Pedro Augusto Ulisses Andrade Lucas Andrade

Katia Volte Para Mim! Cantarei Boate Azul Para Você (>o<)

Contents

1 Utils **2**
1.1 Makefile 2
1.2 Limites 2
1.3 Mini Template Cpp 2
1.4 Template Cpp 3
1.5 Files 3
1.6 Template Python 3

2 Informações **4**
2.1 Vector 4
2.2 Sort 4
2.3 Priority Queue 4

3 .vscode **5**

4 Estruturas **5**
4.1 Bittree 5
4.2 Sparse Table Disjunta 6
4.3 Seg Tree 6

5 Grafos **6**
5.1 Bfs 6
5.2 Dijkstra 7
5.3 Euler Tree 7
5.4 Kruskal 7
5.5 Dfs 8

6 Matematica **8**
6.1 Mdc Multiplo 8
6.2 Factorial 8
6.3 Mmc Multiplo 8
6.4 Fast Exponentiation 8
6.5 Sieve 9
6.6 Fact Grande 9
6.7 Mdc 9
6.8 Primo 9
6.9 Miller Rabin 9
6.10 Fatorial Grande 10
6.11 Sieve Linear 10
6.12 Numeros Grandes 10
6.13 Mmc 11

7 Strings **11**
7.1 Ocorrencias 11
7.2 Upper Case 11
7.3 Palindromo 11
7.4 Split Cria 11
7.5 Remove Acento 12
7.6 Chaves Colchetes Parenteses 12
7.7 Lower Case 12
7.8 Lexicograficamente Minima 12

7.9 Split 12

8 Vector **13**

8.1 Teste 13

8.2 Remove Repetitive 13

8.3 Elemento Mais Frequente 13

9 Outros **13**

9.1 Binario 13

9.2 Horario 13

9.3 Max Subarray Sum 14

9.4 Binary Search 14

9.5 Fibonacci 14

1 Utils

1.1 Makefile

```
CXX = g++
CXXFLAGS = -fsanitize=address,undefined -fno-omit-frame-pointer -g -Wall -Wshadow -std=c++17 -Wno-unused-result -Wno-sign-compare -Wno-char-subscripts #-fuse-ld=gold

compile:
    g++ -g $(f).cpp $(CXXFLAGS) -o $(f)
exec:
    ./$(f)

runc: compile exec
runci: compile
    ./$(f) < $(f).txt

clearexe:
    find . -maxdepth 1 -type f -executable -exec rm {} +
cleartxt:
    find . -type f -name "*.txt" -exec rm -f {} \;
clear: clearexe cleartxt

runp:
    python3 $(f).py
runpt:
    python3 $(f).py < $(f).txt
```

1.2 Limites

```
// LIMITES DE ÇÃREPRESENTAO DE DADOS

       tipo     | bits |      ímnimo .. ámximo      | ãpreciso decimal
----------------+------+----------------------------+------------------
char            |   8  |           0 .. 127         |         2
signed char     |   8  |        -128 .. 127         |         2
unsigned char   |   8  |           0 .. 255         |         2
short           |  16  |     -32.768 .. 32.767      |         4
unsigned short  |  16  |           0 .. 65.535      |         4
int             |  32  |  -2 x 10**9 .. 2 x 10**9   |         9
unsigned int    |  32  |           0 .. 4 x 10**9   |         9
int64_t         |  64  | -9 x 10**18 .. 9 x 10**18  |        18
uint64_t        |  64  |           0 .. 18 x 10**18 |        19

// LIMITES DE MEMORIA

1MB = 1,048,576 bool
1MB =   524,288 char
1MB =   262,144 int32_t
1MB =   131,072 int64_t
1MB =    65,536 float
1MB =    32,768 double
1MB =    16,384 long double


// ESTOURAR TEMPO

1s = 10^8 çõoperaes

// FATORIAL

0!  =                         1
1!  =                         1
2!  =                         2
3!  =                         6
4!  =                        24
5!  =                       120
6!  =                       720
7!  =                     5.040
8!  =                    40.320
9!  =                   362.880
10! =                 3.628.800
11! =                39.916.800
12! =               479.001.600 [limite do (u)int]
13! =             6.227.020.800
14! =            87.178.291.200
15! =         1.307.674.368.000
16! =        20.922.789.888.000
17! =       355.687.428.096.000
18! =     6.402.373.705.728.000
19! =   121.645.100.408.832.000
20! = 2.432.902.008.176.640.000 [limite do (u)int64_t]
```

1.3 Mini Template Cpp

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

#define _ ios_base::sync_with_stdio(0); cin.tie(0);

