d", &arr[i]);
    init(0, n-1, 1);
   while(q--){
    int x;scanf("%d", &x);
    if(t[1].x < x){
        printf("0 ");
```

```
    }else{
        query(0, n-1, 1, x);
    }
   }
   printf("\n");

}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
    //cin>>t;
    while(t--){
        solve();
    }
}


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

const int tam = 200010;
struct Node{
    int x;
    //algo
    Node(int _x = 0/*para poner por default y no necesitar constructor por defecto*/){x = _x;}
    static inline Node merge(const Node&a, const Node& b){
        return Node(max(a.x, b.x));
    }
};

Node t[4*tam];
int arr[tam];

void init(int b, int e, int node){
    if(b == e){
        t[node] = Node(arr[b]);
        return;
    }
    int mid = (b+e)>>1, l = (node << 1) | 1, r = l+1;
    init(b, mid, l);
    init(mid+1, e, r);
    t[node] = Node::merge(t[l], t[r]);
}

Node query(int b, int e, int node, int i, int j){
    if(i<=b and j>=e){
        return t[node];
    }
    int mid = (b+e)>>1, l = (node << 1) | 1, r = l+1;
    if(mid >= j)
        return query(b, mid, l, i, j);
    if(mid < i)
        return query(mid+1, e, r, i, j);
    return Node::merge(query(b, mid, l, i, j), query(mid+1, e, r, i, j));
}

void update(int b, int e, int node, int pos, Node& val){
    if(b == e){
        //reemplazar, sumar, xor, lo que sea
        t[node].x += val.x;
        return;
    }
    int mid = (b+e)>>1, l = (node << 1) | 1, r = l+1;
    if(pos <= mid){
        update(b, mid, l, pos, val);
    }else{
        update(mid+1, e, r, pos, val);
    }
    t[node] = Node::merge(t[l], t[r]);
}

void solve(){
    int n, q;scanf("%d %d", &n, &q);
    for(int i = 0; i<n; i++)scanf(" %d", &arr[i]);
    init(0, n-1, 1);
    while(q--){
        int x;scanf(" %d", &x);
        int l = 0, r = n-1;
        int ans = -1;
        while(l<=r){
            int mid = (l+r)/2;
            if(query(0, n-1, 1, 0, mid).x >=x){
                r = mid-1;
                ans = mid;
            }else{
                l = mid+1;
            }
        }
        if(ans == -1){
            cout<<0<<'\n';
        }else{
            cout<<ans+1<<'\n';
            auto it = Node(-x);
            update(0, n-1, 1, ans, it);
            arr[ans]-=x;
        }
    }
}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    solve();
}
```

8.0.5 List_Removals

```
// You are given a list consisting of n integers. Your task is to remove elements from the list at given positions, and report the removed elements.
#include <bits/stdc++.h>
using namespace std;
```

```cpp
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;

#define lsb(x) ((x) & (-x))
struct BIT {
    // indexado a 1
    vector<int> bit;
    BIT(int N){
        bit.resize(N+1);
    }
    void add(int i, int x) {
        int real = i;
        while (i < sz(bit)) {
            bit[i] += x;
            i += lsb(i);
        }
    }
    int sum(int i) {
        int ans = 0;
        while (i > 0) {
            ans += bit[i];
            i -= lsb(i);
        }
        return ans;
    }
    int sum(int l, int r) {
        if (l > r) return 0;
        return sum(r) - sum(l - 1);
    }
};
void solve(){
   int n;cin>>n;
   vector<int> arr(n);
   BIT bit(n);
   for(int i = 0; i<n; i++){
    int x;cin>>x;
    arr[i] = x;
    bit.add(i+1, 1);
   }
   for(int i = 0; i<n; i++){
    int idx;cin>>idx;
    int l = 1, r = n;
    int pos;
    while(l<=r){
        int mid = (l+r)/2;
        if(bit.sum(1, mid) >= idx){
            pos = mid;
            r = mid-1;
        }else{
            l = mid+1;
        }
    }

    bit.add(pos, -1);
    cout<<arr[pos-1]<<' ';
   }
   cout<<'\n';
}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
 //   cin>>t;
    while(t--){
        solve();
    }
}
```

8.0.6 Missing_Coin_Sum_Queries

```cpp
// You have n coins with positive integer values. The coins are numbered
↪  1,2,\dots,n.
// Your task is to process q queries of the form: "if you can use coins a
↪  \dots b, what is the smallest sum you cannot produce?"
#include <bits/stdc++.h>
using namespace std;
using ll = long long;
#define fore(i, a, b) for(int i = (a); i < (b); ++i)
#define FOR(i, n) for(int i = 0; i < (n); ++i)

int n, q;

// persistent segtree (sum)
struct PST {
    // children and sum arrays
    vector<int> L, R;
    vector<ll> sum;
    vector<int> roots; // roots[0..n]
    PST(int est_nodes = 1){
        L.reserve(est_nodes);
        R.reserve(est_nodes);
        sum.reserve(est_nodes);
        // create node 0 as empty null node
        L.push_back(0); R.push_back(0); sum.push_back(0);
    }
    int new_node(int l=0,int r=0,ll s=0){
        L.push_back(l); R.push_back(r); sum.push_back(s);
        return (int)sum.size() - 1;
    }
    // update position pos by adding val, based on previous node 'cur'
    int update(int cur, int tl, int tr, int pos, ll val){
        int node = new_node(); // clone new
        L[node] = L[cur]; R[node] = R[cur]; sum[node] = sum[cur] + val;
        if(tl == tr) return node;
```

```
        int tm = (tl + tr) >> 1;
        if(pos <= tm){
            int left_child = update(L[cur], tl, tm, pos, val);
            L[node] = left_child;
        } else {
            int right_child = update(R[cur], tm+1, tr, pos, val);
            R[node] = right_child;
        }
        return node;
    }
    // query sum on [lq, rq] on node
    ll query(int node, int tl, int tr, int lq, int rq){
        if(!node || rq < tl || tr < lq) return 0LL;
        if(lq <= tl && tr <= rq) return sum[node];
        int tm = (tl + tr) >> 1;
        return query(L[node], tl, tm, lq, rq) + query(R[node], tm+1, tr,
        ↪  lq, rq);
    }
};

int main(){
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cin >> n >> q;
    vector<int> a(n+1);
    fore(i,1,n+1) cin >> a[i];

    // pairs (value, pos)
    vector<pair<int,int>> vp;
    vp.reserve(n);
    fore(i,1,n+1) vp.emplace_back(a[i], i);
    sort(vp.begin(), vp.end()); // ascending by value

    // build PST versions: root[0] = empty, root[k] = with first k
    ↪  elements of vp added
    // estimate nodes ~ n * (log2 n + 5)
    int est_nodes = n * 22;
    PST pst(est_nodes);
    pst.roots.resize(n+1);
    pst.roots[0] = 0; // empty tree
    for(int k = 1; k <= n; ++k){
        int val = vp[k-1].first;
        int pos = vp[k-1].second;
        pst.roots[k] = pst.update(pst.roots[k-1], 1, n, pos, val);
    }

    // vector of only values for upper_bound
    vector<int> vals(n);
    for(int i=0;i<n;i++) vals[i] = vp[i].first;

    while(q--){
        int L, R; cin >> L >> R;
        ll res = 1;
        while(true){
            // count k = #values ≤ res
            int k = upper_bound(vals.begin(), vals.end(),
            ↪  (int)min<ll>(res, (ll)INT_MAX)) - vals.begin();
            ll s = 0;
            if(k > 0) s = pst.query(pst.roots[k], 1, n, L, R);
            if(s < res){
                cout << res << '\n';
                break;
            }
            if(s + 1 == res){ // safety guard (shouldn't happen)
                cout << res << '\n';
                break;
            }
            res = s + 1;
        }
    }
    return 0;
}
```

8.0.7 Pizzeria_Queries

```
// There are n buildings on a street, numbered 1,2,\dots,n. Each building
↪  has a pizzeria and an apartment.
// The pizza price in building k is p_k. If you order a pizza from
↪  building a to building b, its price (with delivery) is p_a+|a-b|.
// Your task is to process two types of queries:
// The pizza price p_k in building k becomes x.
// You are in building k and want to order a pizza. What is the minimum
↪  price?
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 4e18;
const int tam = 200005;
struct Item{
    //algun atributo
    int x;
    Item(int _x = 2e9){
        x = _x;
    }
    static Item merge(const Item& a, const Item& b){
        return Item(min(a.x, b.x));
    }
};

struct Nodo {
    Item value;
    Nodo *izq = nullptr, *der = nullptr;
};

void update(Nodo*& node, ll inicio, ll fin, ll pos, Item& val) {
```

```
    if (!node) node = new Nodo();
    if (inicio == fin) {//la actualizacion, puede ser suma o asignacion o lo que sea, piensa bien
        node->value = val;
        return;
    }
    ll mid = (inicio + fin) / 2;
    if (pos <= mid)
        update(node->izq, inicio, mid, pos, val);
    else
        update(node->der, mid + 1, fin, pos, val);
    //esto esta interesante, que es el elemento neutro?
    auto it = Item();
    Item& izqVal = node->izq ? node->izq->value : it;
    Item& derVal = node->der ? node->der->value : it;
    node->value = Item::merge(izqVal, derVal);
}

Item query(Nodo* node, ll inicio, ll fin, ll l, ll r) {
    if (!node || r < inicio || l > fin) return Item();
    if (l <= inicio && fin <= r) return node->value;
    ll mid = (inicio + fin) / 2;
    return Item::merge(query(node->izq, inicio, mid, l, r),
           query(node->der, mid + 1, fin, l, r));
}
int arr[tam];
Nodo* izq = new Nodo();
Nodo* der = new Nodo();
void solve(){

   int n, q;scanf("%d %d", &n, &q);
   for(int i = 1; i<=n; i++){
    scanf(" %d", &arr[i]);
    auto itizq = Item(arr[i] - i);
    auto itder =Item(arr[i]+i);
    update(izq, 1, n, i, itizq);
    update(der, 1, n, i, itder);
   }
   while(q--){
    int t;
    scanf(" %d", &t);
    if(t == 1){
        int pos, x;
        scanf(" %d %d", &pos, &x);
        arr[pos] = x;
        auto itizq = Item(arr[pos] - pos);
        auto itder = Item(arr[pos] + pos);
        update(izq, 1, n, pos, itizq);
        update(der, 1, n, pos, itder);

    }else{
        int pos;
        scanf(" %d", &pos);
        auto resIzq = query(izq, 1, n, 1, pos);
        auto resDer = query(der, 1, n, pos, n);
        int ans = min(resIzq.x + pos, resDer.x - pos);
        printf("%d\n", ans);
    }
   }

}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
 //   cin>>t;
    while(t--){
        solve();
    }
}
```

8.0.8 Polynomial_Quieries

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const int tam = 1;

//Segment tree de prograsiones aritmeticas
// Your task is to maintain an array of n values and efficiently process the following types of queries:

// Increase the first value in range [a,b] by 1, the second value by 2, the third value by 3, and so on.
// Calculate the sum of values in range [a,b].



const int MAXN = 200003;
const ll INF = 4e18;

struct Node {
    ll sum;        // Suma del segmento
    ll first_term; // Primer término de la PA (si aplica)
    ll diff;       // Diferencia común (si aplica)
    int size;      // Tamaño del segmento

    Node(ll s = 0, ll f = 0, ll d = 0, int sz = 0)
        : sum(s), first_term(f), diff(d), size(sz) {}

    static Node merge(const Node& a, const Node& b) {
        return Node(a.sum + b.sum, 0, 0, a.size + b.size);
    }
};

