

Índice

1. Preámbulo	1
1.1. Template	1
1.2. Shell	1
2. STL	2
2.1. Resumen	2
2.2. Order statistics multiset	2
3. Estructuras	2
3.1. Prefix table	2
3.2. Sparse table	2
3.3. Segment tree (RMQ)	2
3.4. Disjoint set union	3
4. Algoritmos	3
4.1. Búsqueda binaria	3
5. Matemática	3
5.1. Cuentas	3
5.2. Sqrt	3
5.3. Prime test	3
6. Grafos	3
6.1. Preámbulo	3
6.2. Euler tour	4
6.3. LCA	4
7. Misc.	4
7.1. Operaciones de bits	4

1. Preámbulo

1.1. Template

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 using i64 = int64_t;
4 #define rep(i,N) for (int i = 0; i < int(N); i++)
5 #define scn(k,i,j) for (int k = int(i); k <= int(j); k++)
6 #define pb push_back
7 #define endl '\n'
8 #define mp make_pair
9 #define fst first
10 #define snd second
11 #define forall(it,v) for(auto it = v.begin(); it != v.end(); it++)
12 #define printall(v) forall(x,v){cout << *x << " ";} cout << endl
13 #define printpair(p) cout << "(" << p.fst << ", " << p.snd << ")" << endl
14
15 int main (void) {
16     ios::sync_with_stdio(0); cin.tie(0);
17
18     // :)
19
20     return 0;
21 }
```

1.2. Shell

```
1 // Makefile
2 CPPFLAGS = -std=c++20 -O0 -Wall -g
3 CC = g++
4 // comp.sh: compilar $1 y mostrar primeras $2 lineas de error
5 rm -f $1
6 clear
7 make $1 2>&1 | head -$2
8 // run.sh: correr $1 con $2 como input
9 rm -f $1
10 clear
11 make $1 && ./ $1 < $2
```

2. STL

2.1. Resumen

Función	Params	Descripción
assign	first last / n val	resize y asignación
find	first last val	primer =
is_sorted	first last comp	true si esta ordenado
sort, stable_sort	first last comp	ordena el intervalo
binary_search	first last val comp	true si aparece
lower_bound	first last val comp	primer \geq
upper_bound	first last val comp	primer $>$
next_permutation	first last	sort; do {...} while (next_perm);
prev_permutation	first last	sort; reverse; do {...} while (...);
lexicographical_compare	first last 1,2 comp	"aabbcc" $<$ "aabc"

2.2. Order statistics multiset

Warning: es pesado. Ver time limit del problema.

```

1 #include <ext/pb_ds/assoc_container.hpp>
2 #include <ext/pb_ds/tree_policy.hpp>
3 using namespace __gnu_pbds;
4 struct osms {
5     int t = 0; tree<
6         pair<int,int>, null_type, less<pair<int,int>>,
7         rb_tree_tag, tree_order_statistics_node_update
8     > ms;
9     void add (int x) { ms.insert(mp(x,t++)); }
10    int nle (int x) { return ms.order_of_key(mp(x,-1)); }
11    int nleq (int x) { return ms.order_of_key(mp(x,INT_MAX)); }
12    int cnt (int x) { return nleq(x) - nle(x); }
13    int ith (int i) { return (*ms.find_by_order(i)).fst; }
14    int size (void) { return ms.size(); }
15 };

```

3. Estructuras

3.1. Prefix table

PTable pt = {arr, arr.size()}; pt.make();

```

1 struct PTable {

```

```

2     vector<i64> &arr; int N;
3     vector<i64> pt;
4     void make () {
5         pt.resize(N);
6         rep(i,N) pt[i] = !i ? arr[i] : pt[i-1] + arr[i];
7     }
8     i64 q (int i, int j) { return (--i < 0) ? pt[j] : pt[j] - pt[i]; }
9 };

```

3.2. Sparse table

Operacion asociativa **idempotente**.

STable st = {arr, arr.size()}; st.make();

```

1 struct STable {
2     vector<int>& arr; int N;
3     vector<vector<int>>> st;
4     int op (int a, int b) { return min(a,b); }
5     void make () {
6         st.resize(20, vector<int>(N));
7         st[0] = arr; scn(w,1,19) scn(i,0,N - (1 << w))
8             st[w][i] = op(st[w-1][i], st[w-1][i + (1 << (w-1))]);
9     }
10    int q (int i, int j) {
11        int w = log2fl(j - i + 1);
12        return op(st[w][i], st[w][j - (1 << w) + 1]);
13    }
14 };

```

3.3. Segment tree (RMQ)

SegTree<int>st = {arr, arr.size()}; st.make();

