Competitive Programming Reference

Ajolotetos

September 13, 2025

Contents

1 Plantilla y Utilidades de C++

Bloque inicial para agilizar código en competiciones.

Listing 1: Plantilla y Fast I/O

```cpp
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define endl '\n'
#define all(x) (x).begin(), (x).end()


void solve(int testcase){

}

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

    int t = 1;
    cin >> t;
    for(int i = 0; i<t; i++){
        solve(i);
    }
}
```

2 STL: Contenedores y Algoritmos

Uso de las estructuras y funciones estándar de la biblioteca.

2.1 Contenedores Comunes

```cpp
vector<int> v;                  // acceso din mico
deque<int> dq;                  // inserci n en ambos extremos
set<int> st;                    // elementos  nicos  ordenados
unordered_map<int,int> mp;      // mapa hash
priority_queue<int> pq;         // m ximo por defecto
priority_queue<int,vector<int>,greater<int>> minpq; // m nimo
```

2.2 Algoritmos Útiles

```
sort(all(v));                  // ordenar ascendente
next_permutation(all(v));      // siguiente permutaci n lexicogr fica
lower_bound(all(v), x);        // primer >= x
upper_bound(all(v), x);        // primer > x
binary_search(all(v), x);      // existe x?
accumulate(all(v), 0LL);       // suma de elementos
rotate(v.begin(), v.begin()+k, v.end()); // rotar
partition(all(v), [](int x){return x%2==0;}); // separar pares e
    impares
iota(v.begin(), v.end(), 1);   // llenar con 1,2,...
```

3 Fenwick Tree (BIT)

Estructura para sumas y actualizaciones en rango prefijo en $O(\log n)$.

Listing 2: Fenwick Tree

```
struct Fenwick {
  int n; vector<ll> f;
  Fenwick(int _n): n(_n), f(n+1,0) {}
  void update(int i, ll v) {
    for (; i <= n; i += i&-i) f[i] += v;
  }
  ll query(int i) {
    ll s = 0;
    for (; i > 0; i -= i&-i) s += f[i];
    return s;
  }
  ll range(int l, int r) {
    return query(r) - query(l-1);
  }
};
// Ejemplo:
// Fenwick ft(5);
// ft.update(3,10);
// ft.range(1,3) -> 10
```

4 Binary Search

Busca un `key` en un rango ordenado en tiempo $O(\log n)$.

Listing 3: Binary Search Standard

```
int l = -1, r = n;
while (r - l > 1) {
    int m = (l + r) / 2;
    if (k < a[m]) {
        r = m; // a[l] <= k < a[m] <= a[r]
    } else {
        l = m; // a[l] <= a[m] <= k < a[r]
    }
}
```

5 Binary Exponentiation

Calcula a^b en $O(\log b)$.

Listing 4: Binary Exponentiation

```
long long binexp(long long a, long long b) {
  long long res = 1;
  while (b) {
    if (b & 1) res = res * a;
    a = a * a;
    b >>= 1;
  }
  return res;
}
// Ejemplo:
// binexp(2,10) -> 1024
```

5.1 Modular Exponentiation

Calcula $a^b \bmod m$.

Listing 5: Modular Exponentiation

```cpp
long long modexp(long long a, long long b, long long m) {
  long long res = 1;
  a %= m;
  while (b) {
    if (b & 1) res = (res * a) % m;
    a = (a * a) % m;
    b >>= 1;
  }
  return res;
}
// Ejemplo:
// modexp(2,10,1000) -> 24
```

6 Factorial and Inverse Factorial

Cálculo de factoriales e inversos en tiempo lineal.

Listing 6: Factorials

```cpp
typedef long long ll;
const int N = 1000000;
ll fact[N+1], ifact[N+1], mod = 1000000007;

void init_fact() {
  fact[0] = 1;
  for (int i = 1; i <= N; i++) fact[i] = fact[i-1] * i % mod;
  ifact[N] = modexp(fact[N], mod-2, mod);
  for (int i = N; i > 0; i--) ifact[i-1] = ifact[i] * i % mod;
}
// Uso:
// fact[k] = k! % mod
// ifact[k] = (k!)^{-1} % mod
```

7 Combinatorics

Cálculo de coeficientes binomiales en $O(1)$ tras precomputación.

Listing 7: Binomial Coefficient

```cpp
long long nCk(int n, int k) {
  if (k < 0 || k > n) return 0;
  return fact[n] * ifact[k] % mod * ifact[n-k] % mod;
}
// Ejemplo:
// nCk(5,2) -> 10
```

8 Sieve of Eratosthenes

Genera todos los primos hasta N en $O(n \log \log n)$.

