

# CSES Solutions TeamBook

Un económico pecho mixto para la mesa por favor.

Universidad Mayor de San Simón (UMSS)

## Contents

<b>1 Additional Problems II</b>	<b>3</b>
1.0.1 GCD_Subsets . . . . .	3
<b>2 Bitwise Operations</b>	<b>4</b>
2.0.1 All_Subarray_Xors . . . . .	4
2.0.2 Counting_Bits . . . . .	4
2.0.3 K_Subset_Xors . . . . .	4
2.0.4 Maximum_Xor_Subarray . . . . .	5
2.0.5 Maximum_Xor_Subset . . . . .	6
2.0.6 Number_of_Subset_Xors . . . . .	6
2.0.7 Xor_Pyramid_Peak . . . . .	7
<b>3 Dynamic Programming</b>	<b>7</b>
3.0.1 Book_Shop . . . . .	7
3.0.2 Coin_Combinations . . . . .	8
3.0.3 Coin_Combinations_II . . . . .	8
3.0.4 Dice_Combinations . . . . .	9
3.0.5 Edit_Distance . . . . .	9
3.0.6 Grid_Paths_I . . . . .	10
3.0.7 Longest_Common_Subsequence . . . . .	10
3.0.8 Longest_Common_Subsequences . . . . .	11
3.0.9 Minimizing_Coins . . . . .	11
3.0.10 Money_Sums . . . . .	12
3.0.11 Rectangle_Cutting . . . . .	12
3.0.12 Removing_Digits . . . . .	13
<b>4 Geometry</b>	<b>13</b>
4.0.1 All_Manhattan_Distances . . . . .	13
4.0.2 Area_of_Rectangles . . . . .	13
4.0.3 Convex_Hull . . . . .	14
4.0.4 Intersection_Points . . . . .	15
4.0.5 Lines_And_Questions_I . . . . .	16
4.0.6 Lines_And_Questions_II . . . . .	17
4.0.7 Line_Segment_Intersection . . . . .	18
4.0.8 Line_Segments_Trace_II . . . . .	20
4.0.9 Line_Segment_Trace_I . . . . .	21
4.0.10 Maximum_Manhattan_Distances . . . . .	21
4.0.11 Minimum_Euclidean_Distance . . . . .	22
4.0.12 Point_in_Polygon . . . . .	22
4.0.13 Point_Location_Test . . . . .	24
4.0.14 Polygon_Area . . . . .	25
4.0.15 Polygon_Lattice_Points . . . . .	26

<b>5 Graph Algorithms</b>	<b>27</b>	8.0.6 Missing_Coin_Sum_Questions . . . . .	52
5.0.1 Cycle_Finding . . . . .	27	8.0.7 Pizzeria_Questions . . . . .	53
5.0.2 Flight_Routes . . . . .	28	8.0.8 Polynomial_Querries . . . . .	54
5.0.3 Game_Levels . . . . .	28	8.0.9 Prefix_Sum_Questions . . . . .	56
5.0.4 High_Score . . . . .	29	8.0.10 Range_Interval_Questions . . . . .	57
5.0.5 Longest_Flight_Route . . . . .	30	8.0.11 Range_Questions_and_Copies . . . . .	57
5.0.6 Monsters . . . . .	30	8.0.12 Range_Update_Questions . . . . .	58
5.0.7 Round_Trip . . . . .	32	8.0.13 Salary_Questions . . . . .	59
5.0.8 Round_Trip_II . . . . .	33	8.0.14 Subarray_Sum_Questions . . . . .	60
5.0.9 Shortest_Routes . . . . .	34	8.0.15 Subarray_Sum_Questions_II . . . . .	61
5.0.10 Shortest_Routes_II . . . . .	34		
<b>6 Introductory Problems</b>	<b>35</b>	<b>9 Sliding Window</b>	<b>62</b>
6.0.1 Apple_Division . . . . .	35	9.0.1 Sliding_Window_Median . . . . .	62
6.0.2 Bit.Strings . . . . .	35		
6.0.3 Coin_Piles . . . . .	36		
6.0.4 Creating.Strings . . . . .	36		
6.0.5 Gray_Code . . . . .	36		
6.0.6 Increasing_Array . . . . .	37		
6.0.7 Missing_Number . . . . .	37		
6.0.8 Palindrome_Reorder . . . . .	38		
6.0.9 Permutation . . . . .	38		
6.0.10 Repetitions . . . . .	38		
6.0.11 String_Reorder . . . . .	39		
6.0.12 Trailing_Zeros . . . . .	39		
6.0.13 Two_Knights . . . . .	40		
6.0.14 Two_Sets . . . . .	40		
6.0.15 Weird_Algorithm . . . . .	40		
<b>7 Mathematics</b>	<b>41</b>	<b>10 Sorting and Searching</b>	<b>63</b>
7.0.1 Bracket_Sequences_I . . . . .	41	10.0.1 Array_Division . . . . .	63
7.0.2 Christmas_Party . . . . .	41	10.0.2 Collecting_Numbers . . . . .	63
7.0.3 Common_Divisors . . . . .	42	10.0.3 Collecting_Numbers_II . . . . .	64
7.0.4 Distribuiting_Apples . . . . .	42	10.0.4 Concert_Tickets . . . . .	64
7.0.5 Divisor_Analysis . . . . .	43	10.0.5 Distinct_Numbers . . . . .	65
7.0.6 Exponentiation_II . . . . .	44	10.0.6 Distinct_Values_Subarray . . . . .	65
7.0.7 Next_Prime . . . . .	44	10.0.7 Distinct_Values_Subarrays_II . . . . .	65
7.0.8 Sum_Of_Divisors . . . . .	44	10.0.8 Distinct_Values_Subsequences . . . . .	66
<b>8 Range Queries</b>	<b>45</b>	10.0.9 Factory_Machines . . . . .	66
8.0.1 Distinct_Values_Questions . . . . .	45	10.0.10 Factory_Machines . . . . .	67
8.0.2 Distinct_Values_Questions_II . . . . .	45	10.0.11 Ferris_Wheel . . . . .	67
8.0.3 Forest_Questions . . . . .	49	10.0.12 Josephus_Problem_I . . . . .	67
8.0.4 Hotel_Questions . . . . .	50	10.0.13 Josephus_Problem_II . . . . .	68
8.0.5 List_Removals . . . . .	51	10.0.14 Maximum_Subarray_Sum . . . . .	68
		10.0.15 Maximum_Subarray_Sum_II . . . . .	69
		10.0.16 Missing_Coin_Sum . . . . .	69
		10.0.17 Movie_Festival . . . . .	69
		10.0.18 Movie_Festival_II . . . . .	70
		10.0.19 Nearest_Smaller_Values . . . . .	71
		10.0.20 Nested_Ranges_Check . . . . .	71
		10.0.21 Nested_Ranges_Count . . . . .	72
		10.0.22 Playlist . . . . .	73
		10.0.23 Reading_Books . . . . .	73
		10.0.24 Restaurant_Customers . . . . .	73
		10.0.25 Room_Allocation . . . . .	74
		10.0.26 Stick_Lengths . . . . .	74
		10.0.27 Subarray_Divisibility . . . . .	75
		10.0.28 Subarray_Sums_I . . . . .	75
		10.0.29 Subarray_Sums_II . . . . .	76
		10.0.30 Sum_of_Four_Values . . . . .	76

10.0.31	Sum_of_Three_Values	.....
10.0.32	Sum_of_Two_Values	.....
10.0.33	Tasks_and_Deadlines	.....
10.0.34	Towers	.....
10.0.35	Traffic_Lights	.....

## 11 String Algorithms

11.0.1	Finding_Borders	.....
11.0.2	Longest_Palindrome	.....
11.0.3	Palindrome_Questions_BIT	.....
11.0.4	Palindrome_Questions	.....

## 12 Tree Algorithms

12.0.1	Distinct_Colors	.....
--------	-----------------	-------

# 1 Additional Problems II

## 1.0.1 GCD\_Subsets

---

```

77 // You are given an array of n integers. Your task is to calculate the
78 // number of non-empty subsets whose elements' greatest common divisor is
79 // equal to k for each k = 1, \dots, n.
79 #include <bits/stdc++.h>
79 using namespace std;
80
80 #define MOD 10000000007
80 #define ll long long
81
81 int main() {
84     ios::sync_with_stdio(false);
84     cin.tie(nullptr);
85
85     int n;
86     cin >> n;
87     vector<int> a(n);
88     for (int& x : a) cin >> x;
89
90     // Paso 1: contar cuántos elementos son múltiplos de cada k
91     vector<int> freq(n + 1, 0);
92     for (int x : a) freq[x]++;
93
94     // cnt[k] = cantidad de elementos que son múltiplos de k
95     vector<int> cnt(n + 1, 0);
96     for (int k = 1; k <= n; k++) {
97         for (int mult = k; mult <= n; mult += k) {
98             cnt[k] += freq[mult];
99         }
100    }
101
102    // Precomputar potencias de 2: pow2[i] = 2^i % MOD
103    vector<ll> pow2(n + 1, 1);
104    for (int i = 1; i <= n; i++) {
105        pow2[i] = (pow2[i - 1] * 2) % MOD;
106    }
107
108    // Paso 2 y 3: inclusion-exclusion
109    vector<ll> ans(n + 1, 0);
110    for (int k = n; k >= 1; k--) {
111        ans[k] = (pow2[cnt[k]] - 1 + MOD) % MOD;
112        // Restar contribuciones de múltiplos de k mayores
113        for (int mult = 2 * k; mult <= n; mult += k) {
114            ans[k] = (ans[k] - ans[mult] + MOD) % MOD;
115        }
116    }
117
118    // Imprimir respuestas para k = 1 hasta n
119    for (int k = 1; k <= n; k++) {
120        cout << ans[k] << " ";
121    }
122    cout << "\n";
123
124    return 0;

```

## 2 Bitwise Operations

### 2.0.1 All\_Subarray\_Xors

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 using ull = unsigned long long;
4 void fwht(vector<ull> &a, bool inv) {
5     int n = a.size();
6     for (int len = 1; len < n; len <<= 1) {
7         for (int i = 0; i < n; i += len << 1) {
8             for (int j = 0; j < len; ++j) {
9                 ull u = a[i + j];
10                ull v = a[i + j + len];
11                a[i + j] = u + v;
12                a[i + j + len] = u - v;
13            }
14        }
15    }
16    if (inv) {
17        for (ull &x : a) x /= n;
18    }
19 }
20 signed main() {
21     ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
22     int n;
23     cin >> n;
24     vector<int> a(n);
25     for (int &x : a) cin >> x;
26     // prefijos del xor para los rangos
27     vector<int> p(n + 1, 0);
28     for (int i = 0; i < n; ++i)
29         p[i + 1] = p[i] ^ a[i];
30     // cosas de la fwht
31     int maxVal = *max_element(p.begin(), p.end());
32     int B = 0;
33     while ((1 << B) <= maxVal) ++B;
34     int M = 1 << B;
35     // frecuencias
36     vector<ull> A(M, 0);
37     map<int, int> freq;
38     bool hasZero = false;
39     for (int x : p) {
40         A[x]++;
41         if (++freq[x] >= 2) hasZero = 1;
42     }
43     fwht(A, false);
44     for (int i = 0; i < M; ++i)
45         A[i] = A[i] * A[i];
46     fwht(A, true);
47     // los xor posibles
48     vector<int> res;
49     res.reserve(M);
50     for (int i = 0; i < M; ++i) {

```

```

51         if (A[i] > 0) {
52             if (i == 0 && !hasZero) continue;
53             // i ya es tu respuesta
54             res.push_back(i);
55         }
56     }
57     cout << (int)res.size() << '\n';
58     for (int x : res) cout << x << ' ';
59 }

```

### 2.0.2 Counting\_Bits

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define int long long
4 #define ll long long
5 const int tam = 1;
6
7 int cantbits(int n){
8     if(n == 0) return 0;
9     else{
10         int k = log2(n);
11         int ans = k*(1LL<<(k-1)) + (n - (1LL<<k) + 1);
12         ans+=cantbits(n - (1LL<<k));
13         return ans;
14     }
15 }
16
17 void solve(){
18     int n;cin>>n;
19     cout << cantbits(n) << endl;
20 }
21
22 signed main(){
23     ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
24     int t = 1;
25     // cin >> t;
26     while(t--){
27         solve();
28     }
29 }

```

### 2.0.3 K\_Subset\_Xors

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define ll long long
4 #define ull unsigned long long
5 #define fore(i, a, b) for (int i = (a); i < (b); i++)
6 #define FOR(i, n) for (int i = 0; i < (n); i++)
7 #define all(x) (x).begin(), (x).end()
8 #define sz(x) (int)(x).size()
9 #define pb push_back

```

```

10  using vi = vector<int>;
11  const int MOD = 1000000007;
12  const ll INF = 9223372036854775807LL;
13  const int tam = 31;
14  int basis[tam] = {0};
15  void insert(int x) {
16      for (int i = tam - 1; i >= 0; i--) {
17          if (x & (1 << i)) {
18              if (!basis[i]) {
19                  basis[i] = x;
20                  return;
21              }
22              x ^= basis[i];
23          }
24      }
25  }
26
27  void solve() {
28      //freopen("input.txt", "r", stdin);
29      int n, k;
30      cin >> n >> k;
31      vector<int> arr(n);
32      fore(i, 0, n) cin >> arr[i], insert(arr[i]);
33      vector<int> base;
34      fore(i, 0, tam)
35          if (basis[i])
36              base.push_back(basis[i]);
37      int sz = sz(base);
38      int mul = 1<<(min(n-sz, 30));
39      int comb = 1<<sz;
40      int unitam = min({k/mul + (k%mul>0), 200000, comb});
41      vector<int> unicos;
42      for(int mask = 0; mask<min(comb, 2*unitam); mask++){
43          int actual = 0;
44          for(int i = 0; i<sz; i++){
45              if(mask&(1<<i)){
46                  actual^=base[i];
47              }
48          }
49          unicos.push_back(actual);
50      }
51      sort(all(unicos));
52      int impresos = 0;
53      for(int i = 0; i<sz(unicos) and impresos < k; i++){
54          for(int j = 0; j<mul and impresos < k; j++,
55              → impresos++) cout<<unicos[i]<<' ';
56      }
57      cout<<'\n';
58  }
59
60  signed main() {
61      ios::sync_with_stdio(0);
62      cin.tie(0);
63      cout.tie(0);
64      int t = 1;
65      //cin >> t;
66      while (t--) { solve(); }
67  }

```

66 }

## 2.0.4 Maximum\_Xor\_Subarray

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  #define ll long long
4  #define ull unsigned long long
5  #define fore(i, a, b) for(int i = (a); i<(b); i++)
6  #define FOR(i, n) for(int i = 0; i<(n); i++)
7  #define all(x) (x).begin(), (x).end()
8  #define sz(x) (int)(x).size()
9  #define pb push_back
10 using vi = vector<int>;
11 const int MOD = 1000000007;
12 const ll INF = 9223372036854775807LL;
13 const int tam = 1;
14
15 struct nodo {
16     nodo* hijos[2];
17     nodo() {
18         hijos[0] = nullptr;
19         hijos[1] = nullptr;
20     }
21 };
22 nodo* raiz;
23
24 void insertar(int x) {
25     nodo* it = raiz;
26     for (int j = 30; j >= 0; j--) {
27         int bit = (x & (1 << j)) != 0;
28         if (!it->hijos[bit]) { it->hijos[bit] = new nodo(); }
29         it = it->hijos[bit];
30     }
31 }
32
33 int buscar(int x) {
34     nodo* it = raiz;
35     int ans = 0;
36     for (int j = 30; j >= 0; j--) {
37         int bit = (x & (1 << j)) != 0;
38         if (it->hijos[1 - bit]) {
39             ans |= (1 << j);
40             it = it->hijos[1 - bit];
41         } else {
42             it =
43                 it->hijos[bit];
44         }
45     }
46     return ans;
47 }
48
49 void solve() {
50     int n;
51

```

```

52     cin >> n;
53     raiz = new nodo();
54     vector<int> prefix(n + 1, 0);
55     vector<int> arr(n);
56     for (int i = 0; i < n; i++) {
57         cin >> arr[i];
58         prefix[i + 1] = prefix[i] ^ arr[i];
59     }
60     insertar(0);
61     int ans = 0;
62     for (int i = 1; i <= n; i++) {
63         ans = max(ans, buscar(prefix[i]));
64         insertar(prefix[i]);
65     }
66     cout << ans << endl;
67 }
68
69 signed main(){
70     ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
71     int t = 1;
72     //cin>>t;
73     while(t--){
74         solve();
75     }
76 }
```

## 2.0.5 Maximum\_Xor\_Subset

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define ll long long
4 #define ull unsigned long long
5 #define fore(i, a, b) for (int i = (a); i < (b); i++)
6 #define FOR(i, n) for (int i = 0; i < (n); i++)
7 #define all(x) (x).begin(), (x).end()
8 #define sz(x) (int)(x).size()
9 #define pb push_back
10 using vi = vector<int>;
11 const int MOD = 1000000007;
12 const ll INF = 9223372036854775807LL;
13 const int tam = 32;
14 int basis[tam];
15 void insert(int x) {
16     for (int i = tam - 1; i >= 0; i--) {
17         if (x & (1 << i)) {
18             if (!basis[i]) {
19                 basis[i] = x;
20                 return;
21             }
22             x ^= basis[i];
23         }
24     }
25 }
26 void solve() {
27     int n;
```

```

28     cin >> n;
29     for (int i = 0; i < n; i++) {
30         int x;
31         cin >> x;
32         insert(x);
33     }
34     int ans = 0;
35     for(int i = tam-1; i>=0; i--){
36         ans = max(ans, ans^basis[i]);
37     }
38     cout<<ans<<endl;
39 }
40
41 signed main() {
42     ios::sync_with_stdio(0);
43     cin.tie(0);
44     cout.tie(0);
45     int t = 1;
46     //cin >> t;
47     while (t--) { solve(); }
48 }
```

## 2.0.6 Number\_of\_Subset\_Xors

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define ll long long
4 #define ull unsigned long long
5 #define fore(i, a, b) for (int i = (a); i < (b); i++)
6 #define FOR(i, n) for (int i = 0; i < (n); i++)
7 #define all(x) (x).begin(), (x).end()
8 #define sz(x) (int)(x).size()
9 #define pb push_back
10 using vi = vector<int>;
11 const int MOD = 1000000007;
12 const ll INF = 9223372036854775807LL;
13 const int tam = 32;
14 int basis[tam];
15 void insert(int x) {
16     for (int i = tam - 1; i >= 0; i--) {
17         if (x & (1 << i)) {
18             if (!basis[i]) {
19                 basis[i] = x;
20                 return;
21             }
22             x ^= basis[i];
23         }
24     }
25 }
26 void solve() {
27     int n;
28     cin >> n;
29     for (int i = 0; i < n; i++) {
30         int x;
31         cin >> x;
```

```

32     insert(x);
33 }
34 int ans = 0;
35 for(int i = tam-1; i>=0; i--){
36     ans+=(basis[i] != 0);
37 }
38 cout<<(1<<ans)<<endl;
39 }
40
41 signed main() {
42     ios::sync_with_stdio(0);
43     cin.tie(0);
44     cout.tie(0);
45     int t = 1;
46     //cin >> t;
47     while (t--) { solve(); }
48 }
```

## 2.0.7 Xor\_Pyramid\_Peak

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 using ll = long long;
4
5 int main(){
6     ios::sync_with_stdio(false);
7     cin.tie(nullptr);
8
9     int n;
10    cin >> n;
11    vector<ll> a(n);
12    for(int i = 0; i < n; i++){
13        cin >> a[i];
14    }
15
16    ll ans = 0;
17    int N = n - 1;
18    for(int i = 0; i < n; i++){
19        // C(n-1, i) es impar  $\iff$   $(i \& (n-1)) = i$ 
20        if ((i & N) == i) {
21            ans ^= a[i];
22        }
23    }
24
25    cout << ans << "\n";
26    return 0;
27 }
```

```

2 // You have decided that the total price of your purchases will be at most
3 // x. What is the maximum number of pages you can buy? You can buy each
4 // book at most once.
5 #include <bits/stdc++.h>
6 using namespace std;
7 //##define int long long
8 #define f first
9 #define s second
10 #define pb push_back
11 #define sz(x) (int)(x).size()
12 #define all(a) (a).begin(), (a).end()
13 #define rall(a) (a).rbegin(), (a).rend()
14 #define fore(i, a, n) for (int i = (a); i < (n); i++)
15 #define forb(i, n) for (int i = (n) - 1; i ≥ 0; i--)
16 #define popcount(x) __builtin_popcountll(x);
17 typedef pair<int, int> pii;
18 typedef vector<int> vi;
19 const int MOD = 1000000007;
20 const double EPS = 1e-9;
21 const double PI = acos(-1);
22 const int INF = 1e18;
23 // PLUS ULTRA RECARGADO!!!
24 void solve() {
25     int n, dinero;
26     cin >> n >> dinero;
27     int precio[n], pag[n];
28     fore(i, 0, n) cin >> precio[i];
29     fore(i, 0, n) cin >> pag[i];
30     vector<vi> dp(n+1, vi(dinero+1, 0));
31     for (int i = n - 1; i ≥ 0; i--) {
32         for (int d = 0; d ≤ dinero; d++) {
33             int notomar = dp[i+1][d];
34             int tomar = 0;
35             if(d - precio[i]>=0){
36                 tomar = dp[i+1][d - precio[i]] + pag[i];
37             }
38             dp[i][d] = max(tomar, notomar);
39         }
40     }
41     cout << dp[0][dinero] << endl;
42 }
43 signed main() {
44     ios::sync_with_stdio(0);
45     cin.tie(0);
46     cout.tie(0);
47     // int t; cin >> t; while(t--) solve();
48 }
49 }
```

## 3 Dynamic Programming

### 3.0.1 Book\_Shop

---

1 // You are in a book shop which sells n different books. You know the  
→ price and number of pages of each book.

### 3.0.2 Coin\_Combinations

```

1 // Consider a money system consisting of n coins. Each coin has a positive
2 → integer value. Your task is to calculate the number of distinct ways
3 → you can produce a money sum x using the available coins.
4
5 // For example, if the coins are \{2,3,5\} and the desired sum is 9, there
6 → are 8
7
8 #include <bits/stdc++.h>
9 using namespace std;
10 #define int long long
11 #define f first
12 #define s second
13 #define pb push_back
14 #define sz(x) (int)(x).size()
15 #define all(a) (a).begin(), (a).end()
16 #define rall(a) (a).rbegin(), (a).rend()
17 #define fore(i, a, n) for(int i = (a); i < (n); i++)
18 #define forb(i, n) for (int i = (n) - 1; i ≥ 0; i--)
19 #define popcorn(x) __builtin_popcountll(x);
20
21 typedef pair<int, int> pii;
22 typedef vector<int> vi;
23 const int MOD = 10000000007;
24 const double EPS = 1e-9;
25 const double PI = acos(-1);
26 const int INF = 1e18;
27 //PLUS ULTRA RECARGADO!!!
28 void solve() {
29     int n, x;cin>>n>>x;
30     vi monedas(n);
31     fore(i,0,n)cin>>monedas[i];
32     vi memo(x+1, 0);
33     memo[0] = 1;
34     fore(i, 1, x+1){
35         for(int moneda:monedas){
36             if(i - moneda≥0){
37                 memo[i]+=memo[i-moneda];
38                 memo[i]%=MOD;
39             }
40         }
41     }
42     cout<<memo[x]<<endl;
43 }
44 signed main() {
45     ios::sync_with_stdio(0);
46     cin.tie(0);
47     cout.tie(0);
48     // int t;cin>>t;while(t--)solve();
49     solve();
50 }
```

### 3.0.3 Coin\_Combinations\_II

---

// 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

```

1 // Consider a money system consisting of n coins. Each coin has a positive
2 → integer value. Your task is to calculate the number of distinct
3 → ordered ways you can produce a money sum x using the available coins.
4
5 // For example, if the coins are \{2,3,5\} and the desired sum is 9, there
6 → are 3
7
8 #include <bits/stdc++.h>
9 using namespace std;
10 //##pragma optimize("Ofast")
11 //##define int long long
12 #define f first
13 #define s second
14 #define pb push_back
15 #define sz(x) (int)(x).size()
16 #define all(a) (a).begin(), (a).end()
17 #define rall(a) (a).rbegin(), (a).rend()
18 #define fore(i, a, n) for(int i = (a); i < (n); i++)
19 #define forb(i, n) for (int i = (n) - 1; i ≥ 0; i--)
20 #define popcorn(x) __builtin_popcountll(x);
21
22 typedef pair<int, int> pii;
23 typedef vector<int> vi;
24 const int MOD = 10000000007;
25 const double EPS = 1e-9;
26 const double PI = acos(-1);
27 //PLUS ULTRA RECARGADO!!!
28 void solve() {
29     int n, x;cin>>n>>x;
30     vi monedas(n);
31     fore(i, 0, n)cin>>monedas[i];
32     vector<vi> dp(n+1, vi(x+1));
33     for(int i = 0; i≤n; i++){
34         dp[i][0] = 1;
35     }
36     for(int i = n-1; i≥0; i--){
37         for(int suma = 1; suma≤x; suma++){
38             int notomar = dp[i+1][suma];
39             int tomar = 0;
40             if(suma - monedas[i]≥0){
41                 tomar = dp[i][suma - monedas[i]];
42             }
43             dp[i][suma] = tomar + notomar;
44             dp[i][suma]%=MOD;
45         }
46     }
47     cout<<dp[0][x]<<endl;
48 }
49
50 signed main() {
51     ios::sync_with_stdio(0);
52     cin.tie(0);
53     cout.tie(0);
54     solve();
55 }
```

---

### 3.0.4 Dice\_Combinations

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 #define ll long long
4 #define ull unsigned long long
5 #define fore(i, a, b) for(int i = (a); i<(b); i++)
6 #define FOR(i, n) for(int i = 0; i<(n); i++)
7 #define all(x) (x).begin(), (x).end()
8 #define sz(x) (int)(x).size()
9 #define pb push_back
10 using vi = vector<int>;
11 const int MOD = 1000000007;
12 const ll INF = 9223372036854775807LL;
13 const int tam = 1000000+5;
14
15 void solve(){
16     int x;cin>>x;
17     int memo[tam];
18     memset(memo, 0, sizeof memo);
19     memo[0] = 1;
20     for(int i = 1; i<=x; i++){
21         ll ans = 0;
22         for(int m = 1; m<=i;m++){
23             if(i-m>=0){
24                 ans+=memo[i-m];
25                 ans%=MOD;
26             }
27         }
28         memo[i] = ans;
29     }
30     cout<<memo[x]<<endl;
31 }
32
33 signed main(){
34     ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
35     int t = 1;
36     //cin>>t;
37     while(t--){
38         solve();
39     }
40 }
```

### 3.0.5 Edit\_Distance

```
1 // The edit distance between two strings is the minimum number of
2 // operations required to transform one string into the other.
3 // The allowed operations are:
4 // Add one character to the string.
5 // Remove one character from the string.
6 // Replace one character in the string.
7 // For example, the edit distance between LOVE and MOVIE is 2, because you
8 // can first replace L with M, and then add I.
9 // Your task is to calculate the edit distance between two strings.
```

```
8 #include <bits/stdc++.h>
9 using namespace std;
10 //#define int long long
11 #define f first
12 #define sst stringstream
13 #define s second
14 #define pb push_back
15 #define sz(x) (int)(x).size()
16 #define all(a) (a).begin(), (a).end()
17 #define rall(a) (a).rbegin(), (a).rend()
18 #define fore(i, a, n) for(int i = (a); i < (n); i++)
19 #define forb(i, n) for (int i = (n) - 1; i ≥ 0; i--)
20 #define popcorn(x) __builtin_popcountll(x);
21 typedef pair<int, int> pii;
22 typedef vector<int> vi;
23 const int MOD = 1000000007;
24 const double EPS = 1e-9;
25 const double PI = acos(-1);
26 const int INF = 1e18;
27 //PLUS ULTRA RECARGADO!!!
28 string s, t;
29 int memo[5011][5011];
30
31 int dp(int i, int j) {
32     if (i < 0) return j + 1;
33     if (j < 0) return i + 1;
34     if (memo[i][j] != -1) return memo[i][j];
35
36     if (s[i] == t[j]) {
37         memo[i][j] = dp(i - 1, j - 1);
38     } else {
39         memo[i][j] = min({
40             dp(i - 1, j) + 1,
41             dp(i, j - 1) + 1,
42             dp(i - 1, j - 1) + 1
43         });
44     }
45     return memo[i][j];
46 }
47
48 void solve() {
49     cin >> s >> t;
50     memset(memo, -1, sizeof(memo));
51     int ns = sz(s), nt = sz(t);
52     cout << dp(ns - 1, nt - 1) << endl;
53 }
54
55
56 signed main() {
57     ios::sync_with_stdio(0);
58     cin.tie(0);
59     cout.tie(0);
60     //int t;cin>>t;while(t--)solve();
61     solve();
62 }
```

### 3.0.6 Grid\_Paths\_I

55 }

---

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define int long long
4 #define f first
5 #define sst stringstream
6 #define s second
7 #define pb push_back
8 #define sz(x) (int)(x).size()
9 #define all(a) (a).begin(), (a).end()
10 #define rall(a) (a).rbegin(), (a).rend()
11 #define fore(i, a, n) for (int i = (a); i < (n); i++)
12 #define forb(i, n) for (int i = (n) - 1; i ≥ 0; i--)
13 #define popcorn(x) __builtin_popcountll(x);
14 typedef pair<int, int> pii;
15 typedef vector<int> vi;
16 const int MOD = 1000000007;
17 const double EPS = 1e-9;
18 const double PI = acos(-1);
19 const int INF = 1e18;
20 // PLUS ULTRA RECARGADO!!!
21 void solve() {
22     int n;
23     cin >> n;
24     char mat[n][n];
25     fore(i, 0, n) {
26         fore(j, 0, n) {
27             cin >> mat[i][j];
28         }
29     }
30     int dp[n][n];
31     memset(dp, 0, sizeof dp);
32     if (mat[n - 1][n - 1] != '*') dp[n - 1][n - 1] = 1;
33     for (int i = n - 1; i ≥ 0; i--) {
34         for (int j = n - 1; j ≥ 0; j--) {
35             if (mat[i][j] == '*') {
36                 dp[i][j] = 0;
37                 continue;
38             }
39             if (j + 1 < n) dp[i][j] += dp[i][j + 1];
40             if (i + 1 < n) dp[i][j] += dp[i + 1][j];
41             dp[i][j] %= MOD;
42         }
43     }
44     cout << dp[0][0] << endl;
45 }
46 signed main() {
47     ios::sync_with_stdio(0);
48     cin.tie(0);
49     cout.tie(0);
50     int t;
51     // cin >> t;
52     // while (t--) solve();
53     solve();
54 }
```

### 3.0.7 Longest\_Common\_Subsequence

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define ll long long
4 #define ull unsigned long long
5 #define fore(i, a, b) for(int i = (a); i < (b); i++)
6 #define FOR(i, n) for(int i = 0; i < (n); i++)
7 #define all(x) (x).begin(), (x).end()
8 #define sz(x) (int)(x).size()
9 #define pb push_back
10 using vi = vector<int>;
11 const int MOD = 1000000007;
12 const ll INF = 9223372036854775807LL;
13 const int tam = 1005;
14 // const int n = 1, m = 1;
15 // int dp(int i, int j){
16 //     if(i ≥ n){
17 //         return 0;
18 //     }
19 //     if(j ≥ m){
20 //         return 0;
21 //     }
22 //     if(memo[i][j] == -1){
23 //         int a = 0, b, c;
24 //         if(s[i] == t[j]){
25 //             a = dp(i+1, j+1) + 1;
26 //         }
27 //         b = dp(i+1, j);
28 //         c = dp(i, j+1);
29 //         memo[i][j] = max({a, b, c});
30 //     }
31 //     return memo[i][j];
32 }
33 void solve(){
34     int n, m; cin >> n >> m;
35     int dp[tam][tam];
36     memset(dp, 0, sizeof dp);
37     int s[n], t[m];
38     fore(i, 0, n) cin >> s[i];
39     fore(i, 0, m) cin >> t[i];
40     for(int i = n-1; i ≥ 0; i--){
41         for(int j = m-1; j ≥ 0; j--){
42             int a = 0, b, c;
43             if(s[i] == t[j]){
44                 a = dp[i+1][j+1] + 1;
45             }
46             b = dp[i+1][j];
47             c = dp[i][j+1];
48             dp[i][j] = max({a, b, c});
49         }
50     }
51 }
```

```

52 }
53 cout<<dp[0][0]<<endl;
54 int i = 0, j = 0;
55 vector<int> ans;
56 while(i<n and j<m){
57     if(s[i] == t[j]){
58         ans.push_back(s[i]);
59         ++i, ++j;
60     }else if(dp[i+1][j] >= dp[i][j+1]){
61         ++i;
62     }else{
63         ++j;
64     }
65 }
66 for(int x:ans)cout<<x<<' ';
67 cout<<'\n';
68 }

69 }

70 signed main(){
71     ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
72     int t = 1;
73     // cin>>t;
74     while(t--){
75         solve();
76     }
77 }

```

---

### 3.0.8 Longest\_Common\_Subsequences

```

1 // Given two arrays of integers, find their longest common subsequence. 57
2 // A subsequence is a sequence of array elements from left to right that 58
3 // can contain gaps. A common subsequence is a subsequence that appears 59
4 // in both arrays.
5 #include <bits/stdc++.h>
6 using namespace std;
7 #define ll long long
8 #define ull unsigned long long
9 #define fore(i, a, b) for(int i = (a); i<(b); i++)
10 #define FOR(i, n) for(int i = 0; i<(n); i++)
11 #define all(x) (x).begin(), (x).end()
12 #define sz(x) (int)(x).size()
13 #define pb push_back
14 using vi = vector<int>;
15 const int MOD = 1000000007;
16 const ll INF = 9223372036854775807LL;
17 const int tam = 1005;
18 void solve(){
19     int n, m;cin>>n>>m;
20     int dp[tam][tam];
21     memset(dp, 0, sizeof dp);
22     int s[n], t[m];
23     fore(i, 0, n)cin>>s[i];
24     fore(i, 0, m)cin>>t[i];
25     for(int i = n-1; i>=0; i--) {
26         for(int j = m-1; j>=0; j--) {
27             int a = 0, b, c;
28             if(s[i] == t[j]){
29                 a = dp[i+1][j+1] + 1;
30             }
31             b = dp[i+1][j];
32             c = dp[i][j+1];
33             dp[i][j] = max({a, b, c});
34         }
35         cout<<dp[0][0]<<endl;
36         int i = 0, j = 0;
37         vector<int> ans;
38         while(i<n and j<m){
39             if(s[i] == t[j]){
40                 ans.push_back(s[i]);
41                 ++i, ++j;
42             }else if(dp[i+1][j] >= dp[i][j+1]){
43                 ++i;
44             }else{
45                 ++j;
46             }
47         }
48         for(int x:ans)cout<<x<<' ';
49         cout<<'\n';
50     }
51 }

```

---

```

52 signed main(){
53     ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
54     int t = 1;
55     // cin>>t;
56     while(t--){
57         solve();
58     }
59 }


```

### 3.0.9 Minimizing\_Coins

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define int long long
4 #define f first
5 #define sst stringstream
6 #define s second
7 #define pb push_back
8 #define sz(x) (int)(x).size()
9 #define all(a) (a).begin(), (a).end()
10 #define rall(a) (a).rbegin(), (a).rend()
11 #define fore(i, a, n) for(int i = (a); i < (n); i++)
12 #define forb(i, n) for (int i = (n) - 1; i ≥ 0; i--)
13 #define popcorn(x) __builtin_popcountll(x);
14 typedef pair<int, int> pii;
15 typedef vector<int> vi;
16 const int MOD = 1000000007;

```

```

17 const double EPS = 1e-9;
18 const double PI = acos(-1);
19 const int INF = 1e18;
20 // Consider a money system consisting of n coins. Each coin has a positive
21 // integer value. Your task is to produce a sum of money x using the
22 // available coins in such a way that the number of coins is minimal.
23 // For example, if the coins are \{1,5,7\} and the desired sum is 11, an
24 // optimal solution is 5+5+1 which requires 3 coins.
25
26 void solve() {
27     int n, x; cin >> n >> x;
28     vi monedas(n);
29     fore(i, 0, n) cin >> monedas[i];
30     vi dp(x+1, INF);
31     dp[0] = 0;
32     int ans = 0;
33     fore(i, 1, x+1) {
34         for(int moneda:monedas) {
35             if(i-moneda >= 0) {
36                 dp[i] = min(dp[i], dp[i-moneda]+1);
37             }
38         }
39     }
40     cout << (dp[x] == INF?-1:dp[x]) << endl;
41 }
42
43 signed main() {
44     ios::sync_with_stdio(0);
45     cin.tie(0);
46     cout.tie(0);
47     // int t; cin >> t; while(t--) solve();
48     solve();
49 }
```

### 3.0.10 Money Sums

```

1 #include <cstring>
2 #include <iostream>
3 #include <vector>
4 using namespace std;
5 // You have n coins with certain values. Your task is to find all money
6 // sums you can create using these coins.
7 void solve() {
8     int n;
9     cin >> n;
10    vector<int> monedas(n);
11    int maxSuma = 0;
12    for (int i = 0; i < n; i++) {
13        cin >> monedas[i];
14        maxSuma += monedas[i];
15    }
16    vector<bool> dp(maxSuma + 1, false);
17    dp[0] = true;
18
19    for (int coin : monedas) {
```

```

20        for (int s = maxSuma; s >= coin; s--) {
21            if (dp[s - coin]) dp[s] = true;
22        }
23
24        int count = 0;
25        for (int s = 1; s <= maxSuma; s++) {
26            if (dp[s]) count++;
27        }
28        cout << count << "\n";
29        for (int s = 1; s <= maxSuma; s++) {
30            if (dp[s]) cout << s << " ";
31        }
32        cout << "\n";
33    }
34
35    int main() {
36        ios::sync_with_stdio(false);
37        cin.tie(nullptr);
38
39        int t = 1;
40        //cin >> t;
41        while (t--) { solve(); }
42        return 0;
43    }
44 }
```

### 3.0.11 Rectangle\_Cutting

---