#define all(a)          a.begin(), a.end()
#define int             long long int
#define double          long double
#define endl            "\n"
#define print_v(a)      for(auto x : a) cout << x << " "; cout << endl
#define f(i,s,e)        for(int i=s;i<e;i++)
#define rf(i,e,s)       for(int i=e-1;i>=s;i--)
#define dbg(x) cout << #x << " = " << x << endl;

void solve() {

}

int32_t main() { _

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

    return 0;
}
```

1.4 Template Cpp

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

#define _ ios_base::sync_with_stdio(0); cin.tie(0);

#define all(a)          a.begin(), a.end()
#define int             long long int
#define double          long double
#define vi              vector<int>
#define endl            "\n"
#define print_v(a)      for(auto x : a) cout << x << " "; cout << endl
#define print_vp(a)     for(auto x : a) cout << x.F << " " << x.S << endl
#define f(i,s,e)        for(int i=s;i<e;i++)
#define rf(i,e,s)       for(int i=e-1;i>=s;i--)
#define CEIL(a, b)      ((a) + (b - 1))/b
#define TRUNC(x)        floor(x * 100) / 100

#define dbg(x) cout << #x << " = " << x << " ";
#define dbgl(x) cout << #x << " = " << x << endl;

const int INF =  0x7f3f3f3f;
const int LINF = 0x3f3f3f3f3f3f3f3f; // 0x com 8 3f's
const double PI = acos(-1);
const int MAX = 1e6+10; // 10^6 + 10

void solve() {

}

int32_t main() { _

    clock_t z = clock();

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

    cerr << fixed << "Run Time : " << ((double)(clock() - z) / CLOCKS_PER_SEC) << endl;
    return 0;
}
```

1.5 Files

```bash
#!/bin/bash

for c in {a..f}; do
    cp temp.cpp "$c.cpp"
    echo "$c" > "$c.txt"
    if [ "$c" = "$letter" ]; then
        break
    fi
done
```

1.6 Template Python

```python
import sys
import math
import bisect
from sys import stdin,stdout
from math import gcd,floor,sqrt,log
from collections import defaultdict as dd
from bisect import bisect_left as bl,bisect_right as br

sys.setrecursionlimit(100000000)

inp    =lambda: int(input())
strng  =lambda: input().strip()
jn     =lambda x,l: x.join(map(str,l))
strl   =lambda: list(input().strip())
mul    =lambda: map(int,input().strip().split())
mulf   =lambda: map(float,input().strip().split())
seq    =lambda: list(map(int,input().strip().split()))

ceil   =lambda x: int(x) if(x==int(x)) else int(x)+1
ceildiv=lambda x,d: x//d if(x%d==0) else x//d+1

flush  =lambda: stdout.flush()
stdstr =lambda: stdin.readline()
stdint =lambda: int(stdin.readline())
stdpr  =lambda x: stdout.write(str(x))

mod=1000000007

#main code

a = None
b = None
lista = None

def ident(*args):
    if len(args) == 1:
        return args[0]
    return args


def parsin(*, l=1, vpl=1, s=" "):
    if l == 1:
        if vpl == 1: return ident(input())
        else: return list(map(ident, input().split(s)))
    else:
        if vpl == 1: return [ident(input()) for _ in range(l)]
        else: return [list(map(ident, input().split(s))) for _ in range(l)]


def solve():
    pass

# if __name__ == '__main__':
def main():
    st = clk()

    escolha = "in"
    #escolha = "num"

    match escolha:
        case "in":
            # êl infinitas linhas agrupadas de 2 em 2
            # pra infinitos valores em 1 linha pode armazenar em uma lista
            while True:
                global a, b
                try: a, b = input().split()
                except (EOFError): break #permite ler todas as linahs dentro do .txt
                except (ValueError): pass # consegue ler éat linhas em branco
                else:
                    a, b = int(a), int(b)
                solve()

        case "num":
            global lista
            # int l; cin >> l; while(l--){for(i=0; i<vpl; i++)}
            # retorna listas com inputs de cada linha
            # leia l linhas com vpl valores em cada uma delas
                # caseo seja mais de uma linha, retorna lista com listas de inputs
            lista = parsin(l=2, vpl=5)
            solve()

    sys.stderr.write(f"Run Time : {(clk() - st):.6f} seconds\n")

main()
```