```

```cpp
struct Lazy {
    ll a;  // Primer término de la PA
    ll d;  // Diferencia común

    Lazy(ll a_val = 0, ll d_val = 0) : a(a_val), d(d_val) {}

    bool has_update() const { return a != 0 or d != 0; }
};

Node t[4 * MAXN];
Lazy lazy[4 * MAXN];
ll arr[MAXN];

// Calcula la suma de una PA: n/2 * (2a + (n-1)d)
ll arithmetic_sum(ll a, ll d, int n) {
    return n * (2 * a + (n - 1) * d) / 2;
}

void build(int node, int l, int r) {
    lazy[node] = Lazy();
    if (l == r) {
        t[node] = Node(arr[l], 0, 0, 1);
        return;
    }
    int mid = (l + r) / 2;
    build(2 * node, l, mid);
    build(2 * node + 1, mid + 1, r);
    t[node] = Node::merge(t[2 * node], t[2 * node + 1]);
}

void apply_lazy(Node& node, int l, int r, const Lazy& lz, int seg_start) {
    if (lz.has_update()) {
        // Calcular el primer término para este segmento específico
        // Si el segmento comienza en 'start', el primer término es: a + (start - original_start) * d
        // Pero necesitamos saber el inicio original del update
        ll first_term_for_segment = lz.a + (l - seg_start) * lz.d;

        // Sumar la PA completa del segmento
        node.sum += arithmetic_sum(first_term_for_segment, lz.d, node.size);
    }
}

// Versión alternativa más práctica (sin necesidad de seg_start):
void apply_lazy_simple(Node& node, int l, int r, const Lazy& lz) {
    if (lz.has_update()) {
        // Para un segmento [l, r], la suma de la PA que comienza en l con diferencia d es:
        // Sum = (r-l+1)/2 * (2*a + (r-l)*d)
        // Pero necesitamos ajustar el primer término para este segmento
        node.sum += arithmetic_sum(lz.a, lz.d, node.size);
    }
}

void push(int node, int l, int r) {
    if (lazy[node].has_update()) {
        apply_lazy_simple(t[node], l, r, lazy[node]);

        if (l != r) {
            int mid = (l + r) / 2;

            // Hijo izquierdo: misma PA, mismo primer término
            lazy[2 * node].a += lazy[node].a;
            lazy[2 * node].d += lazy[node].d;

            // Hijo derecho: la PA continúa, primer término ajustado
            ll first_term_right = lazy[node].a + (mid + 1 - l) * lazy[node].d;
            lazy[2 * node + 1].a += first_term_right;
            lazy[2 * node + 1].d += lazy[node].d;
        }
        lazy[node] = Lazy();
    }
}

void update_arithmetic(int node, int l, int r, int ql, int qr, ll a, ll d) {
    push(node, l, r);
    if (l > qr or r < ql) return;

    if (ql <= l and r <= qr) {
        // Calcular el primer término para este segmento
        ll first_term_here = a + (l - ql) * d;
        lazy[node] = Lazy(first_term_here, d);
        push(node, l, r);
        return;
    }

    int mid = (l + r) / 2;
    update_arithmetic(2 * node, l, mid, ql, qr, a, d);
    update_arithmetic(2 * node + 1, mid + 1, r, ql, qr, a, d);
    t[node] = Node::merge(t[2 * node], t[2 * node + 1]);
}

void update_ap(int node, int l, int r, int ql, int qr, ll a, ll d) {
    push(node, l, r);
    if (l > qr or r < ql) return;

    if (ql <= l and r <= qr) {
        // Para el segmento [l, r], el primer término es a + (l - ql) * d
        ll segment_first = a + (l - ql) * d;
        lazy[node] = Lazy(segment_first, d);
        push(node, l, r);
        return;
    }

    int mid = (l + r) / 2;
    update_ap(2 * node, l, mid, ql, qr, a, d);
    update_ap(2 * node + 1, mid + 1, r, ql, qr, a, d);
    t[node] = Node::merge(t[2 * node], t[2 * node + 1]);
}

ll query_sum(int node, int l, int r, int ql, int qr) {
    push(node, l, r);
```

```
    if (l > qr or r < ql) return 0;
    if (ql <= l and r <= qr) return t[node].sum;

    int mid = (l + r) / 2;
    return query_sum(2 * node, l, mid, ql, qr) +
           query_sum(2 * node + 1, mid + 1, r, ql, qr);
}


void solve(){
   int n, q;cin>>n>>q;
   for(int i = 0; i<n; i++)cin>>arr[i];
   build(1, 0, n-1);
   while(q--){
    int t, a, b;cin>>t>>a>>b;
    if(t == 1){
        a--;b--;
        update_arithmetic(1, 0, n-1, a, b, 1, 1);
    }else{
        a--, b--;
        auto res = query_sum(1, 0, n-1, a, b);
        cout<<res<<'\n';
    }
   }
}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
   // cin>>t;
    while(t--){
        solve();
    }
}
```

8.0.9 Prefix_Sum_Queries

```
// Given an array of n integers, your task is to process q queries of the following types:
// update the value at position k to u
// what is the maximum prefix sum in range [a,b]?
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 200005;
ll arr[tam];

struct Item{
    //algun atributo
    ll sum;
    ll maxPref;
    Item(ll s = 0, ll maxP = -(1e18)){
        sum = s;
        maxPref = maxP;
    }
    static Item merge(const Item& a, const Item& b){
        return Item(a.sum + b.sum, max(a.maxPref, a.sum + b.maxPref));
    }
};

struct Nodo {
    Item value;
    Nodo *izq = nullptr, *der = nullptr;
};

void update(Nodo*& node, ll inicio, ll fin, ll pos, Item& val) {
    if (!node) node = new Nodo();
    if (inicio == fin) {//la actualizacion, puede ser suma o asignacion o lo que sea, piensa bien
        node->value = val;
        return;
    }
    ll mid = (inicio + fin) / 2;
    if (pos <= mid)
        update(node->izq, inicio, mid, pos, val);
    else
        update(node->der, mid + 1, fin, pos, val);
    //esto esta interesante, que es el elemento neutro?
    auto it = Item();
    Item& izqVal = node->izq ? node->izq->value : it;
    Item& derVal = node->der ? node->der->value : it;
    node->value = Item::merge(izqVal, derVal);
}

Item query(Nodo* node, ll inicio, ll fin, ll l, ll r) {
    if (!node || r < inicio || l > fin) return Item();
    if (l <= inicio && fin <= r) return node->value;
    ll mid = (inicio + fin) / 2;
    return Item::merge(query(node->izq, inicio, mid, l, r),
           query(node->der, mid + 1, fin, l, r));
}

void solve(){
    Nodo* seg = new Nodo();
   int n, q;cin>>n>>q;
    for(int i = 0; i<n; i++){
        cin>>arr[i];
        auto it = Item(arr[i], arr[i]);
        update(seg, 0, n-1, i, it);
    }
    while(q--){
        int t;cin>>t;
        if(t == 1){
            int pos;
            ll k;
```

```
            cin>>pos>>k;
            --pos;
            auto it = Item(k, k);
            update(seg, 0, n-1, pos, it);
        }else{
            int l, r;cin>>l>>r;
            --l, --r;
            Item res = query(seg, 0, n-1, l, r);
            cout<<max(0LL, res.maxPref)<<'\n';
        }
    }

}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
   // cin>>t;
    while(t--){
        solve();
    }
}
```

8.0.10 Range_Interval_Queries

```
// Given an array x of n integers, your task is to process q queries of the form: how many integers i satisfy a \le i \le b and c \le x_i \le d?
#include <bits/stdc++.h>
using namespace std;
using ll = long long;

struct Fenwick {
    int n;
    vector<int> bit;
    Fenwick(int n=0): n(n), bit(n+1,0) {}
    void add(int idx, int val){
        for(; idx <= n; idx += idx & -idx) bit[idx] += val;
    }
    int sum(int idx){
        int r=0;
        for(; idx>0; idx -= idx & -idx) r += bit[idx];
        return r;
    }
    int rangeSum(int l, int r){
        if(r < l) return 0;
        return sum(r) - sum(l-1);
    }
};

int main(){
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    int n, q;
    if(!(cin >> n >> q)) return 0;
    vector<pair<int,int>> vals(n); // (value, pos)
    for(int i=0;i<n;i++){
        int x; cin >> x;
        vals[i] = {x, i+1}; // 1-based index for BIT
    }
    sort(vals.begin(), vals.end(), [](auto &a, auto &b){
        if(a.first != b.first) return a.first < b.first;
        return a.second < b.second;
    });

    struct Event { int threshold; int l, r; int qi; int sign; };
    vector<Event> events;
    events.reserve(2*q);
    for(int i=0;i<q;i++){
        int a,b,c,d; cin >> a >> b >> c >> d;
        // count ≤ d  (sign +1)
        events.push_back({d, a, b, i, +1});
        // count ≤ c-1 (sign -1)
        events.push_back({c-1, a, b, i, -1});
    }
    sort(events.begin(), events.end(), [](const Event &A, const Event &B){
        return A.threshold < B.threshold;
    });

    vector<ll> ans(q, 0);
    Fenwick fw(n);
    int ptr = 0; // pointer in vals

    for(auto &ev : events){
        while(ptr < n && vals[ptr].first <= ev.threshold){
            fw.add(vals[ptr].second, 1); // activate position
            ptr++;
        }
        int cnt = fw.rangeSum(ev.l, ev.r);
        ans[ev.qi] += 1LL * ev.sign * cnt;
    }

    for(int i=0;i<q;i++) cout << ans[i] << '\n';
    return 0;
}
```

8.0.11 Range_Queries_and_Copies

```
// Your task is to maintain a list of arrays which initially has a single array. You have to process the following types of queries:
// Set the value a in array k to x.
// Calculate the sum of values in range [a,b] in array k.
// Create a copy of array k and add it to the end of the list.
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
```

```cpp
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 200005;

struct Node {
    int val;
    Node *l, *r;

    // Constructor hoja
    Node(int _val) {
        val = _val;
        l = r = nullptr;
    }

    // Constructor combinando hijos
    Node(Node* L, Node* R) {
        l = L; r = R;
        val = L->val + R->val;
    }
};

// Construcción inicial
int arr[tam];
Node* build(int tl, int tr) {
    if (tl == tr)
        return new Node(arr[tl]);
    int tm = (tl + tr) / 2;
    return new Node(build(tl, tm), build(tm+1, tr));
}

Node* update(Node* v, int tl, int tr, int pos, long long new_val) {
    if (tl == tr) return new Node(new_val);
    int tm = (tl + tr) / 2;
    if (pos <= tm) {
        return new Node(update(v->l, tl, tm, pos, new_val), v->r);
    } else {
        return new Node(v->l, update(v->r, tm+1, tr, pos, new_val));
    }
}

Node* merge(Node* a, Node* b) {
    if (!a) return b;
    if (!b) return a;
    return new Node(a, b);
}

// Query en rango [l,r]
Node* query(Node* v, int tl, int tr, int l, int r) {
    if (l > r) return nullptr;
    if (l == tl && r == tr) return v;
    int tm = (tl + tr) / 2;
    return merge(
        query(v->l, tl, tm, l, min(r, tm)),
        query(v->r, tm+1, tr, max(l, tm+1), r)
    );
}

void solve(){
    int n, q;cin>>n>>q;
    for(int i = 0; i<n; i++)cin>>arr[i];
    vector<Node*> segmentos;
    segmentos.push_back(build(0, n-1));
    while(q--){
        int t;cin>>t;
        if(t == 1){
            int k, a, x;cin>>k>>a>>x;
            k--, a--;
            auto we = update(segmentos[k], 0, n-1, a, x);
            segmentos[k] = we;
        }else if(t == 2){
            int k, a, b;cin>>k>>a>>b;
            k--, a--, b--;
            cout<<query(segmentos[k], 0, n-1, a, b)->val<<'\n';
        }else{
            int k;cin>>k;
            k--;
            segmentos.push_back(segmentos[k]);
        }
    }
}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
   // cin>>t;
    while(t--){
        solve();
    }
}
```