```

1 // Given an a \times b rectangle, your task is to cut it into squares. On
2 // each move you can select a rectangle and cut it into two rectangles in
3 // such a way that all side lengths remain integers. What is the minimum
4 // possible number of moves?
5 #include <bits/stdc++.h>
6 using namespace std;
7 #define ll long long
8 #define ull unsigned long long
9 #define fore(i, a, b) for(int i = (a); i < (b); i++)
10 #define FOR(i, n) for(int i = 0; i < (n); i++)
11 #define all(x) (x).begin(), (x).end()
12 #define sz(x) (int)(x).size()
13 using vi = vector<int>;
14 const int MOD = 1000000007;
15 const ll INF = 9223372036854775807LL;
16 int memo[501][501];
17 ll dp(int a, int b) {
18     if (a == b) return 0;
19     if (memo[a][b] != -1) return memo[a][b];
20     ll res = INF;
21     for (int i = 1; i < a; ++i) {
22         res = min(res, dp(i, b) + dp(a - i, b) + 1);
23     }
24     for (int i = 1; i < b; ++i) {
25         res = min(res, dp(a, i) + dp(a, b - i) + 1);
26     }
27     memo[a][b] = res;
28 }
```

```

25     return res;
26 }
27 void solve(){
28     int a, b; cin>>a>>b;
29     memset(memo, -1, sizeof memo);
30     cout<<dp(a, b)<<endl;
31 }
32
33 signed main(){
34     ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
35     solve();
36 }

```

### 3.0.12 Removing\_Digits

```

1 // You are given an integer n. On each step, you may subtract one of the
2 // digits from the number.
3 // How many steps are required to make the number equal to 0?
4 #include <bits/stdc++.h>
5 using namespace std;
6 #define int long long
7 #define f first
8 #define s second
9 #define pb push_back
10 #define sz(x) (int)(x).size()
11 #define all(a) (a).begin(), (a).end()
12 #define rall(a) (a).rbegin(), (a).rend()
13 #define fore(i, a, n) for(int i = (a); i < (n); i++)
14 #define forb(i, n) for(int i = (n) - 1; i ≥ 0; i--)
15 #define popcount(x) __builtin_popcountll(x);
16 typedef pair<int, int> pii;
17 typedef vector<int> vi;
18 const int MOD = 1000000007;
19 const double EPS = 1e-9;
20 const double PI = acos(-1);
21 const int INF = 1e18;
22 //PLUS ULTRA RECARGADO!!!
23 void solve() {
24     int n; cin>>n;
25     vi memo(n+1, INF);
26     memo[0] = 1;
27     for(int i = 1; i<=n; i++){
28         int num = i;
29         while(num){
30             if((i - num%10)>=0){
31                 memo[i] = min(memo[i], memo[i - num%10]+1);
32             }
33             num/=10;
34         }
35     }
36     cout<<memo[n]-1<<endl;
37 }
38 signed main() {
39     ios::sync_with_stdio(0);

```

```

40     cin.tie(0);
41     cout.tie(0);
42     // int t; cin>>t; while(t--) solve();
43     solve();
44 }

```

## 4 Geometry

### 4.0.1 All\_Manhattan\_DIstances

```

1 // Given a set of points, calculate the sum of all Manhattan distances
2 // between two point pairs.
3 import sys
4 d=sys.stdin.readlines()
5 n=int(d[0])
6 x=[0]*n
7 y=[0]*n
8 def f(a):
9     a.sort()
10    s=0
11    for i in range(n):
12        s+=a[i]*(2*i-n+1)
13    return s
14 for i in range(1, n+1):
15    x[i-1],y[i-1]=map(int,d[i].split())
16 print(f(x)+f(y))

```

### 4.0.2 Area\_of\_Rectangles

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define int ll
4 #define ll long long
5 #define ull unsigned ll
6 #define fore(i, a, b) for(int i = (a); i<(b); i++)
7 #define FOR(i, n) for(int i = 0; i<(n); i++)
8 #define all(x) (x).begin(), (x).end()
9 #define sz(x) (int)(x).size()
10 #define pb push_back
11 using vi = vector<int>;
12 const int MOD = 1000000007;
13 const ll INF = 9223372036854775807LL;
14 const int tam = 1;
15
16 struct Node {
17     unique_ptr<Node> left, right;
18     int count = 0;
19     ll length = 0;
20 };
21
22 void recalc(Node* node, ll l, ll r) {
23     if (node->count > 0) {
24         node->length = r - l; // todo cubierto

```

```

25 } else {
26     ll left_len = node->left ? node->left->length : 0;
27     ll right_len = node->right ? node->right->length : 0;
28     node->length = left_len + right_len;
29 }
30 }
31
32 void update(Node* node, ll l, ll r, ll y_l, ll y_r, int val) {
33     if (y_r <= l || r <= y_l) return;
34     if (y_l <= l && r <= y_r) {
35         node->count += val;
36     } else {
37         ll m = l + (r - l)/2;
38         if (!node->left) node->left = make_unique<Node>();
39         if (!node->right) node->right = make_unique<Node>();
40         update(node->left.get(), l, m, y_l, y_r, val);
41         update(node->right.get(), m, r, y_l, y_r, val);
42     }
43     recalc(node, l, r);
44 }
45
46 struct event{
47     int x, y1, y2, tipo; //apertura es +1, cierre es -1
48     event(int a = 0, int b = 0, int c = 0, int d = 1){
49         x = a, y1 = b, y2 = c, tipo = d;
50     }
51     bool operator<(const event& otro) const {
52         return x < otro.x || (x == otro.x && tipo > otro.tipo);
53     }
54 };
55 const int ymin = -1e6-10;
56 const int ymax = 1e6 + 10;
57 void solve(){
58     auto root = make_unique<Node>();
59     int n;cin>>n;
60     vector<event> eventos;
61     for(int i = 0; i<n; i++){
62         int a, b, c, d;cin>>a>>b>>c>>d;
63         eventos.push_back(event(a, b, d, 1));
64         eventos.push_back(event(c, b, d, -1));
65     }
66     sort(all(eventos));
67
68     int ans = 0;
69     int ulti = eventos[0].x;
70     for(auto e:eventos){
71         int dx = e.x - ulti;
72         ulti = e.x;
73         ans+=dx*root->length;
74         update(root.get(), ymin, ymax, e.y1, e.y2, e.tipo);
75     }
76     cout<<ans<<'\n';
77 }
78
79 signed main(){
80     ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
81     int t = 1;
// cin>>t;

```

```

83     while(t--){
84         solve();
85     }
86 }

```

#### 4.0.3 Convex\_Hull

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 struct Punto {
5     long long x, y;
6     bool operator<(const Punto& o) const {
7         return x < o.x || (x == o.x && y < o.y);
8     }
9 };
10
11 // Producto cruzado (para ver si el giro es izquierda/derecha)
12 long long cruz(const Punto& O, const Punto& A, const Punto& B) {
13     return (A.x - O.x) * (B.y - O.y) -
14             (A.y - O.y) * (B.x - O.x);
15 }
16
17 vector<Punto> convexHull(vector<Punto>& pts) {
18     int n = pts.size(), k = 0;
19     if (n <= 1) return pts;
20     sort(pts.begin(), pts.end());
21
22     vector<Punto> hull(2*n);
23
24     // Construir lower hull
25     for (int i = 0; i < n; i++) {
26         while (k >= 2 && cruz(hull[k-2], hull[k-1], pts[i]) <= 0) k--;
27         hull[k++] = pts[i];
28     }
29
30     // Construir upper hull
31     for (int i = n-2, t = k+1; i >= 0; i--) {
32         while (k >= t && cruz(hull[k-2], hull[k-1], pts[i]) <= 0) k--;
33         hull[k++] = pts[i];
34     }
35
36     hull.resize(k-1); // último punto es el mismo que el primero
37     return hull;
38 }
39
40 int main() {
41     int n;
42     cin >> n;
43     vector<Punto> pts(n);
44     for (int i = 0; i < n; i++) cin >> pts[i].x >> pts[i].y;
45
46     vector<Punto> hull = convexHull(pts);
47
48     cout << hull.size() << "\n";

```

```
49     for (auto& p : hull)
50         cout << p.x << " " << p.y << "\n";
51 }
```

#### 4.0.4 Intersection\_Points

```

1 // Given n horizontal and vertical line segments, your task is to
2 // calculate the number of their intersection points.
3 // You can assume that no parallel line segments intersect, and no
4 // endpoint of a line segment is an intersection point.
5 #include <bits/stdc++.h>
6 #define int long long
7 using namespace std;
8 #define ll long long
9 #define ull unsigned long long
10 #define fore(i, a, b) for(int i = (a); i<(b); i++)
11 #define FOR(i, n) for(int i = 0; i<(n); i++)
12 #define all(x) (x).begin(), (x).end()
13 #define sz(x) (int)(x).size()
14 #define pb push_back
15 using vi = vector<int>;
16 const int MOD = 1000000007;
17 const ll INF = 9223372036854775807LL;
18 const int tam = 1;
19 struct segment{
20     int inix, iniy, finx, finy;
21     segment(){}
22     segment(int a, int b, int c, int d){
23         pair<int, int> aa = {a, b};
24         pair<int, int> bb = {c, d};
25         if(aa > bb){
26             swap(aa, bb);
27         }
28         inix = aa.first;
29         iniy = aa.second;
30         finx = bb.first;
31         finy = bb.second;
32     }
33     // bool operator < (const segment& otro) const{
34     //     return make_pair(make_pair(inix, iniy), make_pair(finx,
35     //     < make_pair(make_pair(otro.inix, otro.iniy), make_pair(otro.
36     //     < otro.finy));
37     // }
38 };
39 #define abrir 10
40 #define consulta 20
41 #define cerrar 30
42
43 struct event{
44     int x, y1, y2, tipo;
45     event(int a = 0, int b = 0, int c = 0, int d = 0){
46         x = a, y1 = b, y2 = c, tipo = d;
47     }
48     bool operator < (const event& otro) const{
49         return x < otro.x or (x == otro.x and tipo < otro.tipo);
50     }
51 }
```

```

46
47 } ;
48
49 #define lsb(x) ((x) & (-x))
50
51 struct BIT {
52     // indexado a 1
53     vector<int> bit;
54     BIT(int N) : bit(N + 1) {}
55     void add(int i, int x) {
56         while (i < sz(bit)) {
57             bit[i] += x;
58             i += lsb(i);
59         }
60     }
61     int sum(int i) {
62         int ans = 0;
63         while (i > 0) {
64             ans += bit[i];
65             i -= lsb(i);
66         }
67         return ans;
68     }
69     int sum(int l, int r) {
70         if (l > r) return 0;
71         return sum(r) - sum(l - 1);
72     }
73 };
74
75 void solve(){
76     int n;cin>>n;
77     vector<segment> arr(n);
78     vector<int> ys;
79     ys.reserve(4*n);
80     vector<tuple<int, int, int , int>> we;
81     for(int i = 0; i<n ;i++){
82         int a, b, c, d;cin>>a>>b>>c>>d;
83         we.emplace_back(a, b, c, d);
84         ys.push_back(b);
85         ys.push_back(b+1);
86         ys.push_back(d);
87         ys.push_back(d+1);
88     }
89     sort(all(ys));
90     ys.erase(unique(all(ys)), ys.end());
91     unordered_map<int, int> f;
92     f.reserve(sz(ys)+2);
93     for(int i = 1; i<=sz(ys); i++){
94         f[ys[i-1]] = i;
95     }
96     int idx = 0;
97     for(auto [a, b,c , d]:we){
98         arr[idx++] = segment(a, f[b], c, f[d]);
99     }
100    vector<event> eventos;
101    eventos.reserve(sz(arr));
102    idx = 0;
103    for(auto x:arr){

```

```

104     if(x.inix == x.finx){
105         //es consulta
106         eventos.push_back(event(x.inix, x.iniy, x.finy, consulta));
107     }else{
108         eventos.push_back(event(x.inix, x.iniy, x.iniy, abrir));
109         eventos.push_back(event(x.finx, x.finy, x.finy, cerrar));
110     }
111 }
112 sort(all(eventos));
113 BIT bit(sz(ys) + 10);
114 int ans = 0;
115 for(auto ev:eventos){
116     if(ev.tipo == abrir){
117         bit.add(ev.y1, 1);
118     }else if(ev.tipo == cerrar){
119         bit.add(ev.y1, -1);
120     }else{
121         ans+=bit.sum(ev.y1, ev.y2);
122     }
123 }
124 cout<<ans<<'\n';
125 }
126
127 signed main(){
128     ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
129     int t = 1;
130     // cin>>t;
131     while(t--){
132         solve();
133     }
134 }
```

#### 4.0.5 Lines\_And\_Qualities\_I

```

1 // Your task is to efficiently process the following types of queries:
2 // Add a line ax+b
3 // Find the maximum point in any line at position x
4 #include <bits/stdc++.h>
5 #define int long long
6 using namespace std;
7 #define ll long long
8 #define ull unsigned ll
9 #define fore(i, a, b) for(int i = (a); i<(b); i++)
10 #define FOR(i, n) for(int i = 0; i<(n); i++)
11 #define all(x) (x).begin(), (x).end()
12 #define sz(x) (int)(x).size()
13 #define pb push_back
14 using vi = vector<int>;
15 const int MOD = 1000000007;
16 const ll INF = 4e18;
17 const int tam = 1;
18
19 struct Line {
20     ll a, b;
21     Line(ll _a=0, ll _b=-INF) : a(_a), b(_b) {}
```

```

22     ll eval(ll x) const { return a*x + b; }
23 };
24
25 struct LiChao {
26     ll l, r;
27     Line line;
28     unique_ptr<LiChao> left, right;
29
30     LiChao(ll _l, ll _r) : l(_l), r(_r), line(Line()), left(nullptr),
31     right(nullptr) {}
32
33     // insertar la linea "nueva" en el intervalo [l_, R_] (ambos
34     // ↪ inclusivos)
35     void insert(Line nueva, ll L_, ll R_) {
36         // si no hay intersección entre [l,r] (nodo) y [L_,R_]
37         // ↪ (inserción), cortar
38         if (R_ < l || r < L_) return;
39
40         // Si el nodo está completamente cubierto por la inserción,
41         // ↪ hacemos el procedimiento Li Chao
42         if (L_ <= l && r <= R_) {
43             ll m = (l + r) >> 1;
44             bool leftBetter = nueva.eval(l) > line.eval(l); // > porque
45             // ↪ buscas máximo en tu código original
46             bool midBetter = nueva.eval(m) > line.eval(m);
47
48             if (midBetter) swap(line, nueva);
49
50             if (l == r) return;
51
52             if (leftBetter != midBetter) {
53                 // la intersección está en el lado izquierdo [l,m]
54                 if (!left) left = make_unique<LiChao>(l, m);
55                 left->insert(nueva, L_, R_);
56             } else {
57                 // la intersección está en el lado derecho [m+1,r]
58                 if (!right) right = make_unique<LiChao>(m+1, r);
59                 right->insert(nueva, L_, R_);
60             }
61             return;
62         }
63
64         // Caso parcial: solo recursar a los hijos cuyos intervalos
65         // ↪ intersectan [l_,R_]
66         ll m = (l + r) >> 1;
67         // Si la inserción toca la mitad izquierda
68         if (L_ <= m) {
69             if (!left) left = make_unique<LiChao>(l, m);
70             left->insert(nueva, L_, R_);
71         }
72         // Si la inserción toca la mitad derecha
73         if (R_ > m) {
74             if (!right) right = make_unique<LiChao>(m+1, r);
75             right->insert(nueva, L_, R_);
76         }
77     }
78 }
```

```

73 // consulta punto x (máximo)
74 ll query(ll x) const {
75     ll res = line.eval(x);
76     if (l == r) return res;
77     ll m = (l + r) >> 1;
78     if (x <= m) {
79         if (left) res = max(res, left->query(x));
80     } else {
81         if (right) res = max(res, right->query(x));
82     }
83     return res;
84 }
85
86 void solve(){
87     int q;cin>>q;
88     int maxn = 1e5 + 100;
89     LiChao tree(-maxn, maxn);
90     while(q--){
91         int tipo;cin>>tipo;
92         if(tipo == 1){
93             int a, b;cin>>a>>b;
94             int l = -maxn, r = maxn;
95             tree.insert(Line(a, b), l, r);
96         }else{
97             int x;cin>>x;
98             auto ans = tree.query(x);
99             cout<<ans<<'\n';
100            // cout<<(ans == -INF or ans ==
101            //      INF?"NO":to_string(ans))<<'\n';
102        }
103    }
104 }
105
106 signed main(){
107     ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
108     int t = 1;
109     // cin>>t;
110     while(t--){
111         solve();
112     }
113 }

11  #define all(x) (x).begin(), (x).end()
12  #define sz(x) (int)(x).size()
13  #define pb push_back
14  using vi = vector<int>;
15  const int MOD = 1000000007;
16  const ll INF = 4e18;
17  const int tam = 1;
18
19  struct Line {
20     ll a, b;
21     Line(ll _a=0, ll _b=-INF) : a(_a), b(_b) {}
22     ll eval(ll x) const { return a*x + b; }
23 };
24
25  struct LiChao {
26     ll l, r;
27     Line line;
28     unique_ptr<LiChao> left, right;
29
30     LiChao(ll _l, ll _r) : l(_l), r(_r), line(Line()), left(nullptr),
31     ↪ right(nullptr) {}
32
33     // insertar la línea "nueva" en el intervalo [l_, R_] (ambos
34     ↪ inclusivos)
35     void insert(Line nueva, ll L_, ll R_) {
36         // si no hay intersección entre [l,r] (nodo) y [L_,R_]
37         ↪ (inserción), cortar
38         if (R_ < l || r < L_) return;
39
40         // Si el nodo está completamente cubierto por la inserción,
41         ↪ hacemos el procedimiento Li Chao
42         if (L_ <= l && r <= R_) {
43             ll m = (l + r) >> 1;
44             bool leftBetter = nueva.eval(l) > line.eval(l); // > porque
45             ↪ buscas máximo en tu código original
46             bool midBetter = nueva.eval(m) > line.eval(m);
47
48             if (midBetter) swap(line, nueva);
49
50             if (l == r) return;
51
52             if (leftBetter != midBetter) {
53                 // la intersección está en el lado izquierdo [l,m]
54                 if (!left) left = make_unique<LiChao>(l, m);
55                 left->insert(nueva, L_, R_);
56             } else {
57                 // la intersección está en el lado derecho [m+1,r]
58                 if (!right) right = make_unique<LiChao>(m+1, r);
59                 right->insert(nueva, L_, R_);
60             }
61             return;
62
63             // Caso parcial: solo recursar a los hijos cuyos intervalos
64             ↪ intersectan [L_,R_]
65             ll m = (l + r) >> 1;
66             // Si la inserción toca la mitad izquierda
67         }
68     }
69
70     // Consulta punto x (máximo)
71     ll query(ll x) const {
72         ll res = line.eval(x);
73         if (l == r) return res;
74         ll m = (l + r) >> 1;
75         if (x <= m) {
76             if (left) res = max(res, left->query(x));
77         } else {
78             if (right) res = max(res, right->query(x));
79         }
80         return res;
81     }
82
83     void solve(){
84         int q;cin>>q;
85         int maxn = 1e5 + 100;
86         LiChao tree(-maxn, maxn);
87         while(q--){
88             int tipo;cin>>tipo;
89             if(tipo == 1){
90                 int a, b;cin>>a>>b;
91                 int l = -maxn, r = maxn;
92                 tree.insert(Line(a, b), l, r);
93             }else{
94                 int x;cin>>x;
95                 auto ans = tree.query(x);
96                 cout<<ans<<'\n';
97                 // cout<<(ans == -INF or ans ==
98                 //      INF?"NO":to_string(ans))<<'\n';
99             }
100            }
101        }
102    }
103 }
104
105 signed main(){
106     ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
107     int t = 1;
108     // cin>>t;
109     while(t--){
110         solve();
111     }
112 }


```

#### 4.0.6 Lines And Queries II

```

1 // Your task is to efficiently process the following types of queries:
2 // Add a line ax+b that is active in range [l,r]
3 // Find the maximum point in any active line at position x
4 #include <bits/stdc++.h>
5 #define int long long
6 using namespace std;
7 #define ll long long
8 #define ull unsigned ll
9 #define fore(i, a, b) for(int i = (a); i<(b); i++)
10 #define FOR(i, n) for(int i = 0; i<(n); i++)


```

```

62     if (L_ <= m) {
63         if (!left) left = make_unique<LiChao>(l, m);
64         left->insert(nueva, L_, R_);
65     }
66     // Si la inserción toca la mitad derecha
67     if (R_ > m) {
68         if (!right) right = make_unique<LiChao>(m+1, n);
69         right->insert(nueva, L_, R_);
70     }
71 }
72
73 // consulta punto x (máximo)
74 ll query(ll x) const {
75     ll res = line.eval(x);
76     if (l == r) return res;
77     ll m = (l + r) >> 1;
78     if (x <= m) {
79         if (left) res = max(res, left->query(x));
80     } else {
81         if (right) res = max(res, right->query(x));
82     }
83     return res;
84 }
85
86 void solve(){
87     int q;cin>>q;
88     int maxn = 1e5 + 100;
89     LiChao tree(-maxn, maxn);
90     while(q--){
91         int tipo;cin>>tipo;
92         if(tipo == 1){
93             int a, b;cin>>a>>b;
94             int l, r;cin>>l>>r;
95             tree.insert(Line(a, b), l, r);
96         }else{
97             int x;cin>>x;
98             auto ans = tree.query(x);
99             cout<<(ans == -INF or ans == INF?"NO":to_string(ans))<<'\n';
100        }
101    }
102 }
103
104 signed main(){
105     ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
106     int t = 1;
107     // cin>>t;
108     while(t--){
109         solve();
110     }
111 }
112 }
```

#### 4.0.7 Line\_Segment\_Intersection

---

```

1 // There are two line segments: the first goes through the points
2 → (x_1,y_1) and (x_2,y_2), and the second goes through the points
3 → (x_3,y_3) and (x_4,y_4).
4 // Your task is to determine if the line segments intersect, i.e., they
5 → have at least one common point.
6 #include <bits/stdc++.h>
7
8 using namespace std;
9 typedef long double ld;
10 const ld EPS = 1e-9;
11 #define int long long
12
13 bool ge(int x, int y) {
14     return x + EPS > y;
15 }
16 bool le(int x, int y) {
17     return x - EPS < y;
18 }
19 bool eq(int x, int y) {
20     return ge(x, y) and le(x, y);
21 }
22 int sign(int x) {
23     return ge(x, 0) - le(x, 0);
24 }
25
26 struct point_i {
27     // Construcción
28     int x, y;
29     point_i() : x(0), y(0) {}
30     point_i(int x, int y) : x(x), y(y) {}
31     // Operadores
32     point_i operator-() { return point_i(-x, -y); }
33     point_i operator+(point_i p) { return point_i(x + p.x, y + p.y); }
34     point_i operator-(point_i p) { return point_i(x - p.x, y - p.y); }
35     point_i operator*(int k) { return point_i(k * x, k * y); }
36     // producto punto
37     int operator*(point_i p) { return x * p.x + y * p.y; }
38     // producto cruz
39     int operator%(point_i p) { return x * p.y - y * p.x; }
40     // Comparadores
41     bool operator==(const point_i &p) const { return x == p.x and y == p.y; }
42     bool operator!=(const point_i &p) const { return x != p.x or y != p.y; }
43     bool operator<(const point_i &p) const {
44         return (x < p.x) or (x == p.x and y < p.y);
45     }
46     // 0 ⇒ same direction
47     // 1 ⇒ p is on the left
48     // -1 ⇒ p is on the right
49     int dir(point_i o, point_i p) {
50         int x = (*this - o) % (p - o);
51         return (ge(x, 0)) - le(x, 0);
52     }
53 }
```

```

51
52     bool on_seg(point_i p, point_i q) {
53         if (this->dir(p, q)) return 0;
54         return ge(x, min(p.x, q.x)) and le(x, max(p.x, q.x)) and
55             ge(y, min(p.y, q.y)) and le(y, max(p.y, q.y));
56     }
57
58     // Longitudes y distancias
59     ld abs() { return sqrtl(x * x + y * y); }
60     int abs2() { return x * x + y * y; }
61     ld dist(point_i q) { return (*this - q).abs(); }
62     int dist2(point_i q) { return (*this - q).abs2(); }
63     // Angulo respecto al eje x (1,1) -> 45 (pero en radianes)
64     ld arg() { return atan2l(y, x); }
65     // DOUBLE
66     point_i project(point_i y) { return y * ((*this * y) / (y * y)); }
67     // Proyecta el punto actual sobre la linea definida por los puntos
68     → y
69     // 'y'
70
71     point_i project(point_i x, point_i y) {
72         return x + (*this - x).project(y - x);
73     }
74     // Calcula la distancia del punto actual a la linea infinita que pasa
75     → por x
76     // 'y'
77     ld dist_line(point_i x, point_i y) { return dist(project(x, y)); }
78     // Calcula la distancia del punto actual al segmento [x, y]
79     ld dist_seg(point_i x, point_i y) {
80         return project(x, y).on_seg(x, y) ? dist_line(x, y)
81                                     : min(dist(x), dist(y));
82     }
83     // rotaciones
84     point_i rotate(ld sin, ld cos) {
85         return point_i(cos * x - sin * y, sin * x + cos * y);
86     }
87     point_i rotate(ld a) { return rotate(sinl(a), cosl(a)); }
88
89     // rotar respecto a un punto
90     // USAR DOUBLES POR LA PRECICION AQUI
91     point_i rotate(point_i p) { return rotate(p.y / p.abs(), p.x /
92         → p.abs()); }
93     // Duda
94     int direction(point_i o, point_i p, point_i q) {
95         return p.dir(o, q);
96     }
97     // Rotacion horaria y antihoraria
98     point_i rotate_ccw90(point_i p) {
99         return point_i(-p.y, p.x);
100    }
101    point_i rotate_cw90(point_i p) {
102        return point_i(p.y, -p.x);
103    }
104
105    // for reading purposes avoid using * and % operators, use the functions
106    → below:
107    int dot(point_i p, point_i q) {
108        return p.x * q.x + p.y * q.y;
109    }
110
111    int cross(point_i p, point_i q) {
112        return p.x * q.y - p.y * q.x;
113    }
114
115    // Duda
116    int area_2(point_i a, point_i b, point_i c) {
117        return cross(a, b) + cross(b, c) + cross(c, a);
118    }
119    // Duda
120    int angle_less(const point_i &a1, const point_i &b1, const point_i &a2,
121                  const point_i &b2) {
122        // angle between (a1 and b1) vs angle between (a2 and b2)
123        // 1 : bigger
124        // -1 : smaller
125        // 0 : equal
126        point_i p1(dot(a1, b1), abs(cross(a1, b1)));
127        point_i p2(dot(a2, b2), abs(cross(a2, b2)));
128        if (cross(p1, p2) < 0) return 1;
129        if (cross(p1, p2) > 0) return -1;
130        return 0;
131    }
132
133    ostream &operator<<(ostream &os, const point_i &p) {
134        os << "(" << p.x << "," << p.y << ")";
135        return os;
136    }
137    istream &operator>>(istream &in, point_i &p) {
138        in >> p.x >> p.y;
139        return in;
140    }
141
142    bool dentro(point_i &p, point_i &a, point_i &b) {
143        return min(a.x, b.x) <= p.x and p.x <= max(a.x, b.x) and
144            min(a.y, b.y) <= p.y and p.y <= max(a.y, b.y);
145    }
146
147    void solve() {
148        point_i a, b, c, d;
149        cin >> a >> b >> c >> d;
150        point_i ab = b - a, ac = c - a, ad = d - a;
151        point_i cd = d - c, ca = a - c, cb = b - c;
152        if ((ab%ac > 0 and ab%ad < 0) or (ab%ac < 0 and ab%ad > 0)) and
153            ((cd%ca > 0 and cd%cb < 0) or (cd%ca < 0 and cd%cb > 0)) {
154                cout << "YES\n";
155                return;
156            }
157        bool ok = (ab%ac == 0 and dentro(c, a, b)) or
158            (ab%ad == 0 and dentro(d, a, b)) or
159            (cd%ca == 0 and dentro(a, c, d)) or
160            (cd%cb == 0 and dentro(b, c, d));
161        if (ok) {
162            cout << "YES\n";
163        } else {
164            cout << "NO\n";
165        }
166    }
167
168    signed main(){
169

```

```

163 ios::sync_with_stdio(0);
164 cin.tie(0);
165 cout.tie(0);
166 int t = 1;
167 cin>>t;
168 while(t--){solve();}
169 }



---



#### 4.0.8 Line_Segments_Trace II



---



```

1 // There are n line segments whose endpoints have integer coordinates. 55
2 → Each x-coordinate is between 0 and m. The slope of each segment is an 56
2 → integer. 57
3 // For each x-coordinate 0,1,\dots,m, find the maximum point in any line 58
3 → segment. If there is no segment at some point, the maximum is -1. 59
4 #include <bits/stdc++.h> 60
5 #define int long long 61
6 #define ll long long 62
7 #define ull unsigned ll 63
8 #define fore(i, a, b) for(int i = (a); i<(b); i++) 64
9 #define FOR(i, n) for(int i = 0; i<(n); i++) 65
10 #define all(x) (x).begin(), (x).end() 66
11 #define sz(x) (int)(x).size() 67
12 #define pb push_back 68
13 using vi = vector<int>; 69
14 const int MOD = 1000000007; 70
15 const ll INF = 4e18; 71
16 const int tam = 1; 72
17
18 struct Line { 73
19     ll a, b; 74
20     Line(ll _a=0, ll _b=-INF) : a(_a), b(_b) {} 75
21     ll eval(ll x) { return a*x + b; } 76
22 }; 77
23
24 struct LiChao { 78
25     int l, r; 79
26     Line line; 80
27     unique_ptr<LiChao> left, right; 81
28     LiChao(int _l, int _r) : l(_l), r(_r), line(Line()), left(nullptr), 82
29     → right(nullptr) {} 83
30     void insert(Line nueva, int L, int R) { 84
31         if (R < l or r < L) return; 85
32         if (L <= l and r <= R) { 86
33             int m = (l+r)/2; 87
34             bool izq_bet = nueva.eval(l) > line.eval(l); 88
35             bool mid_bet = nueva.eval(m) > line.eval(m); 89
36
37             if (mid_bet) swap(line, nueva); 90
38
39             if (l == r) return; 91
40
41             if (izq_bet != mid_bet) {
42                 if (!left) left = make_unique<LiChao>(l, m);
43                 left->insert(nueva, L, R);
44             } else {
45                 if (!right) right = make_unique<LiChao>(m+1, r);
46                 right->insert(nueva, L, R);
47             }
48             return;
49         }
50         if (!left) left = make_unique<LiChao>(l, (l+r)/2);
51         if (!right) right = make_unique<LiChao>((l+r)/2+1, r);
52         left->insert(nueva, L, R);
53         right->insert(nueva, L, R);
54     }
55
56     ll query(int x) {
57         if (l == r) return line.eval(x);
58         int m = (l+r)/2;
59         ll res = line.eval(x);
60         if (x <= m and left) res = max(res, left->query(x));
61         if (x > m and right) res = max(res, right->query(x));
62         return res;
63     };
64
65
66     void solve(){
67         int n, m;cin>>n>>m;
68         LiChao tree(0, 1e9);
69         for(int i = 0; i<n; i++){
70             int a, b, c, d;cin>>a>>b>>c>>d;
71             int dy = d - b;
72             int dx = c - a;
73             int pendiente = dy/dx;
74             int constante = b - pendiente*a;
75             tree.insert(Line(pendiente, constante), a, c);
76         }
77         for(int i = 0; i<=m; i++){
78             int ans = tree.query(i);
79
80             cout<<(ans == -INF?-1:ans)<<'\n';
81         }
82     }
83
84     signed main(){
85         ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
86         int t = 1;
87         // cin>>t;
88         while(t--){
89             solve();
90         }
91     }

```



---



```

#### 4.0.9 Line\_Segment\_Trace\_I

```

1 // There are n line segments whose endpoints have integer coordinates. The
2 // left x-coordinate of each segment is 0 and the right x-coordinate is m.
2 // The slope of each segment is an integer.
3 // For each x-coordinate 0,1,...,m, find the maximum point in any line
4 // segment.
5 #include <bits/stdc++.h>
6 using namespace std;
7 #define int long long
8 #define ll long long
9 #define ull unsigned long long
10 #define fore(i, a, b) for (int i = (a); i < (b); i++)
11 #define FOR(i, n) for (int i = 0; i < (n); i++)
12 #define all(x) (x).begin(), (x).end()
13 #define sz(x) (int)(x).size()
14 #define pb push_back
15 using vi = vector<int>;
16 const int MOD = 1000000007;
17 const ll INF = 9223372036854775807LL;
18 const int tam = 1;
19 /**
20 * Author: Simon Lindholm
21 * Date: 2017-04-20
22 * License: CCO
23 * Source: own work
24 * Description: Container where you can add lines of the form kx+m, and
25 // query
26 // maximum values at points x. Useful for dynamic programming. Time:
27 // O(\log N)
28 * Status: tested
29 */
30 struct Line {
31     mutable ll k, m, p;
32     bool operator<(const Line &o) const { return k < o.k; }
33     bool operator<(ll x) const { return p < x; }
34 };
35 struct LineContainer : multiset<Line, less<> {
36     // (for doubles, use inf = 1./0., div(a,b) = a/b)
37     const ll inf = LLONG_MAX;
38     ll div(ll a, ll b) { // floored division
39         return a / b - ((a ^ b) < 0 && a % b);
40     }
41     bool isect(iterator x, iterator y) {
42         if (y == end()) {
43             x->p = inf;
44             return false;
45         }
46         if (x->k == y->k)
47             x->p = x->m > y->m ? inf : -inf;
48         else
49             x->p = div(y->m - x->m, x->k - y->k);
50         return x->p >= y->p;
51     }
52     void add(ll k, ll m) {
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95

```

```

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

```

1 // A set is initially empty and n points are added to it. Calculate the
2 // maximum Manhattan distance of two points after each addition.
3 void solve(){
4     int n; cin >> n;
5     multiset<int> sum, diff;
6     for(int i = 0; i < n; i++){
        int a, b; cin >> a >> b;
        sum.insert(a+b);
        diff.insert(a-b);
    }
    for(int i = 1; i < n; i++){
        int a, b; cin >> a >> b;
        sum.insert(a+b);
        diff.insert(a-b);
        cout << (*sum.rbegin() - *sum.begin()) - (*diff.rbegin() - *diff.begin());
    }
}

```

```
7     sum.insert(a+b);
8     diff.insert(a-b);
9     auto ini_s = sum.begin();
10    auto end_s = sum.end();
11    end_s--;
12    auto ini_d = diff.begin();
13    auto end_d = diff.end();
14    end_d--;
15    cout<<max(*end_s - *ini_s, *end_d - *ini_d)<<'\n';
16
17 }
```

#### 4.0.11 Minimum\_Euclidean\_Distance

```

1 // Given a set of points in the two-dimensional plane, your task is
2 // find the minimum Euclidean distance between two distinct points.
3 // The Euclidean distance of points (x_1,y_1) and (x_2,y_2) is
4 // \sqrt{(x_1-x_2)^2+(y_1-y_2)^2}.
5 #include <bits/stdc++.h>
6 using namespace std;
7 #pragma GCC optimize("Ofast")
8 #pragma GCC target("avx2,bmi,bmi2,lzcnt,popcnt")
9 #pragma GCC optimize("unroll-loops")
10
11 #define int long long
12 #define all(x) (x).begin(), (x).end()
13 const int INF_INT = 2000000000;
14
15 int dis(pair<int,int> a, pair<int,int> b){
16     int dx = a.first - b.first;
17     int dy = a.second - b.second;
18     return dx*dx + dy*dy;
19 }
20
21 void solve(){
22     int n; cin >> n;
23     vector<pair<pair<int,int>, int>> ptos(n); // {{x,y}, id}
24     for(int i=0;i<n;i++){
25         int a,b; cin >> a >> b;
26         ptos[i] = {{a,b}, i};
27     }
28     sort(all(ptos)); // orden por x
29
30     // inicializar best (asegúrate n≥2 según restricciones)
31     int best = dis(ptos[0].first, ptos[1].first);
32     pair<int,int> mejores = {ptos[0].second, ptos[1].second};
33
34     // set ordenado por y: elemento = {{y,x}, id}
35     set<pair<pair<int,int>, int>> ventana;
36
37     int left = 0; // puntero que avanza para cerrar la ventana por X
38     for(int i=0;i<n;i++){
39         int x = ptos[i].first.first;
40         int y = ptos[i].first.second;
41         int id = ptos[i].second;
42
43         // cerrar la ventana
44         while(ventana.size() > 0 && y >= ventana.rbegin().first)
45             ventana.erase(ventana.rbegin());
46
47         // insertar en la ventana
48         ventana.insert({{y,x}, id});
49
50         // calcular el mejor resultado
51         if(best > dis(mejores.first, {{y,x}, id}))
52             mejores = {{y,x}, id};
53
54     }
55
56     cout << best;
57 }
```

```

40
41     // 1) eliminar del set los puntos que quedaron demasiado a la
42     // izquierda (por X)
43     while(left < i){
44         int x_left = ptos[left].first.first;
45         int dx = x - x_left;
46         if(dx*dx > best){
47             int y_left = ptos[left].first.second;
48             int id_left = ptos[left].second;
49             ventana.erase({{y_left, x_left}, id_left});
50             left++;
51         } else break;
52     }
53
54     // 2) limitar por Y usando d = floor(sqrt(best)) para no iterar
55     // todo el set
56     int d = (int)floor(sqrt((long double)best));
57     auto it_low = ventana.lower_bound({{y - d, -INF_INT}, -INF_INT});
58     auto it_high = ventana.upper_bound({{y + d, INF_INT}, INF_INT});
59
60     for(auto it = it_low; it != it_high; ++it){
61         int x2 = it->first.second;
62         int y2 = it->first.first;
63         int id2 = it->second;
64         int dx = x - x2;
65         int dx2 = dx*dx;
66         if(dx2 > best) continue; // filtro extra
67         int dy = y - y2;
68         int dist = dx2 + dy*dy;
69         if(dist < best){
70             best = dist;
71             mejores = {id, id2};
72         }
73     }
74
75     // 3) insertar el punto actual para futuros i
76     ventana.insert({{y, x}, id});
77 }
78
79 cout << best << '\n';
80 // cout << mejores.first << ' ' << mejores.second << '\n';
81
82 signed main(){
83     ios::sync_with_stdio(0);
84     cin.tie(0); cout.tie(0);
85     solve();
86 }