2 Informações

2.1 Vector

```
// INICIALIZAR
vector<int> v (n); // n ócpias de 0
vector<int> v (n, v); // n ócpias de v

// PUSH_BACK
// Complexidade: O(1) amortizado (O(n) se realocar)
v.push_back(x);

// REMOVER
// Complexidade: O(n)
v.erase(v.begin() + i);

// INSERIR
// Complexidade: O(n)
v.insert(v.begin() + i, x);

// ORDENAR
// Complexidade: O(n log(n))
sort(v.begin(), v.end());
sort(all(v));

// BUSCA BINARIA
// Complexidade: O(log(n))
// Retorno: true se existe, false se ãno existe
binary_search(v.begin(), v.end(), x);

// FIND
// Complexidade: O(n)
// Retorno: iterador para o elemento, v.end() se ãno existe
find(v.begin(), v.end(), x);

// CONTAR
// Complexidade: O(n)
// Retorno: únmero de êocorrncias
count(v.begin(), v.end(), x);
```

2.2 Sort

```
vector<int> v;
    // Sort Crescente:
    sort(v.begin(), v.end());
    sort(all(v));

    // Sort Decrescente:
    sort(v.rbegin(), v.rend());
    sort(all(v), greater<int>());

    // Sort por uma çãfuno:
    auto cmp = [](int a, int b) { return a > b; }; // { 2, 3, 1 } -> { 3, 2, 1 }
    auto cmp = [](int a, int b) { return a < b; }; // { 2, 3, 1 } -> { 1, 2, 3 }
    sort(v.begin(), v.end(), cmp);
    sort(all(v), cmp);

    // Sort por uma çãfuno (çãcomparao de pares):
    auto cmp = [](pair<int, int> a, pair<int, int> b) { return a.second > b.second; };
```

2.3 Priority Queue

```
// HEAP CRESCENTE {5,4,3,2,1}
priority_queue<int> pq; // max heap
    // maior elemento:
    pq.top();

// HEAP DECRESCENTE {1,2,3,4,5}
priority_queue<int, vector<int>, greater<int>> pq; // min heap
    // menor elemento:
    pq.top();

// REMOVER ELEMENTO
// Complexidade: O(n)
// Retorno: true se existe, false se ãno existe
```

```
pq.remove(x);

// INSERIR ELEMENTO
// Complexidade: O(log(n))
pq.push(x);

// REMOVER TOP
// Complexidade: O(log(n))
pq.pop();

// TAMANHO
// Complexidade: O(1)
pq.size();

// VAZIO
// Complexidade: O(1)
pq.empty();

// LIMPAR
// Complexidade: O(n)
pq.clear();

// ITERAR
// Complexidade: O(n)
for (auto x : pq) {}

// çãOrdenao por çãfuno customizada passada por parametro ao criar a pq
// Complexidade: O(n log(n))
auto cmp = [](int a, int b) { return a > b; };
priority_queue<int, vector<int>, decltype(cmp)> pq(cmp);
```

3 .vscode

4 Estruturas

4.1 Bittree

```
// C++ code to demonstrate operations of Binary Index Tree
#include <iostream>

using namespace std;

/*       n --> No. of elements present in input array.
    BITree[0..n] --> Array that represents Binary Indexed Tree.
    arr[0..n-1] --> Input array for which prefix sum is evaluated. */

// Returns sum of arr[0..index]. This function assumes
// that the array is preprocessed and partial sums of
// array elements are stored in BITree[].
int getSum(int BITree[], int index)
{
    int sum = 0; // Initialize result

    // index in BITree[] is 1 more than the index in arr[]
    index = index + 1;

    // Traverse ancestors of BITree[index]
    while (index>0)
    {
        // Add current element of BITree to sum
        sum += BITree[index];

        // Move index to parent node in getSum View
        index -= index & (-index);
    }
    return sum;
}

// Updates a node in Binary Index Tree (BITree) at given index
// in BITree. The given value ’val’ is added to BITree[i] and
// all of its ancestors in tree.
void updateBIT(int BITree[], int n, int index, int val)
{
    // index in BITree[] is 1 more than the index in arr[]
    index = index + 1;

    // Traverse all ancestors and add ’val’
    while (index <= n)
    {
    // Add ’val’ to current node of BI Tree
    BITree[index] += val;

    // Update index to that of parent in update View
    index += index & (-index);
    }
}

// Constructs and returns a Binary Indexed Tree for given
// array of size n.
int *constructBITree(int arr[], int n)
{
    // Create and initialize BITree[] as 0
    int *BITree = new int[n+1];
    for (int i=1; i<=n; i++)
        BITree[i] = 0;

    // Store the actual values in BITree[] using update()
    for (int i=0; i<n; i++)
        updateBIT(BITree, n, i, arr[i]);

    // Uncomment below lines to see contents of BITree[]
    //for (int i=1; i<=n; i++)
    //   cout << BITree[i] << " ";

    return BITree;
}


// Driver program to test above functions
int main()
{
    int freq[] = {2, 1, 1, 3, 2, 3, 4, 5, 6, 7, 8, 9};
```

```cpp
    int n = sizeof(freq)/sizeof(freq[0]);
    int *BITree = constructBITree(freq, n);
    cout << "Sum of elements in arr[0..5] is "
        << getSum(BITree, 5);

    // Let use test the update operation
    freq[3] += 6;
    updateBIT(BITree, n, 3, 6); //Update BIT for above change in arr[]

    cout << "\nSum of elements in arr[0..5] after update is "
        << getSum(BITree, 5);

    return 0;
}
```