Listing 8: Sieve

```cpp
vector<int> sieve(int n) {
  vector<bool> isprime(n+1, true);
  vector<int> primes;
  isprime[0] = isprime[1] = false;
  for (int i = 2; i <= n; i++) if (isprime[i]) {
    primes.push_back(i);
    for (long long j = 1LL*i*i; j <= n; j += i) isprime[j] = false;
  }
  return primes;
}
// Ejemplo:
// auto ps = sieve(30);
```

9 Primality Test (Miller–Rabin)

Test probabilístico de primalidad para enteros hasta 64 bits.

Listing 9: Miller–Rabin Test

```cpp
bool isPrime(ll n) {
  if (n < 2) return false;
  for (ll p: {2,3,5,7,11,13,17,19,23,29,31,37}) if (n%p==0) return
     n==p;
  ll d = n-1, s = 0; while ((d&1)==0) d>>=1, s++;
  auto check=[&](ll a){ ll x=modexp(a,d,n); if (x==1||x==n-1) return
     true;
    for(int i=1;i<s;i++){ x=(__int128)x*x%n; if(x==n-1) return true;}
       return false; };
  for (ll a: {2,325,9375,28178,450775,9780504,1795265022})
     if(!check(a)) return false;
  return true;
}
// Ejemplo:
// isPrime(101) -> true
```

10 Disjoint Set Union (DSU)

Unión y búsqueda con compresión de caminos y unión por rango.

Listing 10: DSU con `make_set` y unión por tamaño

```
struct DSU {
    vector<int> parent, sz;

    // Inicializa el conjunto con un solo elemento v
    void make_set(int v) {
        parent[v] = v;
        sz[v] = 1;
    }

    int find_set(int v) {
        if (v == parent[v])
            return v;
        return parent[v] = find_set(parent[v]);
    }

    void union_sets(int a, int b) {
        a = find_set(a);
        b = find_set(b);
        if (a != b) {
            // unir por tama o
            if (sz[a] < sz[b])
                swap(a, b);
            parent[b] = a;
            sz[a] += sz[b];
        }
    }
};
// Ejemplo de uso:
// DSU dsu;
// dsu.parent.resize(n);
// dsu.sz.resize(n);
// for(int v = 0; v < n; v++) dsu.make_set(v);
// dsu.union_sets(1, 2);
// int root = dsu.find_set(2);
```

11 Depth-First Search (DFS)

Recorrido de grafos en $O(n+m)$.

Listing 11: DFS Recursive

```
vector<vector<int>> adj;
vector<bool> vis;
void dfs(int u){ vis[u]=true; for(int v:adj[u]) if(!vis[v]) dfs(v);}
// Ejemplo:
// adj={{1,2},{0,3},{0},{1}}; vis.assign(4,false); dfs(0);
```

12 Breadth-First Search (BFS)

Encuentra distancias mínimas en grafos no ponderados.

Listing 12: BFS

```
vector<int> bfs(int s){ int n=adj.size(); vector<int> d(n,-1);
  queue<int>q; d[s]=0; q.push(s);
  while(!q.empty()){ int u=q.front(); q.pop();
    for(int v:adj[u]) if(d[v]<0){ d[v]=d[u]+1; q.push(v);} }
  return d;
}
// Ejemplo:
// auto dist=bfs(0);
```

13 Kruskal's Algorithm

Construye MST usando DSU en $O(m \log m)$.

Listing 13: Kruskal

```
struct Edge{int u,v; ll w;};
ll kruskal(int n, vector<Edge>& edges){
   sort(edges.begin(),edges.end(),[](auto&a,auto&b){return a.w<b.w;});
  DSU dsu(n); ll cost=0; for(auto&e:edges) if(dsu.unite(e.u,e.v))
     cost+=e.w; return cost;
}
// Ejemplo:
// edges={{0,1,5},{1,2,3},{0,2,7}}; kruskal(3,edges);
```

14 Dijkstra's Algorithm

Calcula distancias mínimas en grafos con pesos no negativos.

Listing 14: Dijkstra

```
vector<ll> dijkstra(int s){ int n=adj.size(); const ll INF=LLONG_MAX;
  vector<ll> dist(n,INF); dist[s]=0;
  priority_queue<pair<ll,int>,vector<pair<ll,int>>,greater<>>pq;
  pq.emplace(0,s);
  while(!pq.empty()){ auto [d,u]=pq.top(); pq.pop(); if(d>dist[u])
     continue;
    for(auto [v,w]:adj[u]) if(dist[v]>d+w){ dist[v]=d+w;
       pq.emplace(dist[v],v);} }
  return dist;
}
// Ejemplo:
// adj={{{1,2},{2,5}},{{0,2},{2,1}},{{0,5},{1,1}}}; dijkstra(0);
```

15 Segment Tree

Consultas y actualizaciones en rango en $O(\log n)$.