```

#### 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.`

```

// 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);
                }
            }
        }
    }
}

```

```

109         // contar solo si la intersección está a la derecha de
110         → x
111         // (rayo hacia +inf)
112         if (x_int > (double)x.x) veces++;
113     }
114 }
115 if (flag == 1) {
116     cout << "BOUNDARY\n";
117 } else {
118     cout << (veces % 2 ? "INSIDE" : "OUTSIDE") << '\n';
119 }
120 }
121 }
122
123 signed main() {
124     ios::sync_with_stdio(0);
125     cin.tie(0);
126     cout.tie(0);
127     int t = 1;
128     // cin>>t;
129     while (t--) { solve(); }
130 }
```

---

```

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;
}
```

#### 4.0.13 Point\_Location\_Test

```

1 // There is a line that goes through the points p_1=(x_1,y_1) and
2 // p_2=(x_2,y_2). There is also a point p_3=(x_3,y_3).
3 // Your task is to determine whether p_3 is located on the left or right
4 // side of the line or if it touches the line when we are looking from
5 // p_1 to p_2.
6 #include <bits/stdc++.h>
7 using namespace std;
8 #define int long long
9 #define f first
10 #define sst stringstream
11 #define s second
12 #define pb push_back
13 #define sz(x) (int)(x).size()
14 #define all(a) (a).begin(), (a).end()
15 #define rall(a) (a).rbegin(), (a).rend()
16 #define fore(i, a, n) for (int i = (a); i < (n); i++)
17 #define forb(i, n) for (int i = (n) - 1; i ≥ 0; i--)
18 #define popcorn(x) __builtin_popcountll(x);
19 typedef pair<int, int> pii;
20 typedef vector<int> vi;
21 const int MOD = 1000000007;
22 const double EPS = 1e-9;
23 const double PI = acos(-1);
24 const int INF = 1e18;
25 // PLUS ULTRA RECARGADO!!!
26
27 struct point {
28     int x, y;
29     point() {}
```

```

84 }
85
86 void solve() {
87     point a, b, c;
88     cin >> a >> b >> c;
89     point ab = b - a;
90     point ac = c - a;
91     int cros = cross(ab, ac);
92     if (cros > 0) {
93         cout << "LEFT\n";
94     } else if (cros < 0) {
95         cout << "RIGHT\n";
96     } else {
97         cout << "TOUCH\n";
98     }
99 }
100 signed main() {
101     ios::sync_with_stdio(0);
102     cin.tie(0);
103     cout.tie(0);
104     int t;
105     cin >> t;
106     while (t--) solve();
107     // solve();
108 }

// Construcción
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(); }
// Ángulo respecto al eje x (1,1) -> 45 (pero en radianes)
ld arg() { return atan2l(y, x); }

// Rotación 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;

```

#### 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 popcorn(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 {


```

```

80     cin >> n;
81     vector<point_i> pol(n);
82     fore(i, 0, n) {
83         cin >> pol[i];
84     }
85     int ans = 0;
86     fore(i, 0, n) {
87         ans += (pol[i].x * pol[(i + 1) % n].y);
88         ans -= (pol[i].y * pol[(i + 1) % n].x);
89     }
90     cout << abs(ans) << endl;
91 }
92 signed main() {
93     ios::sync_with_stdio(0);
94     cin.tie(0);
95     cout.tie(0);
96     int t = 1;
97     // cin>>t;
98     while (t--) solve();
99 }

28     const pto operator*(const pto& otro) const {
29         return pto(x*otro.x, y*otro.y);
30     }
31     const tipo operator%(const pto& otro) const {
32         return x*otro.y - y*otro.x;
33     }
34     const bool operator==(const pto& otro) const{
35         return x == otro.x and y == otro.y;
36     }
37     const pto operator*(const tipo k) const{
38         return pto(x*k, y*k);
39     }
40 }
41
42 ll boundary_points(const vector<pto>& poly, const int n){
43     ll ans = 0;
44     for(int i = 0; i<n; i++){
45         int sig = (i+1)%n;
46         ll dx = abs(poly[i].x - poly[sig].x);
47         ll dy = abs(poly[i].y - poly[sig].y);
48         ans+=gcd(dx, dy);
49     }
50     return ans;
51 }
52

53 void solve(){
54     int n; cin>>n;
55     vector<pto> poly(n);
56     for(int i = 0; i<n; i++){
57         tipo a, b; cin>>a>>b;
58         poly[i] = {a, b};
59     }
60     ll area2 = 0;
61     for(int i = 0; i<n; i++){
62         int sig = (i+1)%n;
63         auto[x1, y1] = poly[i];
64         auto[x2, y2] = poly[sig];
65         area2+=((x1*y2)-(y1*x2));
66     }
67     area2 = abs(area2);
68     ll b = boundary_points(poly, n);
69     ll ans = area2 - b + 2;
70     cout<<ans/2 << ' ' <<b<<'\n';
71 }
72

73 signed main(){
74     ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
75     int t = 1;
76     //cin>>t;
77     while(t--){
78         solve();
79     }
80 }
81

```

#### 4.0.15 Polygon\_Lattice\_Points

```

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

21#define tipo long long
22 struct pto
23 {
24     tipo x, y;
25     pto(tipo _x = 0, tipo _y = 0): x(_x), y(_y){}
26     const pto operator + (const pto& otro) const {
27         return pto(x + otro.x, y + otro.y);
28     }
29     const pto operator - (const pto& otro) const {
30         return pto(x - otro.x, y - otro.y);
31     }
32 }
```

## 5 Graph Algorithms

### 5.0.1 Cycle\_Finding

```

1 #include <bits/stdc++.h>
2 #define ll long long int
3 #define INF 1000000000000000LL
4 //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.
5
6
7 using namespace std;
8
9 // Lista de adyacencia para almacenar el grafo ponderado
10 vector<pair<ll, ll>> grafo[2501];
11
12 // Arreglos para guardar las distancias, padres y el número de
→ relajaciones
13 ll distancias[2501];
14 ll padres[2501];
15 ll contadorRelajaciones[2501];
16
17 ll numNodos, numAristas, nodoInicio, nodoDestino, pesoArista;
18 bool enCola[2501]; // Arreglo para saber si un nodo está en la cola
19 bool visitado[2501]; // Arreglo para saber si un nodo ya fue visitado
20 ll nodoConCiclo; // Nodo involucrado en el ciclo negativo
21
22 // Función que implementa el algoritmo SPFA (Shortest Path Faster
→ Algorithm)
23 bool spfa(ll inicio) {
24     // Inicializamos la distancia del nodo de inicio a 0
25     distancias[inicio] = 0;
26     padres[inicio] = -1; // El nodo de inicio no tiene un parente
27
28     queue<ll> cola; // Cola para procesar los nodos
29     cola.push(inicio);
30     enCola[inicio] = true; // Marcamos como que está en la cola
31
32     while (!cola.empty()) {
33         ll nodoActual = cola.front(); // Sacamos el primer nodo de la co
34         visitado[nodoActual] = true;
35         enCola[nodoActual] = false;
36         cola.pop();
37
38         // Iteramos sobre los vecinos del nodo actual
39         for (auto vecino : grafo[nodoActual]) {
40             // Si la distancia al vecino es mayor que la distancia al no
→ actual más el peso de la arista
41             if (distancias[vecino.first] > distancias[nodoActual] +
→ vecino.second) {
42                 contadorRelajaciones[vecino.first]++; // Aumentamos el
→ contador de relajaciones
43                 // Si el nodo ha sido relajado más de "numNodos" veces,
→ hay un ciclo negativo
44                 if (contadorRelajaciones[vecino.first] > numNodos) {
45                     nodoConCiclo = vecino.first;
46                     padres[vecino.first] = nodoActual;
47                 }
48             }
49         }
50     }
51 }

```

```

47     return false; // Se encuentra un ciclo negativo,
48     // → retornamos falso
49 }
50     // Actualizamos la distancia al vecino
51     distancias[vecino.first] = distancias[nodoActual] +
52     // → vecino.second;
53     // Si el vecino no está en la cola, lo agregamos
54     if (!enCola[vecino.first]) {
55         cola.push(vecino.first);
56         enCola[vecino.first] = true;
57     }
58     // Actualizamos el nodo padre del vecino
59     padres[vecino.first] = nodoActual;
60 }
61 }
62 return true; // Si no encontramos ningún ciclo negativo, retornamos
63 // → true
64 }
65
66 int main() {
67     // Leemos el número de nodos y aristas
68     cin >> numNodos >> numAristas;
69
70     // Inicializamos todas las distancias a infinito (INF)
71     for (int i = 1; i <= numNodos; i++) {
72         distancias[i] = INF;
73     }
74
75     // Leemos las aristas (nodo de inicio, nodo de destino, peso)
76     for (ll i = 0; i < numAristas; i++) {
77         cin >> nodoInicio >> nodoDestino >> pesoArista;
78         // Añadimos la arista al grafo
79         grafo[nodoInicio].push_back({nodoDestino, pesoArista});
80     }
81
82     // Intentamos ejecutar SPFA desde cada nodo
83     for (ll i = 1; i <= numNodos; i++) {
84         // Si encontramos un ciclo negativo desde el nodo "i"
85         if (!spfa(i)) {
86             cout << "YES" << endl; // Imprimimos que hay un ciclo negativo
87             ll nodo = nodoConCiclo;
88             stack<ll> pila;
89             bool enPila[2501] = {}; // Arreglo para verificar si el nodo
90             // → está en la pila
91
92             // Reconstruimos el ciclo negativo utilizando el arreglo de
93             // → padres
94             while (!enPila[nodo]) {
95                 enPila[nodo] = true;
96                 pila.push(nodo);
97                 nodo = padres[nodo];
98             }
99
100            // Imprimimos el ciclo negativo
101            cout << nodo << " "; // Imprimimos el primer nodo del ciclo
102            while (pila.top() != nodo) { // Imprimimos el resto de los
103                // → nodos en el ciclo

```

```

99         cout << pila.top() << " ";
100        pila.pop();
101    }
102    cout << nodo << endl; // Imprimimos el último nodo del ciclo
103    return 0;
104}
105
106 cout << "NO" << endl; // Si no encontramos ciclo negativo, imprimimos
107     ~ "NO"
108}

```

## 5.0.2 Flight\_Routes

```

1 // Your task is to find the k shortest flight routes from Syrjälä to
2     → Metsälä. A route can visit the same city several times.
3 // Note that there can be several routes with the same price and each of
4     → them should be considered (see the example).
5 #include <bits/stdc++.h>
6 using namespace std;
7 #define int long long
8 #define ll long long
9 #define ull unsigned long long
10 #define fore(i, a, b) for(int i = (a); i<(b); i++)
11 #define FOR(i, n) for(int i = 0; i<(n); i++)
12 #define all(x) (x).begin(), (x).end()
13 #define sz(x) (int)(x).size()
14 #define pb push_back
15 using vi = vector<int>;
16 const int MOD = 1000000007;
17 const ll INF = 4e18;
18 const int tam = 1e5+10;
19 int n, m, k;
20 vector<pair<int, int>> g[tam];
21
22 vector<vector<int>> dijkstra(){
23     vector<vector<int>> dist(n+1, vector<int>(k, INF));
24     priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>> pq;
25     dist[1][0] = 0;
26     pq.push({0, 1});
27     while(!pq.empty()){
28         auto[d, u] = pq.top();
29         pq.pop();
30         if(d > dist[u][k-1]) continue;
31         for(auto[v, w]:g[u]){
32             if(d + w < dist[v][k-1]){
33                 dist[v][k-1] = d + w;
34                 pq.push({dist[v][k-1], v});
35             }
36         }
37     }
38     return dist;
39 }

```

```

39
40 void solve(){
41     cin>>n>>m>>k;
42     for(int i = 0; i<m; i++){
43         int a, b, c;cin>>a>>b>>c;
44         g[a].push_back({b, c});
45     }
46     auto d = dijkstra();
47     for(int i = 0; i<k; i++){
48         cout<<d[n][i]<< ' ';
49     }
50     cout<<'\n';
51
52 signed main(){
53     ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
54     int t = 1;
55     // cin>>t;
56     while(t--){
57         solve();
58     }
59 }

```

## 5.0.3 Game\_Levels

```

1 // A game has n levels, connected by m teleporters, and your task is to
2     → get from level 1 to level n. The game has been designed so that there
3     → are no directed cycles in the underlying graph. In how many ways can
4     → you complete the game?
5 #include <bits/stdc++.h>
6 using namespace std;
7 #define int long long
8 #define ll long long
9 const int tam = 100000+10;
10 const int MOD = 1000000000+7;
11 #define sz(x) (int)(x).size()
12 int n, m;
13 vector<vector<int>> adj(tam);
14 vector<int> dist(tam, -1);
15 vector<int> topsort(tam);
16 int dfs(int nodo){
17     if(nodo == n){
18         return 1;
19     }else if(sz(adj[nodo]) == 0){
20         return 0;
21     }else if(dist[nodo] == -1){
22         dist[nodo] = 0;
23         for(int vecino:adj[nodo]){
24             int tam = dfs(vecino);
25             dist[nodo]+=tam;
26             dist[nodo]%=MOD;
27         }
28     }
29     return dist[nodo];
30 }

```

```

28
29 void solve(){
30     cin>>n>>m;
31     for(int i = 0; i<m; i++){
32         int a, b;cin>>a>>b;
33         adj[a].push_back(b);
34     }
35     cout<<dfs(1)<<endl;
36 }
37
38 signed main(){
39     ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
40     int t = 1;
41     // cin>>t;
42     while(t--){
43         solve();
44     }
45 }

31
32     auto [a, b, cost] = e;
33     if (distancia[a] < INF && distancia[a] + cost < distancia[b])
34     {
35         distancia[b] = max(-INF, distancia[a] + cost);
36     }
37 }

38 // Detectar ciclos negativos
39 vector<bool> in_cycle(n, false);
40 for (int i = 0; i < n; ++i) {
41     for (const auto& e : edges) {
42         auto [a, b, cost] = e;
43         if (distancia[a] < INF && distancia[a] + cost < distancia[b])
44         {
45             distancia[b] = -INF; // Marcar nodo afectado por ciclo
46             in_cycle[b] = true;
47         }
48     }
49 }
50 return distancia[n - 1]; // Distancia al último nodo
}

```

#### 5.0.4 High\_Score

```

1 // You play a game consisting of n rooms and m tunnels. Your initial score
2 // is 0, and each tunnel increases your score by x where x may be both
3 // positive or negative. You may go through a tunnel several times.
4 // Your task is to walk from room 1 to room n. What is the maximum score
5 // you can get?
6 #include <bits/stdc++.h>
7 using namespace std;
8 #define int long long
9 const int INF = 1e18;
10
11 // Función para determinar accesibilidad desde un nodo
12 void bfs(int start, vector<vector<int>>& graph, vector<bool>& visited) {
13     queue<int> q;
14     q.push(start);
15     visited[start] = true;
16     while (!q.empty()) {
17         int u = q.front(); q.pop();
18         for (int v : graph[u]) {
19             if (!visited[v]) {
20                 visited[v] = true;
21                 q.push(v);
22             }
23         }
24     }
25     int bellman_ford(int n, int start, const vector<tuple<int, int, int>>&
26     edges) {
27         vector<int> distancia(n, INF);
28         distancia[start] = 0;
29         // Relajar las aristas n-1 veces
30         for (int i = 0; i < n - 1; ++i) {
31             for (const auto& e : edges) {
32                 auto [a, b, cost] = e;
33                 if (distancia[a] < INF && distancia[a] + cost < distancia[b])
34                 {
35                     distancia[b] = max(-INF, distancia[a] + cost);
36                 }
37             }
38             // Detectar ciclos negativos
39             vector<bool> in_cycle(n, false);
40             for (int i = 0; i < n; ++i) {
41                 for (const auto& e : edges) {
42                     auto [a, b, cost] = e;
43                     if (distancia[a] < INF && distancia[a] + cost < distancia[b])
44                     {
45                         distancia[b] = -INF; // Marcar nodo afectado por ciclo
46                         in_cycle[b] = true;
47                     }
48                 }
49             }
50             return distancia[n - 1]; // Distancia al último nodo
51         }
52     }
53
54 void solve() {
55     int n, m;
56     cin >> n >> m;
57
58     vector<tuple<int, int, int>> edges(m);
59     vector<vector<int>> graph(n), reverse_graph(n);
60
61     for (int i = 0; i < m; ++i) {
62         int a, b, c;
63         cin >> a >> b >> c;
64         --a, --b;
65         edges[i] = {a, b, -c}; // Invertimos el signo del costo para
66         // maximizar
67         graph[a].push_back(b);
68         reverse_graph[b].push_back(a);
69     }
70
71     // Verificar accesibilidad desde el inicio y al final
72     vector<bool> reachable_from_start(n, false), reachable_from_end(n,
73     false);
74     bfs(0, graph, reachable_from_start);
75     bfs(n - 1, reverse_graph, reachable_from_end);
76
77     // Ejecutar Bellman-Ford
78     int result = bellman_ford(n, 0, edges);
79
80     // Verificar si un ciclo negativo afecta la solución
81     for (int i = 0; i < n; ++i) {
82         if (reachable_from_start[i] && reachable_from_end[i] && result ==
83             -INF) {
84             cout << -1 << endl;
85             return;
86         }
87     }
88 }

```

```

82 }
83
84 // Si no hay ciclos negativos relevantes, devolver la distancia
85 cout << result << endl;
86 }
87
88 signed main() {
89     ios::sync_with_stdio(0);
90     cin.tie(0);
91     cout.tie(0);
92     solve();
93 }

```

## 5.0.5 Longest\_Flight\_Route

```

1 // Uolevi has won a contest, and the prize is a free flight trip that can
2 // consist of one or more flights through cities. Of course, Uolevi wants
3 // to choose a trip that has as many cities as possible.
4 // Uolevi wants to fly from Syrjälä to Lehmälä so that he visits the
5 // maximum number of cities. You are given the list of possible flights
6 // and you know that there are no directed cycles in the flight network.
7 #include <bits/stdc++.h>
8 using namespace std;
9 #define int long long
10 const int INF = 1000000000000000;
11 const int tam = 100000 + 1;
12 #define sz(x) (int)(x).size()
13 int n;
14 vector<vector<pair<int, int>>> grafo;
15 vector<bool> visi;
16 vector<int> topsort;
17 vector<int> dis;
18 vector<int> padre;
19 void dfs(int nodo) {
20     visi[nodo] = true;
21     for (auto [vecino, peso] : grafo[nodo]) {
22         if (!visi[vecino]) { dfs(vecino); }
23     }
24     topsort.push_back(nodo);
25 }
26
27 void encontrarCaminoMasLargo(int origen) {
28     for (int i = 1; i <= n; i++) {
29         if (!visi[i]) { dfs(i); }
30     }
31     reverse(topsort.begin(), topsort.end());
32     dis.resize(n + 1, -INF);
33     dis[origen] = 0;
34     for (int nodo : topsort) {
35         if (dis[nodo] == -INF) continue;
36         for (auto [vecino, peso] : grafo[nodo]) {
37             if (dis[vecino] < dis[nodo] + peso) {
38                 dis[vecino] = dis[nodo] + peso;
39                 padre[vecino] = nodo;
40             }
41         }
42     }
43 }
44
45 void solve() {
46     int m;
47     cin >> n >> m;
48     padre.resize(n + 1, -1);
49     grafo.resize(n + 1);
50     visi.resize(n + 1, false);
51     for (int i = 0; i < m; i++) {
52         int a, b;
53         cin >> a >> b;
54         grafo[a].push_back({b, 1});
55     }
56     encontrarCaminoMasLargo(1);
57     if (dis[n] == -INF) {
58         cout << "IMPOSSIBLE\n";
59         return;
60     }
61     cout << dis[n]+1 << endl;
62     int it = n;
63     deque<int> path;
64     path.push_front(n);
65     while (padre[it] != -1) {
66         path.push_front(padre[it]);
67         it = padre[it];
68     }
69     for(int i:path)cout<<i<<' ';
70     cout<<"\n";
71 }
72
73 signed main() {
74     // ios::sync_with_stdio(0);
75     // cin.tie(0);
76     // cout.tie(0);
77     int t = 1;
78     // cin >> t;
79     while (t--) { solve(); }
80 }

```

```

37     }
38 }
39
40 void solve() {
41     int m;
42     cin >> n >> m;
43     padre.resize(n + 1, -1);
44     grafo.resize(n + 1);
45     visi.resize(n + 1, false);
46     for (int i = 0; i < m; i++) {
47         int a, b;
48         cin >> a >> b;
49         grafo[a].push_back({b, 1});
50     }
51     encontrarCaminoMasLargo(1);
52     if (dis[n] == -INF) {
53         cout << "IMPOSSIBLE\n";
54         return;
55     }
56     cout << dis[n]+1 << endl;
57     int it = n;
58     deque<int> path;
59     path.push_front(n);
60     while (padre[it] != -1) {
61         path.push_front(padre[it]);
62         it = padre[it];
63     }
64     for(int i:path)cout<<i<<' ';
65     cout<<"\n";
66 }
67
68 signed main() {
69     // ios::sync_with_stdio(0);
70     // cin.tie(0);
71     // cout.tie(0);
72     int t = 1;
73     // cin >> t;
74     while (t--) { solve(); }
75 }

```

## 5.0.6 Monsters

---

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

2 // Your task is to find out if your goal is possible, and if it is, print  
3 // a path that you can follow. Your plan has to work in any situation;  
4 // even if the monsters know your path beforehand.

5 #include <bits/stdc++.h>

6 using namespace std;

7 #define int long long

8

```

9   int n, m;
10  vector<pair<int, int>> monstruos;
11  vector<vector<int>> lava(1000 + 10, vector<int>(1000 + 10, INT_MAX));
12  pair<int, int> inicio, fin;
13  // 0 = U  1 = D  2 = R  3 = L
14  //Estos indices se usan para calc
15  vector<pair<int, int>> mov = {{-1, 0}, {1, 0}, {0, 1}, {0, -1}};
16  char calc(int x){
17    char res;
18    switch (x)
19    {
20      case 0:
21        res = 'U';
22        break;
23      case 1:
24        res = 'D';
25        break;
26      case 2:
27        res = 'R';
28        break;
29      case 3:
30        res = 'L';
31        break;
32    }
33    return res;
34  }
35  vector<vector<char>> direcciones(1000 + 10, vector<char>(1000 + 10, '#'));
36  void reconstruircamino(){
37    vector<char> res;
38    auto[x, y] = fin;
39    while(direcciones[x][y] != '$'){
40      char aux = direcciones[x][y];
41      res.push_back(aux);
42      if(aux == 'U'){
43        x++;
44      }else if(aux == 'D'){
45        x--;
46      }else if(aux == 'L'){
47        y++;
48      }else{
49        y--;
50      }
51      cout<<(int)res.size()<<'\n';
52      reverse(res.begin(), res.end());
53      for(char& it:res){
54        cout<<it;
55      }
56      cout<<'\n';
57    }
58    bool esvalida(int x, int y, int tiempo)
59    {
60      if (x < 0 or y < 0 or x >= n or y >= m)
61      {
62        return false;
63      }
64      if (lava[x][y] <= tiempo)
65      {
66
67        return false;
68      }
69      return true;
70    }
71
72  bool esSalida(int x, int y, int tiempo)
73  {
74    if (!esvalida(x, y, tiempo))
75      return false;
76    if (x == 0 or y == 0 or
77        x == n - 1 or y == m - 1)
78      return true;
79    return false;
80  }
81
82  bool bfsDeSalida()
83  {
84    queue<pair<pair<int, int>, int>> q;
85    q.push(make_pair(inicio, 0));
86    direcciones[inicio.first][inicio.second] = '$';
87    while (!q.empty())
88    {
89      int cx = q.front().first.first;
90      int cy = q.front().first.second;
91      int tiempo = q.front().second;
92      tiempo++;
93      q.pop();
94      for (int i = 0; i<4; i++)
95      {
96        auto mv = mov[i];
97        int tx = cx + mv.first;
98        int ty = cy + mv.second;
99        if (esSalida(tx, ty, tiempo))
100        {
101          direcciones[tx][ty] = calc(i);
102          fin = {tx, ty};
103          return true;
104        }
105        if (esvalida(tx, ty, tiempo))
106        {
107          direcciones[tx][ty] = calc(i);
108          lava[tx][ty] = tiempo;
109          q.push({{tx, ty}, tiempo});
110        }
111      }
112    }
113    return false;
114  }
115
116  void precalculolava()
117  {
118    queue<pair<pair<int, int>, int>> q;
119    for (auto m : monstruos)
120    {
121      q.push(make_pair(m, 0));
122    }
123    while (!q.empty())
124    {

```

```

125     int cx = q.front().first.first;
126     int cy = q.front().first.second;
127     int tiempo = q.front().second;
128     tiempo++;
129     q.pop();
130
131     for (auto mv : mov)
132     {
133         int tx = cx + mv.first;
134         int ty = cy + mv.second;
135         if (esvalida(tx, ty, tiempo))
136         {
137             lava[tx][ty] = tiempo;
138             q.push({{tx, ty}, tiempo});
139         }
140     }
141 }
142
143 signed main()
144 {
145     ios_base::sync_with_stdio(false);
146     cin.tie(NULL);
147     cin >> n >> m;
148     for (int i = 0; i < n; ++i)
149     {
150         for (int j = 0; j < m; ++j)
151         {
152             char c;
153             cin >> c;
154             if (c == '#')
155             {
156                 lava[i][j] = 0;
157             }
158             else if (c == 'M')
159             {
160                 lava[i][j] = 0;
161
162                 monstruos.push_back({i, j});
163             }
164             else if (c == 'A')
165             {
166                 lava[i][j] = 0;
167                 inicio = {i, j};
168             }
169             else
170             {
171                 lava[i][j] = INT_MAX;
172             }
173         }
174     }
175     if (inicio.first == 0 or inicio.second == 0 or inicio.first == n - 1
176     or inicio.second == m - 1)
177     {
178         cout << "YES" << endl;
179         cout << 0;
180         return 0;
181     }

```

```

182     precalculoLava();
183
184     if (!bfsDeSalida())
185     {
186         cout << "NO";
187         return 0;
188     }
189     cout << "YES" << endl;
190     reconstruircamino();
191 }

```

## 5.0.7 Round\_Trip

```

1 // Byteland has n cities and m roads between them. Your task is to design
2 // a round trip that begins in a city, goes through two or more other
3 // cities, and finally returns to the starting city. Every intermediate
4 // city on the route has to be distinct.
5 #include <bits/stdc++.h>
6 using namespace std;
7 #define int long long
8 #define uset unordered_set
9 #define umap unordered_map
10 #define mp make_pair
11 #define pb push_back
12 #define all(a) (a).begin(), (a).end()
13 #define rall(a) (a).rbegin(), (a).rend()
14 #define floatigual(a, b) (fabs(a - b) < EPS)
15 #define mod(a) md(a, MOD)
16 #define FOR(i, n) for (int i = 0; i < (n); ++i)
17 #define FOR3(i, a, b) for (int i = (a); i < (b); ++i)
18 #define FORD(i, n) for (int i = (n) - 1; i ≥ 0; --i)
19 #define FORDD(i, a, b) for (int i = (b) - 1; i ≥ (a); --i)
20 #define techo(a, b) (a / b + (a % b ≠ 0))
21 #define popcount(x) __builtin_popcountll(x);
22 using namespace std;
23 typedef long double ld;
24 typedef unsigned long long ull;
25 typedef pair<int, int> pii;
26 typedef vector<int> vi;
27 typedef vector<bool> vbol;
28
29 void solve() {
30     int n, m;
31     cin >> n >> m;
32     vector<vi> g(n + 1);
33     vbol visi(n + 1);
34     vi losamopadres(n + 1);
35
36     while (m--) {
37         int a, b;
38         cin >> a >> b;
39         g[a].pb(b);
40         g[b].pb(a);
41     }
42 }

```

```

40
41     bool hay = false;
42     int com = -1, fin = -1;
43
44     FOR3(i, 1, n + 1) {
45         if (visi[i]) continue;
46         stack<pii> pila;
47
48         // Nodo actual, nodo padre
49         pila.push({i, -1});
50         losamopadres[i] = -1;
51
52         while (!pila.empty()) {
53             auto [nodo, padre] = pila.top();
54             pila.pop();
55
56             visi[nodo] = true;
57             losamopadres[nodo] = padre;
58
59             for (int vecino : g[nodo]) {
60                 if (vecino == padre) continue;
61
62                 if (visi[vecino]) {
63                     // Si ya fue visitado, detectamos el ciclo
64                     com = vecino;
65                     fin = nodo;
66                     hay = true;
67                     break;
68                 }
69
70                 if (!visi[vecino]) {
71                     pila.push({vecino, nodo});
72                 }
73             }
74             if (hay) break;
75         }
76         if (hay) {
77             // Reconstrucción del ciclo
78             vi res;
79             res.pb(fin); // Nodo donde se detectó el ciclo
80             while (fin != com) {
81                 res.pb(losamopadres[fin]);
82                 fin = losamopadres[fin];
83             }
84             //res.pb(com); // Anadimos el nodo donde comenzó el ciclo
85             res.pb(res[0]); // Para cerrar el ciclo
86
87             cout << res.size() << endl;
88             for (int i : res) {
89                 cout << i << ' ';
90             }
91             return;
92         }
93     cout << "IMPOSSIBLE\n";
94 }
95
96 signed main() {
97     //int t;cin>>t;while(t--)solve();

```

```
98 solve();  
99 }  
  


---

  
5.0.8 Round  
  
1 // Byteland has  
2 → a round trip  
3 → cities, and  
4 → city on the  
5 #include <bits/stdc++.h>  
6 using namespace std;  
7  
8 const int tam = 1000000;  
9 int n, m;  
10 vector<int> g[tam];  
11 int estado[tam];  
12 vector<int> path;  
13 pair<int,int> ese;  
14  
15 bool dfs(int u){  
16     estado[u] = 1;  
17     if(estado[u] == 2) return true;  
18     if(dfs(g[u][0])){  
19         ese = {g[u][0], u};  
20         return true;  
21     } else if(dfs(g[u][1])){  
22         ese = {g[u][1], u};  
23     }  
24     path.push_back(u);  
25     if(path.size() == m) return false;  
26     estado[u] = 2;  
27     return true;  
28 }  
29  
30 void solve(){  
31     cin >> n >> m;  
32     for(int i = 0; i < m; i++){  
33         fill(estado, 0);  
34         path.clear();  
35         ese = {-1, -1};  
36         for(int j = 0; j < n; j++){  
37             int a, b;  
38             cin >> a >> b;  
39             g[a].push_back(b);  
40         }  
41     }  
42     for(int i = 0; i < m; i++){  
43         if(estado[i] == 0){  
44             if(dfs(i))  
45                 if(ese.first != -1)  
46                     if(ese.first == ese.second)  
47                         if(ese.first == i)  
48                             cout << "YES"  
49                         else cout << "NO";  
50                     else cout << "NO";  
51                 else cout << "NO";  
52             else cout << "NO";  
53         }  
54     }  
55 }
```

## 5.0.8 Round\_Trip\_II

```
48     cout << "IMPOSSIBLE\n";
49     return;
50 }
51
52 vector<int> ans;
53 bool started = false;
54 for(int node : path) {
55     if(node == ese.second) started = true;
56     if(started) ans.push_back(node);
57     if(node == ese.first and started) break;
58 }
59 ans.push_back(ese.second);
60
61 cout << ans.size() << "\n";
62 for(int x : ans) cout << x << " ";
63 cout << "\n";
64 }
65
66 int main() {
67     ios::sync_with_stdio(false);
68     cin.tie(nullptr);
69
70     int t = 1;
71     while(t--) solve();
72 }
```

---

```
28     }
29 }
30
31 void solve()
32 {
33     int n, m;cin>>n>>m;
34     vector<vector<pii>> grafo(n+1);
35     while(m--){
36         int a, b, c;cin>>a>>b>>c;
37         grafo[a].pb({b,c});
38     }
39     dijkstra(graf, n, 1);
40     cout<<'\n';
41 }
42
43 signed main()
44 {
45     ios::sync_with_stdio(0);
46     cin.tie(0);
47     cout.tie(0);
48     int t;
49     //cin >> t;
50     //while (t--)
51     //    solve();
52     solve();
53 }
```

### **5.0.9 Shortest\_Routes**

```
1 #include<bits/stdc++.h>
2 using namespace std;
3
4 const int INF = LONG_MAX/100;
5
6
7 void dijkstra(vector<vector<pii>>& grafo, int cant, int ini){
8     vi dis(cant+1, INF);
9     dis[ini] = 0;
10    priority_queue<pii, vector<pii>, greater<pii>> pq;
11    pq.push({0, ini});
12    while(!pq.empty()){
13        pii menor = pq.top(); pq.pop();
14        int w = menor.first;
15        int nodo = menor.second;
16        if(dis[nodo]<w)continue;
17        for(auto a:grafo[nodo]){
18            int v = a.first;
19            int costo = a.second;
20            if(dis[nodo] + costo <dis[v]){
21                dis[v] = dis[nodo]+costo;
22                pq.push({dis[v], v});
23            }
24        }
25    }
26    for(int i = 1;i<=cant; i++){
27        cout< $\langle$ dis[i]< $\rangle$  :
```

### 5.0.10 Shortest\_Routes\_II

```

1 // There are n cities and m roads between them. Your task is to process q
2 → queries where you have to determine the length of the shortest route
3 → between two given cities.
4 #include <bits/stdc++.h>
5 using namespace std;
6 #define int ll
7 #define uset unordered_set
8 #define umap unordered_map
9 #define mp make_pair
10 #define pb push_back
11 #define all(a) (a).begin(), (a).end()
12 #define rall(a) (a).rbegin(), (a).rend()
13 #define floatigual(a, b) (fabs(a - b) < EPS)
14 #define mod(a) md(a, MOD)
15 #define FOR(i, n) for (int i = 0; i < (n); ++i)
16 #define FOR3(i, a, b) for (int i = (a); i < (b); ++i)
17 #define FORD(i, n) for (int i = (n) - 1; i ≥ 0; --i)
18 #define FORDD(i, a, b) for (int i = (b) - 1; i ≥ (a); --i)
19 #define si cout << "YES" << endl
20 #define no cout << "NO" << endl
21
22 vector<vi> floyd(vector<vector<pii>>& grafo, int n){
23     vector<vector<int>> dis(n+1, vi(n+1, INF));
24     for(int i = 1;i<=n; i++){
25         for(int j = 1;j<=n; j++){
26             if(i == j) dis[i][j] = 0;
27             else dis[i][j] = grafo[i-1][j-1].second;
28         }
29     }
30     for(int k = 1;k<=n; k++){
31         for(int i = 1;i<=n; i++){
32             for(int j = 1;j<=n; j++){
33                 if(dis[i][j] > dis[i][k] + dis[k][j]) dis[i][j] = dis[i][k] + dis[k][j];
34             }
35         }
36     }
37     return dis;
38 }
```

```

23     dis[i][i] = 0;
24 }
25 FOR3(u, 1, n+1){
26     for(auto donde:grafo[u]){
27         int v = donde.first; int w = donde.second;
28         dis[u][v] = min(dis[u][v], w);
29         dis[v][u] = min(dis[v][u], w);
30     }
31 }
32 FOR3(k, 1, n+1){
33     FOR3(u, 1, n+1){
34         FOR3(v, 1, n+1){
35             dis[u][v] = min(dis[u][v], dis[u][k] + dis[k][v]);
36         }
37     }
38 }
39 return dis;
40 }
41
42 void solve()
43 {
44 //Foi warshal pe
45 int n, m, q; cin >> n >> m >> q;
46 unionFind dsu(n+1);
47 vector<vector<pii>> grafo(n+1);
48 while(m--){
49     int a, b, c; cin >> a >> b >> c;
50     grafo[a].pb({b, c});
51     grafo[b].pb({a, c});
52     dsu.join(a, b);
53 }
54 auto var = floyd(grafo, n);
55 while(q--){
56     int a, b; cin >> a >> b;
57     if(dsu.find(a) != dsu.find(b)){
58         cout << "-1\n";
59     } else{
60         cout << var[a][b] << '\n';
61     }
62 }
63
64 signed main()
65 {
66
67     ios::sync_with_stdio(0);
68     cin.tie(0);
69     cout.tie(0);
70     int t;
71     //cin >> t;
72     //while (t--)
73     //    solve();
74     solve();
75 }

```

## 6 Introductory Problems

### 6.0.1 Apple\_Division

```

1 #include <bits/stdc++.h>
2 #define int long long
3 using namespace std;
4 int arr[21];
5 void dp(pair<int, int> par, int &mini, int i, int n) {
6     if (i == n) {
7         mini = min(mini, abs(par.first - par.second));
8     } else {
9         par.first += arr[i];
10        dp(par, mini, i + 1, n);
11        par.first -= arr[i];
12        par.second += arr[i];
13        dp(par, mini, i + 1, n);
14    }
15 }
16 const int INF = 1e18;
17 signed main() {
18     int n;
19     cin >> n;
20     for (int i = 0; i < n; i++) cin >> arr[i];
21     pair<int, int> ini = {0, 0};
22     int mini = INF;
23     dp(ini, mini, 0, n);
24     cout << mini << endl;
25 }

```

### 6.0.2 Bit.Strings

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define ll long long
4 #define ull unsigned long long
5 #define fore(i, a, b) for(int i = (a); i < (b); i++)
6 #define FOR(i, n) for(int i = 0; i < (n); i++)
7 #define all(x) (x).begin(), (x).end()
8 #define sz(x) (int)(x).size()
9 #define pb push_back
10 using vi = vector<int>;
11 const int MOD = 1000000007;
12 const ll INF = 9223372036854775807LL;
13 const int tam = 1;
14
15 int expMod(int base, int exponente, int mod) {
16     int res = 1;
17     base %= mod;
18     while (exponente > 0) {
19         if (exponente % 2 == 1)
20             res = (res * base) % mod;
21         exponente >>= 1;
22         base = (base * base) % mod;
23     }