4.2 Sparse Table Disjunta

```cpp
// Sparse Table Disjunta
//
// Resolve qualquer operacao associativa
// MAX2 = log(MAX)
//
// Complexidades:
// build - O(n log(n))
// query - O(1)

namespace sparse {
    int m[MAX2][2*MAX], n, v[2*MAX];
    int op(int a, int b) { return min(a, b); }
    void build(int n2, int* v2) {
        n = n2;
        for (int i = 0; i < n; i++) v[i] = v2[i];
        while (n&(n-1)) n++;
        for (int j = 0; (1<<j) < n; j++) {
            int len = 1<<j;
            for (int c = len; c < n; c += 2*len) {
                m[j][c] = v[c], m[j][c-1] = v[c-1];
                for (int i = c+1; i <  c+len; i++) m[j][i] = op(m[j][i-1], v[i]);
                for (int i = c-2; i >= c-len; i--) m[j][i] = op(v[i], m[j][i+1]);
            }
        }
    }
    int query(int l, int r) {
        if (l == r) return v[l];
        int j = __builtin_clz(l) - __builtin_clz(l^r);
        return op(m[j][l], m[j][r]);
    }
}
```

4.3 Seg Tree

```cpp
// SegTree
//
// Query: soma do range [a, b]
// Update: soma x em cada elemento do range [a, b]
//
// Complexidades:
// build - O(n)
// query - O(log(n))
// update - O(log(n))
namespace seg {

    int seg[4*MAX];
    int n, *v;

    int op(int a, int b) { return a + b; }

    int build(int p=1, int l=0, int r=n-1) {
        if (l == r) return seg[p] = v[l];
        int m = (l+r)/2;
        return seg[p] = op(build(2*p, l, m), build(2*p+1, m+1, r));
    }

    void build(int n2, int* v2) {
        n = n2, v = v2;
        build();
    }

    int query(int a, int b, int p=1, int l=0, int r=n-1) {
        if (a <= l and r <= b) return seg[p];
        if (b < l or r < a) return 0;
        int m = (l+r)/2;
        return op(query(a, b, 2*p, l, m), query(a, b, 2*p+1, m+1, r));
    }

    int update(int a, int b, int x, int p=1, int l=0, int r=n-1) {
        if (a <= l and r <= b) return seg[p];
        if (b < l or r < a) return seg[p];
        int m = (l+r)/2;
        return seg[p] = op(update(a, b, x, 2*p, l, m), update(a, b, x, 2*p+1, m+1, r));
    }
};
```

5 Grafos

5.1 Bfs

```cpp
// BFS com informacoes adicionais sobre a distancia e o pai de cada vertice
// Complexidade: O(V + E), onde V eh o numero de vertices e E o numero de areqas
vector<vector<int>> adj; // liqa de adjacencia
int n, s; // n = numero de vertices, s = vertice inicial

vector<bool> used(n);
vector<int> d(n), p(n);

void bfs(int s) {
    queue<int> q;
    q.push(s);
    used[s] = true;
    d[s] = 0;
    p[s] = -1;

    while (!q.empty()) {
        int v = q.front();
        q.pop();
        for (int u : adj[v]) {
            if (!used[u]) {
                used[u] = true;
                q.push(u);
                d[u] = d[v] + 1;
                p[u] = v;
            }
        }
    }
}

//pra uma bfs que n guarda o backtracking:
void bfs(int p) {
    memset(visited, 0, sizeof visited);
    queue<int> q;
    q.push(p);

    while (!q.empty()) {
        int curr = q.top();
        q.pop();
        if (visited[curr]==1)continue;
        visited[curr]=1;
        // process current node here

        for (auto i : adj[curr]) {
            q.push(i);
        }

    }
}
```