```

1 template<class T> struct SegTree {
2     vector<T>& arr; int N;
3     T op (T a, T b) { return min(a,b); }
4     T id = INT_MAX;
5     vector<T> t;
6     void make () {
7         t.resize(N << 1); rep(i,N) t[i+N] = arr[i];
8         for (int i = N - 1; i; i--) t[i] = op(t[i<<1], t[i<<1|1]);
9     }
10    void set (int i, int v) {

```

```

11     for(t[i += N] = v; i > 1; i >>= 1) t[i>>1] = op(t[i], t[i^1]);
12 }
13 T q (int l, int r) {
14     int res = id;
15     for (l += N, r += N; l < r; l >>= 1, r >>= 1) {
16         if (l&1) res = op(res, t[l++]);
17         if (r&1) res = op(res, t[--r]);
18     } return res;
19 }
20 };

```

3.4. Disjiont set union

DSU dsu(N);

```

1 struct DSU {
2     vector<int> p, w; int nc;
3     DSU (int n) {
4         nc = n, p.resize(n), w.resize(n);
5         rep(i,n) p[i] = i, w[i] = 1;
6     }
7     int f (int x) { return p[x] == x ? x : p[x] = f(p[x]); }
8     void u (int x, int y) {
9         x = f(x), y = f(y);
10        if (x == y) return;
11        if (w[x] > w[y]) swap(x,y);
12        p[x] = y, w[y] += w[x];
13    }
14    bool c (int x, int y) { return f(x) == f(y); }
15 };

```

4. Algoritmos

4.1. Búsqueda binaria

Si existe, idx de primer true Si no, i - 1

```

1 i64 bsearch (i64 i, i64 j, bool (*pred)(i64)) {
2     int d = i-1; while (!(i + 1 == j)) {
3         i64 m = i + ((j - i) >> 1);
4         pred(m) ? j = m : i = m;
5     }
6     if (pred(i)) return i;

```

```

7     if (pred(j)) return j;
8     return d;
9 }

```

5. Matemática

5.1. Cuentas

```

1 #define ceildiv(a,b) ((a+b-1)/b)
2 #define log2fl(x) (x ? 63 - __builtin_clzll(x) : -1)

```

5.2. Sqrt

```

1 i64 isqrt (i64 x) {
2     i64 s = 0; for (i64 k = 1 << 30; k; k >>= 1)
3         if ((s+k)*(s+k) <= x) s += k;
4     return s;
5 }

```

5.3. Prime test

```

1 struct primetest {
2     bool c[1000001]; vector<int> p;
3     primetest () {
4         p.reserve(1<<16); scn(i,2,1000000) if (!c[i]) {
5             p.pb(i); for (int j = 2; i*j < 1000001; j++) c[i*j] = 1;
6         }
7     }
8     bool isprime (int x) {
9         for (int i = 0, d = p[i]; d*d <= x; d = p[++i])
10            if (!(x % d)) return false;
11        return x >= 2;
12    }
13 };

```

6. Grafos

6.1. Preámbulo

```

1 typedef vector<vector<int>> adj;
2 typedef vector<vector<pair<int,i64>>> wadj;

```

6.2. Euler tour

```
ETour et = {G, G.size()}; et.make(0);
```

```
1 struct ETour {
2     adj& G; int N, R;
3     vector<int> t, f, d;
4     void dfs (int u, int de = 0) {
5         d[u] = de, f[u] = t.size(), t.pb(u);
6         for (int v : G[u]) { dfs(v,de+1); t.pb(u); }
7     }
8     void make () { f.resize(N), d.resize(N), dfs(R); }
9 };
```

6.3. LCA

```
LCA lca = {G, G.size(), root}; lca.make();
```

```
1 struct LCA {
2     adj& G; int N, R;
3     int M; vector<int> e, f, d; adj st;
4     void dfs (int u, int de = 0) {
5         d[u] = de, f[u] = e.size(), e.pb(u);
6         for (int v : G[u]) dfs(v,de+1), e.pb(u);
7     }
8     int op (int a, int b) {
9         if (a == -1) return b;
10        if (b == -1) return a;
11        return d[a] < d[b] ? a : b;
12    }
13    void make () {
14        f.resize(N), d.resize(N), dfs(R), M = e.size();
15        st.resize(20, vector<int>(M));
16        st[0] = e; scn(w,1,19) scn(i,0,M - (1 << w))
17            st[w][i] = op(st[w-1][i], st[w-1][i + (1 << (w-1))]);
18    }
19    int q (int u, int v) {
20        int i = f[u], j = f[v];
21        if (i > j) swap(i,j);
22        int w = log2fl(j - i + 1);
23        return op(st[w][i], st[w][j - (1 << w) + 1]);
24    }
25 };
```

7. Misc.

7.1. Operaciones de bits

```
1 #define bits(x) __builtin_popcount(x)
```