```

```

24     return res;
25 }
26
27 void solve(){
28     int n;cin>>n;cout<<expMod(2,n,MOD);
29 }
30
31 signed main(){
32
33     ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
34     //int t;cin>>t;while(t--)solve();
35     solve();
36 }

```

### 6.0.3 Coin\_Piles

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define ll long long
4 #define ull unsigned long long
5 #define fore(i, a, b) for (int i = (a); i < (b); i++)
6 #define FOR(i, n) for (int i = 0; i < (n); i++)
7 #define all(x) (x).begin(), (x).end()
8 #define sz(x) (int)(x).size()
9 #define pb push_back
10 using vi = vector<int>;
11 const int MOD = 1000000007;
12 const ll INF = 9223372036854775807LL;
13 const int tam = 1;
14 //You have two coin piles containing a and b coins.
15 //On each move, you can either remove one coin from the left pile
16 //and two coins from the right pile, or two coins from
17 //the left pile and one coin from the right pile.
18 //Your task is to efficiently find out if you can empty both the piles.
19 void solve() {
20     int a, b;
21     cin >> a >> b;
22     if (a > b) { swap(a, b); }
23     a = a * 2 - b;
24     if (a % 3 == 0 && a >= 0) {
25         cout << "YES\n";
26     } else {
27         cout << "NO\n";
28     }
29 }
30
31 signed main() {
32     ios::sync_with_stdio(0);
33     cin.tie(0);
34     cout.tie(0);
35     int t;
36     cin >> t;
37     while (t--) solve();
38     // solve();

```

```

39   }

```

### 6.0.4 Creating.Strings

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define ll long long
4 #define ull unsigned long long
5 #define fore(i, a, b) for (int i = (a); i < (b); i++)
6 #define FOR(i, n) for (int i = 0; i < (n); i++)
7 #define all(x) (x).begin(), (x).end()
8 #define sz(x) (int)(x).size()
9 #define pb push_back
10 using vi = vector<int>;
11 const int MOD = 1000000007;
12 const ll INF = 9223372036854775807LL;
13 const int tam = 1;
14
15 void solve() {
16     string cad;
17     cin >> cad;
18     sort(all(cad));
19     vector<string> vec;
20     do { vec.pb(cad); } while (next_permutation(all(cad)));
21     cout << vec.size() << endl;
22     sort(all(vec));
23     for (auto &c : vec) { cout << c << endl; }
24 }
25
26 signed main() {
27     ios::sync_with_stdio(0);
28     cin.tie(0);
29     cout.tie(0);
30     // int t;cin>>t;while(t--)solve();
31     solve();
32 }

```

### 6.0.5 Gray\_Code

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define ll long long
4 #define ull unsigned long long
5 #define fore(i, a, b) for (int i = (a); i < (b); i++)
6 #define FOR(i, n) for (int i = 0; i < (n); i++)
7 #define all(x) (x).begin(), (x).end()
8 #define sz(x) (int)(x).size()
9 #define pb push_back
10 using vi = vector<int>;
11 const int MOD = 1000000007;
12 const ll INF = 9223372036854775807LL;
13 const int tam = 1;
14

```

```

15 int binpow(int a, int b) {
16     int res = 1;
17     while (b > 0) {
18         if (b & 1)
19             res = res * a;
20         a = a * a;
21         b >>= 1;
22     }
23     return res;
24 }
25
26 void solve() {
27     int n;
28     cin >> n;
29     int l = binpow(2, n);
30     for (int i = 0; i < l; i++) {
31         int x = i ^ (i >> 1);
32         bitset<32> b(x);
33         string s = b.to_string();
34         cout << s.substr(32 - n) << endl;
35     }
36
37 signed main() {
38     ios::sync_with_stdio(0);
39     cin.tie(0);
40     cout.tie(0);
41     solve();
42     // int t;cin>>t;while(t--)solve();
43 }
44

```

---

## 6.0.6 Increasing\_Array

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define ll long long
4 #define ull unsigned long long
5 #define fore(i, a, b) for(int i = (a); i<(b); i++)
6 #define FOR(i, n) for(int i = 0; i<(n); i++)
7 #define all(x) (x).begin(), (x).end()
8 #define sz(x) (int)(x).size()
9 #define pb push_back
10 using vi = vector<int>;
11 const int MOD = 1000000007;
12 const ll INF = 9223372036854775807LL;
13 const int tam = 1;
14
15 void solve(){
16     int n;cin>>n;
17     vi vec(n);FOR(i, n)cin>>vec[i];
18     int cont = 0;
19     for(int i = 1;i<n; i++){
20         if(vec[i]<vec[i-1]){
21             int mov = vec[i-1]-vec[i];
22             cont+=mov;

```

```

23                 vec[i]+=mov;
24             }
25         }
26         cout<<cont<<'\n';
27     }
28     signed main(){
29         ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
30         //int t;cin>>t;while(t--)solve();
31         solve();
32     }
33

```

---

## 6.0.7 Missing\_Number

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define ll long long
4 #define ull unsigned long long
5 #define fore(i, a, b) for(int i = (a); i<(b); i++)
6 #define FOR(i, n) for(int i = 0; i<(n); i++)
7 #define all(x) (x).begin(), (x).end()
8 #define sz(x) (int)(x).size()
9 #define pb push_back
10 using vi = vector<int>;
11 const int MOD = 1000000007;
12 const ll INF = 9223372036854775807LL;
13 const int tam = 1;
14
15 void solve(){
16     int n;cin>>n;
17     vi vec(n-1);
18     FOR(i, n)cin>>vec[i];
19     sort(all(vec));
20     int mex = 1;
21     for(int a:vec){
22         if(mex == a){
23             mex++;
24         }else{
25             cout<<mex<<endl;
26             return;
27         }
28     }
29     cout<<mex<<endl;
30 }
31
32 signed main(){
33     ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
34     //int t;cin>>t;while(t--)solve();
35 }
36

```

---

### 6.0.8 Palindrome\_Reorder

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 #define ll long long
4 #define ull unsigned long long
5 #define fore(i, a, b) for (int i = (a); i < (b); i++)
6 #define FOR(i, n) for (int i = 0; i < (n); i++)
7 #define all(x) (x).begin(), (x).end()
8 #define sz(x) (int)(x).size()
9 #define pb push_back
10 using vi = vector<int>;
11 const int MOD = 1000000007;
12 const ll INF = 9223372036854775807LL;
13 const int tam = 1;
14
15 string stringear(int a) {
16     char c = (char)a;
17     string res = "";
18     res += c;
19     return res;
20 }
21
22 void solve() {
23     string cad;
24     cin >> cad;
25     int n = (int)cad.size();
26     vector<int> letras(26);
27     for (char &c : cad) { letras[c - 'A']++; }
28     int imp = 0;
29     char algo;
30     for (int i = 0; i < 26; i++) {
31         int rep = letras[i];
32         if (rep % 2 != 0) {
33             imp++;
34             algo = char(i + 'A');
35         }
36     }
37     if (imp > 1) {
38         cout << "NO SOLUTION\n";
39         return;
40     }
41     string res = "";
42     for (int i = 0; i < 26; i++) {
43         if (letras[i] % 2 != 0) { continue; }
44         for (int j = 0; j < letras[i] / 2; j++) {
45             res.append(stringear(i + 'A'));
46         }
47     }
48     if (imp != 0) {
49         cout << res;
50         for (int i = 0; i < letras[algo - 'A']; i++) { cout << algo; }
51     } else {
52         cout << res;
53     }
54     for (int i = (int)res.size() - 1; i >= 0; i--) { cout << res[i]; }
55 }
```

```
56
57 signed main() {
58     ios::sync_with_stdio(0);
59     cin.tie(0);
60     cout.tie(0);
61     // int t;cin>>t;while(t--)solve();
62     solve();
63 }
```

### 6.0.9 Permutation

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 #define ll long long
4 #define ull unsigned long long
5 #define fore(i, a, b) for (int i = (a); i < (b); i++)
6 #define FOR(i, n) for (int i = 0; i < (n); i++)
7 #define all(x) (x).begin(), (x).end()
8 #define sz(x) (int)(x).size()
9 #define pb push_back
10 using vi = vector<int>;
11 const int MOD = 1000000007;
12 const ll INF = 9223372036854775807LL;
13 const int tam = 1;
14
15 void solve() {
16     int n;
17     cin >> n;
18     if (n == 2 || n == 3) {
19         cout << "NO SOLUTION\n";
20     } else if (n == 4) {
21         cout << "2 4 1 3";
22     } else {
23         vi vec(n);
24         int cont = n;
25         for (int i = 0; i < n; i += 2) { vec[i] = cont--; }
26         for (int i = 1; i < n; i += 2) { vec[i] = cont--; }
27         for (auto a : vec) { cout << a << ' ';}
28         cout << '\n';
29     }
30 }
31 signed main() {
32     ios::sync_with_stdio(0);
33     cin.tie(0);
34     cout.tie(0);
35     // int t;cin>>t;while(t--)solve();
36     solve();
37 }
```

### 6.0.10 Repetitions

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

```

3 #define ll long long
4 #define ull unsigned long long
5 #define fore(i, a, b) for(int i = (a); i<(b); i++)
6 #define FOR(i, n) for(int i = 0; i<(n); i++)
7 #define all(x) (x).begin(), (x).end()
8 #define sz(x) (int)(x).size()
9 #define pb push_back
10 using vi = vector<int>;
11 const int MOD = 1000000007;
12 const ll INF = 9223372036854775807LL;
13 const int tam = 1;
14
15 void solve(){
16     string cad;cin>>cad;
17     int n = (int)(cad.size());
18     int cont = 1;
19     int m = 1;
20     for(int i = 1; i < n; i++){
21         if(cad[i] == cad[i - 1]){
22             cont++;
23         }else{
24             m = max(m, cont);
25             cont = 1;
26         }
27     }
28     m = max(m, cont);
29     cout<<m<<endl;
30 }
31 signed main(){
32     ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
33     //int t;cin>>t;while(t--)solve();
34     solve();
35 }
36
19 priority_queue<pair<char, int>, vector<pair<char, int>>, greater<pair<char, int>>>
20     pq;
21     for (auto x : mp) pq.push(x);
22     for (int i = 0; i < n; i++) {
23         deque<pair<char, int>> otro;
24         bool found = false;
25         pair<char, int> elegido;
26
27         while (!pq.empty()) {
28             auto top = pq.top();
29             pq.pop();
30             if (!found && top.first != last) {
31                 elegido = top;
32                 found = true;
33             } else {
34                 otro.push_back(top);
35             }
36         }
37
38         if (!found) {
39             if (otro.empty()) {
40                 elegido = {last, 0};
41                 found = true;
42             } else {
43                 cout << "-1\n";
44                 return;
45             }
46         }
47         ans[i] = elegido.first;
48         last = elegido.first;
49         elegido.second--;
50         if (elegido.second > 0) pq.push(elegido);
51         while (!otro.empty()) pq.push(otro.front()), otro.pop_front();
52     }
53     for (char c : ans) cout << c;
54     cout << '\n';
55 }
56
57
58 signed main() {
59     int t = 1;
60     // cin>>t;
61     while (t--) solve();
62 }

```

### 6.0.11 String\_Reorder

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define sz(x) (int)(x).size()
4 void solve() {
5     string s;
6     cin >> s;
7     vector<char> ans(sz(s));
8     map<char, int> mp;
9     int n = sz(s);
10    int maxi = 0;
11    for (char& c : s) {
12        mp[c]++;
13        maxi = max(maxi, mp[c]);
14    }
15    if (maxi > (n + 1) / 2) {
16        cout << "-1\n";
17    } else {
18        char last = '(';

```

### 6.0.12 Trailing\_Zeros

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define ll long long
4 #define ull unsigned long long
5 #define fore(i, a, b) for (int i = (a); i < (b); i++)
6 #define FOR(i, n) for (int i = 0; i < (n); i++)
7 #define all(x) (x).begin(), (x).end()
8 #define sz(x) (int)(x).size()

```

```

9 #define pb push_back
10 using vi = vector<int>;
11 const int MOD = 1000000007;
12 const ll INF = 9223372036854775807LL;
13 const int tam = 1;
14 //Your task is to calculate the number of trailing zeros in the factorial
15 → n!.
16 //For example, 20! = 2432902008176640000 and it has 4 trailing zeros.
17 void solve() {
18     int n;
19     cin >> n;
20     int cont = 0;
21     for (int i = 5; n / i >= 1; i *= 5) { cont += n / i; }
22     cout << cont << '\n';
23 }
24 signed main() {
25     ios::sync_with_stdio(0);
26     cin.tie(0);
27     cout.tie(0);
28     // int t;cin>>t;while(t--)solve();
29     solve();
30 }
```

### 6.0.13 Two\_Knights

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 #define int long long
4 signed main(){
5     int n;
6     cin>>n;
7     for(int i=1;i<=n;i++){
8         int totManeras = (i*i)*(i*i-1) / 2;
9         int totAtaques = 4*(i-1)*(i-2);
10        cout<<totManeras-totAtaques<<endl;
11    }
12    return 0;
13 }
```

### 6.0.14 Two\_Sets

```

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

```

12 const ll INF = 9223372036854775807LL;
13 const int tam = 1;
14
15 void solve() {
16     ll n;
17     cin >> n;
18     ll suma = n * (n + 1) / 4;
19     if (n * (n + 1) % 4) {
20         cout << "NO\n";
21     } else {
22         cout << "YES\n";
23         ll target = suma;
24         set<int> uno, dos;
25         ll sumUno = 0;
26         for (int i = n; i >= 1; i--) {
27             if (sumUno + i <= target) {
28                 uno.insert(i);
29                 sumUno += i;
30             } else {
31                 dos.insert(i);
32             }
33         }
34         cout << sz(uno) << "\n";
35         for (int x : uno) cout << x << " ";
36         cout << "\n" << sz(dos) << "\n";
37         for (int x : dos) cout << x << " ";
38         cout << "\n";
39     }
40 }
```

```

41 signed main() {
42     ios::sync_with_stdio(0);
43     cin.tie(0);
44     cout.tie(0);
45     int t = 1;
46     // cin>>t;
47     while (t--) { solve(); }
48 }
```

### 6.0.15 Weird\_Algorithm

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define ll long long
4 #define ull unsigned long long
5 #define fore(i, a, b) for(int i = (a); i < (b); i++)
6 #define FOR(i, n) for(int i = 0; i < (n); i++)
7 #define all(x) (x).begin(), (x).end()
8 #define sz(x) (int)(x).size()
9 #define pb push_back
10 using vi = vector<int>;
11 const int MOD = 1000000007;
12 const ll INF = 9223372036854775807LL;
13 const int tam = 1;
14
```

```

15 void collatz(int n){
16     if(n == 1){
17         return;
18     }else if(!(n%2)){
19         cout<<n/2<<' ';
20         collatz(n/2);
21     }else{
22         cout<<3*n + 1<<' ';
23         collatz(3*n+1);
24     }
25 }
26
27 void solve(){
28     int n;cin>>n;
29     cout<<n<<' ';
30     collatz(n);
31 }
32
33 signed main(){
34
35     ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
36     //int t;cin>>t;while(t--)solve();
37     solve();
38 }

```

---

```

27         a = a * a % m;
28         b >>= 1;
29     }
30     return res;
31 }
32 void prec(){
33     fact[0] = 1;
34     for (int i = 1; i < tam; i++) {
35         fact[i] = fact[i - 1] * i % MOD;
36     }
37     invFact[tam - 1] = expmod(fact[tam - 1], MOD - 2, MOD);
38     for (int i = tam - 2; i >= 0; i--) {
39         invFact[i] = invFact[i + 1] * (i + 1) % MOD;
40     }
41 }
42 ll catalan(int n){
43     return fact[2*n] * invFact[n+1] %MOD * invFact[n]%MOD;
44 }
45 void solve(){
46     int n;cin>>n;
47     if(n&1){
48         cout<<"0\n";
49         return;
50     }
51     cout<<catalan(n/2)<<endl;
52 }
53
54 signed main(){
55     prec();
56     ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
57     int t = 1;
58     //cin>>t;
59     while(t--){
60         solve();
61     }
62 }

```

## 7 Mathematics

### 7.0.1 Bracket\_Sequences\_I

```

1 // Your task is to calculate the number of valid bracket sequences of
2 // length n. For example, when n=6, there are 5 sequences:
3 // ()()
4 // ()(())
5 // ((()))
6 // ((()))
7 // ((()())
8 #include <bits/stdc++.h>
9 using namespace std;
10#define ll long long
11#define ull unsigned long long
12#define fore(i, a, b) for(int i = (a); i<(b); i++)
13#define FOR(i, n) for(int i = 0; i<(n); i++)
14#define all(x) (x).begin(), (x).end()
15#define sz(x) (int)(x).size()
16#define pb push_back
17 using vi = vector<int>;
18 const int MOD = 1000000007;
19 const ll INF = 9223372036854775807LL;
20 const int tam = 1000000 + 20;
21 ll fact[tam], invFact[tam];
22 ll expmod(ll a, ll b, ll m) {
23     a %= m;
24     int res = 1;
25     while (b > 0) {
26         if (b & 1)

```

```

            res = res * 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;
    }
}

```

### 7.0.2 Christmas\_Party

```

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

```

```

15 const int tam = 1000000 + 10;
16 ll des[tam];
17
18 void precdes() {
19     des[0] = 1;
20     des[1] = 0;
21     for (int i = 2; i < tam; ++i) {
22         des[i] = (i - 1) * ((des[i - 1] + des[i - 2]) % MOD) % MOD;
23     }
24 }
25
26 void solve() {
27     int n;
28     cin >> n;
29     cout << des[n] << endl;
30 }
31
32 signed main() {
33     precdes();
34     ios::sync_with_stdio(0);
35     cin.tie(0);
36     cout.tie(0);
37
38     int t = 1;
39     // cin >> t;
40     while (t--) { solve(); }
41 }
42

```

---

```

25 // integers such that their greatest common divisor is as large as
26 // possible.
27
28 void solve() {
29     int n;
30     cin >> n;
31     memset(freq, 0, sizeof freq);
32     for (int& i : arr) cin >> i, freq[i]++;
33     memset(cnt, 0, sizeof cnt);
34     for (int i = 1; i < tam; i++) {
35         for (int j = i; j < tam; j += i) { cnt[i] += freq[j]; }
36     }
37     for (int i = tam - 1; i >= 1; i--) {
38         if (cnt[i] >= 2) {
39             cout << i << '\n';
40             return;
41         }
42     }
43
44 signed main() {
45     ios::sync_with_stdio(0);
46     cin.tie(0);
47     cout.tie(0);
48     int t = 1;
49     // cin>>t;
50     while (t--) { solve(); }
51 }

```

### 7.0.3 Common\_Divisors

```

1 #include <bits/stdc++.h>
2 // sin doubles
3 #pragma GCC optimize("Ofast")
4 #pragma GCC target("avx2,bmi,bmi2,lzcnt,popcnt")
5 #pragma GCC optimize("unroll-loops")
6 // con doubles
7 using namespace std;
8 #define int long long
9 #define ll long long
10 #define ull unsigned long long
11 #define fore(i, a, b) for (int i = (a); i < (b); i++)
12 #define FOR(i, n) for (int i = 0; i < (n); i++)
13 #define all(x) (x).begin(), (x).end()
14 #define sz(x) (int)(x).size()
15 #define pb push_back
16 using vi = vector<int>;
17 const int MOD = 1000000007;
18 const ll INF = 9223372036854775807LL;
19 const int tam = 1e6 + 100;
20 int freq[tam];
21 int arr[tam];
22 int cnt[tam];
23
// You are given an array of n positive integers. Your task is to find two

```

### 7.0.4 Distribuiting\_Apples

```

1 // There are n children and m apples that will be distributed to them.
2 // Your task is to count the number of ways this can be done.
3 // For example, if n=3 and m=2, there are 6 ways: [0,0,2], [0,1,1],
4 // [0,2,0], [1,0,1], [1,1,0] and [2,0,0].
5 #include <bits/stdc++.h>
6 using namespace std;
7
8 #define ll long long
9 const int MOD = 1000000007;
10 const int tam = 2 * 1000000 + 10; // Límite para los factoriales
11
12 ll fact[tam], invFact[tam];
13
14 ll expmod(ll a, ll b, ll m) {
15     a %= m;
16     ll res = 1;
17     while (b > 0) {
18         if (b & 1) res = res * a % m;
19         a = a * a % m;
20         b >>= 1;
21     }
22     return res;
23 }
24
void prec() {

```

```

24 fact[0] = 1;
25 for (int i = 1; i < tam; i++) {
26     fact[i] = fact[i - 1] * i % MOD;
27 }
28 invFact[tam - 1] = expmod(fact[tam - 1], MOD - 2, MOD); // Inverso del factorial más grande
29 for (int i = tam - 2; i >= 0; i--) {
30     invFact[i] = invFact[i + 1] * (i + 1) % MOD; // Calculamos los inversos de forma descendente
31 }
32
33 ll nck(int n, int k) {
34     if (k > n) return 0;
35     return fact[n] * invFact[k] % MOD * invFact[n - k] % MOD;
36 }
37
38 int main() {
39     prec(); // Precomputamos los factoriales e inversos
40     int n, m;
41     cin >> n >> m;
42     cout << nck(n + m - 1, m) << endl; // Combinaciones con repetición
43     return 0;
44 }

```

---

```

26         res = res * a % m;
27         a = a * a % m;
28         b >= 1;
29     }
30     return res;
31 }
32 bint prodiv(map<bint, bint>& factores) {
33     const bint MOD1 = (bint)(MOD - 1);
34     const bint MOD2 = MOD1 * 2;
35
36     bint cant_mod2 = 1;
37     for (auto [p, k] : factores) {
38         cant_mod2 = (cant_mod2 * (k + 1)) % MOD2;
39     }
40
41     bint cant_div2_mod1 = (cant_mod2 / 2) % MOD1;
42
43     bint ans = 1;
44     for (auto [p, k] : factores) {
45         bint nuevo = ( (k % MOD1) * cant_div2_mod1 ) % MOD1;
46         ans = ans * expmod(p, nuevo, (bint)MOD) % (bint)MOD;
47     }
48     if (cant_mod2 % 2 == 1) {
49         bint raiz = 1;
50         for (auto [p, k] : factores) {
51             raiz = raiz * expmod(p, k / 2, (bint)MOD) % (bint)MOD;
52         }
53     }
54 }
```

### 7.0.5 Divisor\_Analysis

```

1 // Given an integer, your task is to find the number, sum and product of
2 // its divisors. As an example, let us consider the number 12:
3 // the number of divisors is 6 (they are 1, 2, 3, 4, 6, 12)
4 // the sum of divisors is 1+2+3+4+6+12=28
5 // the product of divisors is 1 \cdot 2 \cdot 3 \cdot 4 \cdot 6 \cdot 12
6 // 1728
7 // Since the input number may be large, it is given as a prime
8 // factorization.
9 #include <bits/stdc++.h>
10 #define int long long
11 #define namespace std;
12 #define ll long long
13 #define ull unsigned long long
14 #define fore(i, a, b) for(int i = (a); i<(b); i++)
15 #define FOR(i, n) for(int i = 0; i<(n); i++)
16 #define all(x) (x).begin(), (x).end()
17 #define sz(x) (int)(x).size()
18 #define pb push_back
19 using vi = vector<int>;
20 const int MOD = 1000000007;
21 const ll INF = 9223372036854775807LL;
22 const int tam = 1;
23 #define bint __int128_t
24 bint expmod(bint a, bint b, bint m) {
25     a %= m;
26     bint res = 1;
27     while (b > 0) {
28         if (b & 1)
29             res *= a;
30         a *= a;
31         b >>= 1;
32     }
33     return res;
34 }
35
36 bint invmod(bint num){
37     return expmod(num, MOD-2, MOD);
38 }
39
40 bint sumdiv(map<bint, bint>& factores){
41     bint ans = 1;
42     for(auto[factor, exp]:factores){
43         ans*=(factor^exp);
44         bint inv = invmod((factor-1+MOD)%MOD);
45         bint numer = (expmod(factor, exp+1, MOD) - 1 + MOD)%MOD;
46         ans*=numer*inv%MOD;
47     }
48     ans%=MOD;
49 }
50
51 bint return ans;
52 }
53
54 bint numdiv(map<bint, bint>& factores){
55     bint ans = 1;
56     for(auto [a,b]:factores){
57         ans*=(b+1);
58         ans%=MOD;
59     }
60 }
61
62 void solve(){
63 }
```

```

84     int n;cin>>n;
85     map<bint, bint> factores;
86     // factores.reserve(n);
87     for(int i = 0; i<n; i++){
88         int a, b;cin>>a>>b;
89         factores[a]+=b;
90     }
91     bint cant = numdiv(factores);
92     cout<<(int)cant<< ' '<<(int)sumdiv(factores)<<' 
93     →   '<<(int)proddiv(factores)<<'\n';
94 }
95
96 signed main(){
97     ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
98     int t = 1;
99     //   cin>>t;
100    while(t--){
101        solve();
102    }
103 }

23     }
24     return 1;
25 }
26 void solve(){
27     int x;cin>>x;
28     int num = x+1;
29     while(!esPrimo(num))++num;
30     cout<<num<<'\n';
31 }
32
33 signed main(){
34     ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
35     int t = 1;
36     cin>>t;
37     while(t--){
38         solve();
39     }
40 }
```

## 7.0.6 Exponentiation\_II

```
1 void solve(){
2     int a, b, c;
3     cin >> a >> b >> c;
4     cout << expMod(a,expMod(b, c, MOD-1), MOD) << endl;
5 }
```

## 7.0.7 Next\_Prime

```
1 // Given a positive integer n, find the next prime number after it.
2 #include <bits/stdc++.h>
3 using namespace std;
4 #define int long long
5 #define ll long long
6 #define ull unsigned long long
7 #define fore(i, a, b) for(int i = (a); i<(b); i++)
8 #define FOR(i, n) for(int i = 0; i<(n); i++)
9 #define all(x) (x).begin(), (x).end()
10 #define sz(x) (int)(x).size()
11 #define pb push_back
12 using vi = vector<int>;
13 const int MOD = 1000000007;
14 const ll INF = 9223372036854775807LL;
15 const int tam = 1;
16 bool esPrimo(ll n) {
17     if (n < 2) return 0;
18     if (n == 2 or n == 3) return 1;
19     if (n % 2 == 0 or n % 3 == 0) return 0;
20     for (ll i = 5; i * i <= n; i += 6) {
21         if (n % i == 0 or n % (i + 2) == 0)
22             return 0;
23     }
24 }
```

### 7.0.8 Sum\_of\_Divisors

```

1 // Let \sigma(n) denote the sum of divisors of an integer n. For example,
2 // \sigma(12)=1+2+3+4+6+12=28.
3 // Your task is to calculate the sum \sum_{i=1}^n \sigma(i) modulo 10^9+7.
4 #include <bits/stdc++.h>
5 using namespace std;
6 #define ll long long
7 #define ull unsigned long long
8 #define fore(i, a, b) for (int i = (a); i < (b); i++)
9 #define FOR(i, n) for (int i = 0; i < (n); i++)
10 #define all(x) (x).begin(), (x).end()
11 #define sz(x) (int)(x).size()
12 #define pb push_back
13 using vi = vector<int>;
14 const ll MOD = 1000000007;
15 const ll INF = 9223372036854775807LL;
16 const int tam = 1;
17 ll expmod(ll a, ll b, ll m) {
18     a %= m;
19     ll res = 1;
20     while (b > 0) {
21         if (b & 1)
22             res = res * a % m;
23         a = a * a % m;
24         b >>= 1;
25     }
26     return res;
27 }
28 ll invmod(int x, int m) { return expmod(x, MOD - 2, MOD); }
29 void solve() {
30     ll n;
31     cin >> n;
32     ll ans = 0;
33     ll d = 1;

```

```

34     while (d <= n) {
35         ll t = n / d;
36         ll r = n / t;           // último d con mismo valor de floor(n/d)
37         ll cnt = r - d + 1;    // tamaño del bloque [d..r]
38         // suma aritmética de d..r = (d + r) * cnt / 2
39         ll sum = ((__int128)(d + r) * cnt / 2) % MOD;
40         ans = (ans + sum * (t % MOD)) % MOD;
41         d = r + 1;             // saltamos al siguiente bloque
42     }
43
44     cout << ans << "\n";
45 }
46
47 signed main() {
48     ios::sync_with_stdio(0);
49     cin.tie(0);
50     cout.tie(0);
51     int t = 1;
52     // cin >> t;
53     while (t--) {
54         solve();
55     }
56 }
```

## 8 Range Queries

### 8.0.1 Distinct\_Values\_Questions

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define ll long long
4 #define ull unsigned long long
5 #define fore(i, a, b) for(int i = (a); i<(b); i++)
6 #define FOR(i, n) for(int i = 0; i<(n); i++)
7 #define all(x) (x).begin(), (x).end()
8 #define sz(x) (int)(x).size()
9 #define pb push_back
10 using vi = vector<int>;
11 const int MOD = 1000000007;
12 const ll INF = 9223372036854775807LL;
13 const int tam = 1;
14 // You are given an array of n integers and q queries of the form: how
15 // many distinct values are there in a range [a,b]?
16 vector<ll> bit;
17 int len;
18
19 void update(int pos, ll delta){
20     while (pos < len) {
21         bit[pos] += delta;
22         pos += pos & -pos;
23     }
24
25 ll query(int pos){
26     ll sum = 0;
27     while (pos > 0) {
```

```

28         sum += bit[pos];
29         pos -= pos & -pos;
30     }
31     return sum;
32 }
33
34 void solve(){
35     int n, q; cin >> n >> q;
36     len = n+1;
37     bit.resize(n+1);
38     //for(int i = 1; i<=n; i++) update(i, 1);
39     vector<int> elementos(n+1);
40     map<int, int> ult;
41     fore(i, 1, n+1)cin >> elementos[i];
42     map<int, vector<pair<int, int>>> querys;
43     fore(i, 0, q){
44         int a, b; cin >> a >> b;
45         querys[b].push_back({a, i});
46     }
47     vector<int> ans(q);
48     for(int i = 1; i<=n; i++){
49         if(ult.find(elementos[i]) == ult.end()){
50             update(i, 1);
51             ult[elementos[i]] = i;
52         }else{
53             update(ult[elementos[i]], -1);
54             ult[elementos[i]] = i;
55             update(i, 1);
56         }
57         for(auto[l, pos]:querys[i]){
58             ans[pos] = query(i) - query(l-1);
59         }
60     }
61     fore(i, 0, q)cout << ans[i] << endl;
62 }
63
64 signed main(){
65     ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
66     int t = 1;
67     //cin >> t;
68     while(t--){
69         solve();
70     }
71 }
```

### 8.0.2 Distinct\_Values\_Questions\_II

---

```

1 // Given an array of n integers, your task is to process q queries of the
2 // following types:
3
4 // update the value at position k to u
5 // check if every value in range [a, b] is distinct
6 #include <bits/stdc++.h>
7 #define ll long long
using namespace std;
```

```

8  using vi = vector<int>;
9  const int INF = 1e9;
10 #define sz(x) (int)(x).size()
11
12 struct BIT {
13     int n;
14     vi bit;
15     BIT() { n = 0; }
16     BIT(int _n) { init(_n); }
17     void init(int _n) { n = _n; bit.assign(n+1, 0); }
18     void add(int i, int delta) { // 1-indexed
19         for (; i <= n; i += i & -i) bit[i] += delta;
20     }
21     int sum(int i) {
22         int s = 0;
23         for (; i > 0; i -= i & -i) s += bit[i];
24         return s;
25     }
26     int rangeSum(int l, int r) { if (r < l) return 0; return sum(r) -
27         sum(l-1); }
28 };
29
30 int n, q;
31 vector<ll> a; // values
32 struct consulta{
33     int tipo, x;
34     ll y;
35 };
36 vector<consulta> queries; // store queries
37
38 // segment tree nodes
39 vector<vi> nodeVals; // sorted unique next-values relevant to this node
40 vector<BIT> nodeBIT; // BIT per node
41
42 // build node values by merging children's vectors
43 void build_nodes(int p, int l, int r, const vector<vi>& valsPerPos) {
44     if (l == r) {
45         nodeVals[p] = valsPerPos[l];
46         sort(nodeVals[p].begin(), nodeVals[p].end());
47         nodeVals[p].erase(unique(nodeVals[p].begin(), nodeVals[p].end())),
48             nodeVals[p].end());
49     } else {
50         int m = (l + r) >> 1;
51         build_nodes(p<<1, l, m, valsPerPos);
52         build_nodes(p<<1|1, m+1, r, valsPerPos);
53         vi &L = nodeVals[p<<1], &R = nodeVals[p<<1|1];
54         nodeVals[p].resize(sz(L) + sz(R));
55         merge(L.begin(), L.end(), R.begin(), R.end(),
56             nodeVals[p].begin());
57         nodeVals[p].erase(unique(nodeVals[p].begin(), nodeVals[p].end())),
58             nodeVals[p].end());
59     }
60     nodeBIT[p].init(sz(nodeVals[p]));
61 }
62
63 // add delta at position posIdx for value val (val must exist in
64 // nodeVals[p])
65 void add_val(int p, int l, int r, int posIdx, int val, int delta) {
66     // find index of val in nodeVals[p]
67     int idx = int(upper_bound(nodeVals[p].begin(), nodeVals[p].end(),
68         val) - nodeVals[p].begin());
69     // we want count of values ≤ val in queries, but for add we need
70     // exact index of val
71     // since nodeVals[p] is unique sorted, find exact position via
72     // lower_bound
73     int idx_exact = int(lower_bound(nodeVals[p].begin(),
74         nodeVals[p].end(), val) - nodeVals[p].begin());
75     if (idx_exact < sz(nodeVals[p]) and nodeVals[p][idx_exact] == val) {
76         nodeBIT[p].add(idx_exact + 1, delta); // BIT is 1-indexed
77     }
78     if (l == r) return;
79     int m = (l + r) >> 1;
80     if (posIdx <= m) add_val(p<<1, l, m, posIdx, val, delta);
81     else add_val(p<<1|1, m+1, r, posIdx, val, delta);
82 }
83
84 // query count of values ≤ bound in positions [i,j]
85 int query_countLE(int p, int l, int r, int i, int j, int bound) {
86     if (r < i or l > j) return 0;
87     if (i <= l and r <= j) {
88         // count how many values ≤ bound in this node: upper_bound index
89         int pos = int(upper_bound(nodeVals[p].begin(), nodeVals[p].end(),
90             bound) - nodeVals[p].begin());
91         return nodeBIT[p].sum(pos); // sum up to pos
92     }
93     int m = (l + r) >> 1;
94     return query_countLE(p<<1, l, m, i, j, bound) + query_countLE(p<<1|1,
95         m+1, r, i, j, bound);
96 }
97
98 int main() {
99     ios::sync_with_stdio(false);
100    cin.tie(nullptr);
101    cin >> n >> q;
102    a.assign(n+1, 0);
103    for (int i = 1; i <= n; ++i) cin >> a[i];
104    queries.reserve(q);
105    // read queries
106    for (int t = 0; t < q; ++t) {
107        int tp; cin >> tp;
108        if (tp == 1) {
109            int k; ll u; cin >> k >> u;
110            queries.push_back({tp, k, u});
111        } else {
112            int l, r; cin >> l >> r;
113            queries.push_back({tp, l, (ll)r});
114        }
115    }
116    const int INF_IDX = n + 1;
117
118    // initial: pos map
119    unordered_map<ll, set<int>> pos;
120    pos.reserve((size_t)(n*2));

```

```

111  for (int i = 1; i <= n; ++i) pos[a[i]].insert(i);
112
113 // nextPos current state for simulation
114 vector<int> nextPos(n+1, INF_IDX);
115 for (auto &kv : pos) {
116     auto &s = kv.second;
117     int prev = -1;
118     for (int idx : s) {
119         if (prev != -1) nextPos[prev] = idx;
120         prev = idx;
121     }
122     if (prev != -1) nextPos[prev] = INF_IDX;
123 }
124
125 // valsPerPos: collect all next-values that will ever appear at
126 // position i
126 vector<vi> valsPerPos(n+1);
127 for (int i = 1; i <= n; ++i) {
128     valsPerPos[i].push_back(nextPos[i]); // initial
129 }
130
131 // --- SIMULATE ALL QUERIES to collect all next-values per position
132 // ---
133 // clone structures for simulation
134 auto posSim = pos;
135 auto aSim = a;
136 auto nextSim = nextPos;
137
138 for (auto &qq : queries) {
139     int tp = qq.tipo;
140     if (tp == 1) {
141         int k = qq.x;
142         ll u = qq.y;
143         if (aSim[k] == u) continue;
144         ll old = aSim[k];
145
146         // remove k from old set
147         auto &sOld = posSim[old];
148         auto it = sOld.find(k);
149         int oldPred = -1, oldNext = INF_IDX;
150         if (it != sOld.end()) {
151             auto itNext = next(it);
152             if (itNext != sOld.end()) oldNext = *itNext;
153             if (it != sOld.begin()) {
154                 auto itPred = prev(it);
155                 oldPred = *itPred;
156             }
157             // after removing k, pred_old's next will become oldNext
158             sOld.erase(it);
159         }
160         if (oldPred != -1) {
161             if (nextSim[oldPred] != oldNext) {
162                 nextSim[oldPred] = oldNext;
163                 valsPerPos[oldPred].push_back(oldNext);
164             }
165         }
166         // k's next will be set according to insertion in new set
167         // prepare new set
168
169         auto &sNew = posSim[u];
170         sNew.insert(k);
171         auto itNew = sNew.find(k);
172         int newPred = -1, newNext = INF_IDX;
173         if (itNew != sNew.begin()) {
174             auto itPred = prev(itNew);
175             newPred = *itPred;
176         }
177         if (itNew != sNew.end()) newNext = *itNew;
178
179         if (nextSim[k] != newNext) {
180             nextSim[k] = newNext;
181             valsPerPos[k].push_back(newNext);
182         }
183         if (newPred != -1) {
184             if (nextSim[newPred] != k) {
185                 nextSim[newPred] = k;
186                 valsPerPos[newPred].push_back(k);
187             }
188         }
189         aSim[k] = u;
190         if (sOld.empty()) posSim.erase(old);
191     } // else type 2: nothing to simulate for structure
192 }
193
194 // now build segment tree nodes
195 int SZ = 4 * (n + 5);
196 nodeVals.assign(SZ, vi());
197 nodeBIT.assign(SZ, BIT());
198 build_nodes(1, 1, n, valsPerPos);
199
200 // now initialize current real structures (pos, nextPos, a)
201 // we already had pos, nextPos, a earlier (from initial)
202 // insert initial nextPos into BITs
203 for (int i = 1; i <= n; ++i) {
204     add_val(1, 1, n, i, nextPos[i], +1);
205 }
206
207 // process queries online, updating pos, nextPos, a, and the BITs
208 for (auto &qq : queries) {
209     int tp = qq.tipo;
210     if (tp == 1) {
211         int k = qq.x;
212         ll u = qq.y;
213         if (a[k] == u) continue;
214         ll old = a[k];
215
216         auto &sOld = pos[old];
217         auto it = sOld.find(k);
218         int oldPred = -1, oldNext = INF_IDX;
219         if (it != sOld.end()) {
220             auto itNext = next(it);
221             if (itNext != sOld.end()) oldNext = *itNext;
222             if (it != sOld.begin()) {
223                 auto itPred = prev(it);
224                 oldPred = *itPred;

```