5.2 Dijkstra

```
vector<vector<pair<int, int>>> adj; // adj[a] = [{b, w}]
int n;

vector<int> dist(n, LLINF);
vector<int> parent(n, -1);
vector<bool> used(n);

//Complexidade: O((V + E)logV)
void dijkstra(int s) {

    dist[s] = 0;

    priority_queue<pair<int, int>> q;
    q.push({0, s});

    while (!q.empty()) {
        int a = q.top().second; q.pop();

        if (used[a]) continue;
        used[a] = true;

        for (auto [b, w] : adj[a]) {
            if (dist[a] + w < dist[b]) {
                dist[b] = dist[a] + w;
                parent[b] = a;
                q.push({-dist[b], b});
            }
        }
    }
}

//Complexidade: O(V)
vector<int> restorePath(int v) {
    vector<int> path;
    for (int u = v; u != -1; u = parent[u])
        path.push_back(u);
    reverse(path.begin(), path.end());
    return path;
}
```

5.3 Euler Tree

```
vector<vector<int>> adj(MAX);
vector<int> vis(MAX, 0);
vector<int> euTree(MAX);

void eulerTree(int u, int &index)
{
    vis[u] = 1;
    euTree[index++] = u;
    for (auto it : adj[u]) {
        if (!vis[it]) {
            eulerTree(it, index);
            euTree[index++] = u;
        }
    }
}

int main() {

    f(i,0,n-1) {
        int a, b; cin >> a >> b;
        adj[a].push_back(b);
        adj[b].push_back(a);
    }

    int index = 0;
    eulerTree(1, index);
}
```

5.4 Kruskal

```
// Kruskal
//
// Gera e retorna uma AGM e seu custo total a partir do vetor de arestas (edg)
// do grafo
//
// O(m log(m) + m a(m))

vector<tuple<int, int, int>> edg; // {peso,x,y}
vector<int> id, sz;

int find(int p){ // O(a(N)) amortizado
    return id[p] = (id[p] == p ? p : find(id[p]));
}

void uni(int p, int q) { // O(a(N)) amortizado
    p = find(p), q = find(q);
    if(p == q) return;
    if(sz[p] > sz[q]) swap(p,q);
    id[p] = q, sz[q] += sz[p];
}

pair<int, vector<tuple<int, int, int>>> kruskal() {

    sort(edg.begin(), edg.end());

    int cost = 0;
    vector<tuple<int, int, int>> mst; // opcional
    for (auto [w,x,y] : edg) if (find(x) != find(y)) {
        mst.emplace_back(w, x, y); // opcional
        cost += w;
        uni(x,y);
    }
    return {cost, mst};
}
```

5.5 Dfs

```

vector<int> adj[MAXN];
int visited[MAXN];

void dfs(int p) {
    memset(visited, 0, sizeof visited);
    stack<int> st;
    st.push(p);

    while (!st.empty()) {
        int curr = st.top();
        st.pop();
        if (visited[curr]==1)continue;
        visited[curr]=1;
        // process current node here

        for (auto i : adj[curr]) {
            st.push(i);
        }
    }
}
```

6 Matematica

6.1 Mdc Multiplo

```
// Calcula o mdc de varios numeros, ideal ser utilizado para n > 2

int mdc_many(vector<int> arr) {
   int result = arr[0];

   for (int& num : arr) {
       result = mdc(num, result);

       if(result == 1) return 1;
   }
   return result;
}
```

6.2 Factorial

```
unordered_map<int, int> memo;

int factorial(int n) {
    if (n == 0 || n == 1)  return 1;
    if (memo.find(n) != memo.end()) return memo[n];
    return memo[n] = n * factorial(n - 1);
}
```

6.3 Mmc Multiplo

```
// calcula mmc de varios numeros passados em um array, recomendado para n > 2

int mmc_many(vector<int> arr) {
   int result = arr[0];

   for(int& num : arr)
       result = (num * result / mmc(num, result ));
   return ans;
}
```

6.4 Fast Exponentiation

```
const int mod = 1e9+7;
int fexp(int a, int b)
{
    int ans = 1;
    while (b)
    {
        if (b & 1)
            ans = ans * a % mod;
        a = a * a % mod;
        b >>= 1;
    }
    return ans;
}
```