8.0.12 Range_Update_Queries

```cpp
// Given an array of n integers, your task is to process q queries of the following types:
// increase each value in range [a,b] by u
// what is the value at position k?
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
int tam;
```

```cpp
vector<ll> bit;
vector<ll> arr;

void update(int pos, ll delta){
    while (pos < tam) {
        bit[pos] += delta;
        pos += pos & -pos;
    }
}

ll query(int pos){
    ll sum = 0;
    while (pos > 0) {
        sum += bit[pos];
        pos -= pos & -pos;
    }
    return sum;
}

void range_add(int l, int r, ll val) {
    update(l, val);
    update(r + 1, -val);
}

ll point_query(int i) {
    return query(i);
}

void solve(){
    int n, q; cin >> n >> q;
    tam = n+2;
    arr.resize(n + 1);
    bit.resize(n + 2, 0);
    fore(i, 1, n + 1){
        cin >> arr[i];
        range_add(i, i, arr[i]);
    }
    while (q--){
        int type; cin >> type;
        if (type == 1){
            int l, r, val; cin >> l >> r >> val;
            range_add(l, r, val);
        } else {
            int index; cin >> index;
            cout << point_query(index) << '\n';
        }
    }
}
signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
    //cin>>t;
    while(t--){
        solve();
    }
}
```

8.0.13 Salary_Querys

```cpp
// A company has n employees with certain salaries. Your task is to keep track of the salaries and process queries.
// Input
// The first input line contains two integers n and q: the number of employees and queries. The employees are numbered 1,2,\ldots,n.
// The next line has n integers p_1,p_2,\ldots,p_n: each employee's salary.
// After this, there are q lines describing the queries. Each line has one of the following forms:
// ! k x: change the salary of employee k to x
// ? a b: count the number of employees whose salary is between a \ldots b
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for (int i = (a); i < (b); i++)
#define FOR(i, n) for (int i = 0; i < (n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;

#define lsb(x) ((x) & (-x))
struct BIT {
    vector<int> bit;
    BIT(int N) : bit(N + 1) {}
    void add(int i, int x) {
        while (i < sz(bit)) {
            bit[i] += x;
            i += lsb(i);
        }
    }
    int sum(int i) {
        int ans = 0;
        while (i > 0) {
            ans += bit[i];
            i -= lsb(i);
        }
        return ans;
    }
    int sum(int l, int r) {
        if (l > r) return 0;
        return sum(r) - sum(l - 1);
    }
};

void solve() {
    int n, q;cin>>n>>q;
    vector<int> arr(n);
    int we = 0;
    vector<int> v;
    v.reserve(n+q + 1000);
```

```
    for (int i = 0; i < n; i++, we++)
    {
        cin>>arr[i];
        v.push_back(arr[i]);

    }
    vector<tuple<char, int, int>> queries(q);
    for(int i = 0; i<q; i++){
        char t;
        int l, r;
        cin>>t>>l>>r;
        queries[i] = {t, l , r};
        v.push_back(l);
        v.push_back(l+1);
        v.push_back(l-1);
        v.push_back(r);
        v.push_back(r-1);
        v.push_back(r+1);
    }
    sort(all(v));
    v.erase(unique(all(v)), v.end());
    unordered_map<int, int> mp;
    mp.reserve((n+q)*2);
    for(int i = 1; i<=sz(v); i++){
        mp[v[i-1]] = i;
    }
    BIT bit(sz(v));
    for(int i = 1; i<=n; i++){
        bit.add(mp[arr[i-1]], 1);
    }
    for(auto[tipo, l, r]:queries){
        if(tipo == '?'){
            cout<<bit.sum(mp[l], mp[r])<<'\n';
        }else{
            bit.add(mp[arr[l-1]], -1);
            arr[l-1] = r;
            bit.add(mp[r], 1);
        }
    }
}

signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    int t = 1;
    //   cin>>t;
    while (t--) { solve(); }
}
```

8.0.14 Subarray_Sum_Queries

```
// There is an array consisting of n integers. Some values of the array will be updated, and after each update, your task is to report the maximum subarray sum in the array.
// Input
// The first input line contains integers n and m: the size of the array and the number of updates. The array is indexed 1,2,\ldots,n.
// The next line has n integers: x_1,x_2,\ldots,x_n: the initial contents of the array.
// Then there are m lines describing the changes. Each line has two integers k and x: the value at position k becomes x.
#include <bits/stdc++.h>
using namespace std;
#define int long long

struct Node {
    int total_sum, prefix_sum, suffix_sum, max_sum;

    Node() : total_sum(0), prefix_sum(0), suffix_sum(0), max_sum(0) {}
    Node(int val) : total_sum(val), prefix_sum(val), suffix_sum(val), max_sum(val) {}
};

// Función para combinar dos nodos
Node combine(const Node &left, const Node &right) {
    Node res;
    res.total_sum = left.total_sum + right.total_sum;
    res.prefix_sum = max(left.prefix_sum, left.total_sum + right.prefix_sum);
    res.suffix_sum = max(right.suffix_sum, right.total_sum + left.suffix_sum);
    res.max_sum = max({left.max_sum, right.max_sum, left.suffix_sum + right.prefix_sum});
    return res;
}

class SegmentTree {
private:
    vector<Node> tree;
    int n;

    void build(const vector<int> &arr, int node, int start, int end) {
        if (start == end) {
            tree[node] = Node(arr[start]);
        } else {
            int mid = (start + end) / 2;
            build(arr, 2 * node + 1, start, mid);
            build(arr, 2 * node + 2, mid + 1, end);
            tree[node] = combine(tree[2 * node + 1], tree[2 * node + 2]);
        }
    }

    Node query(int node, int start, int end, int L, int R) {
        if (R < start || L > end) {
            return Node(INT_MIN); // Nodo neutro para max_sum
        }
        if (L <= start && end <= R) {
            return tree[node];
        }
        int mid = (start + end) / 2;
        Node left = query(2 * node + 1, start, mid, L, R);
```

```
        Node right = query(2 * node + 2, mid + 1, end, L, R);
        return combine(left, right);
    }

    void update(int node, int start, int end, int pos, int new_val) {
        if (start == end) {
            tree[node] = Node(new_val);
        } else {
            int mid = (start + end) / 2;
            if (pos <= mid) {
                update(2 * node + 1, start, mid, pos, new_val);
            } else {
                update(2 * node + 2, mid + 1, end, pos, new_val);
            }
            tree[node] = combine(tree[2 * node + 1], tree[2 * node + 2]);
        }
    }

public:
    SegmentTree(const vector<int> &arr) {
        n = arr.size();
        tree.resize(4 * n);
        build(arr, 0, 0, n - 1);
    }

    int max_sum_query(int L, int R) {
        Node res = query(0, 0, n - 1, L, R);
        return res.max_sum;
    }

    void update(int pos, int new_val) {
        update(0, 0, n - 1, pos, new_val);
    }
};

signed main() {
    int n, q;
    cin >> n >> q;
    vector<int> arr(n);
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }

    SegmentTree segTree(arr);

    while (q--) {
        int pos, x;cin>>pos>>x;pos--;
    segTree.update(pos, x);
    int ans = segTree.max_sum_query(0, n-1);
    ans = max(0LL, ans);
    cout<<ans<<endl;
    }
    return 0;
}
```

8.0.15 Subarray_Sum_Queries_II

```
// You are given an array of n integers and q queries. In each query, your task is to calculate the maximum subarray sum in the range [a,b].
// Empty subarrays (with sum 0) are allowed.
#include <bits/stdc++.h>
#define int long long
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;

struct Node {
    int total_sum, max_sum, min_sum, prefix_sum, suffix_sum;
    Node() : total_sum(0), max_sum(INT_MIN), min_sum(INT_MAX), prefix_sum(0), suffix_sum(0) {}
};

class SegmentTree {
private:
    vector<Node> tree;
    int n;

    Node combine(const Node &left, const Node &right) {
        Node res;
        res.total_sum = left.total_sum + right.total_sum;
        res.prefix_sum = max(left.prefix_sum, left.total_sum + right.prefix_sum);
        res.suffix_sum = max(right.suffix_sum, right.total_sum + left.suffix_sum);
        res.max_sum = max({left.max_sum, right.max_sum, left.suffix_sum + right.prefix_sum});

        res.min_sum = min({left.min_sum, right.min_sum, left.suffix_sum + right.prefix_sum});
        return res;
    }

    void build(const vector<int> &arr, int v, int tl, int tr) {
        if (tl == tr) {
            tree[v].total_sum = arr[tl];
            tree[v].max_sum = arr[tl];
            tree[v].min_sum = arr[tl];
            tree[v].prefix_sum = arr[tl];
            tree[v].suffix_sum = arr[tl];
        } else {
            int tm = (tl + tr) / 2;
            build(arr, v * 2, tl, tm);
```

```
            build(arr, v * 2 + 1, tm + 1, tr);
            tree[v] = combine(tree[v * 2], tree[v * 2 + 1]);
        }
    }

    Node query(int v, int tl, int tr, int l, int r) {
        if (l > r) return Node();
        if (l == tl && r == tr) return tree[v];
        int tm = (tl + tr) / 2;
        Node left_result = query(v * 2, tl, tm, l, min(r, tm));
        Node right_result = query(v * 2 + 1, tm + 1, tr, max(l, tm + 1),
        ↪  r);
        return combine(left_result, right_result);
    }

public:
    SegmentTree(const vector<int> &arr) {
        n = arr.size();
        tree.resize(n * 4);
        build(arr, 1, 0, n - 1);
    }

    int getMaxSum(int l, int r) {
        return query(1, 0, n - 1, l, r).max_sum;
    }

    int getMinSum(int l, int r) {
        return query(1, 0, n - 1, l, r).min_sum;
    }
};


void solve(){
    int n, q;cin>>n>>q;
    vector<int> arr(n);
    for(int i = 0; i<n; i++)cin>>arr[i];

    SegmentTree st(arr);
    /*cuando falta poco tiempo, es mejor que los 3 nos enfoquemos en un
    ↪  problema, ya no lo vale*/
    while(q--){
        int l, r;cin>>l>>r;
        l--, r--;
        cout<<st.getMaxSum(l, r)<<'\n';
    }
}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
   // cin>>t;
    while(t--){
        solve();
    }
}
```