```

225 }
226 // remove k's current next from BITS
227 add_val(1, 1, n, k, nextPos[k], -1);
228 sOld.erase(it);
229 }
230 if (oldPred != -1) {
231     // pred_old: next changes from nextPos[oldPred] to oldNext
232     int before = nextPos[oldPred];
233     if (before != oldNext) {
234         add_val(1, 1, n, oldPred, before, -1);
235         nextPos[oldPred] = oldNext;
236         add_val(1, 1, n, oldPred, nextPos[oldPred], +1);
237     }
238 }
239 // insert k into new set
240 auto &sNew = pos[u];
241 sNew.insert(k);
242 auto itNew = sNew.find(k);
243 int newPred = -1, newNext = INF_IDX;
244 if (itNew != sNew.begin()) {
245     auto itPred = prev(itNew);
246     newPred = *itPred;
247 }
248 {
249     auto itN = next(itNew);
250     if (itN != sNew.end()) newNext = *itN;
251 }
252 // set next[k] = newNext
253 nextPos[k] = newNext;
254 add_val(1, 1, n, k, nextPos[k], +1);
255
256 if (newPred != -1) {
257     int before = nextPos[newPred];
258     if (before != k) {
259         add_val(1, 1, n, newPred, before, -1);
260         nextPos[newPred] = k;
261         add_val(1, 1, n, newPred, nextPos[newPred], +1);
262     }
263 }
264
265 a[k] = u;
266 if (sOld.empty()) pos.erase(old);
267 }
268 } else {
269     int l = qq.x;
270     int r = (int)qq.y;
271     int countNextLER = query_countLE(1, 1, n, l, r, r);
272     int distinct = (r - l + 1) - countNextLER;
273     cout << (distinct == (r-l+1)? "YES": "NO") << '\n';
274 }
275 }
276
277 return 0;
278 }
279 //-----
280 #include <bits/stdc++.h>
281 using namespace std;
282

```

```

283 struct SegTree {
284     int n;
285     vector<int> st;
286     SegTree(int _n = 0) { init(_n); }
287     void init(int _n) {
288         n = _n;
289         st.assign(4*n + 5, INT_MAX);
290     }
291     void build(int p, int l, int r, const vector<int>& a) {
292         if (l == r) { st[p] = a[l]; return; }
293         int m = (l + r) >> 1;
294         build(p<<1, l, m, a);
295         build(p<<1|1, m+1, r, a);
296         st[p] = min(st[p<<1], st[p<<1|1]);
297     }
298     int queryMin(int p, int l, int r, int i, int j) {
299         if (r < i || l > j) return INT_MAX;
300         if (i <= l && r <= j) return st[p];
301         int m = (l + r) >> 1;
302         return min(queryMin(p<<1, l, m, i, j), queryMin(p<<1|1, m+1, r,
303             i, j));
304     }
305     void update(int p, int l, int r, int idx, int val) {
306         if (l == r) { st[p] = val; return; }
307         int m = (l + r) >> 1;
308         if (idx <= m) update(p<<1, l, m, idx, val);
309         else update(p<<1|1, m+1, r, idx, val);
310         st[p] = min(st[p<<1], st[p<<1|1]);
311     }
312
313 int main() {
314     ios::sync_with_stdio(false);
315     cin.tie(nullptr);
316     int n, q;
317     if (!(cin >> n >> q)) return 0;
318     vector<long long> a(n+1);
319     for (int i = 1; i <= n; ++i) cin >> a[i];
320
321     const int INF = n + 1;
322     unordered_map<long long, set<int>> pos;
323     pos.reserve(n * 2);
324
325     for (int i = 1; i <= n; ++i) pos[a[i]].insert(i);
326
327     vector<int> nextPos(n+1, INF);
328     for (auto &kv : pos) {
329         auto &s = kv.second;
330         int prev = -1;
331         for (int idx : s) {
332             if (prev != -1) nextPos[prev] = idx;
333             prev = idx;
334         }
335         if (prev != -1) nextPos[prev] = INF;
336     }
337
338     SegTree seg(n);
339     seg.build(1, 1, n, nextPos);

```

```

340
341     while (q--) {
342         int type;
343         cin >> type;
344         if (type == 1) {
345             int k; long long u;
346             cin >> k >> u;
347             if (a[k] == u) continue; // nada que hacer
348             long long old = a[k];
349
350             // --- quitar k del conjunto antiguo ---
351             auto &sOld = pos[old];
352             auto it = sOld.find(k);
353             int oldPred = -1, oldNext = INF;
354             if (it != sOld.end()) {
355                 auto itNext = next(it);
356                 if (itNext != sOld.end()) oldNext = *itNext;
357                 if (it != sOld.begin()) {
358                     auto itPred = prev(it);
359                     oldPred = *itPred;
360                 }
361                 sOld.erase(it);
362             }
363             // actualizar next de oldPred (si existía)
364             if (oldPred != -1) {
365                 nextPos[oldPred] = (oldNext == INF ? INF : oldNext);
366                 seg.update(1, 1, n, oldPred, nextPos[oldPred]);
367             }
368             // temporalmente el next de k será INF (se va a insertar en
369             // → nuevo conjunto)
370             nextPos[k] = INF;
371             seg.update(1, 1, n, k, nextPos[k]);
372
373             // --- insertar k en el conjunto nuevo ---
374             auto &sNew = pos[u];
375             auto resItPair = sNew.insert(k);
376             auto itNew = resItPair.first;
377             int newPred = -1, newNext = INF;
378             if (itNew != sNew.begin()) {
379                 auto itPred = prev(itNew);
380                 newPred = *itPred;
381             }
382             auto itN = next(itNew);
383             if (itN != sNew.end()) newNext = *itN;
384
385             if (newPred != -1) {
386                 // el next del predecesor ahora apunta a k
387                 nextPos[newPred] = k;
388                 seg.update(1, 1, n, newPred, nextPos[newPred]);
389             }
390             // next de k es newNext (o INF)
391             nextPos[k] = (newNext == INF ? INF : newNext);
392             seg.update(1, 1, n, k, nextPos[k]);
393             a[k] = u;
394
395             // (opcional) borrar entrada de pos[old] si quedó vacía

```

```

397         if (sOld.empty()) pos.erase(old);
398
399     } else if (type == 2) {
400         int l, r;
401         cin >> l >> r;
402         int mn = seg.queryMin(1, 1, n, l, r);
403         if (mn > r) cout << "YES\n";
404         else cout << "NO\n";
405     }
406
407     return 0;
408 }

```

### 8.0.3 Forest\_Quieries

```

1 #include "bits/stdc++.h"
2 using namespace std;
3 #define int long long
4 const int tam = 1009;
5
6 int BIT[tam][tam];
7 int A[tam][tam]; // Matriz original
8 int n, m;
9
10 void update(int row, int col, int val)
11 {
12     row++; col++;
13     for (int i = row; i <= n; i += (i & -i))
14     {
15         for (int j = col; j <= m; j += (j & -j))
16         {
17             BIT[i][j] += val;
18         }
19     }
20 }
21
22 // Consulta acumulativa desde (0,0) hasta (row, col)
23 int query(int row, int col)
24 {
25     int res = 0;
26     row++; col++;
27     for (int i = row; i > 0; i -= (i & -i))
28     {
29         for (int j = col; j > 0; j -= (j & -j))
30         {
31             res += BIT[i][j];
32         }
33     }
34     return res;
35 }
36
37 int queryRange(int x1, int y1, int x2, int y2) {
38     return query(x2, y2) - query(x1 - 1, y2) - query(x2, y1 - 1) +
39     query(x1 - 1, y1 - 1);

```

```

40
41 // Actualiza completamente un valor en la posición (i, j)
42 void setValue(int i, int j, int nuevo) {
43     int diff = nuevo - A[i][j]; // Calcula la diferencia
44     update(i, j, diff); // Aplica la diferencia al BIT
45     A[i][j] = nuevo; // Actualiza el valor en la matriz
46     ↵ original
47 }
48
49 signed main() {
50     memset(BIT, 0, sizeof BIT);
51     int q;
52     cin >> n >> q;
53     m = n;
54     for (int i = 0; i < n; i++) {
55         for (int j = 0; j < m; j++) {
56             char val;
57             cin >> val;
58             A[i][j] = (val == '*');
59             update(i, j, val == '*'); // Construcción inicial
60     }
61     while(q--){
62         int a, b, c, d;cin>>a>>b>>c>>d;
63         cout<<queryRange(a-1, b-1, c-1, d-1)<<endl;
64     }
65
66     return 0;
67 }

```

#### 8.0.4 Hotel\_Questions

```

1 // There are n hotels on a street. For each hotel you know the number of
2 // free rooms. Your task is to assign hotel rooms for groups of tourists
3 // All members of a group want to stay in the same hotel.
4 // The groups will come to you one after another, and you know for each
5 // group the number of rooms it requires. You always assign a group to
6 // the first hotel having enough rooms. After this, the number of free
7 // rooms in the hotel decreases.
8 #include <bits/stdc++.h>
9 using namespace std;
10#define ll long long
11#define ull unsigned long long
12#define fore(i, a, b) for(int i = (a); i<(b); i++)
13#define FOR(i, n) for(int i = 0; i<(n); i++)
14#define all(x) (x).begin(), (x).end()
15#define sz(x) (int)(x).size()
16#define pb push_back
17 using vi = vector<int>;
18 const int MOD = 1000000007;
19 const ll INF = 9223372036854775807LL;
20 const int tam = 200010;
21 struct Node{
22     int x;
23     //algoritmo
24 }
```

```

19     Node(int _x = 0/*para poner por default y no necesitar constructor por
20     → defecto*/){x = _x;}
21     static inline Node merge(const Node&a, const Node& b){
22         return Node(max(a.x, b.x));
23     }
24
25     Node t[4*tam];
26     int arr[tam];
27
28     void init(int b, int e, int node){
29         if(b == e){
30             t[node] = Node(arr[b]);
31             return;
32         }
33         int mid = (b+e)>>1, l = (node << 1) | 1, r = l+1;
34         init(b, mid, l);
35         init(mid+1, e, r);
36         t[node] = Node::merge(t[l], t[r]);
37     }
38
39     void query(int b, int e, int node, int x){
40         if(b == e){
41             t[node].x-=x;
42             printf("%d ", b+1);
43             return;
44         }
45         int mid = (b+e)>>1, l = (node << 1) | 1, r = l+1;
46         if(t[l].x >=x){
47             query(b, mid, l, x);
48         }else{
49             query(mid+1, e, r, x);
50         }
51         t[node] = Node::merge(t[l], t[r]);
52     }
53
54     void update(int b, int e, int node, int pos, Node& val){
55         if(b == e){
56             //reemplazar, sumar, xor, lo que sea
57             t[node] = val;
58             return;
59         }
60         int mid = (b+e)>>1, l = (node << 1) | 1, r = l+1;
61         if(pos <= mid){
62             update(b, mid, l, pos, val);
63         }else{
64             update(mid+1, e, r, pos, val);
65         }
66         t[node] = Node::merge(t[l], t[r]);
67     }
68     void solve(){
69         int n, q;scanf("%d %d", &n, &q);
70         for(int i = 0; i<n; i++)scanf("%d", &arr[i]);
71         init(0, n-1, 1);
72         while(q--){
73             int x;scanf("%d", &x);
74             if(t[1].x < x){
75                 printf("0 ");
76             }
77         }
78     }
79 }
```

```

76 }else{
77     query(0, n-1, 1, x);
78 }
79 printf("\n");
80 }
81 }
82 }
83
84 signed main(){
85     ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
86     int t = 1;
87     //cin>>t;
88     while(t--){
89         solve();
90     }
91 }
92
93
94 #include<bits/stdc++.h>
95 using namespace std;
96
97 const int tam = 200010;
98 struct Node{
99     int x;
100    //algo
101    Node(int _x = 0/*para poner por default y no necesitar constructor por defecto*/){x = _x;}
102    static inline Node merge(const Node&a, const Node&b){
103        return Node(max(a.x, b.x));
104    }
105 };
106
107 Node t[4*tam];
108 int arr[tam];
109
110 void init(int b, int e, int node){
111     if(b == e){
112         t[node] = Node(arr[b]);
113         return;
114     }
115     int mid = (b+e)>>1, l = (node << 1) | 1, r = l+1;
116     init(b, mid, l);
117     init(mid+1, e, r);
118     t[node] = Node::merge(t[l], t[r]);
119 }
120
121 Node query(int b, int e, int node, int i, int j){
122     if(i<=b and j>=e){
123         return t[node];
124     }
125     int mid = (b+e)>>1, l = (node << 1) | 1, r = l+1;
126     if(mid >= j)
127         return query(b, mid, l, i, j);
128     if(mid < i)
129         return query(mid+1, e, r, i, j);
130     return Node::merge(query(b, mid, l, i, j), query(mid+1, e, r, i, j));
131 }
132
133 void update(int b, int e, int node, int pos, Node& val){
134     if(b == e){
135         //reemplazar, sumar, xor, lo que sea
136         t[node].x += val.x;
137         return;
138     }
139     int mid = (b+e)>>1, l = (node << 1) | 1, r = l+1;
140     if(pos <= mid){
141         update(b, mid, l, pos, val);
142     }else{
143         update(mid+1, e, r, pos, val);
144     }
145     t[node] = Node::merge(t[l], t[r]);
146 }
147
148 void solve(){
149     int n, q;scanf("%d %d", &n, &q);
150     for(int i = 0; i<n; i++)scanf(" %d", &arr[i]);
151     init(0, n-1, 1);
152     while(q--){
153         int x;scanf(" %d", &x);
154         int l = 0, r = n-1;
155         int ans = -1;
156         while(l<=r){
157             int mid = (l+r)/2;
158             if(query(0, n-1, 1, 0, mid).x >=x){
159                 r = mid-1;
160                 ans = mid;
161             }else{
162                 l = mid+1;
163             }
164         }
165         if(ans == -1){
166             cout<<0<<'\n';
167         }else{
168             cout<<ans+1<<'\n';
169             auto it = Node(-x);
170             update(0, n-1, 1, ans, it);
171             arr[ans]-=x;
172         }
173     }
174 }
175
176 signed main(){
177     ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
178     solve();
179 }

```

---

## 8.0.5 List\_Removals

```

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

```

```

4 #define ll long long
5 #define ull unsigned long long
6 #define fore(i, a, b) for(int i = (a); i<(b); i++)
7 #define FOR(i, n) for(int i = 0; i<(n); i++)
8 #define all(x) (x).begin(), (x).end()
9 #define sz(x) (int)(x).size()
10 #define pb push_back
11 using vi = vector<int>;
12 const int MOD = 1000000007;
13 const ll INF = 9223372036854775807LL;
14 const int tam = 1;
15
16 #define lsb(x) ((x) & (-x))
17 struct BIT {
18     // indexado a 1
19     vector<int> bit;
20     BIT(int N){
21         bit.resize(N+1);
22     }
23     void add(int i, int x) {
24         int real = i;
25         while (i < sz(bit)) {
26             bit[i] += x;
27             i += lsb(i);
28         }
29     }
30     int sum(int i) {
31         int ans = 0;
32         while (i > 0) {
33             ans += bit[i];
34             i -= lsb(i);
35         }
36         return ans;
37     }
38     int sum(int l, int r) {
39         if (l > r) return 0;
40         return sum(r) - sum(l - 1);
41     }
42 };
43 void solve(){
44     int n;cin>>n;
45     vector<int> arr(n);
46     BIT bit(n);
47     for(int i = 0; i<n; i++){
48         int x;cin>>x;
49         arr[i] = x;
50         bit.add(i+1, 1);
51     }
52     for(int i = 0; i<n; i++){
53         int idx;cin>>idx;
54         int l = 1, r = n;
55         int pos;
56         while(l<=r){
57             int mid = (l+r)/2;
58             if(bit.sum(1, mid) >= idx){
59                 pos = mid;
60                 r = mid-1;
61             }else{
62             }
63         }
64     }
65 }
```

```
    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

```

// 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;
    }
}
```

```

33     int tm = (tl + tr) >> 1;
34     if(pos <= tm){
35         int left_child = update(L[cur], tl, tm, pos, val);
36         L[node] = left_child;
37     } else {
38         int right_child = update(R[cur], tm+1, tr, pos, val);
39         R[node] = right_child;
40     }
41     return node;
42 }
43 // query sum on [lq, rq] on node
44 ll query(int node, int tl, int tr, int lq, int rq){
45     if(!node || rq < tl || tr < lq) return 0LL;
46     if(lq <= tl && tr <= rq) return sum[node];
47     int tm = (tl + tr) >> 1;
48     return query(L[node], tl, tm, lq, rq) + query(R[node], tm+1, tr,
49             lq, rq);
50 }
51
52 int main(){
53     ios::sync_with_stdio(false);
54     cin.tie(nullptr);
55     cin >> n >> q;
56     vector<int> a(n+1);
57     fore(i,1,n+1) cin >> a[i];
58
59     // pairs (value, pos)
60     vector<pair<int,int>> vp;
61     vp.reserve(n);
62     fore(i,1,n+1) vp.emplace_back(a[i], i);
63     sort(vp.begin(), vp.end()); // ascending by value
64
65     // build PST versions: root[0] = empty, root[k] = with first k
66     // → elements of vp added
67     // estimate nodes ~ n * (log2 n + 5)
68     int est_nodes = n * 22;
69     PST pst(est_nodes);
70     pst.roots.resize(n+1);
71     pst.roots[0] = 0; // empty tree
72     for(int k = 1; k <= n; ++k){
73         int val = vp[k-1].first;
74         int pos = vp[k-1].second;
75         pst.roots[k] = pst.update(pst.roots[k-1], 1, n, pos, val);
76     }
77
78     // vector of only values for upper_bound
79     vector<int> vals(n);
80     for(int i=0;i<n;i++) vals[i] = vp[i].first;
81
82     while(q--){
83         int L, R; cin >> L >> R;
84         ll res = 1;
85         while(true){
86             // count k = #values ≤ res
87             int k = upper_bound(vals.begin(), vals.end(),
88                                 (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

```

1 // There are n buildings on a street, numbered 1,2,\dots,n. Each building
2 // has a pizzeria and an apartment.
3 // The pizza price in building k is p_k. If you order a pizza from
4 // building a to building b, its price (with delivery) is p_a+|a-b|.
5 // Your task is to process two types of queries:
6 // The pizza price p_k in building k becomes x.
7 // You are in building k and want to order a pizza. What is the minimum
8 // price?
9 #include <bits/stdc++.h>
10 using namespace std;
11 #define ll long long
12 #define ull unsigned long long
13 #define fore(i, a, b) for(int i = (a); i<(b); i++)
14 #define all(x) (x).begin(), (x).end()
15 #define sz(x) (int)(x).size()
16 #define pb push_back
17 using vi = vector<int>;
18 const int MOD = 1000000007;
19 const ll INF = 4e18;
20 const int tam = 200005;
21 struct Item{
22     //algún atributo
23     int x;
24     Item(int _x = 2e9){
25         x = _x;
26     }
27     static Item merge(const Item& a, const Item& b){
28         return Item(min(a.x, b.x));
29     }
30 };
31 struct Nodo {
32     Item value;
33     Nodo *izq = nullptr, *der = nullptr;
34 };
35 void update(Nodo*& node, ll inicio, ll fin, ll pos, Item& val) {
36     if (node == nullptr) {
37         node = new Nodo();
38     }
39     if (inicio > fin) {
40         return;
41     }
42     if (pos < inicio) {
43         update(node->izq, inicio, fin, pos, val);
44     } else if (pos > fin) {
45         update(node->der, inicio, fin, pos, val);
46     } else {
47         node->value = val;
48     }
49 }
```

```

35     if (!node) node = new Nodo();
36     if (inicio == fin) { //la actualizacion, puede ser suma o asignacion
37         → lo que sea, piensa bien
38         node->value = val;
39         return;
40     }
41     ll mid = (inicio + fin) / 2;
42     if (pos <= mid)
43         update(node->izq, inicio, mid, pos, val);
44     else
45         update(node->der, mid + 1, fin, pos, val);
46     //esto esta interesante, que es el elemento neutro?
47     auto it = Item();
48     Item& izqVal = node->izq ? node->izq->value : it;
49     Item& derVal = node->der ? node->der->value : it;
50     node->value = Item::merge(izqVal, derVal);
51 }
52
53 Item query(Nodo* node, ll inicio, ll fin, ll l, ll r) {
54     if (!node || r < inicio || l > fin) return Item();
55     if (l <= inicio && fin <= r) return node->value;
56     ll mid = (inicio + fin) / 2;
57     return Item::merge(query(node->izq, inicio, mid, l, r),
58                         query(node->der, mid + 1, fin, l, r));
59 }
60 int arr[tam];
61 Nodo* izq = new Nodo();
62 Nodo* der = new Nodo();
63 void solve(){
64     int n, q;scanf("%d %d", &n, &q);
65     for(int i = 1; i<=n; i++){
66         scanf(" %d", &arr[i]);
67         auto itizq = Item(arr[i] - i);
68         auto itder = Item(arr[i]+i);
69         update(izq, 1, n, i, itizq);
70         update(der, 1, n, i, itder);
71     }
72     while(q--){
73         int t;
74         scanf(" %d", &t);
75         if(t == 1){
76             int pos, x;
77             scanf(" %d %d", &pos, &x);
78             arr[pos] = x;
79             auto itizq = Item(arr[pos] - pos);
80             auto itder = Item(arr[pos] + pos);
81             update(izq, 1, n, pos, itizq);
82             update(der, 1, n, pos, itder);
83         }else{
84             int pos;
85             scanf(" %d", &pos);
86             auto resIzq = query(izq, 1, n, 1, pos);
87             auto resDer = query(der, 1, n, pos, n);
88             int ans = min(resIzq.x + pos, resDer.x - pos);
89             printf("%d\n", ans);
90         }
91     }
92 }
93
94
95
96 signed main(){
97     ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
98     int t = 1;
99     // cin>>t;
100    while(t--){
101        solve();
102    }
103 }

```

## 8.0.8 Polynomial\_Quieries

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define ll long long
4 #define ull unsigned long long
5 #define fore(i, a, b) for(int i = (a); i<(b); i++)
6 #define FOR(i, n) for(int i = 0; i<(n); i++)
7 #define all(x) (x).begin(), (x).end()
8 #define sz(x) (int)(x).size()
9 #define pb push_back
10 using vi = vector<int>;
11 const int MOD = 1000000007;
12 const int tam = 1;
13
14 //Segment tree de programaciones aritmeticas
15 // Your task is to maintain an array of n values and efficiently process
16 // the following types of queries:
17 // Increase the first value in range [a,b] by 1, the second value by 2,
18 // the third value by 3, and so on.
19 // Calculate the sum of values in range [a,b].
20
21
22 const int MAXN = 200003;
23 const ll INF = 4e18;
24
25 struct Node {
26     ll sum;           // Suma del segmento
27     ll first_term;   // Primer término de la PA (si aplica)
28     ll diff;          // Diferencia común (si aplica)
29     int size;         // Tamaño del segmento
30
31     Node(ll s = 0, ll f = 0, ll d = 0, int sz = 0)
32         : sum(s), first_term(f), diff(d), size(sz) {}
33
34     static Node merge(const Node& a, const Node& b) {
35         return Node(a.sum + b.sum, 0, 0, a.size + b.size);
36     }
37 };
38

```

```

39 struct Lazy {
40     ll a; // Primer término de la PA
41     ll d; // Diferencia común
42
43     Lazy(ll a_val = 0, ll d_val = 0) : a(a_val), d(d_val) {}
44
45     bool has_update() const { return a != 0 or d != 0; }
46 };
47
48 Node t[4 * MAXN];
49 Lazy lazy[4 * MAXN];
50 ll arr[MAXN];
51
52 // Calcula la suma de una PA: n/2 * (2a + (n-1)d)
53 ll arithmetic_sum(ll a, ll d, int n) {
54     return n * (2 * a + (n - 1) * d) / 2;
55 }
56
57 void build(int node, int l, int r) {
58     lazy[node] = Lazy();
59     if (l == r) {
60         t[node] = Node(arr[l], 0, 0, 1);
61         return;
62     }
63     int mid = (l + r) / 2;
64     build(2 * node, l, mid);
65     build(2 * node + 1, mid + 1, r);
66     t[node] = Node::merge(t[2 * node], t[2 * node + 1]);
67 }
68
69 void apply_lazy(Node& node, int l, int r, const Lazy& lz, int seg_start) {
70     if (lz.has_update()) {
71         // Calcular el primer término para este segmento específico
72         // Si el segmento comienza en 'start', el primer término es: a + (start - original_start) * d
73         // Pero necesitamos saber el inicio original del update
74         ll first_term_for_segment = lz.a + (l - seg_start) * lz.d;
75
76         // Sumar la PA completa del segmento
77         node.sum += arithmetic_sum(first_term_for_segment, lz.d,
78                                     → node.size);
79     }
80
81 // Versión alternativa más práctica (sin necesidad de seg_start):
82 void apply_lazy_simple(Node& node, int l, int r, const Lazy& lz) {
83     if (lz.has_update()) {
84         // Para un segmento [l, r], la suma de la PA que comienza en l con diferencia d es:
85         // Sum = (r-l+1)/2 * (2*a + (r-l)*d)
86         // Pero necesitamos ajustar el primer término para este segmento
87         node.sum += arithmetic_sum(lz.a, lz.d, node.size);
88     }
89
90     void push(int node, int l, int r) {
91         if (lazy[node].has_update()) {
92
93             apply_lazy_simple(t[node], l, r, lazy[node]);
94
95             if (l != r) {
96                 int mid = (l + r) / 2;
97
98                 // Hijo izquierdo: misma PA, mismo primer término
99                 lazy[2 * node].a += lazy[node].a;
100                lazy[2 * node].d += lazy[node].d;
101
102                 // Hijo derecho: la PA continúa, primer término ajustado
103                 ll first_term_right = lazy[node].a + (mid + 1 - l) *
104                     → lazy[node].d;
105                 lazy[2 * node + 1].a += first_term_right;
106                 lazy[2 * node + 1].d += lazy[node].d;
107             }
108             lazy[node] = Lazy();
109         }
110
111         void update_arithmetic(int node, int l, int r, int ql, int qr, ll a, ll
112         → d) {
113             push(node, l, r);
114             if (l > qr or r < ql) return;
115
116             if (ql <= l and r <= qr) {
117                 // Calcular el primer término para este segmento
118                 ll first_term_here = a + (l - ql) * d;
119                 lazy[node] = Lazy(first_term_here, d);
120                 push(node, l, r);
121                 return;
122
123             int mid = (l + r) / 2;
124             update_arithmetic(2 * node, l, mid, ql, qr, a, d);
125             update_arithmetic(2 * node + 1, mid + 1, r, ql, qr, a, d);
126             t[node] = Node::merge(t[2 * node], t[2 * node + 1]);
127         }
128
129         void update_ap(int node, int l, int r, int ql, int qr, ll a, ll d) {
130             push(node, l, r);
131             if (l > qr or r < ql) return;
132
133             if (ql <= l and r <= qr) {
134                 // Para el segmento [l, r], el primer término es a + (l - ql) * d
135                 ll segment_first = a + (l - ql) * d;
136                 lazy[node] = Lazy(segment_first, d);
137                 push(node, l, r);
138                 return;
139
140             int mid = (l + r) / 2;
141             update_ap(2 * node, l, mid, ql, qr, a, d);
142             update_ap(2 * node + 1, mid + 1, r, ql, qr, a, d);
143             t[node] = Node::merge(t[2 * node], t[2 * node + 1]);
144         }
145
146         ll query_sum(int node, int l, int r, int ql, int qr) {
147             push(node, l, r);
148

```

```

149 if (l > qr or r < ql) return 0;
150 if (ql <= l and r <= qr) return t[node].sum;
151
152 int mid = (l + r) / 2;
153 return query_sum(2 * node, l, mid, ql, qr) +
154     query_sum(2 * node + 1, mid + 1, r, ql, qr);
155 }
156
157 void solve(){
158     int n, q; cin >> n >> q;
159     for(int i = 0; i < n; i++) cin >> arr[i];
160     build(1, 0, n - 1);
161     while(q--){
162         int t, a, b; cin >> t >> a >> b;
163         if(t == 1){
164             a--; b--;
165             update_arithmetic(1, 0, n - 1, a, b, 1, 1);
166         }else{
167             a--, b--;
168             auto res = query_sum(1, 0, n - 1, a, b);
169             cout << res << '\n';
170         }
171     }
172 }
173
174 signed main(){
175     ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
176     int t = 1;
177     // cin >> t;
178     while(t--){
179         solve();
180     }
181 }
182

```

## 8.0.9 Prefix\_Sum\_Questions

---

// 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]?

```

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

```

```

18 struct Item{
19     //algún atributo
20     ll sum;
21     ll maxPref;
22     Item(ll s = 0, ll maxP = -(1e18)){
23         sum = s;
24         maxPref = maxP;
25     }
26     static Item merge(const Item& a, const Item& b){
27         return Item(a.sum + b.sum, max(a.maxPref, a.sum + b.maxPref));
28     }
29 };
30
31 struct Nodo {
32     Item value;
33     Nodo *izq = nullptr, *der = nullptr;
34 };
35
36 void update(Nodo*& node, ll inicio, ll fin, ll pos, Item& val) {
37     if (!node) node = new Nodo();
38     if (inicio == fin) {//la actualización, puede ser suma o asignación o
39         // lo que sea, piensa bien
40         node->value = val;
41         return;
42     }
43     ll mid = (inicio + fin) / 2;
44     if (pos <= mid)
45         update(node->izq, inicio, mid, pos, val);
46     else
47         update(node->der, mid + 1, fin, pos, val);
48     //esto está interesante, que es el elemento neutro?
49     auto it = Item();
50     Item& izqVal = node->izq ? node->izq->value : it;
51     Item& derVal = node->der ? node->der->value : it;
52     node->value = Item::merge(izqVal, derVal);
53 }
54
55 Item query(Nodo* node, ll inicio, ll fin, ll l, ll r) {
56     if (!node || r < inicio || l > fin) return Item();
57     if (l <= inicio && fin <= r) return node->value;
58     ll mid = (inicio + fin) / 2;
59     return Item::merge(query(node->izq, inicio, mid, l, r),
60                         query(node->der, mid + 1, fin, l, r));
61 }
62
63 void solve(){
64     Nodo* seg = new Nodo();
65     int n, q; cin >> n >> q;
66     for(int i = 0; i < n; i++){
67         cin >> arr[i];
68         auto it = Item(arr[i], arr[i]);
69         update(seg, 0, n - 1, i, it);
70     }
71     while(q--){
72         int t; cin >> t;
73         if(t == 1){
74             int pos;

```

```

75     cin>>pos>>k;
76     --pos;
77     auto it = Item(k, k);
78     update(seg, 0, n-1, pos, it);
79 } else{
80     int l, r;cin>>l>>r;
81     --l, --r;
82     Item res = query(seg, 0, n-1, l, r);
83     cout<<max(0LL, res.maxPref)<<'\n';
84 }
85 }
86 }
87 }
88
89 signed main(){
90     ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
91     int t = 1;
92     // cin>>t;
93     while(t--){
94         solve();
95     }
96 }

29     vector<pair<int,int>> vals(n); // (value, pos)
30     for(int i=0;i<n;i++){
31         int x; cin >> x;
32         vals[i] = {x, i+1}; // 1-based index for BIT
33     }
34     sort(vals.begin(), vals.end(), [](auto &a, auto &b){
35         if(a.first != b.first) return a.first < b.first;
36         return a.second < b.second;
37     });
38
39     struct Event { int threshold; int l, r; int qi; int sign; };
40     vector<Event> events;
41     events.reserve(2*q);
42     for(int i=0;i<q;i++){
43         int a,b,c,d; cin >> a >> b >> c >> d;
44         // count <= d (sign +1)
45         events.push_back({d, a, b, i, +1});
46         // count <= c-1 (sign -1)
47         events.push_back({c-1, a, b, i, -1});
48     }
49     sort(events.begin(), events.end(), [](const Event &A, const Event &B){
50         return A.threshold < B.threshold;
51     });
52
53     vector<ll> ans(q, 0);
54     Fenwick fw(n);
55     int ptr = 0; // pointer in vals
56
57     for(auto &ev : events){
58         while(ptr < n && vals[ptr].first <= ev.threshold){
59             fw.add(vals[ptr].second, 1); // activate position
60             ptr++;
61         }
62         int cnt = fw.rangeSum(ev.l, ev.r);
63         ans[ev.qi] += 1LL * ev.sign * cnt;
64     }
65
66     for(int i=0;i<q;i++) cout << ans[i] << '\n';
67 }
68 }
```

### 8.0.10 Range\_Interval\_Qualities

```

1 // Given an array x of n integers, your task is to process q queries of
2 // the form: how many integers i satisfy a \le i \le b and c \le x_i \le
3 // d?
4 #include <bits/stdc++.h>
5 using namespace std;
6 using ll = long long;
7
8 struct Fenwick {
9     int n;
10    vector<int> bit;
11    Fenwick(int n=0): n(n), bit(n+1, 0) {}
12    void add(int idx, int val){
13        for(; idx <= n; idx += idx & -idx) bit[idx] += val;
14    }
15    int sum(int idx){
16        int r=0;
17        for(; idx>0; idx -= idx & -idx) r += bit[idx];
18        return r;
19    }
20    int rangeSum(int l, int r){
21        if(r < l) return 0;
22        return sum(r) - sum(l-1);
23    }
24 }
25
26 int main(){
27     ios::sync_with_stdio(false);
28     cin.tie(nullptr);
29     int n, q;
30     if(!(cin >> n >> q)) return 0;
31
32     cin>>pos>>k;
33     --pos;
34     auto it = Item(k, k);
35     update(seg, 0, n-1, pos, it);
36 } else{
37     int l, r;cin>>l>>r;
38     --l, --r;
39     Item res = query(seg, 0, n-1, l, r);
40     cout<<max(0LL, res.maxPref)<<'\n';
41 }
42 }
```

```

29     vector<pair<int,int>> vals(n); // (value, pos)
30     for(int i=0;i<n;i++){
31         int x; cin >> x;
32         vals[i] = {x, i+1}; // 1-based index for BIT
33     }
34     sort(vals.begin(), vals.end(), [](auto &a, auto &b){
35         if(a.first != b.first) return a.first < b.first;
36         return a.second < b.second;
37     });
38
39     struct Event { int threshold; int l, r; int qi; int sign; };
40     vector<Event> events;
41     events.reserve(2*q);
42     for(int i=0;i<q;i++){
43         int a,b,c,d; cin >> a >> b >> c >> d;
44         // count <= d (sign +1)
45         events.push_back({d, a, b, i, +1});
46         // count <= c-1 (sign -1)
47         events.push_back({c-1, a, b, i, -1});
48     }
49     sort(events.begin(), events.end(), [](const Event &A, const Event &B){
50         return A.threshold < B.threshold;
51     });
52
53     vector<ll> ans(q, 0);
54     Fenwick fw(n);
55     int ptr = 0; // pointer in vals
56
57     for(auto &ev : events){
58         while(ptr < n && vals[ptr].first <= ev.threshold){
59             fw.add(vals[ptr].second, 1); // activate position
60             ptr++;
61         }
62         int cnt = fw.rangeSum(ev.l, ev.r);
63         ans[ev.qi] += 1LL * ev.sign * cnt;
64     }
65
66     for(int i=0;i<q;i++) cout << ans[i] << '\n';
67 }
68 }
```