6.5 Sieve

```
// Crivo de óEratstenes para gerar primos éat um limite ’lim’
// Complexidade: O(n log log n), onde n é o limite
const int ms = 1e6 + 5;
bool notPrime[ms];   // notPrime[i] é verdadeiro se i ãno é um únmero primo
int primes[ms], qnt; // primes[] armazena os únmeros primos e qnt é a quantidade de primos encontrados

void sieve(int lim)
{
  primes[qnt++] = 1; // adiciona 1 como um únmero primo se ele for ávlido no problema
  for (int i = 2; i <= lim; i++)
  {
    if (notPrime[i])
      continue;                           // se i ãno é primo, pula
    primes[qnt++] = i;                    // i é primo, adiciona em primes[]
    for (int j = i + i; j <= lim; j += i) // marca todos os úmltiplos de i como ãno primos
      notPrime[j] = true;
  }
}
```

6.6 Fact Grande

```
#include <iostream>
#include <vector>

using namespace std;

vector<int> multiply(const vector<int>& num, int multiplier) {
    vector<int> result;
    int carry = 0;

    for (size_t i = 0; i < num.size(); ++i) {
        int product = num[i] * multiplier + carry;
        result.push_back(product % 10);
        carry = product / 10;
    }

    while (carry) {
        result.push_back(carry % 10);
        carry /= 10;
    }

    return result;
}

vector<int> factorial(int n) {
    vector<int> result;
    result.push_back(1);

    for (int i = 2; i <= n; ++i) {
        result = multiply(result, i);
    }

    reverse(result.begin(), result.end());
    return result;
}
```

6.7 Mdc

```
int mdc(int x, int y) {
    return y ? mdc(y, x % y) : abs(x);
}
```

6.8 Primo

```
bool prime(int a) {
    if (a == 1)
        return 0;
    for (int i = 2; i <= round(sqrt(a)); ++i)
        if (a % i == 0)
            return 0;
    return 1;
}
```

6.9 Miller Rabin

```
// Miinter-Rabin
//
// Testa se n eh primo, n <= 3 * 10^18
//
// O(log(n)), considerando multiplicacao
// e exponenciacao constantes

int mul(int a, int b, int m) {
    int ret = a*b - int((long double)1/m*a*b+0.5)*m;
    return ret < 0 ? ret+m : ret;
}

int pow(int x, int y, int m) {
    if (!y) return 1;
    int ans = pow(mul(x, x, m), y/2, m);
    return y%2 ? mul(x, ans, m) : ans;
}

bool prime(int n) {
    if (n < 2) return 0;
    if (n <= 3) return 1;
    if (n % 2 == 0) return 0;
    int r = __builtin_ctzint(n - 1), d = n >> r;

    // com esses primos, o teste funciona garantido para n <= 2^64
    // funciona para n <= 3*10^24 com os primos ate 41
    for (int a : {2, 325, 9375, 28178, 450775, 9780504, 795265022}) {
        int x = pow(a, d, n);
        if (x == 1 or x == n - 1 or a % n == 0) continue;

        for (int j = 0; j < r - 1; j++) {
            x = mul(x, x, n);
            if (x == n - 1) break;
        }
        if (x != n - 1) return 0;
    }
    return 1;
}
```

6.10 Fatorial Grande

```
void multiply(vector<int>& num, int x) {
   int carry = 0;
   for (int i = 0; i < num.size(); i++) {
       int prod = num[i] * x + carry;
       num[i] = prod % 10;
       carry = prod / 10;
   }
   while (carry != 0) {
       num.push_back(carry % 10);
       carry /= 10;
   }
}

vector<int> factorial(int n) {
   vector<int> result;
   result.push_back(1);
   for (int i = 2; i <= n; i++) {
       multiply(result, i);
   }
   return result;
}
```

6.11 Sieve Linear

```
// Sieve de Eratosthenes com linear sieve
// Encontra todos os únmeros primos no intervalo [2, N]
// Complexidade: O(N)

const int N = 10000000;
vector<int> lp(N + 1); // lp[i] = menor fator primo de i
vector<int> pr;        // vetor de primos

for (int i = 2; i <= N; ++i)
{
    if (lp[i] == 0)
    {
        lp[i] = i;
        pr.push_back(i);
    }
    for (int j = 0; i * pr[j] <= N; ++j)
    {
        lp[i * pr[j]] = pr[j];
        if (pr[j] == lp[i])
        {
            break;
        }
    }
}
```