9 Sliding Window

9.0.1 Sliding_Window_Median

```
#include <bits/stdc++.h>
using namespace std;
// You are given an array of n integers. Your task is to calculate the
↪  median of each window of k elements, from left to right.
// The median is the middle element when the elements are sorted. If the
↪  number of elements is even, there are two possible medians and we
↪  assume that the median is the smaller of them.
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
typedef pair<int, int> pii;
typedef tree<pii, null_type, less<pair<int, int>>, rb_tree_tag,
             tree_order_statistics_node_update> ordered_ms;


struct ordMultiset {
    int count;
    ordered_ms oset;
    ordMultiset() { count = 0; }

    int count_occurrences(int x) {
        return oset.order_of_key({x + 1, 0}) - oset.order_of_key({x, 0});
    }

    void erase_one(int x) {
        auto it = oset.lower_bound({x, 0});
        if (it != oset.end() && it->first == x) { oset.erase(it); }
    }

    int kth_element(int k) { return oset.find_by_order(k)->first; }

    void insert(int x) {
        oset.insert({x, count++});
    }
};

void solve() {
    int n, k;cin>>n>>k;
    vector<int> arr(n);
    for(int i = 0; i<n; i++)cin>>arr[i];
    ordMultiset oms;
    int r;
    for(r = 0; r<k-1; r++){
        oms.insert(arr[r]);
    }
    int l = 0;
    for(; r<n; r++){
        oms.insert(arr[r]);
        cout<<oms.kth_element((r-l)/2)<<' ';;
        oms.erase_one(arr[l]);
        l++;
    }
    cout<<endl;
```

```
}

signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    int t = 1;
    //cin >> t;
    while (t--) { solve(); }
}
```

10 Sorting and Searching

10.0.1 Array_Division

```
#include<bits/stdc++.h>
using namespace std;
#define int long long
bool f(vector<int>& arr, int maxSum, int k) {
    int acu = 0;
    int sub = 1;
    for (int i = 0; i<(int)arr.size(); i++) {
        if (acu + arr[i] > maxSum) {
            sub++;
            acu = arr[i];
            if (sub > k) {
                return false;
            }
        } else {
            acu += arr[i];
        }
    }
    return sub <= k;
}
// You are given an array containing n positive integers.
// Your task is to divide the array into k subarrays so that the maximum sum in a subarray is as small as possible.
signed main() {
    int n, k;
    cin >> n >> k;
    vector<int> arr(n);
    for (int i = 0; i < n; i++) cin >> arr[i];

    int l = *max_element(arr.begin(), arr.end());  // Ajuste: el valor mínimo debe ser el mayor elemento
    int r = accumulate(arr.begin(), arr.end(), 0LL); // Ajuste: el valor máximo puede ser la suma de todos los elementos

    int res = -1;
    while (l <= r) {
        int mid = (l + r) / 2;
        if (f(arr, mid, k)) {
            res = mid;
            r = mid - 1;
        } else {
            l = mid + 1;
        }
    }
    cout << res << endl;
}
```

10.0.2 Collecting_Numbers

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define f first
#define sst stringstream
#define s second
#define pb push_back
#define sz(x) (int)(x).size()
#define all(a) (a).begin(), (a).end()
#define rall(a) (a).rbegin(), (a).rend()
#define fore(i, a, n) for(int i = (a); i < (n); i++)
#define forb(i, n) for(int i = (n) - 1; i >= 0; i--)
#define popcount(x) __builtin_popcountll(x);
typedef pair<int, int> pii;
typedef vector<int> vi;
const int MOD = 1000000007;
const double EPS = 1e-9;
const double PI = acos(-1);
const int INF = 1e18;
// PLUS ULTRA RECARGADO!!!
// You are given an array that contains each number between 1 \dots n exactly once. Your task is to collect the numbers from 1 to n in increasing order.
// On each round, you go through the array from left to right and collect as many numbers as possible. What will be the total number of rounds?

void solve() {
    int n;
    cin >> n;
    vector<pair<int, int>> aux(n);
    for(int i = 0; i < n; i++) {
        int x;cin >> x;
        aux[i] = {x, i+1};
    }
    sort(all(aux));
    int ans = 1;
    int prev = 0;
    for(int i = 0; i<n; i++){
        int act = aux[i].second;
        if(!(act>prev)){
            ans++;
        }
        prev = act;
    }
    cout<<ans<<endl;
}
signed main() {
    ios::sync_with_stdio(0);
```

```
    cin.tie(0);
    cout.tie(0);
    //   int t;cin>>t;while(t--)solve();
    solve();
}
```

10.0.3 Collecting_Numbers_II

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;
// You are given an array that contains each number between 1 \dots n exactly once. Your task is to collect the numbers from 1 to n in increasing order.
// On each round, you go through the array from left to right and collect as many numbers as possible.
// Given m operations that swap two numbers in the array, your task is to report the number of rounds after each operation.
void solve(){
    int n, q;cin>>n>>q;
    vector<int> arr(n+1), pos(n+1);
    fore(i, 1, n+1){
        cin>>arr[i];
        pos[arr[i]] = i;
    }
    int inv = 1;
    for(int i = 2; i<=n; i++){
        if(pos[i] < pos[i-1])inv++;
    }

    while(q--){
        set<pair<int, int>> mod;
        int x, y;cin>>x>>y;
        int xelem = arr[x];
        int yelem = arr[y];
        if(xelem > 1){
            mod.insert({xelem-1, xelem});
        }
        if(xelem < n){
            mod.insert({xelem, xelem+1});
        }
        if(yelem > 1){
            mod.insert({yelem-1, yelem});
        }
        if(yelem < n){
            mod.insert({yelem, yelem+1});
        }
        for(auto[left, right]:mod){
            if(pos[left] > pos[right]){
                inv--;
            }
        }
        swap(arr[x], arr[y]);
        pos[xelem] = y;
        pos[yelem] = x;
        for(auto[left, right]:mod){
            if(pos[left] > pos[right]){
                inv++;
            }
        }
        cout<<inv<<endl;
    }

}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
    //cin>>t;
    while(t--){
        solve();
    }
}
```

10.0.4 Concert_Tickets

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
// There are n concert tickets available, each with a certain price. Then, m customers arrive, one after another.
// Each customer announces the maximum price they are willing to pay for a ticket, and after this, they will get a ticket with the nearest possible price such that it does not exceed the maximum price.

signed main() {
    int n, m;
    cin >> n >> m;
    set<pair<int, int>> st;
    for (int i = 0; i < n; i++) {
        int x;
        cin >> x;
        st.insert({x, i});
    }
    while (m--) {
        int q;
        cin >> q;
        if (st.empty()) {
            cout << -1 << endl;
            continue;
```

```
        }
        pair<int, int> c = {q, 0};
        auto it = st.lower_bound(c);
        //cout << "para " << m << " :" << (it->first) << endl;
        if (it == st.end()) {
            it--;
        }
        if (it == st.begin()) {
            if (it->first <= q) {
                cout << (it->first) << endl;
                st.erase(it);
            } else {
                cout << -1 << endl;
            }
        } else {
            if (it->first <= q) {
                cout << (it->first) << endl;
                st.erase(it);
            } else {
                it--;
                cout << (it->first) << endl;
                st.erase(it);
            }
        }
    }
}
```

10.0.5 Distinct_Numbers

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
const int tam = 1;

void solve() {
    int n;
    cin >> n;
    set<int> st;
    int x;
    for (int i = 0; i < n; i++) {
        cin >> x;
        st.insert(x);
    }
    cout << (int)(st.size()) << endl;
}

signed main() {
    // ios::sync_with_stdio(0);
    // cin.tie(0);
    // cout.tie(0);
    int t = 1;
    // cin>>t;
    while (t--) { solve(); }
}
```

10.0.6 Distinct_Values_Subarray

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
//Given an array of n integers, count the number of subarrays where each
↪  element is dictinct.
int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    cin >> n;
    vector<ll> a(n);
    for (int i = 0; i < n; i++) cin >> a[i];

    unordered_map<ll,int> last;
    ll ans = 0;
    int l = 0;

    for (int r = 0; r < n; r++) {
        if (last.count(a[r]) && last[a[r]] >= l) {
            l = last[a[r]] + 1;
        }
        ans += (r - l + 1);
        last[a[r]] = r;
    }

    cout << ans << "\n";
}
```

10.0.7 Distinct_Values_Subarrays_II

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for (int i = (a); i < (b); i++)
#define FOR(i, n) for (int i = 0; i < (n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;
ll g(ll tam) { return tam * (tam + 1) / 2; }
// Given an array of n integers, your task is to calculate the number of
↪  subarrays that have at most k distinct values.

void solve() {
    int n, k;
    cin >> n >> k;
    vi arr(n);
```

```
    fore(i, 0, n) cin >> arr[i];

    map<int, int> freq;
    int l = 0;
    ll ans = 0;

    fore(r, 0, n) {
        freq[arr[r]]++;
        while ((int)freq.size() > k) {
            freq[arr[l]]--;
            if (freq[arr[l]] == 0) freq.erase(arr[l]);
            l++;
        }
        ans += (r - l + 1);
    }
    cout << ans << '\n';
}


signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    int t = 1;
    //   cin>>t;
    while (t--) { solve(); }
}
```

10.0.8 Distinct_Values_Subsequences

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;
unordered_map<int, int> mp;
// Given an array of n integers, count the number of subsequences where each element is dictinct.
// A subsequence is a sequence of array elements from left to right that may have gaps.

void solve(){
   int n;cin>>n;
   mp.reserve(n+10);
   for(int i = 0; i<n; i++){
    int x;cin>>x;
    mp[x]++;
   }
   int ans = 1;
   for(auto p:mp){
    ans*=(p.second+1);
    ans%=MOD;
   }
   cout<<ans-1<<'\n';
}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
 //   cin>>t;
    while(t--){
        solve();
    }
}
```

10.0.9 Factory_Machines

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;
bool f(vector<int>& arr, int k, int mid){
    int suma = 0;
    for(int&x:arr){
        suma+=(mid/x);
        if(suma>=k)return 1;
    }
    return 0;
}
void solve(){
    int n, k;cin>>n>>k;
    vector<int> arr(n);
    for(int i =0; i<n; i++)cin>>arr[i];
    ll l = 1, r = 1000000000000000000LL, ans;
    while(l<=r){
        ll mid = l + (r-l)/2;
        if(f(arr, k, mid)){
            ans = mid;
            r = mid-1;
        }else{
            l = mid+1;
```

```
        }
    }
    cout<<ans<<endl;
}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
 //   cin>>t;
    while(t--){
        solve();
    }
}
```

10.0.10 Factory_Machines

```
import sys
# A factory has n machines which can be used to make products. Your goal is to make a total of t products.
# For each machine, you know the number of seconds it needs to make a single product. The machines can work simultaneously, and you can freely decide their schedule.
# What is the shortest time needed to make t products?
input = sys.stdin.readline
def f(arr, k, mid):
    suma = sum(mid // x for x in arr)
    return suma>=k

n, k = map(int, input().split())
arr = list(map(int, input().split()))
l = 1
r  = max(arr) * k
ans = 0
while l<=r:
    mid = (l+r)//2
    if f(arr, k, mid):
       ans = mid
       r = mid-1
    else:
        l = mid+1
print(ans)
```