### 8.0.11 Range\_Qualities\_and\_Copies

```

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

```

12 #define all(x) (x).begin(), (x).end()
13 #define sz(x) (int)(x).size()
14 #define pb push_back
15 using vi = vector<int>;
16 const int MOD = 1000000007;
17 const ll INF = 9223372036854775807LL;
18 const int tam = 200005;
19
20 struct Node {
21     int val;
22     Node *l, *r;
23
24     // Constructor hoja
25     Node(int _val) {
26         val = _val;
27         l = r = nullptr;
28     }
29
30     // Constructor combinando hijos
31     Node(Node* L, Node* R) {
32         l = L; r = R;
33         val = L->val + R->val;
34     }
35 };
36
37 // Construcción inicial
38 int arr[tam];
39 Node* build(int tl, int tr) {
40     if (tl == tr)
41         return new Node(arr[tl]);
42     int tm = (tl + tr) / 2;
43     return new Node(build(tl, tm), build(tm+1, tr));
44 }
45
46 Node* update(Node* v, int tl, int tr, int pos, long long new_val) {
47     if (tl == tr) return new Node(new_val);
48     int tm = (tl + tr) / 2;
49     if (pos <= tm) {
50         return new Node(update(v->l, tl, tm, pos, new_val), v->r);
51     } else {
52         return new Node(v->l, update(v->r, tm+1, tr, pos, new_val));
53     }
54 }
55
56 Node* merge(Node* a, Node* b) {
57     if (!a) return b;
58     if (!b) return a;
59     return new Node(a, b);
60 }
61
62 // Query en rango [l,r]
63 Node* query(Node* v, int tl, int tr, int l, int r) {
64     if (l > r) return nullptr;
65     if (l == tl && r == tr) return v;
66     int tm = (tl + tr) / 2;
67     return merge(
68         query(v->l, tl, tm, l, min(r, tm)),
69         query(v->r, tm+1, tr, max(l, tm+1), r)
70     );
71 }
72
73 void solve(){
74     int n, q;cin>>n>>q;
75     for(int i = 0; i<n; i++)cin>>arr[i];
76     vector<Node*> segmentos;
77     segmentos.push_back(build(0, n-1));
78     while(q--){
79         int t;cin>>t;
80         if(t == 1){
81             int k, a, x;cin>>k>>a>>x;
82             k--, a--;
83             auto we = update(segmentos[k], 0, n-1, a, x);
84             segmentos[k] = we;
85         }else if(t == 2){
86             int k, a, b;cin>>k>>a>>b;
87             k--, a--, b--;
88             cout<<query(segmentos[k], 0, n-1, a, b)->val<<'\n';
89         }else{
90             int k;cin>>k;
91             k--;
92             segmentos.push_back(segmentos[k]);
93         }
94     }
95 }
96
97 signed main(){
98     ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
99     int t = 1;
100    // cin>>t;
101    while(t--){
102        solve();
103    }
104 }

```

---

### 8.0.12 Range\_Update\_Queries

```

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

```

```

17 vector<ll> bit;
18 vector<ll> arr;
19
20 void update(int pos, ll delta){
21     while (pos < tam) {
22         bit[pos] += delta;
23         pos += pos & -pos;
24     }
25 }
26
27 ll query(int pos){
28     ll sum = 0;
29     while (pos > 0) {
30         sum += bit[pos];
31         pos -= pos & -pos;
32     }
33     return sum;
34 }
35
36 void range_add(int l, int r, ll val) {
37     update(l, val);
38     update(r + 1, -val);
39 }
40
41 ll point_query(int i) {
42     return query(i);
43 }
44
45 void solve(){
46     int n, q; cin >> n >> q;
47     tam = n+2;
48     arr.resize(n + 1);
49     bit.resize(n + 2, 0);
50     fore(i, 1, n + 1){
51         cin >> arr[i];
52         range_add(i, i, arr[i]);
53     }
54     while (q--){
55         int type; cin >> type;
56         if (type == 1){
57             int l, r, val; cin >> l >> r >> val;
58             range_add(l, r, val);
59         } else {
60             int index; cin >> index;
61             cout << point_query(index) << '\n';
62         }
63     }
64 }
65 signed main(){
66     ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
67     int t = 1;
68     //cin>>t;
69     while(t--){
70         solve();
71     }
72 }

```

### 8.0.13 Salary\_Queries

---

```

1 // A company has n employees with certain salaries. Your task is to keep
2 // track of the salaries and process queries.
3 // Input
4 // The first input line contains two integers n and q: the number of
5 // employees and queries. The employees are numbered 1,2,\ldots,n.
6 // The next line has n integers p_1,p_2,\ldots,p_n: each employee's
7 // salary.
8 // After this, there are q lines describing the queries. Each line has one
9 // of the following forms:
10 // ! k x: change the salary of employee k to x
11 // ? a b: count the number of employees whose salary is between a \ldots b
12 #include <bits/stdc++.h>
13 using namespace std;
14 #define ll long long
15 #define ull unsigned long long
16 #define fore(i, a, b) for (int i = (a); i < (b); i++)
17 #define FOR(i, n) for (int i = 0; i < (n); i++)
18 #define all(x) (x).begin(), (x).end()
19 #define sz(x) (int)(x).size()
20 #define pb push_back
21
22 using vi = vector<int>;
23 const int MOD = 1000000007;
24 const ll INF = 9223372036854775807LL;
25 const int tam = 1;
26
27 #define lsb(x) ((x) & (-x))
28
29 struct BIT {
30     vector<int> bit;
31     BIT(int N) : bit(N + 1) {}
32     void add(int i, int x) {
33         while (i < sz(bit)) {
34             bit[i] += x;
35             i += lsb(i);
36         }
37     }
38     int sum(int i) {
39         int ans = 0;
40         while (i > 0) {
41             ans += bit[i];
42             i -= lsb(i);
43         }
44         return ans;
45     }
46     int sum(int l, int r) {
47         if (l > r) return 0;
48         return sum(r) - sum(l - 1);
49     };
50
51     void solve() {
52         int n, q; cin >> n >> q;
53         vector<int> arr(n);
54         int we = 0;
55         vector<int> v;
56         v.reserve(n+q + 1000);
57     }
58 }

```

```

52 // Input
53 // The first input line contains integers n and m: the size of the array
54 // and the number of updates. The array is indexed 1,2,\ldots,n.
55 // The next line has n integers: x_1,x_2,\ldots,x_n: the initial contents
56 // of the array.
57 // Then there are m lines describing the changes. Each line has two
58 // integers k and x: the value at position k becomes x.
59 #include <bits/stdc++.h>
60 using namespace std;
61 #define int long long
62
63 struct Node {
64     int total_sum, prefix_sum, suffix_sum, max_sum;
65
66     Node() : total_sum(0), prefix_sum(0), suffix_sum(0), max_sum(0) {}
67     Node(int val) : total_sum(val), prefix_sum(val), suffix_sum(val),
68                     max_sum(val) {}
69 };
70
71 // Función para combinar dos nodos
72 Node combine(const Node &left, const Node &right) {
73     Node res;
74     res.total_sum = left.total_sum + right.total_sum;
75     res.prefix_sum = max(left.prefix_sum, left.total_sum +
76                          right.prefix_sum);
77     res.suffix_sum = max(right.suffix_sum, right.total_sum +
78                          left.suffix_sum);
79     res.max_sum = max({left.max_sum, right.max_sum, left.suffix_sum +
80                        right.prefix_sum});
81     return res;
82 }
83
84 class SegmentTree {
85 private:
86     vector<Node> tree;
87     int n;
88
89 void build(const vector<int> &arr, int node, int start, int end) {
90     if (start == end) {
91         tree[node] = Node(arr[start]);
92     } else {
93         int mid = (start + end) / 2;
94         build(arr, 2 * node + 1, start, mid);
95         build(arr, 2 * node + 2, mid + 1, end);
96         tree[node] = combine(tree[2 * node + 1], tree[2 * node + 2]);
97     }
98 }
99
100 Node query(int node, int start, int end, int l, int r) {
101 }
```

### 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.

60

```

52     Node right = query(2 * node + 2, mid + 1, end, L, R);
53     return combine(left, right);
54 }
55
56 void update(int node, int start, int end, int pos, int new_val) {
57     if (start == end) {
58         tree[node] = Node(new_val);
59     } else {
60         int mid = (start + end) / 2;
61         if (pos <= mid) {
62             update(2 * node + 1, start, mid, pos, new_val);
63         } else {
64             update(2 * node + 2, mid + 1, end, pos, new_val);
65         }
66         tree[node] = combine(tree[2 * node + 1], tree[2 * node + 2]);
67     }
68 }
69
70 public:
71 SegmentTree(const vector<int> &arr) {
72     n = arr.size();
73     tree.resize(4 * n);
74     build(arr, 0, 0, n - 1);
75 }
76
77 int max_sum_query(int L, int R) {
78     Node res = query(0, 0, n - 1, L, R);
79     return res.max_sum;
80 }
81
82 void update(int pos, int new_val) {
83     update(0, 0, n - 1, pos, new_val);
84 }
85
86 signed main() {
87     int n, q;
88     cin >> n >> q;
89     vector<int> arr(n);
90     for (int i = 0; i < n; i++) {
91         cin >> arr[i];
92     }
93
94     SegmentTree segTree(arr);
95
96     while (q--) {
97         int pos, x; cin >> pos >> x; pos--;
98         segTree.update(pos, x);
99         int ans = segTree.max_sum_query(0, n - 1);
100        ans = max(0LL, ans);
101        cout << ans << endl;
102    }
103    return 0;
104 }

```

## 8.0.15 Subarray\_Sum\_Questions\_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.

```

1 // You are given an array of n integers and q queries. In each query, your
2 // task is to calculate the maximum subarray sum in the range [a,b].
3 // Empty subarrays (with sum 0) are allowed.
4 #include <bits/stdc++.h>
5 #define int long long
6 using namespace std;
7 #define ll long long
8 #define ull unsigned long long
9 #define fore(i, a, b) for(int i = (a); i < (b); i++)
10 #define FOR(i, n) for(int i = 0; i < (n); i++)
11 #define all(x) (x).begin(), (x).end()
12 #define sz(x) (int)(x).size()
13 #define pb push_back
14 using vi = vector<int>;
15 const int MOD = 1000000007;
16 const ll INF = 9223372036854775807LL;
17 const int tam = 1;
18
19 struct Node {
20     int total_sum, max_sum, min_sum, prefix_sum, suffix_sum;
21     Node() : total_sum(0), max_sum(INT_MIN), min_sum(INT_MAX),
22             prefix_sum(0), suffix_sum(0) {}
23 };
24
25 class SegmentTree {
26 private:
27     vector<Node> tree;
28     int n;
29
30     Node combine(const Node &left, const Node &right) {
31         Node res;
32         res.total_sum = left.total_sum + right.total_sum;
33         res.prefix_sum = max(left.prefix_sum, left.total_sum +
34                             right.prefix_sum);
35         res.suffix_sum = max(right.suffix_sum, right.total_sum +
36                             left.suffix_sum);
37         res.max_sum = max({left.max_sum, right.max_sum, left.suffix_sum +
38                            right.prefix_sum});
39
39         res.min_sum = min({left.min_sum, right.min_sum, left.suffix_sum +
40                            right.prefix_sum});
41         return res;
42     }
43
44     void build(const vector<int> &arr, int v, int tl, int tr) {
45         if (tl == tr) {
46             tree[v].total_sum = arr[tl];
47             tree[v].max_sum = arr[tl];
48             tree[v].min_sum = arr[tl];
49             tree[v].prefix_sum = arr[tl];
50             tree[v].suffix_sum = arr[tl];
51         } else {
52             int tm = (tl + tr) / 2;
53             build(arr, v * 2, tl, tm);
54             build(arr, v * 2 + 1, tm + 1, tr);
55             tree[v] = combine(tree[v * 2], tree[v * 2 + 1]);
56         }
57     }
58
59     Node query(int v, int tl, int tr, int l, int r) {
60         if (l > r) return Node();
61         if (l == tl && r == tr) return tree[v];
62         int tm = (tl + tr) / 2;
63         Node left = query(v * 2, tl, tm, l, min(r, tm));
64         Node right = query(v * 2 + 1, tm + 1, tr, max(l, tm + 1), r);
65         return combine(left, right);
66     }
67
68     int max_sum_query(int L, int R) {
69         Node res = query(0, 0, n - 1, L, R);
70         return res.max_sum;
71     }
72
73     void update(int pos, int new_val) {
74         update(0, 0, n - 1, pos, new_val);
75     }
76
77     Node query(int L, int R) {
78         Node res = query(0, 0, n - 1, L, R);
79         return res;
80     }
81
82     Node combine(Node &left, Node &right) {
83         Node res;
84         res.total_sum = left.total_sum + right.total_sum;
85         res.prefix_sum = max(left.prefix_sum, left.total_sum +
86                             right.prefix_sum);
87         res.suffix_sum = max(right.suffix_sum, right.total_sum +
88                             left.suffix_sum);
89         res.max_sum = max({left.max_sum, right.max_sum, left.suffix_sum +
90                            right.prefix_sum});
91
92         res.min_sum = min({left.min_sum, right.min_sum, left.suffix_sum +
93                            right.prefix_sum});
94         return res;
95     }
96
97     void build(const vector<int> &arr, int v, int tl, int tr) {
98         if (tl == tr) {
99             tree[v].total_sum = arr[tl];
100            tree[v].max_sum = arr[tl];
101            tree[v].min_sum = arr[tl];
102            tree[v].prefix_sum = arr[tl];
103            tree[v].suffix_sum = arr[tl];
104        } else {
105            int tm = (tl + tr) / 2;
106            build(arr, v * 2, tl, tm);
107            build(arr, v * 2 + 1, tm + 1, tr);
108            tree[v] = combine(tree[v * 2], tree[v * 2 + 1]);
109        }
110    }
111
112    Node query(int v, int tl, int tr, int l, int r) {
113        if (l > r) return Node();
114        if (l == tl && r == tr) return tree[v];
115        int tm = (tl + tr) / 2;
116        Node left = query(v * 2, tl, tm, l, min(r, tm));
117        Node right = query(v * 2 + 1, tm + 1, tr, max(l, tm + 1), r);
118        return combine(left, right);
119    }
120
121    int max_sum_query(int L, int R) {
122        Node res = query(0, 0, n - 1, L, R);
123        return res.max_sum;
124    }
125
126    void update(int pos, int new_val) {
127        update(0, 0, n - 1, pos, new_val);
128    }
129
130    Node query(int L, int R) {
131        Node res = query(0, 0, n - 1, L, R);
132        return res;
133    }
134
135    Node combine(Node &left, Node &right) {
136        Node res;
137        res.total_sum = left.total_sum + right.total_sum;
138        res.prefix_sum = max(left.prefix_sum, left.total_sum +
139                            right.prefix_sum);
140        res.suffix_sum = max(right.suffix_sum, right.total_sum +
141                            left.suffix_sum);
142        res.max_sum = max({left.max_sum, right.max_sum, left.suffix_sum +
143                           right.prefix_sum});
144
145        res.min_sum = min({left.min_sum, right.min_sum, left.suffix_sum +
146                           right.prefix_sum});
147        return res;
148    }
149
150    void build(const vector<int> &arr, int v, int tl, int tr) {
151        if (tl == tr) {
152            tree[v].total_sum = arr[tl];
153            tree[v].max_sum = arr[tl];
154            tree[v].min_sum = arr[tl];
155            tree[v].prefix_sum = arr[tl];
156            tree[v].suffix_sum = arr[tl];
157        } else {
158            int tm = (tl + tr) / 2;
159            build(arr, v * 2, tl, tm);
160            build(arr, v * 2 + 1, tm + 1, tr);
161            tree[v] = combine(tree[v * 2], tree[v * 2 + 1]);
162        }
163    }
164
165    Node query(int v, int tl, int tr, int l, int r) {
166        if (l > r) return Node();
167        if (l == tl && r == tr) return tree[v];
168        int tm = (tl + tr) / 2;
169        Node left = query(v * 2, tl, tm, l, min(r, tm));
170        Node right = query(v * 2 + 1, tm + 1, tr, max(l, tm + 1), r);
171        return combine(left, right);
172    }
173
174    int max_sum_query(int L, int R) {
175        Node res = query(0, 0, n - 1, L, R);
176        return res.max_sum;
177    }
178
179    void update(int pos, int new_val) {
180        update(0, 0, n - 1, pos, new_val);
181    }
182
183    Node query(int L, int R) {
184        Node res = query(0, 0, n - 1, L, R);
185        return res;
186    }
187
188    Node combine(Node &left, Node &right) {
189        Node res;
190        res.total_sum = left.total_sum + right.total_sum;
191        res.prefix_sum = max(left.prefix_sum, left.total_sum +
192                            right.prefix_sum);
193        res.suffix_sum = max(right.suffix_sum, right.total_sum +
194                            left.suffix_sum);
195        res.max_sum = max({left.max_sum, right.max_sum, left.suffix_sum +
196                           right.prefix_sum});
197
198        res.min_sum = min({left.min_sum, right.min_sum, left.suffix_sum +
199                           right.prefix_sum});
200        return res;
201    }
202
203    void build(const vector<int> &arr, int v, int tl, int tr) {
204        if (tl == tr) {
205            tree[v].total_sum = arr[tl];
206            tree[v].max_sum = arr[tl];
207            tree[v].min_sum = arr[tl];
208            tree[v].prefix_sum = arr[tl];
209            tree[v].suffix_sum = arr[tl];
210        } else {
211            int tm = (tl + tr) / 2;
212            build(arr, v * 2, tl, tm);
213            build(arr, v * 2 + 1, tm + 1, tr);
214            tree[v] = combine(tree[v * 2], tree[v * 2 + 1]);
215        }
216    }
217
218    Node query(int v, int tl, int tr, int l, int r) {
219        if (l > r) return Node();
220        if (l == tl && r == tr) return tree[v];
221        int tm = (tl + tr) / 2;
222        Node left = query(v * 2, tl, tm, l, min(r, tm));
223        Node right = query(v * 2 + 1, tm + 1, tr, max(l, tm + 1), r);
224        return combine(left, right);
225    }
226
227    int max_sum_query(int L, int R) {
228        Node res = query(0, 0, n - 1, L, R);
229        return res.max_sum;
230    }
231
232    void update(int pos, int new_val) {
233        update(0, 0, n - 1, pos, new_val);
234    }
235
236    Node query(int L, int R) {
237        Node res = query(0, 0, n - 1, L, R);
238        return res;
239    }
240
241    Node combine(Node &left, Node &right) {
242        Node res;
243        res.total_sum = left.total_sum + right.total_sum;
244        res.prefix_sum = max(left.prefix_sum, left.total_sum +
245                            right.prefix_sum);
246        res.suffix_sum = max(right.suffix_sum, right.total_sum +
247                            left.suffix_sum);
248        res.max_sum = max({left.max_sum, right.max_sum, left.suffix_sum +
249                           right.prefix_sum});
250
251        res.min_sum = min({left.min_sum, right.min_sum, left.suffix_sum +
252                           right.prefix_sum});
253        return res;
254    }
255
256    void build(const vector<int> &arr, int v, int tl, int tr) {
257        if (tl == tr) {
258            tree[v].total_sum = arr[tl];
259            tree[v].max_sum = arr[tl];
260            tree[v].min_sum = arr[tl];
261            tree[v].prefix_sum = arr[tl];
262            tree[v].suffix_sum = arr[tl];
263        } else {
264            int tm = (tl + tr) / 2;
265            build(arr, v * 2, tl, tm);
266            build(arr, v * 2 + 1, tm + 1, tr);
267            tree[v] = combine(tree[v * 2], tree[v * 2 + 1]);
268        }
269    }
270
271    Node query(int v, int tl, int tr, int l, int r) {
272        if (l > r) return Node();
273        if (l == tl && r == tr) return tree[v];
274        int tm = (tl + tr) / 2;
275        Node left = query(v * 2, tl, tm, l, min(r, tm));
276        Node right = query(v * 2 + 1, tm + 1, tr, max(l, tm + 1), r);
277        return combine(left, right);
278    }
279
280    int max_sum_query(int L, int R) {
281        Node res = query(0, 0, n - 1, L, R);
282        return res.max_sum;
283    }
284
285    void update(int pos, int new_val) {
286        update(0, 0, n - 1, pos, new_val);
287    }
288
289    Node query(int L, int R) {
290        Node res = query(0, 0, n - 1, L, R);
291        return res;
292    }
293
294    Node combine(Node &left, Node &right) {
295        Node res;
296        res.total_sum = left.total_sum + right.total_sum;
297        res.prefix_sum = max(left.prefix_sum, left.total_sum +
298                            right.prefix_sum);
299        res.suffix_sum = max(right.suffix_sum, right.total_sum +
300                            left.suffix_sum);
301        res.max_sum = max({left.max_sum, right.max_sum, left.suffix_sum +
302                           right.prefix_sum});
303
304        res.min_sum = min({left.min_sum, right.min_sum, left.suffix_sum +
305                           right.prefix_sum});
306        return res;
307    }
308
309    void build(const vector<int> &arr, int v, int tl, int tr) {
310        if (tl == tr) {
311            tree[v].total_sum = arr[tl];
312            tree[v].max_sum = arr[tl];
313            tree[v].min_sum = arr[tl];
314            tree[v].prefix_sum = arr[tl];
315            tree[v].suffix_sum = arr[tl];
316        } else {
317            int tm = (tl + tr) / 2;
318            build(arr, v * 2, tl, tm);
319            build(arr, v * 2 + 1, tm + 1, tr);
320            tree[v] = combine(tree[v * 2], tree[v * 2 + 1]);
321        }
322    }
323
324    Node query(int v, int tl, int tr, int l, int r) {
325        if (l > r) return Node();
326        if (l == tl && r == tr) return tree[v];
327        int tm = (tl + tr) / 2;
328        Node left = query(v * 2, tl, tm, l, min(r, tm));
329        Node right = query(v * 2 + 1, tm + 1, tr, max(l, tm + 1), r);
330        return combine(left, right);
331    }
332
333    int max_sum_query(int L, int R) {
334        Node res = query(0, 0, n - 1, L, R);
335        return res.max_sum;
336    }
337
338    void update(int pos, int new_val) {
339        update(0, 0, n - 1, pos, new_val);
340    }
341
342    Node query(int L, int R) {
343        Node res = query(0, 0, n - 1, L, R);
344        return res;
345    }
346
347    Node combine(Node &left, Node &right) {
348        Node res;
349        res.total_sum = left.total_sum + right.total_sum;
350        res.prefix_sum = max(left.prefix_sum, left.total_sum +
351                            right.prefix_sum);
352        res.suffix_sum = max(right.suffix_sum, right.total_sum +
353                            left.suffix_sum);
354        res.max_sum = max({left.max_sum, right.max_sum, left.suffix_sum +
355                           right.prefix_sum});
356
357        res.min_sum = min({left.min_sum, right.min_sum, left.suffix_sum +
358                           right.prefix_sum});
359        return res;
360    }
361
362    void build(const vector<int> &arr, int v, int tl, int tr) {
363        if (tl == tr) {
364            tree[v].total_sum = arr[tl];
365            tree[v].max_sum = arr[tl];
366            tree[v].min_sum = arr[tl];
367            tree[v].prefix_sum = arr[tl];
368            tree[v].suffix_sum = arr[tl];
369        } else {
370            int tm = (tl + tr) / 2;
371            build(arr, v * 2, tl, tm);
372            build(arr, v * 2 + 1, tm + 1, tr);
373            tree[v] = combine(tree[v * 2], tree[v * 2 + 1]);
374        }
375    }
376
377    Node query(int v, int tl, int tr, int l, int r) {
378        if (l > r) return Node();
379        if (l == tl && r == tr) return tree[v];
380        int tm = (tl + tr) / 2;
381        Node left = query(v * 2, tl, tm, l, min(r, tm));
382        Node right = query(v * 2 + 1, tm + 1, tr, max(l, tm + 1), r);
383        return combine(left, right);
384    }
385
386    int max_sum_query(int L, int R) {
387        Node res = query(0, 0, n - 1, L, R);
388        return res.max_sum;
389    }
390
391    void update(int pos, int new_val) {
392        update(0, 0, n - 1, pos, new_val);
393    }
394
395    Node query(int L, int R) {
396        Node res = query(0, 0, n - 1, L, R);
397        return res;
398    }
399
400    Node combine(Node &left, Node &right) {
401        Node res;
402        res.total_sum = left.total_sum + right.total_sum;
403        res.prefix_sum = max(left.prefix_sum, left.total_sum +
404                            right.prefix_sum);
405        res.suffix_sum = max(right.suffix_sum, right.total_sum +
406                            left.suffix_sum);
407        res.max_sum = max({left.max_sum, right.max_sum, left.suffix_sum +
408                           right.prefix_sum});
409
410        res.min_sum = min({left.min_sum, right.min_sum, left.suffix_sum +
411                           right.prefix_sum});
412        return res;
413    }
414
415    void build(const vector<int> &arr, int v, int tl, int tr) {
416        if (tl == tr) {
417            tree[v].total_sum = arr[tl];
418            tree[v].max_sum = arr[tl];
419            tree[v].min_sum = arr[tl];
420            tree[v].prefix_sum = arr[tl];
421            tree[v].suffix_sum = arr[tl];
422        } else {
423            int tm = (tl + tr) / 2;
424            build(arr, v * 2, tl, tm);
425            build(arr, v * 2 + 1, tm + 1, tr);
426            tree[v] = combine(tree[v * 2], tree[v * 2 + 1]);
427        }
428    }
429
430    Node query(int v, int tl, int tr, int l, int r) {
431        if (l > r) return Node();
432        if (l == tl && r == tr) return tree[v];
433        int tm = (tl + tr) / 2;
434        Node left = query(v * 2, tl, tm, l, min(r, tm));
435        Node right = query(v * 2 + 1, tm + 1, tr, max(l, tm + 1), r);
436        return combine(left, right);
437    }
438
439    int max_sum_query(int L, int R) {
440        Node res = query(0, 0, n - 1, L, R);
441        return res.max_sum;
442    }
443
444    void update(int pos, int new_val) {
445        update(0, 0, n - 1, pos, new_val);
446    }
447
448    Node query(int L, int R) {
449        Node res = query(0, 0, n - 1, L, R);
450        return res;
451    }
452
453    Node combine(Node &left, Node &right) {
454        Node res;
455        res.total_sum = left.total_sum + right.total_sum;
456        res.prefix_sum = max(left.prefix_sum, left.total_sum +
457                            right.prefix_sum);
458        res.suffix_sum = max(right.suffix_sum, right.total_sum +
459                            left.suffix_sum);
460        res.max_sum = max({left.max_sum, right.max_sum, left.suffix_sum +
461                           right.prefix_sum});
462
463        res.min_sum = min({left.min_sum, right.min_sum, left.suffix_sum +
464                           right.prefix_sum});
465        return res;
466    }
467
468    void build(const vector<int> &arr, int v, int tl, int tr) {
469        if (tl == tr) {
470            tree[v].total_sum = arr[tl];
471            tree[v].max_sum = arr[tl];
472            tree[v].min_sum = arr[tl];
473            tree[v].prefix_sum = arr[tl];
474            tree[v].suffix_sum = arr[tl];
475        } else {
476            int tm = (tl + tr) / 2;
477            build(arr, v * 2, tl, tm);
478            build(arr, v * 2 + 1, tm + 1, tr);
479            tree[v] = combine(tree[v * 2], tree[v * 2 + 1]);
480        }
481    }
482
483    Node query(int v, int tl, int tr, int l, int r) {
484        if (l > r) return Node();
485        if (l == tl && r == tr) return tree[v];
486        int tm = (tl + tr) / 2;
487        Node left = query(v * 2, tl, tm, l, min(r, tm));
488        Node right = query(v * 2 + 1, tm + 1, tr, max(l, tm + 1), r);
489        return combine(left, right);
490    }
491
492    int max_sum_query(int L, int R) {
493        Node res = query(0, 0, n - 1, L, R);
494        return res.max_sum;
495    }
496
497    void update(int pos, int new_val) {
498        update(0, 0, n - 1, pos, new_val);
499    }
500
501    Node query(int L, int R) {
502        Node res = query(0, 0, n - 1, L, R);
503        return res;
504    }
505
506    Node combine(Node &left, Node &right) {
507        Node res;
508        res.total_sum = left.total_sum + right.total_sum;
509        res.prefix_sum = max(left.prefix_sum, left.total_sum +
510                            right.prefix_sum);
511        res.suffix_sum = max(right.suffix_sum, right.total_sum +
512                            left.suffix_sum);
513        res.max_sum = max({left.max_sum, right.max_sum, left.suffix_sum +
514                           right.prefix_sum});
515
516        res.min_sum = min({left.min_sum, right.min_sum, left.suffix_sum +
517                           right.prefix_sum});
518        return res;
519    }
520
521    void build(const vector<int> &arr, int v, int tl, int tr) {
522        if (tl == tr) {
523            tree[v].total_sum = arr[tl];
524            tree[v].max_sum = arr[tl];
525            tree[v].min_sum = arr[tl];
526            tree[v].prefix_sum = arr[tl];
527            tree[v].suffix_sum = arr[tl];
528        } else {
529            int tm = (tl + tr) / 2;
530            build(arr, v * 2, tl, tm);
531            build(arr, v * 2 + 1, tm + 1, tr);
532            tree[v] = combine(tree[v * 2], tree[v * 2 + 1]);
533        }
534    }
535
536    Node query(int v, int tl, int tr, int l, int r) {
537        if (l > r) return Node();
538        if (l == tl && r == tr) return tree[v];
539        int tm = (tl + tr) / 2;
540        Node left = query(v * 2, tl, tm, l, min(r, tm));
541        Node right = query(v * 2 + 1, tm + 1, tr, max(l, tm + 1), r);
542        return combine(left, right);
543    }
544
545    int max_sum_query(int L, int R) {
546        Node res = query(0, 0, n - 1, L, R);
547        return res.max_sum;
548    }
549
550    void update(int pos, int new_val) {
551        update(0, 0, n - 1, pos, new_val);
552    }
553
554    Node query(int L, int R) {
555        Node res = query(0, 0, n - 1, L, R);
556        return res;
557    }
558
559    Node combine(Node &left, Node &right) {
560        Node res;
561        res.total_sum = left.total_sum + right.total_sum;
562        res.prefix_sum = max(left.prefix_sum, left.total_sum +
563                            right.prefix_sum);
564        res.suffix_sum = max(right.suffix_sum, right.total_sum +
565                            left.suffix_sum);
566        res.max_sum = max({left.max_sum, right.max_sum, left.suffix_sum +
567                           right.prefix_sum});
568
569        res.min_sum = min({left.min_sum, right.min_sum, left.suffix_sum +
570                           right.prefix_sum});
571        return res;
572    }
573
574    void build(const vector<int> &arr, int v, int tl, int tr) {
575        if (tl == tr) {
576            tree[v].total_sum = arr[tl];
577            tree[v].max_sum = arr[tl];
578            tree[v].min_sum = arr[tl];
579            tree[v].prefix_sum = arr[tl];
580            tree[v].suffix_sum = arr[tl];
581        } else {
582            int tm = (tl + tr) / 2;
583            build(arr, v * 2, tl, tm);
584            build(arr, v * 2 + 1, tm + 1, tr);
585            tree[v] = combine(tree[v * 2], tree[v * 2 + 1]);
586        }
587    }
588
589    Node query(int v, int tl, int tr, int l, int r) {
590        if (l > r) return Node();
591        if (l == tl && r == tr) return tree[v];
592        int tm = (tl + tr) / 2;
593        Node left = query(v * 2, tl, tm, l, min(r, tm));
594        Node right = query(v * 2 + 1, tm + 1, tr, max(l, tm + 1), r);
595        return combine(left, right);
596    }
597
598    int max_sum_query(int L, int R) {
599        Node res = query(0, 0, n - 1, L, R);
600        return res.max_sum;
601    }
602
603    void update(int pos, int new_val) {
604        update(0, 0, n - 1, pos, new_val);
605    }
606
607    Node query(int L, int R) {
608        Node res = query(0, 0, n - 1, L, R);
609        return res;
610    }
611
612    Node combine(Node &left, Node &right) {
613        Node res;
614        res.total_sum = left.total_sum + right.total_sum;
615        res.prefix_sum = max(left.prefix_sum, left.total_sum +
616                            right.prefix_sum);
617        res.suffix_sum = max(right.suffix_sum, right.total_sum +
618                            left.suffix_sum);
619        res.max_sum = max({left.max_sum, right.max_sum, left.suffix_sum +
620                           right.prefix_sum});
621
622        res.min_sum = min({left.min_sum, right.min_sum, left.suffix_sum +
623                           right.prefix_sum});
624        return res;
625    }
626
627    void build(const vector<int> &arr, int v, int tl, int tr) {
628        if (tl == tr) {
629            tree[v].total_sum = arr[tl];
630            tree[v].max_sum = arr[tl];
631            tree[v].min_sum = arr[tl];
632            tree[v].prefix_sum = arr[tl];
633            tree[v].suffix_sum = arr[tl];
634        } else {
635            int tm = (tl + tr) / 2;
636            build(arr, v * 2, tl, tm);
637            build(arr, v * 2 + 1, tm + 1, tr);
638            tree[v] = combine(tree[v * 2], tree[v * 2 + 1]);
639        }
640    }
641
642    Node query(int v, int tl, int tr, int l, int r) {
643        if (l > r) return Node();
644        if (l == tl && r == tr) return tree[v];
645        int tm = (tl + tr) / 2;
646        Node left = query(v * 2, tl, tm, l, min(r, tm));
647        Node right = query(v * 2 + 1, tm + 1, tr, max(l, tm + 1), r);
648        return combine(left, right);
649    }
650
651    int max_sum_query(int L, int R) {
652        Node res = query(0, 0, n - 1, L, R);
653        return res.max_sum;
654    }
655
656    void update(int pos, int new_val) {
657        update(0, 0, n - 1, pos, new_val);
658    }
659
660    Node query(int L, int R) {
661        Node res = query(0, 0, n - 1, L, R);
662        return res;
663    }
664
665    Node combine(Node &left, Node &right) {
666        Node res;
667        res.total_sum = left.total_sum + right.total_sum;
668        res.prefix_sum = max(left.prefix_sum, left.total_sum +
669                            right.prefix_sum);
670        res.suffix_sum = max(right.suffix_sum, right.total_sum +
671                            left.suffix_sum);
672        res.max_sum = max({left.max_sum, right.max_sum, left.suffix_sum +
673                           right.prefix_sum});
674
675        res.min_sum = min({left.min_sum, right.min_sum, left.suffix_sum +
676                           right.prefix_sum});
677        return res;
678    }
679
680    void build(const vector<int> &arr, int v, int tl, int tr) {
681        if (tl == tr) {
682            tree[v].total_sum = arr[tl];
683            tree[v].max_sum = arr[tl];
684            tree[v].min_sum = arr[tl];
685            tree[v].prefix_sum = arr[tl];
686            tree[v].suffix_sum = arr[tl];
687        } else {
688            int tm = (tl + tr) / 2;
689            build(arr, v * 2, tl, tm);
690            build(arr, v * 2 + 1, tm + 1, tr);
691            tree[v] = combine(tree[v * 2], tree[v * 2 + 1]);
692        }
693    }
694
695    Node query(int v, int tl, int tr, int l, int r) {
696        if (l > r) return Node();
697        if (l == tl && r == tr) return tree[v];
698        int tm = (tl + tr) / 2;
699        Node left = query(v * 2, tl, tm, l, min(r, tm));
700        Node right = query(v * 2 + 1, tm + 1, tr, max(l, tm + 1), r);
701        return combine(left, right);
702    }
703
704    int max_sum_query(int L, int R) {
705        Node res = query(0, 0, n - 1, L, R);
706        return res.max_sum;
707    }
708
709    void update(int pos, int new_val) {
710        update(0, 0, n - 1, pos, new_val);
711    }
712
713    Node query(int L, int R) {
714        Node res = query(0, 0, n - 1, L, R);
715        return res;
716    }
717
718    Node combine(Node &left, Node &right) {
719        Node res;
720        res.total_sum = left.total_sum + right.total_sum;
721        res.prefix_sum = max(left.prefix_sum, left.total_sum +
722                            right.prefix_sum);
723        res.suffix_sum = max(right.suffix_sum, right.total_sum +
724                            left.suffix_sum);
725        res.max_sum = max({left.max_sum, right.max_sum, left.suffix_sum +
726                           right.prefix_sum});
727
728        res.min_sum = min({left.min_sum, right.min_sum, left.suffix_sum +
729                           right.prefix_sum});
730        return res;
731    }
732
733    void build(const vector<int> &arr, int v, int tl, int tr) {
734        if (tl == tr) {
735            tree[v].total_sum = arr[tl];
736            tree[v].max_sum = arr[tl];
737            tree[v].min_sum = arr[tl];
738            tree[v].prefix_sum = arr[tl];
739            tree[v].suffix_sum = arr[tl];
740        } else {
741            int tm = (tl + tr) / 2;
742            build(arr, v * 2, tl, tm);
743            build(arr, v * 2 + 1, tm + 1, tr);
744            tree[v] = combine(tree[v * 2], tree[v * 2 + 1]);
745        }
746    }
747
748    Node query(int v, int tl, int tr, int l, int r) {
749        if (l > r) return Node();
750        if (l == tl && r == tr) return tree[v];
751        int tm = (tl + tr) / 2;
752        Node left = query(v * 2, tl, tm, l, min(r, tm));
753        Node right = query(v * 2 + 1, tm + 1, tr, max(l, tm + 1), r);
754        return combine(left, right);
755    }
756
757    int max_sum_query(int L, int R) {
758        Node res = query(0, 0, n - 1, L, R);
759        return res.max_sum;
760    }
761
762    void update(int pos, int new_val) {
763        update(0, 0, n - 1, pos, new_val);
764    }
765
766    Node query(int L, int R) {
767        Node res = query(0, 0, n - 1, L, R);
768        return res;
769    }
770
771    Node combine(Node &left, Node &right) {
772        Node res;
773        res.total_sum = left.total_sum + right.total_sum;
774        res.prefix_sum = max(left.prefix_sum, left.total_sum +
775                            right.prefix_sum);
776        res.suffix_sum = max(right.suffix_sum, right.total_sum +
777                            left.suffix_sum);
778        res.max_sum = max({left.max_sum, right.max_sum, left.suffix_sum +
779                           right.prefix_sum});
780
781        res.min_sum = min({left.min_sum, right.min_sum, left.suffix_sum +
782                           right.prefix_sum});
783        return res;
784    }
785
786    void build(const vector<int> &arr, int v, int tl, int tr) {
787        if (tl == tr) {
788            tree[v].total_sum = arr[tl];
789            tree[v].max_sum = arr[tl];
790            tree[v].min_sum = arr[tl];
791            tree[v].prefix_sum = arr[tl];
792            tree[v].suffix_sum = arr[tl];
793        } else {
794            int tm = (tl + tr) / 2;
795            build(arr, v * 2, tl, tm);
796            build(arr, v * 2 + 1, tm + 1, tr);
797            tree[v] = combine(tree[v * 2], tree[v * 2 + 1]);
798        }
799    }
800
801    Node query(int v, int tl, int tr, int l, int r) {
802        if (l > r) return Node();
803        if (l == tl && r == tr) return tree[v];
804        int tm = (tl + tr) / 2;
805        Node left = query(v * 2, tl, tm, l, min(r, tm));
806        Node right = query(v * 2 + 1, tm + 1, tr, max(l, tm + 1), r);
807        return combine(left, right);
808    }
809
810    int max_sum_query(int L, int R) {
811        Node res = query(0, 0, n - 1, L, R);
812        return res.max_sum;
813    }
814
815    void update(int pos, int new_val) {
816        update(0, 0, n - 1, pos, new_val);
817    }
818
819    Node query(int L, int R) {
820        Node res = query(0, 0, n - 1, L, R);
821        return res;
822    }
823
824    Node combine(Node &left, Node &right) {
825        Node res;
826        res.total_sum = left.total_sum + right.total_sum;
827        res.prefix_sum = max(left.prefix_sum, left.total_sum +
828                            right.prefix_sum);
829
```