6.12 Numeros Grandes

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

void normalize(vector<int>& num) {
    int carry = 0;
    for (int i = 0; i < num.size(); ++i) {
        num[i] += carry;
        carry = num[i] / 10;
        num[i] %= 10;
    }

    while (carry > 0) {
        num.push_back(carry % 10);
        carry /= 10;
    }
}

pair<int, vector<int>> makePair(int sign, const vector<int>& magnitude) {
    return {sign, magnitude};
}

pair<int, vector<int>> bigSum(const pair<int, vector<int>>& a, const pair<int, vector<int>>& b) {
    if (a.first == b.first) {
        vector<int> result(max(a.second.size(), b.second.size()), 0);
        transform(a.second.begin(), a.second.end(), b.second.begin(), result.begin(), plus<int>());
        normalize(result);
        return makePair(a.first, result);
    } else {
        // If signs are different, perform subtraction
        vector<int> result(max(a.second.size(), b.second.size()), 0);
        transform(a.second.begin(), a.second.end(), b.second.begin(), result.begin(), minus<int>());
        normalize(result);
        return makePair(a.first, result);
    }
}

pair<int, vector<int>> bigSub(const pair<int, vector<int>>& a, const pair<int, vector<int>>& b) {
    return bigSum(a, makePair(-b.first, b.second));
}

```

```cpp
pair<int, vector<int>> bigMult(const pair<int, vector<int>>& a, const pair<int, vector<int>>& b) {
    vector<int> result(a.second.size() + b.second.size(), 0);

    for (int i = 0; i < a.second.size(); ++i) {
        for (int j = 0; j < b.second.size(); ++j) {
            result[i + j] += a.second[i] * b.second[j];
        }
    }

    normalize(result);
    return makePair(a.first * b.first, result);
}



void printNumber(const pair<int, vector<int>>& num) {
    if (num.first == -1) {
        cout << '-';
    }

    for (auto it = num.second.rbegin(); it != num.second.rend(); ++it) {
        cout << *it;
    }
    cout << endl;
}

int main() {
    // Example usage
    pair<int, vector<int>> num1 = makePair(1, {1, 2, 3});  // Representing +321
    pair<int, vector<int>> num2 = makePair(-1, {4, 5, 6});  // Representing -654

    pair<int, vector<int>> sum = bigSum(num1, num2);
    pair<int, vector<int>> difference = bigSub(num1, num2);
    pair<int, vector<int>> product = bigMult(num1, num2);

    cout << "Sum: ";
    printNumber(sum);

    cout << "Difference: ";
    printNumber(difference);

    cout << "Product: ";
    printNumber(product);

    return 0;
}
```

6.13 Mmc

```cpp
int mmc(int x, int y) {
   return (x && y ? (return abs(x) / mdc(x, y) * abs(y)) : abs(x | y));
}
```

7 Strings

7.1 Ocorrencias

```cpp
/**
 *  @brief  str.find() aprimorado
 *  @param str  string to be analised
 *  @param sub  substring to be searched
 *  @return  vector<int> com indices de todas as êocorrncias de uma substring em uma string
 */
vector<int> ocorrencias(string str,string sub){

    vector<int> ret;
    int index = str.find(sub);
    while(index!=-1){
        ret.push_back(index);
        index = str.find(sub,index+1);
    }

    return ret;
}
```

7.2 Upper Case

```cpp
string to_upper(string a) {
   for (int i=0;i<(int)a.size();++i)
      if (a[i]>='a' && a[i]<='z')
         a[i]-='a'-'A';
   return a;
}

// para checar se e uppercase: isupper(c);
```

7.3 Palindromo

```cpp
bool isPalindrome(string str) {
    for (int i = 0; i < str.length() / 2; i++) {
        if (str[i] != str[str.length() - i - 1]) {
            return false;
        }
    }
    return true;
}
```

7.4 Split Cria

```cpp
vector<string> split(string s, string del = " ") {
   vector<string> retorno;
   int start, end = -1*del.size();
   do {
       start = end + del.size();
       end = s.find(del, start);
       retorno.push_back(s.substr(start, end - start));
   } while (end != -1);
```

```
    return retorno;
}
```

7.5 Remove Acento

```
string removeAcentro(string str) {

    string comAcento = "áéíóúâêôãõà";
    string semAcento = "aeiouaeoaoa";

    for(int i = 0; i < str.size(); i++){
        for(int j = 0; j < comAcento.size(); j++){
            if(str[i] == comAcento[j]){
                str[i] = semAcento[j];
                break;
            }
        }
    }

    return str;
}
```

7.6 Chaves Colchetes Parenteses

```
def balanced(string) -> bool:
    stack = []

    for i in string:
        if i in ’([{’: stack.append(i)

        elif i in ’)]}’:
            if (not stack) or ((stack[-1],i) not in [(’(’,’)’), (’[’,’]’), (’{’,’}’)]):
                return False
            else:
                stack.pop()

    return not stack
```

7.7 Lower Case

```
string to_lower(string a) {
   for (int i=0;i<(int)a.size();++i)
      if (a[i]>=’A’ && a[i]<=’Z’)
         a[i]+=’a’-’A’;
   return a;
}