10.0.11 Ferris_Wheel

```
#include<bits/stdc++.h>
using namespace std;
#define int long long
//There are n children who want to go to a Ferris wheel, and your task is to find a gondola for each child.
//Each gondola may have one or two children in it, and in addition, the total weight in a gondola may not exceed x. You know the weight of every child.
//What is the minimum number of gondolas needed for the children?
signed main(){
    int n, x;cin>>n>>x;
    vector<int> arr(n);
    for(int &i:arr){cin>>i;}
    sort(arr.begin(), arr.end());
    int l = 0, r = n-1;
    int cont = 0;
    while(l<=r){
        if(arr[l] + arr[r] > x){
            r--;
        }else{
            r--;l++;
        }
        cont++;
    }
    cout<<cont<<endl;
}
```

10.0.12 Josephus_Problem_I

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;
//Consider a game where there are n children (numbered 1,2,\dots,n) in a circle. During the game, every other child is removed from the circle until there are no children left. In which order will the children be removed?
void solve(){
    int n;cin>>n;
    deque<int> dq;
    fore(i, 1, n+1){
        dq.push_back(i);
    }
    int ans;
    while(sz(dq) >= 2){
        int vive = dq.front();
        dq.pop_front();
        int muere = dq.front();
        dq.pop_front();
        cout<<muere<<' ';
        dq.push_back(vive);
    }
    cout<<dq.front();
    cout<<'\n';
}
```

```
signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
 //   cin>>t;
    while(t--){
        solve();
    }
}
```

10.0.13 Josephus_Problem_II

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;
#include <bits/stdc++.h>
using namespace std;
//Consider a game where there are n children (numbered 1,2,\dots,n) in a circle. During the game, repeatedly k children are skipped and one child is removed from the circle. In which order will the children be removed?

#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_statistics_node_update>
//greater or less


//cantidad de elementos en un rango:
//o_set.order_of_key(r+1) - o_set.order_of_key(l)
void solve(){
    int n, k;cin>>n>>k;
    ordered_set os;
    for(int i = 1; i<=n; i++){
        os.insert(i);
    }
    int idx = 0;
    while(n){
        idx+=k;
        idx%=n;
        auto it = os.find_by_order(idx);
        cout<<*it<<' ';
        n--;
        os.erase(it);
    }
    cout<<'\n';
}


signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
    //cin>>t;
    while(t--){
        solve();
    }
}
```

10.0.14 Maximum_Subarray_Sum

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define f first
#define sst stringstream
#define s second
#define pb push_back
#define sz(x) (int)(x).size()
#define all(a) (a).begin(), (a).end()
#define rall(a) (a).rbegin(), (a).rend()
#define fore(i, a, n) for(int i = (a); i < (n); i++)
#define forb(i, n) for(int i = (n) - 1; i >= 0; i--)
#define popcount(x) __builtin_popcountll(x);
typedef pair<int, int> pii;
typedef vector<int> vi;
const int MOD = 1000000007;
const double EPS = 1e-9;
const double PI = acos(-1);
const int INF = 1000000000000;
// PLUS ULTRA RECARGADO!!!
void solve() {
    int n;
    cin >> n;
    vector<int> vec(n);
    for(int i = 0; i < n; i++) {
        cin >> vec[i];
    }
    int suma = 0, ans = -INF;
    for(int i = 0; i < n; i++) {
        suma += vec[i];
        ans = max(ans, suma);
        if(suma < 0) suma = 0;
    }
    if(suma != 0) ans = max(ans, suma);
    cout << ans << endl;
}
signed main() {
```

```
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    //   int t;cin>>t;while(t--)solve();
    solve();
}
```

10.0.15 Maximum_Subarray_Sum_II

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 92233720368547758O7LL;
const int tam = 1;
//Given an array of n integers, your task is to find the maximum sum of values in a contiguous subarray with length between a and b.

void solve(){
    int n, a, b;
    cin >> n >> a >> b;
    vector<int> arr(n+1), pref(n+1, 0);
    for(int i = 1; i <= n; i++){
        cin >> arr[i];
        pref[i] = pref[i-1] + arr[i];
    }

    multiset<int> ms;
    int ans = -INF;
    for(int i = a; i <= n; i++){
        ms.insert(pref[i - a]);
        if(i - b - 1 >= 0){
            ms.erase(ms.find(pref[i - b - 1]));
        }
        ans = max(ans, pref[i] - *ms.begin());
    }
    cout << ans << "\n";
}


signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
 //   cin>>t;
    while(t--){
        solve();
    }
}
```

10.0.16 Missing_Coin_Sum

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
// You have n coins with positive integer values. What is the smallest sum you cannot create using a subset of the coins?

void solve(){
    int n;cin>>n;
    int arr[n];
    fore(i, 0, n)cin>>arr[i];
    sort(arr, arr+n);
    ll k = 0;
    fore(i, 0, n){
        if(k+1>=arr[i]){
            k+=arr[i];
        }else{
            cout<<k+1<<endl;
            return;
        }
    }
    cout<<k+1<<endl;
}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
    //cin>>t;
    while(t--){
        solve();
    }
}
```

10.0.17 Movie_Festival

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define f first
#define sst stringstream
#define s second
#define pb push_back
```

```cpp
#define sz(x) (int)(x).size()
#define all(a) (a).begin(), (a).end()
#define rall(a) (a).rbegin(), (a).rend()
#define fore(i, a, n) for(int i = (a); i < (n); i++)
#define forb(i, n) for (int i = (n) - 1; i >= 0; i--)
#define popcount(x) __builtin_popcountll(x);
typedef long double ld;
typedef pair<int, int> pii;
typedef vector<int> vi;
const int MOD = 1000000007;
const double EPS = 1e-9;
const double PI = acos(-1);
const int INF = 1e18;
//PLUS ULTRA RECARGADO!!!
bool cmp(pii& a, pii& b){
    if(a.s  == b.s)return a.f < b.f;
    return a.s < b.s;
}
// In a movie festival n movies will be shown. You know the starting and ending time of each movie. What is the maximum number of movies you can watch entirely?


void solve() {
    int n;cin>>n;
    vector<pii> v(n);
    fore(i,0,n){
        cin>>v[i].f>>v[i].s;
    }
    sort(all(v), cmp);
    int ans = 0;
    int last  = -1;
    for(auto[ini, fin]:v){
        if(last <= ini){
            last = fin;
            ans++;
        }
    }
    cout<<ans<<endl;
}
signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    int t = 1;
//    cin>>t;
    while(t--)solve();
}
```

10.0.18 Movie_Festival_II

```cpp
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;
bool cmp(pair<int, int>& a, pair<int, int>& b){
    if(a.second  == b.second)return a.first < b.first;
    return a.second < b.second;
}
//In a movie festival, n movies will be shown. Syrjälä's movie club consists of k members, who will be all attending the festival.
//You know the starting and ending time of each movie. What is the maximum total number of movies the club members can watch entirely if they act optimally?
void solve(){
    int n, k;cin>>n>>k;
    vector<pair<int, int>> ran(n);
    fore(i, 0, n){
        int l, r;cin>>l>>r;
        ran[i] = {l, r};
    }
    sort(all(ran), cmp);
    int ans = 1;
    set<pair<int, int>> ultimos;
    int i;
    // for(i = 0; i<min(n, k); i++){
    //     ultimos.insert(ran[i].second);
    //     ans++;
    // }
    ultimos.insert({ran[0].second, 0});
    int cont = 1;
    for(i = 1; i<n; i++){
        //si hay alguien menor a su inicio a ese lo ponemos a chambear
        auto it = ultimos.lower_bound({ran[i].first+1, 0});
        if(it == ultimos.begin()){
            if(cont < k){
                ans++;
                ultimos.insert({ran[i].second, i});
                cont++;
            }
        }else{
            it--;
            ultimos.erase(it);
            ultimos.insert({ran[i].second, i});
            ans++;
        }
    }
    cout<<ans<<endl;
}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
 //   cin>>t;
```

```
    while(t--){
        solve();
    }
}
```

10.0.19 Nearest_Smaller_Values

```
#include <bits/stdc++.h>
#include<unistd.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;
//  Given an array of n integers, your task is to find for each array
↪  position the nearest position to its left having a smaller value.
void solve(){
    int n;cin>>n;
    stack<pair<int, int>> pila;
    for(int i = 0; i<n; i++){
        int x;cin>>x;
        int ans = -1;
        while(!pila.empty() and pila.top().first >= x){
            pila.pop();
        }
        if(!pila.empty()){
            cout<<pila.top().second<<' ';
        }else{
            cout<<0<<' ';
        }
        pila.push({x, i+1});
    }
    cout<<'\n';
}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
    //cin>>t;
    while(t--){
        solve();
    }
}
```

10.0.20 Nested_Ranges_Check

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;
//Given n ranges, your task is to determine for each range if it contains
↪  some other range and if some other range contains it.
//Range [a,b] contains range [c,d] if a \le c and d \le b.
bool cmp(tuple<int, int, int>& a, tuple<int, int, int>& b){

    if(get<0>(a) == get<0>(b))return get<1>(a) > get<1>(b);
    return get<0>(a) < get<0>(b);
}
void solve(){
    int n;cin>>n;
    vector<tuple<int, int, int>> arr(n);
    fore(i, 0, n){
        int x, y;cin>>x>>y;
        arr[i] = {x, y, i};
    }
    sort(all(arr), cmp);
    vector<int> minder(n);
    minder[n-1] = get<1>(arr.back());
    for(int i = n-2; i>=0; i--){
        minder[i] = min(minder[i+1], get<1>(arr[i]));
    }
    vector<int> maxizq(n);
    maxizq[0] = get<1>(arr[0]);
    for(int i = 1; i<n; i++){
        maxizq[i] = max(maxizq[i-1], get<1>(arr[i]));
    }
    vector<bool> contiene(n);
    for(int i = 0;i<n-1; i++){
        if(minder[i+1]<=get<1>(arr[i])){
            contiene[get<2>(arr[i])] = 1;
        }
    }
    vector<bool> contenido(n);
    for(int i = 1; i<n; i++){
        if(maxizq[i-1] >= get<1>(arr[i])){
            contenido[get<2>(arr[i])] = 1;
        }
    }
    for(bool i:contiene)cout<<i<<' ';
    cout<<'\n';
    for(bool i:contenido)cout<<i<<' ';
    cout<<'\n';
```