```

49         build(arr, v * 2 + 1, tm + 1, tr);
50         tree[v] = combine(tree[v * 2], tree[v * 2 + 1]);
51     }
52 }
53
54 Node query(int v, int tl, int tr, int l, int r) {
55     if (l > r) return Node();
56     if (l == tl && r == tr) return tree[v];
57     int tm = (tl + tr) / 2;
58     Node left_result = query(v * 2, tl, tm, l, min(r, tm));
59     Node right_result = query(v * 2 + 1, tm + 1, tr, max(l, tm + 1),
60                                r);
61     return combine(left_result, right_result);
62 }
63
64 public:
65     SegmentTree(const vector<int> &arr) {
66         n = arr.size();
67         tree.resize(n * 4);
68         build(arr, 1, 0, n - 1);
69     }
70
71     int getMaxSum(int l, int r) {
72         return query(1, 0, n - 1, l, r).max_sum;
73     }
74
75     int getMinSum(int l, int r) {
76         return query(1, 0, n - 1, l, r).min_sum;
77     }
78
79     void solve(){
80         int n, q; cin >> n >> q;
81         vector<int> arr(n);
82         for(int i = 0; i < n; i++) cin >> arr[i];
83
84         SegmentTree st(arr);
85         /*cuando falta poco tiempo, es mejor que los 3 nos enfoquemos en un
86          ↵ problema, ya no lo vale*/
87         while(q--){
88             int l, r; cin >> l >> r;
89             l--, r--;
90             cout << st.getMaxSum(l, r) << '\n';
91         }
92     }
93
94     signed main(){
95         ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
96         int t = 1;
97         // cin >> t;
98         while(t--){
99             solve();
100        }
101    }

```

## 9 Sliding Window

### 9.0.1 Sliding\_Window\_Median

---

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 // You are given an array of n integers. Your task is to calculate the
4 // → median of each window of k elements, from left to right.
5 // The median is the middle element when the elements are sorted. If the
6 // → number of elements is even, there are two possible medians and we
7 // → assume that the median is the smaller of them.
8 #include <ext/pb_ds/assoc_container.hpp>
9 #include <ext/pb_ds/tree_policy.hpp>
10 using namespace __gnu_pbds;
11 typedef pair<int, int> pii;
12 typedef tree<pii, null_type, less<pair<int, int>>, rb_tree_tag,
13             tree_order_statistics_node_update> ordered_ms;
14
15 struct ordMultiset {
16     int count;
17     ordered_ms oset;
18     ordMultiset() { count = 0; }
19
20     int count_occurrences(int x) {
21         return oset.order_of_key({x + 1, 0}) - oset.order_of_key({x, 0});
22     }
23
24     void erase_one(int x) {
25         auto it = oset.lower_bound({x, 0});
26         if (it != oset.end() && it->first == x) { oset.erase(it); }
27
28         int kth_element(int k) { return oset.find_by_order(k)->first; }
29
30         void insert(int x) {
31             oset.insert({x, count++});
32         }
33
34     void solve() {
35         int n, k; cin >> n >> k;
36         vector<int> arr(n);
37         for(int i = 0; i < n; i++) cin >> arr[i];
38         ordMultiset oms;
39         int r;
40         for(r = 0; r < k - 1; r++){
41             oms.insert(arr[r]);
42         }
43         int l = 0;
44         for(; r < n; r++){
45             oms.insert(arr[r]);
46             cout << oms.kth_element((r - l) / 2) << ' ';
47             oms.erase_one(arr[l]);
48             l++;
49         }
50         cout << endl;
51     }
52 }

```

```

51 }
52
53 signed main() {
54     ios::sync_with_stdio(0);
55     cin.tie(0);
56     cout.tie(0);
57     int t = 1;
58     //cin >> t;
59     while (t--) { solve(); }
60 }
```

## 10 Sorting and Searching

### 10.0.1 Array\_Division

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 #define int long long
4 bool f(vector<int>& arr, int maxSum, int k) {
5     int acu = 0;
6     int sub = 1;
7     for (int i = 0; i < (int)arr.size(); i++) {
8         if (acu + arr[i] > maxSum) {
9             sub++;
10            acu = arr[i];
11            if (sub > k) {
12                return false;
13            }
14        } else {
15            acu += arr[i];
16        }
17    }
18    return sub <= k;
19 }
20 // You are given an array containing n positive integers.
21 // Your task is to divide the array into k subarrays so that the maximum
22 // sum in a subarray is as small as possible.
23 signed main() {
24     int n, k;
25     cin >> n >> k;
26     vector<int> arr(n);
27     for (int i = 0; i < n; i++) cin >> arr[i];
28     int l = *max_element(arr.begin(), arr.end()); // Ajuste: el valor
29     // minimo debe ser el mayor elemento
30     int r = accumulate(arr.begin(), arr.end(), 0LL); // Ajuste: el valor
31     // maximo puede ser la suma de todos los elementos
32     int res = -1;
33     while (l <= r) {
34         int mid = (l + r) / 2;
35         if (f(arr, mid, k)) {
36             res = mid;
37             r = mid - 1;
38         } else {
39             l = mid + 1;
40     }
41     }
42 }
```

---

```

39     }
40 }
41 cout << res << endl;
42 }
```

---

### 10.0.2 Collecting\_Numbers

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define int long long
4 #define f first
5 #define sst stringstream
6 #define s second
7 #define pb push_back
8 #define sz(x) (int)(x).size()
9 #define all(a) (a).begin(), (a).end()
10 #define rall(a) (a).rbegin(), (a).rend()
11 #define fore(i, a, n) for(int i = (a); i < (n); i++)
12 #define forb(i, n) for(int i = (n) - 1; i >= 0; i--)
13 #define popcorn(x) __builtin_popcountll(x);
14 typedef pair<int, int> pii;
15 typedef vector<int> vi;
16 const int MOD = 1000000007;
17 const double EPS = 1e-9;
18 const double PI = acos(-1);
19 const int INF = 1e18;
20 // PLUS ULTRA RECARGADO!!!
21 // You are given an array that contains each number between 1 \dots n
22 // exactly once. Your task is to collect the numbers from 1 to n in
23 // increasing order.
24 // On each round, you go through the array from left to right and collect
25 // as many numbers as possible. What will be the total number of rounds?
26
27 void solve() {
28     int n;
29     cin >> n;
30     vector<pair<int, int>> aux(n);
31     for (int i = 0; i < n; i++) {
32         int x; cin >> x;
33         aux[i] = {x, i+1};
34     }
35     sort(all(aux));
36     int ans = 1;
37     int prev = 0;
38     for (int i = 0; i < n; i++) {
39         int act = aux[i].second;
40         if (!(act > prev)) {
41             ans++;
42         }
43         prev = act;
44     }
45     cout << ans << endl;
46 }
47 signed main() {
48     ios::sync_with_stdio(0);
49 }
```

```

46    cin.tie(0);
47    cout.tie(0);
48    // int t;cin>>t;while(t--)solve();
49    solve();
50 }

```

### 10.0.3 Collecting\_Numbers\_II

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define ll long long
4 #define ull unsigned long long
5 #define fore(i, a, b) for(int i = (a); i<(b); i++)
6 #define FOR(i, n) for(int i = 0; i<(n); i++)
7 #define all(x) (x).begin(), (x).end()
8 #define sz(x) (int)(x).size()
9 #define pb push_back
10 using vi = vector<int>;
11 const int MOD = 1000000007;
12 const ll INF = 9223372036854775807LL;
13 const int tam = 1;
14 // You are given an array that contains each number between 1 \dots n
15 // exactly once. Your task is to collect the numbers from 1 to n in
16 // increasing order.
17 // On each round, you go through the array from left to right and collect
18 // as many numbers as possible.
19 // Given m operations that swap two numbers in the array, your task is to
20 // report the number of rounds after each operation.
21 void solve(){
22     int n, q;cin>>n>>q;
23     vector<int> arr(n+1), pos(n+1);
24     fore(i, 1, n+1){
25         cin>>arr[i];
26         pos[arr[i]] = i;
27     }
28     int inv = 1;
29     for(int i = 2; i<=n; i++){
30         if(pos[i] < pos[i-1])inv++;
31     }
32     while(q--){
33         set<pair<int, int>> mod;
34         int x, y;cin>>x>>y;
35         int xelem = arr[x];
36         int yelem = arr[y];
37         if(xelem > 1){
38             mod.insert({xelem-1, xelem});
39         }
40         if(xelem < n){
41             mod.insert({xelem, xelem+1});
42         }
43         if(yelem > 1){
44             mod.insert({yelem-1, yelem});
45         }
46         if(yelem < n){
47             mod.insert({yelem, yelem+1});
48         }
49     }
50 }

```

```

44     mod.insert({yelem, yelem+1});
45 }
46 for(auto[left, right]:mod){
47     if(pos[left] > pos[right]){
48         inv--;
49     }
50 }
51 swap(arr[x], arr[y]);
52 pos[xelem] = y;
53 pos[yelem] = x;
54 for(auto[left, right]:mod){
55     if(pos[left] > pos[right]){
56         inv++;
57     }
58 }
59 cout<<inv<<endl;
60 }
61 }
62 }
63 }
64 signed main(){
65     ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
66     int t = 1;
67     //cin>>t;
68     while(t--){
69         solve();
70     }
71 }

```

### 10.0.4 Concert\_Tickets

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define int long long
4 // There are n concert tickets available, each with a certain price. Then,
4 // m customers arrive, one after another.
5 // Each customer announces the maximum price they are willing to pay for a
5 // ticket, and after this, they will get a ticket with the nearest
5 // possible price such that it does not exceed the maximum price.
6
7 signed main() {
8     int n, m;
9     cin >> n >> m;
10    set<pair<int, int>> st;
11    for (int i = 0; i < n; i++) {
12        int x;
13        cin >> x;
14        st.insert({x, i});
15    }
16    while (m--) {
17        int q;
18        cin >> q;
19        if (st.empty()) {
20            cout << -1 << endl;
21            continue;
22        }
23        auto it = st.lower_bound(q);
24        if (it == st.begin())
25            cout << -1 << endl;
26        else
27            cout << (*it).second << endl;
28    }
29 }

```

```

22
23     }
24     pair<int, int> c = {q, 0};
25     auto it = st.lower_bound(c);
26     //cout << "para " << m << ":" << (it->first) << endl;
27     if (it == st.end()) {
28         it--;
29     }
30     if (it == st.begin()) {
31         if (it->first <= q) {
32             cout << (it->first) << endl;
33             st.erase(it);
34         } else {
35             cout << -1 << endl;
36         }
37     } else {
38         if (it->first <= q) {
39             cout << (it->first) << endl;
40             st.erase(it);
41         } else {
42             it--;
43             cout << (it->first) << endl;
44             st.erase(it);
45         }
46     }
47 }
```

## 10.0.5 Distinct\_Numbers

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define ll long long
4 const int tam = 1;
5
6 void solve() {
7     int n;
8     cin >> n;
9     set<int> st;
10    int x;
11    for (int i = 0; i < n; i++) {
12        cin >> x;
13        st.insert(x);
14    }
15    cout << (int)(st.size()) << endl;
16 }
17
18 signed main() {
19     // ios::sync_with_stdio(0);
20     // cin.tie(0);
21     // cout.tie(0);
22     int t = 1;
23     // cin>t;
24     while (t--) { solve(); }
25 }
```

## 10.0.6 Distinct\_Values\_Subarray

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define ll long long
4 //Given an array of n integers, count the number of subarrays where each
5 //element is distinct.
6 int main() {
7     ios::sync_with_stdio(false);
8     cin.tie(nullptr);
9
10    int n;
11    cin >> n;
12    vector<ll> a(n);
13    for (int i = 0; i < n; i++) cin >> a[i];
14
15    unordered_map<ll,int> last;
16    ll ans = 0;
17    int l = 0;
18
19    for (int r = 0; r < n; r++) {
20        if (last.count(a[r]) && last[a[r]] >= l) {
21            l = last[a[r]] + 1;
22        }
23        ans += (r - l + 1);
24        last[a[r]] = r;
25    }
26
27    cout << ans << "\n";
}
```

## 10.0.7 Distinct\_Values\_Subarrays\_II

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define ll long long
4 #define ull unsigned long long
5 #define fore(i, a, b) for (int i = (a); i < (b); i++)
6 #define FOR(i, n) for (int i = 0; i < (n); i++)
7 #define all(x) (x).begin(), (x).end()
8 #define sz(x) (int)(x).size()
9 #define pb push_back
10 using vi = vector<int>;
11 const int MOD = 1000000007;
12 const ll INF = 9223372036854775807LL;
13 const int tam = 1;
14 ll g(ll tam) { return tam * (tam + 1) / 2; }
15 // Given an array of n integers, your task is to calculate the number of
16 // subarrays that have at most k distinct values.
17
18 void solve() {
19     int n, k;
20     cin >> n >> k;
21     vi arr(n);
```

```

21 fore(i, 0, n) cin >> arr[i];
22
23 map<int, int> freq;
24 int l = 0;
25 ll ans = 0;
26
27 fore(r, 0, n) {
28     freq[arr[r]]++;
29     while ((int)freq.size() > k) {
30         freq[arr[l]]--;
31         if (freq[arr[l]] == 0) freq.erase(arr[l]);
32         l++;
33     }
34     ans += (r - l + 1);
35 }
36 cout << ans << '\n';
37
38
39
40 signed main() {
41     ios::sync_with_stdio(0);
42     cin.tie(0);
43     cout.tie(0);
44     int t = 1;
45     // cin>>t;
46     while (t--) { solve(); }
47 }

```

---

## 10.0.8 Distinct\_Values\_Subsequences

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define int long long
4 #define ll long long
5 #define ull unsigned long long
6 #define fore(i, a, b) for(int i = (a); i<(b); i++)
7 #define FOR(i, n) for(int i = 0; i<(n); i++)
8 #define all(x) (x).begin(), (x).end()
9 #define sz(x) (int)(x).size()
10 #define pb push_back
11 using vi = vector<int>;
12 const int MOD = 1000000007;
13 const ll INF = 9223372036854775807LL;
14 const int tam = 1;
15 unordered_map<int, int> mp;
16 // Given an array of n integers, count the number of subsequences where
17 // each element is distinct.
18 // A subsequence is a sequence of array elements from left to right that
19 // may have gaps.
20 void solve(){
21     int n;cin>>n;
22     mp.reserve(n+10);
23     for(int i = 0; i<n; i++){
24         int x;cin>>x;
25         if(mp.find(x) != mp.end()){
26             mp[x]++;
27         } else {
28             mp[x] = 1;
29         }
30     }
31     int ans = 1;
32     for(auto p:mp){
33         ans*=p.second+1;
34         ans%=MOD;
35     }
36     cout<<ans-1<<'\n';
37 }
38
39
40 signed main(){
41     ios::sync_with_stdio(0);
42     cin.tie(0);
43     cout.tie(0);
44     int t = 1;
45     // cin>>t;
46     while(t--){
47         solve();
48     }
49 }

```

```

24     mp[x]++;
25 }
26
27 int ans = 1;
28 for(auto p:mp){
29     ans*=(p.second+1);
30     ans%=MOD;
31 }
32 cout<<ans-1<<'\n';
33
34 signed main(){
35     ios::sync_with_stdio(0);
36     cin.tie(0);
37     cout.tie(0);
38     int t = 1;
39     // cin>>t;
40     while(t--){
41         solve();
42     }
43 }

```

---

## 10.0.9 Factory\_Machines

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define int long long
4 #define ll long long
5 #define ull unsigned long long
6 #define fore(i, a, b) for(int i = (a); i<(b); i++)
7 #define FOR(i, n) for(int i = 0; i<(n); i++)
8 #define all(x) (x).begin(), (x).end()
9 #define sz(x) (int)(x).size()
10 #define pb push_back
11 using vi = vector<int>;
12 const int MOD = 1000000007;
13 const ll INF = 9223372036854775807LL;
14 const int tam = 1;
15 bool f(vector<int>& arr, int k, int mid){
16     int suma = 0;
17     for(int&x:arr){
18         suma+=(mid/x);
19         if(suma>=k) return 1;
20     }
21     return 0;
22 }
23 void solve(){
24     int n, k;cin>>n>>k;
25     vector<int> arr(n);
26     for(int i = 0; i<n; i++) cin>>arr[i];
27     ll l = 1, r = 1000000000000000000LL, ans;
28     while(l<=r){
29         ll mid = l + (r-l)/2;
30         if(f(arr, k, mid)){
31             ans = mid;
32             r = mid-1;
33         } else{
34             l = mid+1;
35         }
36     }
37 }
38
39
40 signed main(){
41     ios::sync_with_stdio(0);
42     cin.tie(0);
43     cout.tie(0);
44     int t = 1;
45     // cin>>t;
46     while(t--){
47         solve();
48     }
49 }

```

```

35     }
36 }
37 cout<<ans<<endl;
38 }
39
40 signed main(){
41     ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
42     int t = 1;
43 //    cin>>t;
44     while(t--){
45         solve();
46     }
47 }
```

### 10.0.10 Factory\_Machines

```

1 import sys
2 # A factory has n machines which can be used to make products. Your goal
3 # is to make a total of t products.
4 # For each machine, you know the number of seconds it needs to make a
5 # single product. The machines can work simultaneously, and you can
6 # freely decide their schedule.
7 # What is the shortest time needed to make t products?
8 input = sys.stdin.readline
9 def f(arr, k, mid):
10    suma = sum(mid // x for x in arr)
11    return suma>=k
12
13 n, k = map(int, input().split())
14 arr = list(map(int, input().split()))
15 l = 1
16 r = max(arr) * k
17 ans = 0
18 while l<=r:
19     mid = (l+r)//2
20     if f(arr, k, mid):
21         ans = mid
22         r = mid-1
23     else:
24         l = mid+1
25 print(ans)
```

### 10.0.11 Ferris\_Wheel

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 #define int long long
4 //There are n children who want to go to a Ferris wheel, and your task is
5 //to find a gondola for each child.
6 //Each gondola may have one or two children in it, and in addition, the
7 //total weight in a gondola may not exceed x. You know the weight of
8 //every child.
9 //What is the minimum number of gondolas needed for the children?
```

```

7 signed main(){
8     int n, x;cin>>n>>x;
9     vector<int> arr(n);
10    for(int &i:arr){cin>>i;}
11    sort(arr.begin(), arr.end());
12    int l = 0, r = n-1;
13    int cont = 0;
14    while(l<=r){
15        if(arr[l] + arr[r] > x){
16            r--;
17        }else{
18            r--;l++;
19        }
20        cont++;
21    }
22    cout<<cont<<endl;
23 }
```

### 10.0.12 Josephus\_Problem\_I

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define ll long long
4 #define ull unsigned long long
5 #define fore(i, a, b) for(int i = (a); i<(b); i++)
6 #define FOR(i, n) for(int i = 0; i<(n); i++)
7 #define all(x) (x).begin(), (x).end()
8 #define sz(x) (int)(x).size()
9 #define pb push_back
10 using vi = vector<int>;
11 const int MOD = 1000000007;
12 const ll INF = 9223372036854775807LL;
13 const int tam = 1;
14 //Consider a game where there are n children (numbered 1,2,\dots,n) in a
15 //circle. During the game, every other child is removed from the circle
16 //until there are no children left. In which order will the children be
17 //removed?
18 void solve(){
19     int n;cin>>n;
20     deque<int> dq;
21     fore(i, 1, n+1){
22         dq.push_back(i);
23     }
24     int ans;
25     while(sz(dq) >= 2){
26         int vive = dq.front();
27         dq.pop_front();
28         int muere = dq.front();
29         dq.pop_front();
30         cout<<muere<<' ';
31         dq.push_back(vive);
32     }
33     cout<<dq.front();
34     cout<<'\n';
35 }
```

```

33
34 signed main(){
35     ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
36     int t = 1;
37     // cin>>t;
38     while(t--){
39         solve();
40     }
41 }

40
41     n--;
42     os.erase(it);
43 }
44 cout<<'\n';
45
46
47 signed main(){
48     ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
49     int t = 1;
50     //cin>>t;
51     while(t--){
52         solve();
53     }
54 }

```

### 10.0.13 Josephus\_Problem\_II

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define ll long long
4 #define ull unsigned long long
5 #define fore(i, a, b) for(int i = (a); i<(b); i++)
6 #define FOR(i, n) for(int i = 0; i<(n); i++)
7 #define all(x) (x).begin(), (x).end()
8 #define sz(x) (int(x).size())
9 #define pb push_back
10 using vi = vector<int>;
11 const int MOD = 1000000007;
12 const ll INF = 9223372036854775807LL;
13 const int tam = 1;
14 #include <bits/stdc++.h>
15 using namespace std;
16 //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?
17
18 #include <ext/pb_ds/assoc_container.hpp>
19 #include <ext/pb_ds/tree_policy.hpp>
20 using namespace __gnu_pbds;
21
22 #define ordered_set tree<int, null_type, less<int>,
23 // rb_tree_tag,tree_order_statistics_node_update>
24 //greater or less
25
26 //cantidad de elementos en un rango:
27 //o_set.order_of_key(r+1) - o_set.order_of_key(l)
28 void solve(){
29     int n, k;cin>>n>>k;
30     ordered_set os;
31     for(int i = 1; i<=n; i++){
32         os.insert(i);
33     }
34     int idx = 0;
35     while(n){
36         idx+=k;
37         idx%=n;
38         auto it = os.find_by_order(idx);
39         cout<<*it<< ' ';

```

### 10.0.14 Maximum\_Subarray\_Sum

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define int long long
4 #define f first
5 #define sst stringstream
6 #define s second
7 #define pb push_back
8 #define sz(x) (int(x).size())
9 #define all(a) (a).begin(), (a).end()
10 #define rall(a) (a).rbegin(), (a).rend()
11 #define fore(i, a, n) for(int i = (a); i < (n); i++)
12 #define forb(i, n) for(int i = (n) - 1; i ≥ 0; i--)
13 #define popcount(x) __builtin_popcountll(x);
14 typedef pair<int, inttypedef vector<int> vi;
16 const int MOD = 1000000007;
17 const double EPS = 1e-9;
18 const double PI = acos(-1);
19 const int INF = 1000000000000000000;
// PLUS ULTRA RECARGADO!!!
20 void solve() {
21     int n;
22     cin >> n;
23     vector<int> vec(n);
24     for(int i = 0; i < n; i++) {
25         cin >> vec[i];
26     }
27     int suma = 0, ans = -INF;
28     for(int i = 0; i < n; i++) {
29         suma += vec[i];
30         ans = max(ans, suma);
31         if(suma < 0) suma = 0;
32     }
33     if(suma != 0) ans = max(ans, suma);
34     cout << ans << endl;
35 }
36 signed main() {

```

```

38 ios::sync_with_stdio(0);
39 cin.tie(0);
40 cout.tie(0);
41 // int t;cin>>t;while(t--)solve();
42 solve();
43 }

```

### 10.0.15 Maximum\_Subarray\_Sum\_II

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define int long long
4 #define ll long long
5 #define ull unsigned long long
6 #define fore(i, a, b) for(int i = (a); i<(b); i++)
7 #define FOR(i, n) for(int i = 0; i<(n); i++)
8 #define all(x) (x).begin(), (x).end()
9 #define sz(x) (int)(x).size()
10 #define pb push_back
11 using vi = vector<int>;
12 const int MOD = 1000000007;
13 const ll INF = 9223372036854775807LL;
14 const int tam = 1;
15 //Given an array of n integers, your task is to find the maximum sum of
16 // values in a contiguous subarray with length between a and b.
17 void solve(){
18     int n, a, b;
19     cin >> n >> a >> b;
20     vector<int> arr(n+1), pref(n+1, 0);
21     for(int i = 1; i <= n; i++){
22         cin >> arr[i];
23         pref[i] = pref[i-1] + arr[i];
24     }
25
26     multiset<int> ms;
27     int ans = -INF;
28     for(int i = a; i <= n; i++){
29         ms.insert(pref[i - a]);
30         if(i - b - 1 >= 0){
31             ms.erase(ms.find(pref[i - b - 1]));
32         }
33         ans = max(ans, pref[i] - *ms.begin());
34     }
35     cout << ans << "\n";
36 }
37
38 signed main(){
39     ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
40     int t = 1;
41     // cin>>t;
42     while(t--){
43         solve();
44     }
45 }

```

```

46    }

```

---

### 10.0.16 Missing\_Coin\_Sum

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define ll long long
4 #define ull unsigned long long
5 #define fore(i, a, b) for(int i = (a); i<(b); i++)
6 #define FOR(i, n) for(int i = 0; i<(n); i++)
7 #define all(x) (x).begin(), (x).end()
8 #define sz(x) (int)(x).size()
9 using vi = vector<int>;
10 const int MOD = 1000000007;
11 const ll INF = 9223372036854775807LL;
12 // You have n coins with positive integer values. What is the smallest sum
13 // → you cannot create using a subset of the coins?
14
15 void solve(){
16     int n;cin>>n;
17     int arr[n];
18     fore(i, 0, n)cin>>arr[i];
19     sort(arr, arr+n);
20     ll k = 0;
21     fore(i, 0, n){
22         if(k+1>=arr[i]){
23             k+=arr[i];
24         }else{
25             cout<<k+1<<endl;
26             return;
27         }
28     }
29     cout<<k+1<<endl;
30 }
31
32 signed main(){
33     ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
34     int t = 1;
35     //cin>>t;
36     while(t--){
37         solve();
38     }
39 }

```

---

### 10.0.17 Movie\_Festival

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define int long long
4 #define f first
5 #define sst stringstream
6 #define s second
7 #define pb push_back

```

```

8 #define sz(x) (int)(x).size()
9 #define all(a) (a).begin(), (a).end()
10 #define rall(a) (a).rbegin(), (a).rend()
11 #define fore(i, a, n) for(int i = (a); i < (n); i++)
12 #define forb(i, n) for (int i = (n) - 1; i ≥ 0; i--)
13 #define popcorn(x) __builtin_popcountll(x);
14 typedef long double ld;
15 typedef pair<int, int> pii;
16 typedef vector<int> vi;
17 const int MOD = 1000000007;
18 const double EPS = 1e-9;
19 const double PI = acos(-1);
20 const int INF = 1e18;
21 //PLUS ULTRA RECARGADO!!!
22 bool cmp(pii& a, pii& b){
23     if(a.s == b.s) return a.f < b.f;
24     return a.s < b.s;
25 }
26 // In a movie festival n movies will be shown. You know the starting and
27 // ending time of each movie. What is the maximum number of movies you
28 // can watch entirely?
29 void solve() {
30     int n;cin>>n;
31     vector<pii> v(n);
32     fore(i,0,n){
33         cin>>v[i].f>>v[i].s;
34     }
35     sort(all(v), cmp);
36     int ans = 0;
37     int last = -1;
38     for(auto[ini, fin]:v){
39         if(last <= ini){
40             last = fin;
41             ans++;
42         }
43     }
44     cout<<ans<<endl;
45 }
46 signed main() {
47     ios::sync_with_stdio(0);
48     cin.tie(0);
49     cout.tie(0);
50     int t = 1;
51 //    cin>>t;
52     while(t--)solve();
53 }

5  #define fore(i, a, b) for(int i = (a); i<(b); i++)
6  #define FOR(i, n) for(int i = 0; i<(n); i++)
7  #define all(x) (x).begin(), (x).end()
8  #define sz(x) (int)(x).size()
9  #define pb push_back
10 using vi = vector<int>;
11 const int MOD = 1000000007;
12 const ll INF = 9223372036854775807LL;
13 const int tam = 1;
14 bool cmp(pair<int, int>& a, pair<int, int>& b){
15     if(a.second == b.second) return a.first < b.first;
16     return a.second < b.second;
17 }
18 //In a movie festival, n movies will be shown. Syrjälä's movie club
19 //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
19 //total number of movies the club members can watch entirely if they act
19 //optimally?
20 void solve(){
21     int n, k;cin>>n>>k;
22     vector<pair<int, int>> ran(n);
23     fore(i, 0, n){
24         int l, r;cin>>l>>r;
25         ran[i] = {l, r};
26     }
27     sort(all(ran), cmp);
28     int ans = 1;
29     set<pair<int, int>> ultimos;
30     int i;
31     // for(i = 0; i<min(n, k); i++){
32     //     ultimos.insert(ran[i].second);
33     //     ans++;
34     // }
35     ultimos.insert({ran[0].second, 0});
36     int cont = 1;
37     for(i = 1; i<n; i++){
38         //si hay alguien menor a su inicio a ese lo ponemos a chambear
39         auto it = ultimos.lower_bound({ran[i].first+1, 0});
40         if(it == ultimos.begin()){
41             if(cont < k){
42                 ans++;
43                 ultimos.insert({ran[i].second, i});
44                 cont++;
45             }
46         }else{
47             it--;
48             ultimos.erase(it);
49             ultimos.insert({ran[i].second, i});
50             ans++;
51         }
52     }
53     cout<<ans<<endl;
54 }

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

```

### 10.0.18 Movie\_Festival\_II

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define ll long long
4 #define ull unsigned long long

```

```

60     while(t--){
61         solve();
62     }
63 }
```

### 10.0.19 Nearest\_Smaller\_Values

```

1 #include <bits/stdc++.h>
2 #include<iostream.h>
3 using namespace std;
4 #define ll long long
5 #define ull unsigned long long
6 #define fore(i, a, b) for(int i = (a); i<(b); i++)
7 #define FOR(i, n) for(int i = 0; i<(n); i++)
8 #define all(x) (x).begin(), (x).end()
9 #define sz(x) (int)(x).size()
10 #define pb push_back
11 using vi = vector<int>;
12 const int MOD = 1000000007;
13 const ll INF = 9223372036854775807LL;
14 const int tam = 1;
15 // Given an array of n integers, your task is to find for each array
16 // position the nearest position to its left having a smaller value.
17 void solve(){
18     int n;cin>>n;
19     stack<pair<int, int>> pila;
20     for(int i = 0; i<n; i++){
21         int x;cin>>x;
22         int ans = -1;
23         while(!pila.empty() and pila.top().first >= x){
24             pila.pop();
25         }
26         if(!pila.empty()){
27             cout<<pila.top().second<<' ';
28         }else{
29             cout<<0<<' ';
30         }
31         pila.push({x, i+1});
32     }
33     cout<<'\n';
34 }
35 signed main(){
36     ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
37     int t = 1;
38     //cin>>t;
39     while(t--){
40         solve();
41     }
42 }
```

### 10.0.20 Nested\_Ranges\_Check

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define ll long long
4 #define ull unsigned long long
5 #define fore(i, a, b) for(int i = (a); i<(b); i++)
6 #define FOR(i, n) for(int i = 0; i<(n); i++)
7 #define all(x) (x).begin(), (x).end()
8 #define sz(x) (int)(x).size()
9 #define pb push_back
10 using vi = vector<int>;
11 const int MOD = 1000000007;
12 const ll INF = 9223372036854775807LL;
13 const int tam = 1;
14 //Given n ranges, your task is to determine for each range if it contains
15 // some other range and if some other range contains it.
16 //Range [a,b] contains range [c,d] if a \le c and d \le b.
17 bool cmp(tuple<int, int, int>& a, tuple<int, int, int>& b){
18     if(get<0>(a) == get<0>(b))return get<1>(a) > get<1>(b);
19     return get<0>(a) < get<0>(b);
20 }
21 void solve(){
22     int n;cin>>n;
23     vector<tuple<int, int, int>> arr(n);
24     fore(i, 0, n){
25         int x, y;cin>>x>>y;
26         arr[i] = {x, y, i};
27     }
28     sort(all(arr), cmp);
29     vector<int> minder(n);
30     minder[n-1] = get<1>(arr.back());
31     for(int i = n-2; i>=0; i--){
32         minder[i] = min(minder[i+1], get<1>(arr[i]));
33     }
34     vector<int> maxizq(n);
35     maxizq[0] = get<1>(arr[0]);
36     for(int i = 1; i<n; i++){
37         maxizq[i] = max(maxizq[i-1], get<1>(arr[i]));
38     }
39     vector<bool> contiene(n);
40     for(int i = 0;i<n-1; i++){
41         if(minder[i+1]<=get<1>(arr[i])){
42             contiene[get<2>(arr[i])] = 1;
43         }
44     }
45     vector<bool> contenido(n);
46     for(int i = 1; i<n; i++){
47         if(maxizq[i-1] >= get<1>(arr[i])){
48             contenido[get<2>(arr[i])] = 1;
49         }
50     }
51     for(bool i:contiene)cout<<i<<' ';
52     cout<<'\n';
53     for(bool i:contenido)cout<<i<<' ';
54     cout<<'\n';
```

```

55
56 }
57
58 signed main(){
59     ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
60     int t = 1;
61 //    cin>>t;
62     while(t--){
63         solve();
64     }
65 }
```

### 10.0.21 Nested\_Ranges\_Count

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define ll long long
4 #define ull unsigned long long
5 #define fore(i, a, b) for (int i = (a); i < (b); i++)
6 #define FOR(i, n) for (int i = 0; i < (n); i++)
7 #define all(x) (x).begin(), (x).end()
8 #define sz(x) (int)(x).size()
9 #define pb push_back
10 using vi = vector<int>;
11 const int MOD = 1000000007;
12 const ll INF = 9223372036854775807LL;
13 const int tam = 1;
14 //Given n ranges, your task is to count for each range how many other
15 //→ ranges it contains and how many other ranges contain it.
16 //Range [a,b] contains range [c,d] if a \le c and d \le b.
17 // primero bits/stdc, std y luego las demas librerias
18 using namespace std;
19 #include <ext/pb_ds/assoc_container.hpp>
20 #include <ext/pb_ds/tree_policy.hpp>
21 using namespace __gnu_pbds;
22 typedef pair<int, int> pii;
23
24 typedef tree<pii, null_type, less<pii>, rb_tree_tag,
25             tree_order_statistics_node_update>
26 ordered_ms;
27
28 struct ordMultiset {
29     int count;
30     ordered_ms oset;
31
32     ordMultiset() { count = 0; }
33     // o que posicion le corresponde
34     int count_oc(int x) {
35         return oset.order_of_key({x + 1, 0}) - oset.order_of_key({x, 0});
36     }
37     void erase_one(int x) {
38         auto it = oset.lower_bound({x, 0});
39         if (it != oset.end() && it->first == x) { oset.erase(it); }
40     }
41
42     // O indexed
43     int kth_element(int k) { return oset.find_by_order(k)->first; }
44
45     int count_greater_equal(int x) {
46         return (int)oset.size() - oset.order_of_key({x, 0});
47     }
48
49     void insert(int x) { oset.insert({x, count++}); }
50 };
51
52 struct tupla{
53     int l, r, idx;
54     tupla(){}
55     tupla(int _l, int _r, int _idx){
56         l = _l;
57         r = _r;
58         idx = _idx;
59     }
60 };
61 bool cmp(tupla& a, tupla& b){
62     if(a.l == b.l) return a.r > b.r;
63     return a.l < b.l;
64 }
65 void solve() {
66     int n;cin>>n;
67     vector<tupla> rangos(n);
68     for(int i = 0; i < n; i++){
69         int a, b;cin>>a>>b;
70         rangos[i] = {a, b, i};
71     }
72     sort(all(rangos), cmp);
73     ordMultiset os;
74     os.insert(rangos[0].r);
75     vector<int> contenido(n);
76     for(int i = 1; i < n; i++){
77         contenido[rangos[i].idx] = os.count_greater_equal(rangos[i].r);
78         os.insert(rangos[i].r);
79     }
80     vector<int> contiene(n);
81     os.oset.clear();
82     os.count = 0;
83     os.insert(rangos.back().r);
84     for(int i = n-2; i >= 0; i--){
85         contiene[rangos[i].idx] = os.oset.order_of_key({rangos[i].r+1,
86             0});
87         os.insert(rangos[i].r);
88     }
89     for(int x:contiene)cout<<x<<' ';
90     cout<<'\n';
91     for(int x:contenido)cout<<x<<' ';
92     cout<<'\n';
93 }
94
95 signed main() {
96     ios::sync_with_stdio(0);
97     cin.tie(0);
98     cout.tie(0);
99     int t = 1;
100 //cin >> t;
```

```

98     while (t--) { solve(); }
99 }
```

## 10.0.22 Playlist

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define ll long long
4 #define ull unsigned long long
5 #define fore(i, a, b) for (int i = (a); i < (b); i++)
6 #define FOR(i, n) for (int i = 0; i < (n); i++)
7 #define all(x) (x).begin(), (x).end()
8 #define sz(x) (int)(x).size()
9 using vi = vector<int>;
10 const int MOD = 1000000007;
11 const ll INF = 9223372036854775807LL;
12
13 // You are given a playlist of a radio station since its establishment.
14 // What is the longest sequence of successive songs where each song is
15 // unique?
16 void solve() {
17     int n;
18     cin >> n;
19     int arr[n];
20     fore(i, 0, n) { cin >> arr[i]; }
21     map<int, int> mp;
22     int l = 0, r = 0;
23     int ans = 0;
24     while (r < n) {
25         mp[arr[r]]++;
26         if(mp[arr[r]]>1){
27             while(mp[arr[r]]>1){
28                 mp[arr[l]]--;
29                 l++;
30             }
31         }
32         ans = max(ans, r-l+1);
33         r++;
34     }
35 }
36 cout << ans << endl;
37
38 signed main() {
39     ios::sync_with_stdio(0);
40     cin.tie(0);
41     cout.tie(0);
42     int t = 1;
43     // cin>>t;
44     while (t--) {
45         solve();
46     }
47 }
```

```

48     }
```