Listing 15: Segment Tree

```cpp
#include <bits/stdc++.h>
using namespace std;
using ll = long long;
#define endl '\n'
#define sz(x) int((x).size())

vector<ll> tree, lazy;

void apply(int node, int l, int h, ll v){
    lazy[node] += v;
    tree[node] += v * ll(h - l + 1);
}

void push(int node, int l, int h){
    if(l == h || lazy[node] == 0) return;
    int m = (l + h) >> 1;
    apply(node * 2, l, m, lazy[node]);
    apply(node * 2 + 1, m+1, h, lazy[node]);
    lazy[node] = 0;
}

ll query(int node, int l, int h, int q_l, int q_h){
    if(h < q_l || l > q_h) return 0;           // no overlap
    if(q_l <= l && h <= q_h) return tree[node]; // full cover
    push(node, l, h);
    int m = (l + h) >> 1;
    return query(node * 2, l, m, q_l, q_h)
         + query(node * 2 + 1, m+1, h, q_l, q_h);
}

void update(int node, int l, int h, int q_l, int q_h, ll v){
    if(h < q_l || l > q_h) return;             // no overlap
    if(q_l <= l && h <= q_h){
        apply(node, l, h, v);
        return;
    }
    push(node, l, h);
    int m = (l + h) >> 1;
    update(node * 2, l, m, q_l, q_h, v);
    update(node * 2 + 1, m+1, h, q_l, q_h, v);
    tree[node] = tree[node*2] + tree[node*2 + 1];
}

void solve(int testcase){
    int n_input, q;
    if(!(cin >> n_input >> q)) return;
    vector<ll> A(n_input);
    for(auto &a : A) cin >> a;

    int n = 1;
    while(n < (int)A.size()) n <<= 1;

    tree.assign(2 * n, 0LL);
```

```
    lazy.assign(2 * n, 0LL);

    for(int i = 0; i < sz(A); i++){
        tree[n + i] = A[i];
    }
    for(int i = n - 1; i >= 1; i--){
        tree[i] = tree[2 * i] + tree[2 * i + 1];
    }

    for(int i = 0; i < q; ++i){
        int type; cin >> type;
        if(type == 1){
            int a, b; ll v;
            cin >> a >> b >> v;
            a--; b--;
            if(a > b) swap(a, b);
            if(a < 0) a = 0;
            if(b >= n) b = n-1;
            if(a <= b) update(1, 0, n - 1, a, b, v);
        } else {
            int a; cin >> a;
            a--;
            if(a < 0) a = 0;
            if(a >= n) a = n-1;
            cout << query(1, 0, n - 1, a, a) << endl;
        }
    }
}

int main(){
    ios::sync_with_stdio(false); cin.tie(nullptr);
    int t = 1;
    // cin >> t;
    for(int tc = 0; tc < t; ++tc) solve(tc);
    return 0;
}
}
```

16 Ejemplo de DP: Knapsack 0/1

Mochila 0/1 en $O(nW)$.

Listing 16: Knapsack

```
ll knapsack(int n,int W,vector<int>& wt, vector<int>& val){ vector<ll>
   dp(W+1);
  for(int i=0;i<n;i++) for(int w=W;w>=wt[i];w--)
     dp[w]=max(dp[w],dp[w-wt[i]]+val[i]);
  return dp[W]; }
// Ejemplo:
// wt={3,4,5},val={30,50,60},W=8; knapsack(3,8,wt,val);
```

17 Bitwise Hacks y Built-ins

Operaciones de bits y funciones integradas en $O(1)$.

```
int popcnt = __builtin_popcount(x);      // bits a 1 en int
int lz = __builtin_clz(x);               // ceros iniciales en 32-bit
int tz = __builtin_ctz(x);               // ceros finales
int p = x & -x;                          // LSbit activo
bool isPow2 = (x & (x-1)) == 0;          // potencia de dos?
// Ejemplo:
// x=20 (10100b): popcnt=2, lz=27, tz=2, p=4, isPow2=false
```

18 Ejemplo con Máscara de Bits

Muestra cómo usar máscaras de bits para iterar subconjuntos de un conjunto de tamaño n en $O(2^n \cdot n)$.

Listing 17: Iteración de Subconjuntos

```
// Dado un conjunto representado por n elementos, cada subconjunto es
   un entero mask de 0 a (1<<n)-1.
int n = 3;
vector<string> items = {"A", "B", "C"};
for (int mask = 0; mask < (1 << n); mask++) {
  cout << "Subconjunto { ";
  for (int i = 0; i < n; i++) {
    if (mask & (1 << i)) cout << items[i] << " ";
  }
  cout << "}\n";
}
// Salida:
// {}
// { A }
// { B }
// { A B }
// { C }
// { A C }
// { B C }
// { A B C }
```