```
}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
//    cin>>t;
    while(t--){
        solve();
    }
}
```

10.0.21 Nested_Ranges_Count

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for (int i = (a); i < (b); i++)
#define FOR(i, n) for (int i = 0; i < (n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;
//Given n ranges, your task is to count for each range how many other ranges it contains and how many other ranges contain it.
//Range [a,b] contains range [c,d] if a \le c and d \le b.

// primero bits/stdc, std y luego las demas libs#include <bits/stdc++.h>
using namespace std;
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
typedef pair<int, int> pii;

typedef tree<pii, null_type, less<pii>, rb_tree_tag,
             tree_order_statistics_node_update>
    ordered_ms;

struct ordMultiset {
    int count;
    ordered_ms oset;

    ordMultiset() { count = 0; }
    // o que posicion le corresponde
    int count_oc(int x) {
        return oset.order_of_key({x + 1, 0}) - oset.order_of_key({x, 0});
    }
    void erase_one(int x) {
        auto it = oset.lower_bound({x, 0});
        if (it != oset.end() && it->first == x) { oset.erase(it); }
    }
    // 0 indexed
    int kth_element(int k) { return oset.find_by_order(k)->first; }

    int count_greater_equal(int x) {
        return (int)oset.size() - oset.order_of_key({x, 0});
    }

    void insert(int x) { oset.insert({x, count++}); }
};
struct tupla{
    int l, r, idx;
    tupla(){}
    tupla(int _l, int _r, int _idx){
        l = _l;
        r = _r;
        idx = _idx;
    }
};
bool cmp(tupla& a, tupla& b){
    if(a.l == b.l)return a.r > b.r;
    return a.l<b.l;
}
void solve() {
    int n;cin>>n;
    vector<tupla> rangos(n);
    for(int i = 0; i<n; i++){
        int a, b;cin>>a>>b;
        rangos[i] = {a, b, i};
    }
    sort(all(rangos), cmp);
    ordMultiset os;
    os.insert(rangos[0].r);
    vector<int> contenido(n);
    for(int i = 1; i<n; i++){
        contenido[rangos[i].idx] = os.count_greater_equal(rangos[i].r);
        os.insert(rangos[i].r);
    }
    vector<int> contiene(n);
    os.oset.clear();
    os.count = 0;
    os.insert(rangos.back().r);
    for(int i = n-2; i>=0; i--){
        contiene[rangos[i].idx] = os.oset.order_of_key({rangos[i].r+1, 0});
        os.insert(rangos[i].r);
    }
    for(int x:contiene)cout<<x<<' ';
    cout<<'\n';
    for(int x:contenido)cout<<x<<' ';
    cout<<'\n';
}

signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    int t = 1;
    //cin >> t;
```

```
    while (t--) { solve(); }
}
```

10.0.22 Playlist

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for (int i = (a); i < (b); i++)
#define FOR(i, n) for (int i = 0; i < (n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;

// You are given a playlist of a radio station since its establishment.
↪  The playlist has a total of n songs.
// What is the longest sequence of successive songs where each song is
↪  unique?

void solve() {
  int n;
  cin >> n;
  int arr[n];
  fore(i, 0, n) { cin >> arr[i]; }
  map<int, int> mp;
  int l = 0, r = 0;
  int ans = 0;
  while (r < n) {
    mp[arr[r]]++;
    if(mp[arr[r]]>1){
        while(mp[arr[r]]>1){
            mp[arr[l]]--;
            l++;
        }
    }

    ans = max(ans, r-l+1);
    r++;
}
  cout << ans << endl;
}

signed main() {
  ios::sync_with_stdio(0);
  cin.tie(0);
  cout.tie(0);
  int t = 1;
  // cin>>t;
  while (t--) {
    solve();
  }
}
```

10.0.23 Reading_Books

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;
// There are n books, and Kotivalo and Justiina are going to read them
↪  all. For each book, you know the time it takes to read it.
// They both read each book from beginning to end, and they cannot read a
↪  book at the same time. What is the minimum total time required?

void solve(){
   int n;cin>>n;
   vector<int> arr(n);
   fore(i, 0, n)cin>>arr[i];
   cout<<max(2LL*(*max_element(all(arr))), accumulate(all(arr), 0LL));
}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
 //   cin>>t;
    while(t--){
        solve();
    }
}
```

10.0.24 Restaurant_Customers

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define f first
#define sst stringstream
#define s second
#define pb push_back
#define sz(x) (int)(x).size()
#define all(a) (a).begin(), (a).end()
#define rall(a) (a).rbegin(), (a).rend()
#define fore(i, a, n) for(int i = (a); i < (n); i++)
#define forb(i, n) for (int i = (n) - 1; i >= 0; i--)
```

```cpp
#define popcount(x) __builtin_popcountll(x);
typedef pair<int, int> pii;
typedef vector<int> vi;
const int MOD = 1000000007;
const double EPS = 1e-9;
const double PI = acos(-1);
const int INF = 1e18;
//PLUS ULTRA RECARGADO!!!
void solve() {
   int n;cin>>n;
   vector<pair<int, int>> vec;
   for(int i = 0; i<n; i++){
    int a, b;cin>>a>>b;
    vec.push_back({a, +1});
    vec.push_back({b, -1});
   }
   int ans = 0;
   int c = 0;
   sort(all(vec));
   for(int i = 0; i<sz(vec); i++){
    c+=vec[i].s;
    ans = max(ans, c);
   }
   cout<<ans<<endl;

}
signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
 //   int t;cin>>t;while(t--)solve();
    solve();
}
```

10.0.25 Room_Allocation

```cpp
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const int INF = 922337203;
const int tam = 1;
struct tupla
{
    int l, r, index;

};
bool cmp(tupla& a, tupla&b){
    return make_pair(a.l, a.r) < make_pair(b.l, b.r);
}
//There is a large hotel, and n customers will arrive soon. Each customer wants to have a single room.
// You know each customer's arrival and departure day. Two customers can stay in the same room if the departure day of the first customer is earlier than the arrival day of the second customer.
// What is the minimum number of rooms that are needed to accommodate all customers? And how can the rooms be allocated?
void solve(){
    int n;cin>>n;
    vector<tupla> ran(n);
    for(int i = 0; i<n;i++){
        int l, r;cin>>l>>r;
        ran[i] = {l, r, i};
    }
    sort(all(ran), cmp);
    int k = 0;
    vector<int> ans(n);
    set<pair<int, int>> uso;
    for(int i = 0; i<n; i++){
        auto it = uso.upper_bound({ran[i].l, 0});
        if(it != uso.begin() and sz(uso) > 0){
            it = uso.begin();
            int idx = it->second;
            uso.erase(it);
            uso.insert({ran[i].r, idx});
            ans[ran[i].index] = idx;
        }else{
            k++;
            ans[ran[i].index] = k;
            uso.insert({ran[i].r, k});
        }
    }
    cout<<k<<endl;
    for(int x:ans)cout<<x<<' ';
    cout<<'\n';
}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
 //   cin>>t;
    while(t--){
        solve();
    }
}
```

10.0.26 Stick_Lengths

```cpp
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define f first
#define sst stringstream
```

```cpp
#define s second
#define pb push_back
#define sz(x) (int)(x).size()
#define all(a) (a).begin(), (a).end()
#define rall(a) (a).rbegin(), (a).rend()
#define fore(i, a, n) for(int i = (a); i < (n); i++)
#define forb(i, n) for(int i = (n) - 1; i >= 0; i--)
#define popcount(x) __builtin_popcountll(x);
typedef pair<int, int> pii;
typedef vector<int> vi;
const int MOD = 1000000007;
const double EPS = 1e-9;
const double PI = acos(-1);
const int INF = 1e18;
// PLUS ULTRA RECARGADO!!!
// There are n sticks with some lengths. Your task is to modify the sticks so that each stick has the same length.
// You can either lengthen and shorten each stick. Both operations cost x where x is the difference between the new and original length.
// What is the minimum total cost?
void solve() {
    int n;
    cin >> n;
    vector<int> a(n);
    for(int i = 0; i < n; i++) {
        cin >> a[i];
    }
    sort(all(a));
    int mediana;
    if(!(n & 1)) {
        mediana = a[(n - 1) / 2] + a[n / 2];
        mediana = (mediana +1)/2;
    }else{
        mediana = a[n/2];
    }
    int ans = 0;
    for(int i:a){
        ans+=abs(i-mediana);
    }
    cout<<ans<<endl;
}
signed main() {
    solve();
}
```

10.0.27 Subarray_Divisibility

```cpp
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;
// Given an array of n integers, your task is to count the number of subarrays where the sum of values is divisible by n.

void solve() {
    int n, k;
    cin >> n;
    int suma = 0;
    map<int, int> prefijos;
    prefijos[0] = 1;
    int ans = 0;
    fore(i, 0, n) {
        int x;cin>>x;
        suma+=x;
        while(suma < 0){
            suma+=(n*10LL);
        }
        suma%=n;
        if(prefijos.find(suma) != prefijos.end()){
            ans+=prefijos[suma];
        }
        prefijos[suma]++;
    }
    cout<<ans<<endl;
}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
 //   cin>>t;
    while(t--){
        solve();
    }
}
```

10.0.28 Subarray_Sums_I

```cpp
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define rec(arr, n) vector<int> arr(n);fore(i, 0, n)cin>>arr[i];
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
```

```
const int tam = 1;

//  Given an array of n positive integers, your task is to count the
↪  number of subarrays having sum x.
void solve(){
    int n, k;cin>>n>>k;
    rec(arr, n);
    int l = 0;
    int ans =0, sum = 0;
    fore(r, 0, n){
        sum+=arr[r];
        if(sum == k)ans++;
        while(sum > k and l<r){
            sum-=arr[l];
            l++;
            if(sum == k)ans++;
        }
    }
    cout<<ans<<endl;

}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
    //cin>>t;
    while(t--){
        solve();
    }
}
```

10.0.29 Subarray_Sums_II

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for (int i = (a); i < (b); i++)
#define FOR(i, n) for (int i = 0; i < (n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;
 //Given an array of n integers, your task is to count the number of
 ↪  subarrays having sum x.
void solve() {
    int n, k;
    cin >> n >> k;
    int suma = 0;
    map<int, int> prefijos;
    prefijos[0] = 1;
    int ans = 0;
    fore(i, 0, n) {
        int x;cin>>x;
        suma+=x;
        int falta = suma - k;
        if(prefijos[falta] > 0){
            ans+=prefijos[falta];
        }
        prefijos[suma]++;
    }
    cout<<ans<<endl;
}

signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    int t = 1;
    //   cin>>t;
    while (t--) { solve(); }
}
```

10.0.30 Sum_of_Four_Values

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;
// You are given an array of n integers, and your task is to find four
↪  values (at distinct positions) whose sum is x.
void solve(){
    int n, k;cin>>n>>k;
    vector<int> arr(n);
    fore(i, 0, n)cin>>arr[i];
    map<int, pair<int, int>> mp;
    if(n < 4){
        cout<<"IMPOSSIBLE\n";
        return;
    }
    mp[arr[0] + arr[1]] = {0, 1};
    for(int l = 2; l<n; l++){
        for(int r = l+1; r<n; r++){
            int valor = k - arr[l] - arr[r];
            if(mp.find(valor) != mp.end()){
                //ya encontre
```

```
                cout<<l+1<<' '<<r+1<<' '<<mp[valor].first+1<<' '<<mp[valor].second+1<<'\n';
                return;
            }
        }
        for(int i = 0; i<l; i++){
            mp[arr[i] + arr[l]] = {i, l};
        }
    }
    cout<<"IMPOSSIBLE\n";

}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
    //cin>>t;
    while(t--){
        solve();
    }
}
```

10.0.31 Sum_of_Three_Values