---

## 10.0.23 Reading\_Books

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define int long long
4 #define ll long long
5 #define ull unsigned long long
6 #define fore(i, a, b) for(int i = (a); i<(b); i++)
7 #define FOR(i, n) for(int i = 0; i<(n); i++)
8 #define all(x) (x).begin(), (x).end()
9 #define sz(x) (int)(x).size()
10 #define pb push_back
11 using vi = vector<int>;
12 const int MOD = 1000000007;
13 const ll INF = 9223372036854775807LL;
14 const int tam = 1;
15 // There are n books, and Kotivalo and Justiina are going to read them
16 // all. For each book, you know the time it takes to read it.
17 // They both read each book from beginning to end, and they cannot read a
18 // book at the same time. What is the minimum total time required?
19
20 void solve(){
21     int n;cin>>n;
22     vector<int> arr(n);
23     fore(i, 0, n)cin>>arr[i];
24     cout<<max(2LL*(*max_element(all(arr))), accumulate(all(arr), 0LL));
25 }
26
27 signed main(){
28     ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
29     int t = 1;
30     // cin>>t;
31     while(t--){
32         solve();
33     }
34 }
```

---

## 10.0.24 Restaurant\_Customers

```

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

```

13 #define popcorn(x) __builtin_popcountll(x);
14 typedef pair<int, int> pii;
15 typedef vector<int> vi;
16 const int MOD = 1000000007;
17 const double EPS = 1e-9;
18 const double PI = acos(-1);
19 const int INF = 1e18;
20 //PLUS ULTRA RECARGADO!!!
21 void solve() {
22     int n;cin>>n;
23     vector<pair<int, int>> vec;
24     for(int i = 0; i<n; i++){
25         int a, b;cin>>a>>b;
26         vec.push_back({a, +1});
27         vec.push_back({b, -1});
28     }
29     int ans = 0;
30     int c = 0;
31     sort(all(vec));
32     for(int i = 0; i<sz(vec); i++){
33         c+=vec[i].s;
34         ans = max(ans, c);
35     }
36     cout<<ans<<endl;
37 }
38 signed main() {
39     ios::sync_with_stdio(0);
40     cin.tie(0);
41     cout.tie(0);
42     // int t;cin>>t;while(t--)solve();
43     solve();
44 }

```

---

```

20     return make_pair(a.l, a.r) < make_pair(b.l, b.r);
21 }
22 //There is a large hotel, and n customers will arrive soon. Each customer
23 // wants to have a single room.
24 // You know each customer's arrival and departure day. Two customers can
25 // stay in the same room if the departure day of the first customer is
26 // earlier than the arrival day of the second customer.
27 // What is the minimum number of rooms that are needed to accommodate all
28 // customers? And how can the rooms be allocated?
29 void solve(){
30     int n;cin>>n;
31     vector<tuple> ran(n);
32     for(int i = 0; i<n;i++){
33         int l, r;cin>>l>>r;
34         ran[i] = {l, r, i};
35     }
36     sort(all(ran), cmp);
37     int k = 0;
38     vector<int> ans(n);
39     set<pair<int, int>> uso;
40     for(int i = 0; i<n; i++){
41         auto it = uso.upper_bound({ran[i].l, 0});
42         if(it != uso.begin() and sz(uso) > 0){
43             it = uso.begin();
44             int idx = it->second;
45             uso.erase(it);
46             uso.insert({ran[i].r, idx});
47             ans[ran[i].index] = idx;
48         } else{
49             k++;
50             ans[ran[i].index] = k;
51             uso.insert({ran[i].r, k});
52         }
53     }
54     cout<<k<<endl;
55     for(int x:ans)cout<<x<< ' ';
56     cout<<'\n';
57 }
58
59 signed main(){
60     ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
61     int t = 1;
62     // cin>>t;
63     while(t--){
64         solve();
65     }
66 }
```

### 10.0.25 Room\_Allocation

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define ll long long
4 #define ull unsigned long long
5 #define fore(i, a, b) for(int i = (a); i<(b); i++)
6 #define FOR(i, n) for(int i = 0; i<(n); i++)
7 #define all(x) (x).begin(), (x).end()
8 #define sz(x) (int)(x).size()
9 #define pb push_back
10 using vi = vector<int>;
11 const int MOD = 1000000007;
12 const int INF = 922337203;
13 const int tam = 1;
14 struct tupla
15 {
16     int l, r, index;
17 };
18 };
19 bool cmp(tupla& a, tupla&b){
```

### 10.0.26 Stick\_Lengths

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define int long long
4 #define f first
5 #define sst stringstream
```

```

6 #define s second
7 #define pb push_back
8 #define sz(x) (int)(x).size()
9 #define all(a) (a).begin(), (a).end()
10 #define rall(a) (a).rbegin(), (a).rend()
11 #define fore(i, a, n) for(int i = (a); i < (n); i++)
12 #define forb(i, n) for(int i = (n) - 1; i ≥ 0; i--)
13 #define popcorn(x) __builtin_popcountll(x);
14 typedef pair<int, int> pii;
15 typedef vector<int> vi;
16 const int MOD = 1000000007;
17 const double EPS = 1e-9;
18 const double PI = acos(-1);
19 const int INF = 1e18;
20 // PLUS ULTRA RECARGADO!!!
21 // There are n sticks with some lengths. Your task is to modify the sticks
22 // so that each stick has the same length.
23 // You can either lengthen and shorten each stick. Both operations cost x
24 // where x is the difference between the new and original length.
25 // What is the minimum total cost?
26 void solve() {
27     int n;
28     cin >> n;
29     vector<int> a(n);
30     for(int i = 0; i < n; i++) {
31         cin >> a[i];
32     }
33     sort(all(a));
34     int mediana;
35     if(!(n & 1)) {
36         mediana = a[(n - 1) / 2] + a[n / 2];
37         mediana = (mediana + 1)/2;
38     } else{
39         mediana = a[n/2];
40     }
41     int ans = 0;
42     for(int i:a){
43         ans+=abs(i-mediana);
44     }
45     cout<<ans<<endl;
46 }
47 signed main() {
    solve();
}

```

---

```

9 #define sz(x) (int)(x).size()
10 #define pb push_back
11 using vi = vector<int>;
12 const int MOD = 1000000007;
13 const ll INF = 9223372036854775807LL;
14 const int tam = 1;
15 // Given an array of n integers, your task is to count the number of
16 // subarrays where the sum of values is divisible by n.
17 void solve() {
18     int n, k;
19     cin >> n;
20     int suma = 0;
21     map<int, int> prefijos;
22     prefijos[0] = 1;
23     int ans = 0;
24     fore(i, 0, n) {
25         int x;cin>>x;
26         suma+=x;
27         while(suma < 0){
28             suma+=(n*10LL);
29         }
30         suma%=n;
31         if(prefijos.find(suma) != prefijos.end()){
32             ans+=prefijos[suma];
33         }
34         prefijos[suma]++;
35     }
36     cout<<ans<<endl;
37 }
38
39 signed main(){
40     ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
41     int t = 1;
42     // cin>>t;
43     while(t--){
44         solve();
45     }
46 }

```

---

### 10.0.27 Subarray\_Divisibility

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define int long long
4 #define ll long long
5 #define ull unsigned long long
6 #define fore(i, a, b) for(int i = (a); i<(b); i++)
7 #define FOR(i, n) for(int i = 0; i<(n); i++)
8 #define all(x) (x).begin(), (x).end()
9 #define sz(x) (int)(x).size()
10 #define rec(arr, n) vector<int> arr(n);fore(i, 0, n)cin>>arr[i];
11 #define pb push_back
12 using vi = vector<int>;
13 const int MOD = 1000000007;
14 const ll INF = 9223372036854775807LL;

```

---

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define ll long long
4 #define ull unsigned long long
5 #define fore(i, a, b) for(int i = (a); i<(b); i++)
6 #define FOR(i, n) for(int i = 0; i<(n); i++)
7 #define all(x) (x).begin(), (x).end()
8 #define sz(x) (int)(x).size()
9 #define rec(arr, n) vector<int> arr(n);fore(i, 0, n)cin>>arr[i];
10 #define pb push_back
11 using vi = vector<int>;
12 const int MOD = 1000000007;
13 const ll INF = 9223372036854775807LL;

```

```

14 const int tam = 1;
15
16 // Given an array of n positive integers, your task is to count the
17 // number of subarrays having sum x.
18 void solve(){
19     int n, k; cin >> n >> k;
20     rec(arr, n);
21     int l = 0;
22     int ans = 0, sum = 0;
23     fore(r, 0, n){
24         sum += arr[r];
25         if(sum == k) ans++;
26         while(sum > k and l < r){
27             sum -= arr[l];
28             l++;
29             if(sum == k) ans++;
30         }
31     }
32     cout << ans << endl;
33 }
34
35 signed main(){
36     ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
37     int t = 1;
38     //cin >> t;
39     while(t--){
40         solve();
41     }
42 }
```

### 10.0.29 Subarray\_Sums\_II

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define int long long
4 #define ll long long
5 #define ull unsigned long long
6 #define fore(i, a, b) for (int i = (a); i < (b); i++)
7 #define FOR(i, n) for (int i = 0; i < (n); i++)
8 #define all(x) (x).begin(), (x).end()
9 #define sz(x) (int)(x).size()
10 #define pb push_back
11 using vi = vector<int>;
12 const int MOD = 1000000007;
13 const ll INF = 9223372036854775807LL;
14 const int tam = 1;
15 //Given an array of n integers, your task is to count the number of
16 // subarrays having sum x.
17 void solve() {
18     int n, k;
19     cin >> n >> k;
20     int suma = 0;
21     map<int, int> prefijos;
22     prefijos[0] = 1;
```

```

22     int ans = 0;
23     fore(i, 0, n) {
24         int x; cin >> x;
25         suma += x;
26         int falta = suma - k;
27         if(prefijos[falta] > 0){
28             ans += prefijos[falta];
29         }
30         prefijos[suma]++;
31     }
32     cout << ans << endl;
33 }
34
35 signed main() {
36     ios::sync_with_stdio(0);
37     cin.tie(0);
38     cout.tie(0);
39     int t = 1;
40     // cin >> t;
41     while (t--) { solve(); }
42 }
```

### 10.0.30 Sum\_of\_Four\_Values

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define int long long
4 #define ll long long
5 #define ull unsigned long long
6 #define fore(i, a, b) for(int i = (a); i < (b); i++)
7 #define FOR(i, n) for(int i = 0; i < (n); i++)
8 #define all(x) (x).begin(), (x).end()
9 #define sz(x) (int)(x).size()
10 #define pb push_back
11 using vi = vector<int>;
12 const int MOD = 1000000007;
13 const ll INF = 9223372036854775807LL;
14 const int tam = 1;
15 // You are given an array of n integers, and your task is to find four
16 // values (at distinct positions) whose sum is x.
17 void solve(){
18     int n, k; cin >> n >> k;
19     vector<int> arr(n);
20     fore(i, 0, n) cin >> arr[i];
21     map<int, pair<int, int>> mp;
22     if(n < 4){
23         cout << "IMPOSSIBLE\n";
24         return;
25     }
26     mp[arr[0] + arr[1]] = {0, 1};
27     for(int l = 2; l < n; l++){
28         for(int r = l+1; r < n; r++){
29             int valor = k - arr[l] - arr[r];
30             if(mp.find(valor) != mp.end()){
31                 //ya encontre
```

```

31         cout<<l+1<<' '<<r+1<<' '<<mp[valor].first+1<<' 
32             '<<mp[valor].second+1<<'\n';
33     }
34 }
35 for(int i = 0; i<l; i++){
36     mp[arr[i] + arr[l]] = {i, l};
37 }
38 cout<<"IMPOSSIBLE\n";
39
40 }
41
42 signed main(){
43     ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
44     int t = 1;
45     //cin>>t;
46     while(t--){
47         solve();
48     }
49 }

```

---

### 10.0.31 Sum\_of\_Three\_Values

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 // #include <ext/pb_ds/assoc_container.hpp>
4 // #include <ext/pb_ds/assoc_container.hpp>
5 // #include <ext/pb_ds/tree_policy.hpp>
6 // #include <ext/rope>
7 #define int long long
8 #define uset unordered_set
9 #define f first
10 #define s second
11 #define umap unordered_map
12 #define mp make_pair
13 #define pb push_back
14 #define sz(x) (int)(x).size()
15 #define all(a) (a).begin(), (a).end()
16 #define rall(a) (a).rbegin(), (a).rend()
17 #define floatigual(a, b) (fabs(a - b) < EPS)
18 #define mod(a) md(a, MOD)
19 #define FOR(i, n) for (int i = 0; i < (n); ++i)
20 #define FOR3(i, a, b) for (int i = (a); i < (b); ++i)
21 #define FORD(i, n) for (int i = (n) - 1; i ≥ 0; --i)
22 #define FORDD(i, a, b) for (int i = (b) - 1; i ≥ (a); --i)
23 #define techo(a, b) (a / b + (a % b ≠ 0))
24 #define popcount(x) __builtin_popcountll(x);
25 using namespace std;
26 // You are given an array of n integers, and your task is to find three
27 // values (at distinct positions) whose sum is x.
28 bool f(int a, int b, int c){
29     return a!=b and b!=c and a!=c;
30 }
31 void solve() {
32
33     int n, k;cin>>n>>k;
34     vector<pii> a(n);
35     FOR(i, n){
36         int x;cin>>x;
37         a[i] = {x, i};
38     }
39     sort(all(a));
40     FOR(i, n){
41         FOR3(j, i+1, n){
42             int buscado = k- a[i].first - a[j].first;
43             int l = 0, r = n-1;
44             while(l<=r){
45                 int mid = (l+r)/2;
46                 if(a[mid].f == buscado and f(a[mid].s, a[i].s, a[j].s)){
47                     cout<<a[mid].s+1<<' '<<a[i].s+1<<' '<<a[j].s+1<<'\n';
48                     return;
49                 }else if(buscado<a[mid].f){
50                     r = mid-1;
51                 }else{
52                     l = mid+1;
53                 }
54             }
55             cout<<"IMPOSSIBLE\n";
56         }
57     }
58     signed main() {
59         ios::sync_with_stdio(0);
60         cin.tie(0);
61         cout.tie(0);
62         // int t;cin>>t;while(t--)solve();
63         solve();
64     }

```

---

### 10.0.32 Sum\_of\_Two\_Values

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define ll long long
4 #define ull unsigned long long
5 #define fore(i, a, b) for(int i = (a); i<(b); i++)
6 #define FOR(i, n) for(int i = 0; i<(n); i++)
7 #define all(x) (x).begin(), (x).end()
8 #define sz(x) (int)(x).size()
9 #define pb push_back
10 using vi = vector<int>;
11 const int MOD = 1000000007;
12 const ll INF = 9223372036854775807LL;
13 const int tam = 1;
14 using pii = pair<int, int>;
15 void solve() {
16     int n, k;cin>>n>>k;
17     vector<pii> a(n);
18     FOR(i, n){

```

```

19
20     int x;cin>>x;
21     a[i] = {x, i};
22 }
23 sort(all(a));
24 FOR(i, n){
25     int buscado = k - a[i].first;
26     int l = 0; int r = n-1;
27     int res = -1;
28     while(l<=r){
29         int mid = (l+r)/2;
30         if(a[mid].first == buscado and a[i].second != a[mid].second){
31             cout<<a[mid].second+1<<' '<<a[i].second+1<<'\n';
32             return;
33         }else if(a[mid].first<buscado){
34             l = mid+1;
35         }else{
36             r = mid-1;
37         }
38     }
39 }
40 cout<<"IMPOSSIBLE\n";
41 }
42 signed main() {
43     ios::sync_with_stdio(0);
44     cin.tie(0);
45     cout.tie(0);
46 //     int t;cin>>t;while(t--)solve();
47     solve();
48 }
49

18
19     int b = actual.second;
20     tiempo += a;
21     res += (b - tiempo);
22 }
23 cout << res << endl;
24 }
25
26 signed main()
27 {
28     ios::sync_with_stdio(0);
29     cin.tie(0);
30     cout.tie(0);
31     //int t;
32     //cin >> t;
33     //while (t--)
34     //    solve();
35     solve();
36 }
37



---



### 10.0.34 Towers



---


1 #include <bits/stdc++.h>
2 using namespace std;
3 #define ll long long
4 const int tam = 1;
5 // You are given n cubes in a

```

### 10.0.33 Tasks\_and\_Deadlines

```
1 // You have to process n tasks. Each task has a duration and a deadline,
2 // and you will process the tasks in some order one after another. Your
3 // reward for a task is d-f where d is its deadline and f is your
4 // finishing time. (The starting time is 0, and you have to process all
5 // tasks even if a task would yield negative reward.)
6 // What is your maximum reward if you act optimally?
7 void solve() {
8     int n;
9     cin >> n;
10    vector<pii> v(n);
11    FOR(i, n){
12        cin >> v[i].first >> v[i].second;
13    }
14
15    sort(all(v));
16
17    int tiempo = 0;
18    int res = 0;
19
20    for (pii& actual : v) {
21        int a = actual.first;
22        int b = actual.second;
23
24        if (a <= tiempo) {
25            res += b;
26            tiempo = b;
27        } else {
28            if (a - tiempo > b) {
29                res += b;
30                tiempo = a;
31            } else {
32                res += a - tiempo;
33                tiempo = a;
34            }
35        }
36    }
37
38    cout << res;
39}
```

### 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;
```

```

29     while(t--){
30         solve();
31     }
32 }
```

### 10.0.35 Traffic\_Lights

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define ll long long
4 #define ull unsigned long long
5 #define fore(i, a, b) for(int i = (a); i < (b); i++)
6 #define FOR(i, n) fore(i, 0, n)
7 #define all(x) (x).begin(), (x).end()
8 #define sz(x) (int)(x).size()
9 using vi = vector<int>;
10 const int MOD = 1000000007;
11 const ll INF = 9223372036854775807LL;
12 // There is a street of length x whose positions are numbered
13 // 0,1,\ldots,x. Initially there are no traffic lights, but n sets of
14 // traffic lights are added to the street one after another.
15 // Your task is to calculate the length of the longest passage without
16 // traffic lights after each addition.
17
18 void solve() {
19     ll x, n;
20     cin >> x >> n;
21     set<ll> positions = {0, x};
22     multiset<ll> intervals = {x};
23     fore(i, 0, n){
24         ll p;
25         cin >> p;
26         auto it = positions.insert(p).first;
27         auto left = *prev(it);
28         auto right = *next(it);
29         intervals.erase(intervals.find(right - left));
30         intervals.insert(p - left);
31         intervals.insert(right - p);
32         cout << *intervals.rbegin() << " ";
33     }
34
35     int main() {
36         ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
37         int t = 1;
38         // cin >> t;
39         while(t--) {
40             solve();
41         }
42     }
43 }
```

## 11 String Algorithms

### 11.0.1 Finding\_Borders

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define ll long long
4 #define ull unsigned long long
5 #define fore(i, a, b) for(int i = (a); i < (b); i++)
6 #define FOR(i, n) for(int i = 0; i < (n); i++)
7 #define all(x) (x).begin(), (x).end()
8 #define sz(x) (int)(x).size()
9 #define pb push_back
10 using vi = vector<int>;
11 const int MOD = 1000000007;
12 const ll INF = 9223372036854775807LL;
13 const int tam = 1;
14 vector<int> kmp(string& s) {
15     int n = sz(s);
16     vector<int> pi(n);
17     for (int i = 1; i < n; i++) {
18         int j = pi[i - 1];
19         while (j > 0 and s[i] != s[j]) j = pi[j - 1];
20         if (s[i] == s[j]) j++;
21         pi[i] = j;
22     }
23     return pi;
24 }
25 void solve(){
26     string s;cin>>s;
27     auto pi = kmp(s);
28     int idx = sz(s)-1;
29     vi ans;
30     while(idx > 0){
31         ans.push_back(pi[idx]);
32         idx = pi[idx]-1;
33     }
34     sort(all(ans));
35     for(int x:ans)if(x)cout<<x<<' ';
36     cout<<'\n';
37 }
38
39 signed main(){
40     ios::sync_with_stdio(0);cin.tie(0);cout.tie(0);
41     int t = 1;
42     //cin >> t;
43     while(t--){
44         solve();
45     }
46 }
```

## 11.0.2 Longest\_Palindrome

```

1 // Given a string, your task is to determine the longest palindromic
2 // substring of the string. For example, the longest palindrome in
3 // aybabtu is bab.
4 #include <bits/stdc++.h>
5 using namespace std;
6 #define int long long
7 #define f first
8 #define s second
9 #define pb push_back
10 #define sz(x) (int)(x).size()
11 #define all(a) (a).begin(), (a).end()
12 #define rall(a) (a).rbegin(), (a).rend()
13 #define fore(i, a, n) for(int i = (a); i < (n); i++)
14 #define forb(i, n) for (int i = (n) - 1; i ≥ 0; i--)
15 #define popcount(x) __builtin_popcountll(x);
16 typedef pair<int, int> pii;
17 typedef vector<int> vi;
18 const int MOD = 1000000007;
19 const double EPS = 1e-9;
20 const double PI = acos(-1);
21 const int INF = 1e18;
22 //PLUS ULTRA RECARGADO!!!
23
24 vector<int> manacher_odd(string s) {
25     int n = s.size();
26     s = "$" + s + "^";
27     vector<int> p(n + 2);
28     int l = 1, r = 1;
29     for(int i = 1; i <= n; i++) {
30         p[i] = max(0LL, min(r - i, p[l + (r - i)]));
31         while(s[i - p[i]] == s[i + p[i]]) {
32             p[i]++;
33         }
34         if(i + p[i] > r) {
35             l = i - p[i], r = i + p[i];
36         }
37     }
38     return vector<int>(begin(p) + 1, end(p) - 1);
39 }
40
41 vector<int> manacher(string s) {
42     string t = "";
43     for(auto c: s) {
44         t.push_back('#');
45         t.push_back(c);
46     }
47     auto res = manacher_odd(t + "#");
48     return vector<int>(begin(res) + 1, end(res) - 1);
49 }
50
51 void solve() {
52     string s;cin>>s;
53     vi mani = manacher(s);

```

```

54
55
56     int posi = -1;
57     int maxi = 0;
58     for(int i = 0; i<sz(mani); i++){
59         if(mani[i] > maxi){
60             maxi = mani[i];
61             posi = i;
62         }
63     }
64     posi = (posi - maxi)/2;
65     cout<<s.substr(posi+1, maxi-1);
66 }
67 signed main() {
68     ios::sync_with_stdio(0);
69     cin.tie(0);
70     cout.tie(0);
71     //int t;cin>>t;while(t--)solve();
72     solve();
73 }

```

## 11.0.3 Palindrome\_Qualities\_BIT

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 #define ll long long
4 #define bint __int128
5
6 #define int long long
7
8 bint MOD = 212345678987654321LL;
9 bint P = 1777771;
10 bint PI = 106955741089659571LL;
11
12 struct Fenwick {
13     int n;
14     vector<bint> bit;
15     Fenwick(int n=0){ init(n); }
16     void init(int _n){
17         n = _n;
18         bit.assign(n+1, 0);
19     }
20     void add(int idx, bint val){
21         while(idx <= n){
22             bit[idx] += val;
23             bit[idx] %= MOD;
24             idx += idx & -idx;
25         }
26     }
27     bint sum(int idx){
28         bint res = 0;
29         while(idx > 0){
30             res += bit[idx];
31             res %= MOD;
32             idx -= idx & -idx;
33         }
34         return res;
35     }
36 }

```

```

35 }
36 bint rangeSum(int l, int r){
37     if(r < l) return 0;
38     bint res = sum(r) - sum(l-1);
39     res = (res % MOD + MOD) % MOD;
40     return res;
41 };
42
43 intToIntChar(char c){ return (int)(c); }
44
45 void solve(){
46     ios::sync_with_stdio(false);
47     cin.tie(nullptr);
48
49     int n, q;
50     if(!(cin >> n >> q)) return;
51     string s; cin >> s;
52
53     // Precompute powers and inverse powers (indexado a 0..n)
54     vector<bint> pows(n+1), pi(n+1);
55     pows[0] = 1; pi[0] = 1;
56     for(int i = 1; i <= n; ++i){
57         pows[i] = (pows[i-1] * P) % MOD;
58         pi[i] = (pi[i-1] * PI) % MOD;
59     }
60
61     // Arrays actuales (indexado 1..n)
62     vector<int> cur(n+1), curRev(n+1);
63     for(int i = 1; i <= n; ++i) cur[i] = toIntChar(s[i-1]);
64     for(int i = 1; i <= n; ++i) curRev[i] = toIntChar(s[n - i]); // ← invertida
65
66     // Fenwicks
67     Fenwick bit(n), bitRev(n);
68     bit.init(n); bitRev.init(n);
69
70     // Inicializar: en BIT guardamos a[i] * P^{i-1}
71     for(int i = 1; i <= n; ++i){
72         bint val = (bint)cur[i] * pows[i-1] % MOD;
73         bit.add(i, val);
74     }
75     for(int i = 1; i <= n; ++i){
76         bint val = (bint)curRev[i] * pows[i-1] % MOD;
77         bitRev.add(i, val);
78     }
79
80     auto getHash = [&](Fenwick &B, int l, int r)->long long {
81         // raw = sum_{i=l..r} a[i]*P^{i-1}
82         bint raw = B.rangeSum(l, r);
83         // normalizar: multiplicar por PI^{l-1} para que exponente empiece
84         // en 0
85         raw = (raw * pi[l-1]) % MOD;
86         raw = (raw + MOD) % MOD;
87         return (long long) raw;
88     };
89
90     while(q--){
91
92         int t; cin >> t;
93         if(t == 1){
94             int pos; char x; cin >> pos >> x;
95             int newv = toIntChar(x);
96             // normal
97             if(newv != cur[pos]){
98                 bint diff = (bint)newv - (bint)cur[pos] % MOD;
99                 diff = (diff + MOD) % MOD;
100                bint delta = (diff * pows[pos-1]) % MOD;
101                bit.add(pos, delta);
102                cur[pos] = newv;
103            }
104            // invertido en posicion n-pos+1
105            int posr = n - pos + 1;
106            if(newv != curRev[posr]){
107                bint diff2 = (bint)newv - (bint)curRev[posr] % MOD;
108                diff2 = (diff2 + MOD) % MOD;
109                bint delta2 = (diff2 * pows[posr-1]) % MOD;
110                bitRev.add(posr, delta2);
111                curRev[posr] = newv;
112            }
113            } else {
114                int l, r; cin >> l >> r;
115                long long h1 = getHash(bit, l, r);
116                // para la invertida: correspondencia [n-r+1, n-l+1]
117                int rl = n - r + 1;
118                int rr = n - l + 1;
119                long long h2 = getHash(bitRev, rl, rr);
120                cout << (h1 == h2 ? "YES" : "NO") << '\n';
121            }
122        }
123
124    signed main(){
125        solve();
126        return 0;
127    }

```

---

#### 11.0.4 Palindrome\_Queries

---

```

1 // You are given a string that consists of n characters between a-z. The
2 // positions of the string are indexed 1,2,\dots,n.
3 // Your task is to process m operations of the following types:
4
5 // Change the character at position k to x
6 // Check if the substring from position a to position b is a palindrome
7 #include <bits/stdc++.h>
8 using namespace std;
9 using ull = unsigned long long;
10 const ull MOD1 = 1000000007ULL;
11 const ull MOD2 = 1000000009ULL;
12 const ull P1 = 1777771ULL;
13 const ull P2 = 19260817ULL;
14 struct SegmentTree {
15     struct Node {

```

```

15     ull h1, h2;
16     int len;
17     Node(ull _h1 = 0, ull _h2 = 0, int _len = 0)
18       : h1(_h1), h2(_h2), len(_len) {}
19   };
20
21   int n;
22   vector<Node> st;
23   vector<ull> p1, p2;
24
25   SegmentTree(const vector<int>& a) {
26     n = (int)a.size();
27     st.assign(4 * n, Node());
28     p1.assign(n + 1, 1);
29     p2.assign(n + 1, 1);
30     precomputePowers();
31     build(1, 0, n - 1, a);
32   }
33
34   void precomputePowers() {
35     p1[0] = p2[0] = 1;
36     for (int i = 1; i <= n; ++i) {
37       p1[i] = (p1[i - 1] * P1) % MOD1;
38       p2[i] = (p2[i - 1] * P2) % MOD2;
39     }
40   }
41
42   Node merge(const Node& L, const Node& R) {
43     ull nh1 = ((L.h1 + (R.h1 * p1[L.len]) % MOD1)) % MOD1;
44     ull nh2 = ((L.h2 + (R.h2 * p2[L.len]) % MOD2)) % MOD2;
45     return Node(nh1, nh2, L.len + R.len);
46   }
47
48   void build(int p, int l, int r, const vector<int>& a) {
49     if (l == r) {
50       ull v1 = (ull)(a[l] % (int)MOD1);
51       ull v2 = (ull)(a[l] % (int)MOD2);
52       st[p] = Node(v1, v2, 1);
53       return;
54     }
55     int m = (l + r) >> 1;
56     build(p << 1, l, m, a);
57     build(p << 1 | 1, m + 1, r, a);
58     st[p] = merge(st[p << 1], st[p << 1 | 1]);
59   }
60
61   void update(int p, int l, int r, int idx, int val) {
62     if (l == r) {
63       ull v1 = (ull)(val % (int)MOD1);
64       ull v2 = (ull)(val % (int)MOD2);
65       st[p] = Node(v1, v2, 1);
66       return;
67     }
68     int m = (l + r) >> 1;
69     if (idx <= m)
70       update(p << 1, l, m, idx, val);
71     else
72       update(p << 1 | 1, m + 1, r, idx, val);
73
74     st[p] = merge(st[p << 1], st[p << 1 | 1]);
75   }
76
77   void update(int idx, int val) { update(1, 0, n - 1, idx, val); }
78
79   Node queryNode(int p, int l, int r, int ql, int qr) {
80     if (qr < l || ql > r) return Node(0, 0, 0);
81     if (ql <= l & r <= qr) return st[p];
82     int m = (l + r) >> 1;
83     Node L = queryNode(p << 1, l, m, ql, qr);
84     Node R = queryNode(p << 1 | 1, m + 1, r, ql, qr);
85     return merge(L, R);
86   }
87
88   pair<ull, ull> query(int l, int r) {
89     Node res = queryNode(1, 0, n - 1, l, r);
90     return {res.h1, res.h2};
91   }
92
93   void solve() {
94     int n, q;
95     cin >> n >> q;
96     string s;
97     cin >> s;
98     vector<int> a(n), b(n);
99     for (int i = 0, j = n - 1; i < n; ++i, --j) {
100       a[i] = (int)s[i];
101       b[i] = (int)s[j];
102     }
103
104     SegmentTree normal(a), invertido(b);
105
106     while (q--) {
107       int t;
108       cin >> t;
109       if (t == 1) {
110         int pos;
111         char x;
112         cin >> pos >> x;
113         pos--;
114         normal.update(pos, (int)x);
115         invertido.update(n - 1 - pos, (int)x);
116       } else {
117         int l, r;
118         cin >> l >> r;
119         l--, r--;
120         auto h1 = normal.query(l, r);
121         auto h2 = invertido.query(n - 1 - r, n - 1 - l);
122         cout << (h1 == h2 ? "YES" : "NO") << '\n';
123       }
124     }
125
126     int main() {
127       ios::sync_with_stdio(false);
128       cin.tie(nullptr);
129       cout.tie(0);
130       solve();
131       return 0;
132     }

```

```

131 }
132 //=====
133 #include <bits/stdc++.h>
134 using namespace std;
135 using ull = unsigned long long;
136
137 struct SegmentTree {
138     const ull MOD1 = 1000000007ULL;
139     const ull MOD2 = 1000000009ULL;
140     const ull P1 = 1777771ULL;
141     const ull P2 = 19260817ULL;
142
143     struct Node {
144         ull h1, h2;
145         int len;
146         Node(ull _h1=0, ull _h2=0, int _len=0): h1(_h1), h2(_h2),
147             → len(_len) {}
148     };
149
150     int n;
151     vector<Node> st;
152     vector<ull> p1, p2;
153
154     SegmentTree(const vector<int>& a) {
155         n = (int)a.size();
156         st.assign(4*n, Node());
157         p1.assign(n+1, 1);
158         p2.assign(n+1, 1);
159         precomputePowers();
160         build(1, 0, n-1, a);
161     }
162
163     void precomputePowers() {
164         p1[0] = p2[0] = 1;
165         for (int i = 1; i <= n; ++i) {
166             p1[i] = (p1[i-1] * P1) % MOD1;
167             p2[i] = (p2[i-1] * P2) % MOD2;
168         }
169     }
170
171     // MERGE: left.h + right.h * P^{left.len}
172     Node merge(const Node &L, const Node &R) {
173         ull nh1 = ( (L.h1 + (R.h1 * p1[L.len]) % MOD1) ) % MOD1;
174         ull nh2 = ( (L.h2 + (R.h2 * p2[L.len]) % MOD2) ) % MOD2;
175         return Node(nh1, nh2, L.len + R.len);
176     }
177
178     void build(int p, int l, int r, const vector<int>& a) {
179         if (l == r) {
180             ull v1 = (ull)(a[l] % (int)MOD1);
181             ull v2 = (ull)(a[l] % (int)MOD2);
182             st[p] = Node(v1, v2, 1);
183             return;
184         }
185         int m = (l + r) >> 1;
186         build(p<<1, l, m, a);
187         build(p<<1|1, m+1, r, a);
188
189         st[p] = merge(st[p<<1], st[p<<1|1]);
190     }
191
192     void update(int p, int l, int r, int idx, int val) {
193         if (l == r) {
194             ull v1 = (ull)(val % (int)MOD1);
195             ull v2 = (ull)(val % (int)MOD2);
196             st[p] = Node(v1, v2, 1);
197             return;
198         }
199         int m = (l + r) >> 1;
200         if (idx <= m) update(p<<1, l, m, idx, val);
201         else update(p<<1|1, m+1, r, idx, val);
202         st[p] = merge(st[p<<1], st[p<<1|1]);
203     }
204
205     void update(int idx, int val) { update(1, 0, n-1, idx, val); }
206
207     Node queryNode(int p, int l, int r, int ql, int qr) {
208         if (qr < l || ql > r) return Node(0,0,0);
209         if (ql <= l && r <= qr) return st[p];
210         int m = (l + r) >> 1;
211         Node L = queryNode(p<<1, l, m, ql, qr);
212         Node R = queryNode(p<<1|1, m+1, r, ql, qr);
213         return merge(L, R);
214     }
215
216     pair<ull, ull> query(int l, int r) {
217         Node res = queryNode(1, 0, n-1, l, r);
218         return {res.h1, res.h2};
219     }
220
221     void solve(){
222         int n, q; cin >> n >> q;
223         string s; cin >> s;
224         vector<int> a(n), b(n);
225         for (int i = 0, j = n-1; i < n; ++i, --j) {
226             a[i] = (int)(unsigned char)s[i];
227             b[i] = (int)(unsigned char)s[j];
228         }
229
230         SegmentTree normal(a), invertido(b);
231
232         while (q--) {
233             int t; cin >> t;
234             if (t == 1) {
235                 int pos; char x; cin >> pos >> x;
236                 pos--; // 0-based
237                 normal.update(pos, (int)(unsigned char)x);
238                 invertido.update(n-1-pos, (int)(unsigned char)x);
239             } else {
240                 int l, r; cin >> l >> r;
241                 l--, r--;
242                 auto h1 = normal.query(l, r);
243                 // en el invertido pedimos la ventana correspondiente (ya
244                 // → 0-based)
245                 auto h2 = invertido.query(n-1-r, n-1-l);
246                 cout << (h1 == h2 ? "YES" : "NO") << '\n';
247             }
248         }
249     }
250 }
```

```

245     }
246   }
247 }
248
249 int main(){
250   ios::sync_with_stdio(false);
251   cin.tie(nullptr);
252   cout.tie(0);
253   solve();
254   return 0;
255 }
```

## 12 Tree Algorithms

### 12.0.1 Distinct\_Colors

---

```

1 //dusu
2 // You are given a rooted tree consisting of n nodes. The nodes are
3 // numbered 1,2,\ldots,n, and node 1 is the root. Each node has a color
4 // Your task is to determine for each node the number of distinct colors
5 // in the subtree of the node.
6 #include <bits/stdc++.h>
7 using namespace std;
8 #define ll long long
9 #define ull unsigned long long
10 #define fore(i, a, b) for (int i = (a); i < (b); i++)
11 #define FOR(i, n) for (int i = 0; i < (n); i++)
12 #define all(x) (x).begin(), (x).end()
13 #define sz(x) (int)(x).size()
14 #define pb push_back
15 using vi = vector<int>;
16 const int MOD = 1000000007;
17 const ll INF = 9223372036854775807LL;
18 const int tam = 2 * 1e5 + 10;
19 vector<set<int>> conjuntos(tam);
20 vector<int> ans(tam);
21 vector<int> color(tam);
22 void dfs(int nodo, int padre, vector<vi>& g) {
23   for (int vecino : g[nodo]) {
24     if (vecino != padre) { dfs(vecino, nodo, g); }
25   }
26   int bigchild = -1;
27   for (int vecino : g[nodo]) {
28     if (vecino == padre) continue;
29     if (bigchild == -1 or sz(conjuntos[vecino]) >
30         sz(conjuntos[bigchild])) {
31       bigchild = vecino;
32     }
33   }
34   if (bigchild != -1) { swap(conjuntos[nodo], conjuntos[bigchild]); }
35   conjuntos[nodo].insert(color[nodo]);
36   for (int vecino : g[nodo]) {
37     if (vecino == padre or vecino == bigchild) continue;
38     for (int colorsito : conjuntos[vecino]) {
```

```

39       conjuntos[nodo].insert(color[vecino]);
40     }
41   }
42 }
43 void solve() {
44   int n;
45   cin >> n;
46   fore(i, 1, n+1) cin >> color[i];
47   vector<vector<int>> g(n + 1);
48   fore(i, 0, n - 1) {
49     int a, b;
50     cin >> a >> b;
51     g[a].push_back(b);
52     g[b].push_back(a);
53   }
54   dfs(1, -1, g);
55   fore(i, 1, n+1) cout << ans[i] << " ";
56 }
57
58 signed main() {
59   ios::sync_with_stdio(0);
60   cin.tie(0);
61   cout.tie(0);
62   int t = 1;
63   //cin >> t;
64   while (t--) { solve(); }
65 }
```

---