// para checar se é lowercase: islower(c);
```

7.8 Lexicograficamente Minima

```
// A simple C++ program to find lexicographically minimum rotation of a given string
#include <iostream>
#include <algorithm>
using namespace std;

// This functionr return lexicographically minimum rotation of str
string minLexRotation(string str)
{
    // Find length of given string
    int n = str.length();

    // Create an array of strings to store all rotations
    string arr[n];

    // Create a concatenation of string with itself
    string concat = str + str;

    // One by one store all rotations of str in array.
    // A rotation is obtained by getting a substring of concat
    for (int i = 0; i < n; i++)
        arr[i] = concat.substr(i, n);

    // Sort all rotations
    sort(arr, arr+n);

    // Return the first rotation from the sorted array
    return arr[0];
}

// Driver program to test above function
int main()
{
    cout << minLexRotation("GEEKSFORGEEKS") << endl;
    cout << minLexRotation("GEEKSQUIZ") << endl;
    cout << minLexRotation("BCABDADAB") << endl;
}
```

7.9 Split

```
//split a string with a delimiter
//eg.: split("áOl, tudo bem?", " ") -> ["áOl,", "tudo", "bem?"]

vector<string> split(string in, string delimiter){
    vector<string> numbers;
    string token = "";
    int pos;
    while(true){
        pos = in.find(delimiter);
        if(pos == -1) break;
        token = in.substr(0, pos);
        numbers.push_back(token);
        in = in.erase(0, pos + delimiter.length());
    }
    numbers.push_back(in);
    return numbers;
}
```

8 Vector

8.1 Teste

8.2 Remove Repetitive

```
vector<int> removeRepetitive(const vector<int>& vec) {

    unordered_set<int> s;
    s.reserve(vec.size());

    vector<int> ans;

    for (int num : vec) {
        if (s.insert(num).second)
            v.push_back(num);
    }

    return ans;
}
```

8.3 Elemento Mais Frequente

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

// Encontra o unico elemento mais frequente em um vetor
// Complexidade: O(n)
int maxFreq1(vector<int> v) {
    int res = 0;
    int count = 1;

    for(int i = 1; i < v.size(); i++) {

        if(v[i] == v[res])
            count++;
        else
            count--;

        if(count == 0) {
            res = i;
            count = 1;
        }
    }

    return v[res];
}

// Encontra os elemento mais frequente em um vetor
// Complexidade: O(n)
vector<int> maxFreqn(vector<int> v)
{
    unordered_map<int, int> hash;
    for (int i = 0; i < v.size(); i++)
        hash[v[i]]++;

    int max_count = 0, res = -1;
    for (auto i : hash) {
        if (max_count < i.second) {
            res = i.first;
            max_count = i.second;
        }
    }

    vector<int> ans;
    for (auto i : hash) {
        if (max_count == i.second) {
            ans.push_back(i.first);
        }
    }

    return ans;
}
```

9 Outros

9.1 Binario

```
string decimal_to_binary(int dec) {
    string binary = "";
    while (dec > 0) {
        int bit = dec % 2;
        binary = to_string(bit) + binary;
        dec /= 2;
    }
    return binary;
}

int binary_to_decimal(string binary) {
    int dec = 0;
    int power = 0;
    for (int i = binary.length() - 1; i >= 0; i--) {
        int bit = binary[i] - ’0’;
        dec += bit * pow(2, power);
        power++;
    }
    return dec;
}
```

9.2 Horario

```
int cts(int h, int m, int s) {
    int total = (h * 3600) + (m * 60) + s;
    return total;
}

tuple<int, int, int> cth(int total_seconds) {
    int h = total_seconds / 3600;
    int m = (total_seconds % 3600) / 60;
    int s = total_seconds % 60;
    return make_tuple(h, m, s);
}
```

9.3 Max Subarray Sum

```
int maxSubarraySum(vector<int> x){

    int best = 0, sum = 0;
    for (int k = 0; k < n; k++) {
        sum = max(x[k],sum+x[k]);
        best = max(best,sum);
    }
    return best;
}
```

9.4 Binary Search

```
int BinarySearch(<vector>int arr, int x){
    int k = 0;
    int n = arr.size();

    for (int b = n/2; b >= 1; b /= 2) {
        while (k+b < n && arr[k+b] <= x) k += b;
    }
    if (arr[k] == x) {
        return k;
    }
}
```

9.5 Fibonacci

```
int fib(int n){
    if(n <= 1){
        return n;
    }
    return fib(n - 1) + fib(n - 2);
}
```