```
#include <bits/stdc++.h>
using namespace std;
// #include <ext/pb_ds/assoc_container.hpp>
// #include <ext/pb_ds/assoc_container.hpp>
// #include <ext/pb_ds/tree_policy.hpp>
// #include <ext/rope>
#define int long long
#define uset unordered_set
#define f first
#define s second
#define umap unordered_map
#define mp make_pair
#define pb push_back
#define sz(x) (int)(x).size()
#define all(a) (a).begin(), (a).end()
#define rall(a) (a).rbegin(), (a).rend()
#define floatigual(a, b) (fabs(a - b) < EPS)
#define mod(a) md(a, MOD)
#define FOR(i, n) for (int i = 0; i < (n); ++i)
#define FOR3(i, a, b) for (int i = (a); i < (b); ++i)
#define FORD(i, n) for (int i = (n) - 1; i >= 0; --i)
#define FORDD(i, a, b) for (int i = (b) - 1; i >= (a); --i)
#define techo(a, b) (a / b + (a % b != 0))
#define popcount(x) __builtin_popcountll(x);
using namespace std;
// You are given an array of n integers, and your task is to find three values (at distinct positions) whose sum is x.
bool f(int a, int b, int c){
    return a!=b and b!=c and a!=c;
}
void solve() {
    int n, k;cin>>n>>k;
    vector<pii> a(n);
    FOR(i, n){
      int x;cin>>x;
      a[i] = {x, i};
    }
    sort(all(a));
    FOR(i, n){
     FOR3(j, i+1, n){
         int buscado = k- a[i].first - a[j].first;
         int l = 0, r = n-1;
         while(l<=r){
             int mid = (l+r)/2;
             if(a[mid].f == buscado and f(a[mid].s, a[i].s, a[j].s)){
                 cout<<a[mid].s+1<<' '<<a[i].s+1<<' '<<a[j].s+1<<'\n';
                 return;
             }else if(buscado<a[mid].f){
                 r = mid-1;
             }else{
                 l = mid+1;
             }
         }
     }
    }
    cout<<"IMPOSSIBLE\n";

}
signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
 //   int t;cin>>t;while(t--)solve();
    solve();
}
```

10.0.32 Sum_of_Two_Values

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;
using pii = pair<int, int>;
void solve() {
    int n, k;cin>>n>>k;
    vector<pii> a(n);
    FOR(i, n){
```

```
        int x;cin>>x;
        a[i] = {x, i};
    }
    sort(all(a));
    FOR(i, n){
        int buscado = k - a[i].first;
        int l = 0; int r = n-1;
        int res = -1;
        while(l<=r){
            int mid = (l+r)/2;
            if(a[mid].first == buscado and a[i].second != a[mid].second){
                cout<<a[mid].second+1<<' '<<a[i].second+1<<'\n';
                return;
            }else if(a[mid].first<buscado){
                l = mid+1;
            }else{
                r = mid-1;
            }
        }

    }
    cout<<"IMPOSSIBLE\n";
}
signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
 //   int t;cin>>t;while(t--)solve();
    solve();
}
```

10.0.33 Tasks_and_Deadlines

```
// You have to process n tasks. Each task has a duration and a deadline, and you will process the tasks in some order one after another. Your reward for a task is d-f where d is its deadline and f is your finishing time. (The starting time is 0, and you have to process all tasks even if a task would yield negative reward.)
// What is your maximum reward if you act optimally?
void solve() {
    int n;
    cin >> n;
    vector<pii> v(n);
    FOR(i, n){
        cin >> v[i].first >> v[i].second;
    }

    sort(all(v));

    int tiempo = 0;
    int res = 0;

    for (pii& actual : v) {
        int a = actual.first;
        int b = actual.second;
        tiempo += a;
        res += (b - tiempo);
    }

    cout << res << endl;
}

signed main()
{

    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    //int t;
    //cin >> t;
    //while (t--)
      //  solve();
    solve();
}
```

10.0.34 Towers

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
const int tam = 1;
// You are given n cubes in a certain order, and your task is to build towers using them. Whenever two cubes are one on top of the other, the upper cube must be smaller than the lower cube.
// You must process the cubes in the given order. You can always either place the cube on top of an existing tower, or begin a new tower. What is the minimum possible number of towers?
void solve(){
    int n;cin>>n;
    multiset<int> ms;
    int ans = 0;
    for(int i = 0; i<n; i++){
        int x;cin>>x;
        auto it = ms.upper_bound(x);
        if(it == ms.end()){
            ans++;
            ms.insert(x);
        }else{
            ms.erase(it);
            ms.insert(x);
        }
    }
    cout<<ans<<endl;
}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
    //cin>>t;
```

```
    while(t--){
        solve();
    }
}
```

10.0.35 Traffic_Lights

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i < (b); i++)
#define FOR(i, n) fore(i, 0, n)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
// There is a street of length x whose positions are numbered 0,1,\ldots,x. Initially there are no traffic lights, but n sets of traffic lights are added to the street one after another.
// Your task is to calculate the length of the longest passage without traffic lights after each addition.


void solve() {
    ll x, n;
    cin >> x >> n;
    set<ll> positions = {0, x};
    multiset<ll> intervals = {x};
    fore(i, 0, n){
        ll p;
        cin >> p;
        auto it = positions.insert(p).first;
        auto left = *prev(it);
        auto right = *next(it);
        intervals.erase(intervals.find(right - left));
        intervals.insert(p - left);
        intervals.insert(right - p);
        cout << *intervals.rbegin() << " ";
    }
}

int main() {
    ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
    int t = 1;
    // cin >> t;
    while(t--) {
        solve();
    }
}
```

11 String Algorithms

11.0.1 Finding_Borders

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for(int i = (a); i<(b); i++)
#define FOR(i, n) for(int i = 0; i<(n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 1;
vector<int> kmp(string& s) {
    int n = sz(s);
    vector<int> pi(n);
    for (int i = 1; i < n; i++) {
        int j = pi[i - 1];
        while (j > 0 and s[i] != s[j]) j = pi[j - 1];
        if (s[i] == s[j]) j++;
        pi[i] = j;
    }
    return pi;
}
void solve(){
    string s;cin>>s;
    auto pi = kmp(s);
    int idx = sz(s)-1;
    vi ans;
    while(idx > 0){
        ans.push_back(pi[idx]);
        idx = pi[idx]-1;
    }
    sort(all(ans));
    for(int x:ans)if(x)cout<<x<<' ';
    cout<<'\n';
}

signed main(){
    ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t = 1;
    //cin>>t;
    while(t--){
        solve();
    }
}
```

11.0.2 Longest_Palindrome

```cpp
// Given a string, your task is to determine the longest palindromic substring of the string. For example, the longest palindrome in aybabtu is bab.
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define f first
#define sst stringstream
#define s second
#define pb push_back
#define sz(x) (int)(x).size()
#define all(a) (a).begin(), (a).end()
#define rall(a) (a).rbegin(), (a).rend()
#define fore(i, a, n) for(int i = (a); i < (n); i++)
#define forb(i, n) for (int i = (n) - 1; i >= 0; i--)
#define popcount(x) __builtin_popcountll(x);
typedef pair<int, int> pii;
typedef vector<int> vi;
const int MOD = 1000000007;
const double EPS = 1e-9;
const double PI = acos(-1);
const int INF = 1e18;
//PLUS ULTRA RECARGADO!!!

vector<int> manacher_odd(string s) {
    int n = s.size();
    s = "$" + s + "^";
    vector<int> p(n + 2);
    int l = 1, r = 1;
    for(int i = 1; i <= n; i++) {
        p[i] = max(0LL, min(r - i, p[l + (r - i)]));
        while(s[i - p[i]] == s[i + p[i]]) {
            p[i]++;
        }
        if(i + p[i] > r) {
            l = i - p[i], r = i + p[i];
        }
    }
    return vector<int>(begin(p) + 1, end(p) - 1);
}

vector<int> manacher(string s) {
    string t = "";
    for(auto c: s) {
        t.push_back('#');
        t.push_back(c);
    }
    auto res = manacher_odd(t + "#");
    return vector<int>(begin(res) + 1, end(res) - 1);
}


void solve() {
    string s;cin>>s;
    vi mani = manacher(s);
    int posi = -1;
    int maxi = 0;
    for(int i = 0; i<sz(mani); i++){
        if(mani[i]  > maxi){
            maxi = mani[i];
            posi = i;
        }
    }
    posi = (posi - maxi)/2;
    cout<<s.substr(posi+1, maxi-1);
}
signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    //int t;cin>>t;while(t--)solve();
    solve();
}
```

11.0.3 Palindrome_Queries_BIT

```cpp
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define bint __int128

#define int long long

bint MOD = 212345678987654321LL;
bint P   = 1777771;
bint PI  = 106955741089659571LL;

struct Fenwick {
    int n;
    vector<bint> bit;
    Fenwick(int n=0){ init(n); }
    void init(int _n){
        n = _n;
        bit.assign(n+1, 0);
    }
    void add(int idx, bint val){
        while(idx <= n){
            bit[idx] += val;
            bit[idx] %= MOD;
            idx += idx & -idx;
        }
    }
    bint sum(int idx){
        bint res = 0;
        while(idx > 0){
            res += bit[idx];
            res %= MOD;
            idx -= idx & -idx;
        }
        return res;
```

```
    }
    bint rangeSum(int l, int r){
        if(r < l) return 0;
        bint res = sum(r) - sum(l-1);
        res = (res % MOD + MOD) % MOD;
        return res;
    }
};

int toIntChar(char c){ return (int)(c); }

void solve(){
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, q;
    if(!(cin >> n >> q)) return;
    string s; cin >> s;

    // Precompute powers and inverse powers (indexado a 0..n)
    vector<bint> pows(n+1), pi(n+1);
    pows[0] = 1; pi[0] = 1;
    for(int i = 1; i <= n; ++i){
        pows[i] = (pows[i-1] * P) % MOD;
        pi[i]   = (pi[i-1] * PI) % MOD;
    }

    // Arrays actuales (indexado 1..n)
    vector<int> cur(n+1), curRev(n+1);
    for(int i = 1; i <= n; ++i) cur[i] = toIntChar(s[i-1]);
    for(int i = 1; i <= n; ++i) curRev[i] = toIntChar(s[n - i]); //
    ↪  invertida

    // Fenwicks
    Fenwick bit(n), bitRev(n);
    bit.init(n); bitRev.init(n);

    // Inicializar: en BIT guardamos a[i] * P^{i-1}
    for(int i = 1; i <= n; ++i){
        bint val = ( (bint)cur[i] * pows[i-1] ) % MOD;
        bit.add(i, val);
    }
    for(int i = 1; i <= n; ++i){
        bint val = ( (bint)curRev[i] * pows[i-1] ) % MOD;
        bitRev.add(i, val);
    }

    auto getHash = [&](Fenwick &B, int l, int r)->long long {
        // raw = sum_{i=l..r} a[i]*P^{i-1}
        bint raw = B.rangeSum(l, r);
        // normalizar: multiplicar por PI^{l-1} para que exponente empiece
        ↪  en 0
        raw = (raw * pi[l-1]) % MOD;
        raw = (raw + MOD) % MOD;
        return (long long) raw;
    };

    while(q--){
        int t; cin >> t;
        if(t == 1){
            int pos; char x; cin >> pos >> x;
            int newv = toIntChar(x);
            // normal
            if(newv != cur[pos]){
                bint diff = ( (bint)newv - (bint)cur[pos] ) % MOD;
                diff = (diff + MOD) % MOD;
                bint delta = (diff * pows[pos-1]) % MOD;
                bit.add(pos, delta);
                cur[pos] = newv;
            }
            // invertido en posicion n-pos+1
            int posr = n - pos + 1;
            if(newv != curRev[posr]){
                bint diff2 = ( (bint)newv - (bint)curRev[posr] ) % MOD;
                diff2 = (diff2 + MOD) % MOD;
                bint delta2 = (diff2 * pows[posr-1]) % MOD;
                bitRev.add(posr, delta2);
                curRev[posr] = newv;
            }
        } else {
            int l, r; cin >> l >> r;
            long long h1 = getHash(bit, l, r);
            // para la invertida: correspondencia [n-r+1, n-l+1]
            int rl = n - r + 1;
            int rr = n - l + 1;
            long long h2 = getHash(bitRev, rl, rr);
            cout << (h1 == h2 ? "YES" : "NO") << '\n';
        }
    }
}

signed main(){
    solve();
    return 0;
}
```

11.0.4 Palindrome_Queries

```
// You are given a string that consists of n characters between a-z. The
↪  positions of the string are indexed 1,2,\dots,n.
// Your task is to process m operations of the following types:

// Change the character at position k to x
// Check if the substring from position a to position b is a palindrome
#include <bits/stdc++.h>
using namespace std;
using ull = unsigned long long;
const ull MOD1 = 1000000007ULL;
const ull MOD2 = 1000000009ULL;
const ull P1 = 1777771ULL;
const ull P2 = 19260817ULL;
struct SegmentTree {
    struct Node {
```

```
    ull h1, h2;
    int len;
    Node(ull _h1 = 0, ull _h2 = 0, int _len = 0)
        : h1(_h1), h2(_h2), len(_len) {}
};

int n;
vector<Node> st;
vector<ull> p1, p2;

SegmentTree(const vector<int>& a) {
    n = (int)a.size();
    st.assign(4 * n, Node());
    p1.assign(n + 1, 1);
    p2.assign(n + 1, 1);
    precomputePowers();
    build(1, 0, n - 1, a);
}

void precomputePowers() {
    p1[0] = p2[0] = 1;
    for (int i = 1; i <= n; ++i) {
        p1[i] = (p1[i - 1] * P1) % MOD1;
        p2[i] = (p2[i - 1] * P2) % MOD2;
    }
}

Node merge(const Node& L, const Node& R) {
    ull nh1 = ((L.h1 + (R.h1 * p1[L.len]) % MOD1)) % MOD1;
    ull nh2 = ((L.h2 + (R.h2 * p2[L.len]) % MOD2)) % MOD2;
    return Node(nh1, nh2, L.len + R.len);
}

void build(int p, int l, int r, const vector<int>& a) {
    if (l == r) {
        ull v1 = (ull)(a[l] % (int)MOD1);
        ull v2 = (ull)(a[l] % (int)MOD2);
        st[p] = Node(v1, v2, 1);
        return;
    }
    int m = (l + r) >> 1;
    build(p << 1, l, m, a);
    build(p << 1 | 1, m + 1, r, a);
    st[p] = merge(st[p << 1], st[p << 1 | 1]);
}

void update(int p, int l, int r, int idx, int val) {
    if (l == r) {
        ull v1 = (ull)(val % (int)MOD1);
        ull v2 = (ull)(val % (int)MOD2);
        st[p] = Node(v1, v2, 1);
        return;
    }
    int m = (l + r) >> 1;
    if (idx <= m)
        update(p << 1, l, m, idx, val);
    else
        update(p << 1 | 1, m + 1, r, idx, val);
    st[p] = merge(st[p << 1], st[p << 1 | 1]);
}
void update(int idx, int val) { update(1, 0, n - 1, idx, val); }

Node queryNode(int p, int l, int r, int ql, int qr) {
    if (qr < l or ql > r) return Node(0, 0, 0);
    if (ql <= l and r <= qr) return st[p];
    int m = (l + r) >> 1;
    Node L = queryNode(p << 1, l, m, ql, qr);
    Node R = queryNode(p << 1 | 1, m + 1, r, ql, qr);
    return merge(L, R);
}
pair<ull, ull> query(int l, int r) {
    Node res = queryNode(1, 0, n - 1, l, r);
    return {res.h1, res.h2};
}
};

void solve() {
    int n, q;
    cin >> n >> q;
    string s;
    cin >> s;
    vector<int> a(n), b(n);
    for (int i = 0, j = n - 1; i < n; ++i, --j) {
        a[i] = (int)s[i];
        b[i] = (int)s[j];
    }

    SegmentTree normal(a), invertido(b);

    while (q--) {
        int t;
        cin >> t;
        if (t == 1) {
            int pos;
            char x;
            cin >> pos >> x;
            pos--;
            normal.update(pos, (int)x);
            invertido.update(n - 1 - pos, (int)x);
        } else {
            int l, r;
            cin >> l >> r;
            l--, r--;
            auto h1 = normal.query(l, r);
            auto h2 = invertido.query(n - 1 - r, n - 1 - l);
            cout << (h1 == h2 ? "YES" : "NO") << '\n';
        }
    }
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(0);
    solve();
    return 0;
```

```
}

//=====================================================
#include <bits/stdc++.h>
using namespace std;
using ull = unsigned long long;

struct SegmentTree {
    const ull MOD1 = 1000000007ULL;
    const ull MOD2 = 1000000009ULL;
    const ull P1 = 1777771ULL;
    const ull P2 = 19260817ULL;

    struct Node {
        ull h1, h2;
        int len;
        Node(ull _h1=0, ull _h2=0, int _len=0): h1(_h1), h2(_h2),
        ↪  len(_len) {}
    };

    int n;
    vector<Node> st;
    vector<ull> p1, p2;

    SegmentTree(const vector<int>& a) {
        n = (int)a.size();
        st.assign(4*n, Node());
        p1.assign(n+1, 1);
        p2.assign(n+1, 1);
        precomputePowers();
        build(1, 0, n-1, a);
    }

    void precomputePowers() {
        p1[0] = p2[0] = 1;
        for (int i = 1; i <= n; ++i) {
            p1[i] = (p1[i-1] * P1) % MOD1;
            p2[i] = (p2[i-1] * P2) % MOD2;
        }
    }

    // MERGE: left.h + right.h * P^{left.len}
    Node merge(const Node &L, const Node &R) {
        ull nh1 = ( (L.h1 + (R.h1 * p1[L.len]) % MOD1) ) % MOD1;
        ull nh2 = ( (L.h2 + (R.h2 * p2[L.len]) % MOD2) ) % MOD2;
        return Node(nh1, nh2, L.len + R.len);
    }

    void build(int p, int l, int r, const vector<int>& a) {
        if (l == r) {
            ull v1 = (ull)(a[l] % (int)MOD1);
            ull v2 = (ull)(a[l] % (int)MOD2);
            st[p] = Node(v1, v2, 1);
            return;
        }
        int m = (l + r) >> 1;
        build(p<<1, l, m, a);
        build(p<<1|1, m+1, r, a);
        st[p] = merge(st[p<<1], st[p<<1|1]);
    }

    void update(int p, int l, int r, int idx, int val) {
        if (l == r) {
            ull v1 = (ull)(val % (int)MOD1);
            ull v2 = (ull)(val % (int)MOD2);
            st[p] = Node(v1, v2, 1);
            return;
        }
        int m = (l + r) >> 1;
        if (idx <= m) update(p<<1, l, m, idx, val);
        else update(p<<1|1, m+1, r, idx, val);
        st[p] = merge(st[p<<1], st[p<<1|1]);
    }
    void update(int idx, int val) { update(1, 0, n-1, idx, val); }

    Node queryNode(int p, int l, int r, int ql, int qr) {
        if (qr < l || ql > r) return Node(0,0,0);
        if (ql <= l && r <= qr) return st[p];
        int m = (l + r) >> 1;
        Node L = queryNode(p<<1, l, m, ql, qr);
        Node R = queryNode(p<<1|1, m+1, r, ql, qr);
        return merge(L, R);
    }
    pair<ull, ull> query(int l, int r) {
        Node res = queryNode(1, 0, n-1, l, r);
        return {res.h1, res.h2};
    }
};

void solve(){

    int n, q; cin >> n >> q;
    string s; cin >> s;
    vector<int> a(n), b(n);
    for (int i = 0, j = n-1; i < n; ++i, --j) {
        a[i] = (int)(unsigned char)s[i];
        b[i] = (int)(unsigned char)s[j];
    }

    SegmentTree normal(a), invertido(b);

    while (q--) {
        int t; cin >> t;
        if (t == 1) {
            int pos; char x; cin >> pos >> x;
            pos--; // 0-based
            normal.update(pos, (int)(unsigned char)x);
            invertido.update(n-1-pos, (int)(unsigned char)x);
        } else {
            int l, r; cin >> l >> r;
            l--, r--;
            auto h1 = normal.query(l, r);
            // en el invertido pedimos la ventana correspondiente (ya
            ↪  0-based)
            auto h2 = invertido.query(n-1-r, n-1-l);
            cout << (h1 == h2 ? "YES" : "NO") << '\n';
```

```
        }
    }
}

int main(){
       ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(0);
    solve();
    return 0;
}
```

12 Tree Algorithms

12.0.1 Distinct_Colors

```
//dusu
// You are given a rooted tree consisting of n nodes. The nodes are numbered 1,2,\ldots,n, and node 1 is the root. Each node has a color
// Your task is to determine for each node the number of distinct colors in the subtree of the node.
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
#define fore(i, a, b) for (int i = (a); i < (b); i++)
#define FOR(i, n) for (int i = 0; i < (n); i++)
#define all(x) (x).begin(), (x).end()
#define sz(x) (int)(x).size()
#define pb push_back
using vi = vector<int>;
const int MOD = 1000000007;
const ll INF = 9223372036854775807LL;
const int tam = 2 * 1e5 + 10;
vector<set<int>> conjuntos(tam);
vector<int> ans(tam);
vector<int> color(tam);
void dfs(int nodo, int padre, vector<vi>& g) {
    for (int vecino : g[nodo]) {
        if (vecino != padre) { dfs(vecino, nodo, g); }
    }
    int bigchild = -1;
    for (int vecino : g[nodo]) {
        if (vecino == padre) continue;
        if (bigchild == -1 or sz(conjuntos[vecino]) > sz(conjuntos[bigchild])) {
            bigchild = vecino;
        }
    }

    if (bigchild != -1) { swap(conjuntos[nodo], conjuntos[bigchild]); }
    conjuntos[nodo].insert(color[nodo]);
    for (int vecino : g[nodo]) {
        if (vecino == padre or vecino == bigchild) continue;
        for (int colorsito : conjuntos[vecino]) {
            conjuntos[nodo].insert(colorsito);
        }
    }
    ans[nodo] = sz(conjuntos[nodo]);
}

void solve() {
    int n;
    cin >> n;
    fore(i, 1, n+1) cin >> color[i];
    vector<vector<int>> g(n + 1);
    fore(i, 0, n - 1) {
        int a, b;
        cin >> a >> b;
        g[a].push_back(b);
        g[b].push_back(a);
    }
    dfs(1, -1, g);
    fore(i, 1, n+1)cout<<ans[i]<<" ";
}

signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    int t = 1;
    //cin >> t;
    while (t--) { solve(); }
}
```