



Pedro Augusto Ulisses Andrade Lucas Andrade

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

Contents

1 Utils	2	4 Grafos	6
1.1 Makefile	2	4.1 Bfs	6
1.2 Mini Template Cpp	2	4.2 Dijkstra	6
1.3 Template Cpp	2	4.3 Kruskal	7
1.4 Files	3	4.4 Dfs	7
1.5 Template Python	3	5 Outros	7
2 Strings	3	5.1 Binarysearch	7
2.1 Ocorrencias	3	5.2 Hoursconvert	7
2.2 Chaves Colchetes Parenteses	4	5.3 Maxsubarraysum	7
2.3 Split	4	5.4 Fibonacci	8
2.4 Uppercase	4	5.5 Binaryconvert	8
2.5 Ispalindrome	4		
2.6 Lowercase	4		
3 Matematica	4		
3.1 Mdc Multiplo	4		
3.2 Mmc Multiplo	4		
3.3 Fast Exponentiation	4		
3.4 Sieve	5		
3.5 Miller-rabin	5		
3.6 Mdc	5		
3.7 Fatorial Grande	5		
3.8 Sieve Linear	5		
3.9 Mmc	6		

1 Utils

1.1 Makefile

```
1 CXX = g++
2 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
3
4 clear:
5     find . -maxdepth 1 -type f -executable -exec rm {} +
6
7 runc:
8     g++ -g $(f).cpp $(CPPFLAGS) -o $(f)
9     ./$(f)
10
11 runci:
12     g++ -g $(f).cpp $(CPPFLAGS) -o $(f)
13     ./$(f) < $(f).txt
14
15 runp:
16     python3 $(f).py
17
18 runpt:
19     python3 $(f).py < $(f).txt
```

1.2 Mini Template Cpp

```
1 #include <bits/stdc++.h>
2
3 using namespace std;
4
5 #define _ ios_base::sync_with_stdio(0); cin.tie(0);
6
7 #define int          long long int
8 #define double       long double
9 #define endl         "\n"
10 #define print_v(a)   for(auto x : a) cout << x << " "; cout << endl
11 #define f(i,s,e)     for(int i=s;i<e;i++)
12 #define rf(i,e,s)    for(int i=e-1;i>=s;i--)
13
14 #define dbg(x) cout << #x << " = " << x << endl;
15
16 void solve() {}
17
18 int32_t main() { _
19
20     int t = 1; // cin >> t;
21     while (t--)
22         //while(cin >> a >> b)
23             solve();
24
25
26     return 0;
27 }
```

1.3 Template Cpp

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 #define _ ios_base::sync_with_stdio(0); cin.tie(0);
5
6 #define int          long long int
7 #define double       long double
8 #define endl         "\n"
9 #define print_v(a)   for(auto x : a) cout << x << " "; cout << endl
10 #define print_vp(a)  for(auto x : a) cout << x.F << " " << x.S << endl
11 #define print2(a,x,y) for(int i = x; i < y; i++) cout<< a[i]<< " "; cout
12                       << endl
13 #define f(i,s,e)     for(int i=s;i<e;i++)
14 #define rf(i,e,s)    for(int i=e-1;i>=s;i--)
15
16 #define dbg(x) cout << #x << " = " << x << endl;
17 #define bug(...)    __f (__VA_ARGS__, __VA_ARGS__)
18
19 const int INF = 0x7f3f3f3f;
20 const int MAX = 1e8+10; // 10^6 + 10
21
22 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; }
23 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; }
24 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; }
25
26 template <typename Arg1> void __f (const char* name, Arg1&& arg1) { cout << name << " : " << arg1 << endl; }
27 template <typename Arg1, typename... Args> void __f (const char* names, Arg1&& arg1, Args&&... args) {
28     const char* comma = strchr (names + 1, ',');
29     cout.write (names, comma - names) << " : " << arg1 << " | "; __f (comma + 1, args...);
30 }
31
32 int a, b;
33 vector<vector<int>>> graph;
34 vector<bool> vis;
35
36 void solve() {
37
38 }
39
40 int32_t main() { _
41
42     clock_t z = clock();
43
44     int t = 1; // cin >> t;
45     while (t--)
46         //while(cin >> a >> b)
47             solve();
48 }
```

```

49     cerr << fixed << "Run Time : " << ((double)(clock() - z) /
CLOCKS_PER_SEC) << endl;
50     return 0;
51 }

```

1.4 Files

```

1 #!/bin/bash
2
3 for c in {a..f}; do
4     cp temp.cpp "$c.cpp"
5     echo "$c" > "$c.txt"
6     if [ "$c" = "$letter" ]; then
7         break
8     fi
9 done

```

1.5 Template Python

```

1 import sys
2 import math
3 import bisect
4 from sys import stdin, stdout
5 from math import gcd, floor, sqrt, log
6 from collections import defaultdict as dd
7 from bisect import bisect_left as bl, bisect_right as br
8
9 sys.setrecursionlimit(100000000)
10
11 inp = lambda: int(input())
12 strng = lambda: input().strip()
13 jn = lambda x, l: x.join(map(str, l))
14 strl = lambda: list(input().strip())
15 mul = lambda: map(int, input().strip().split())
16 mulf = lambda: map(float, input().strip().split())
17 seq = lambda: list(map(int, input().strip().split()))
18
19 ceil = lambda x: int(x) if (x == int(x)) else int(x) + 1
20 ceildiv = lambda x, d: x // d if (x % d == 0) else x // d + 1
21
22 flush = lambda: stdout.flush()
23 stdstr = lambda: stdin.readline()
24 stdint = lambda: int(stdin.readline())
25 stdpr = lambda x: stdout.write(str(x))
26
27 mod = 1000000007
28
29 #main code
30
31 a = None
32 b = None
33 lista = None
34
35 def ident(*args):
36     if len(args) == 1:
37         return args[0]

```

```

38     return args
39
40
41 def parsin(*, l=1, vpl=1, s=" "):
42     if l == 1:
43         if vpl == 1: return ident(input())
44         else: return list(map(ident, input().split(s)))
45     else:
46         if vpl == 1: return [ident(input()) for _ in range(l)]
47         else: return [list(map(ident, input().split(s))) for _ in range(l)]
48
49
50 def solve():
51     pass
52
53 # if __name__ == '__main__':
54 def main():
55     st = clk()
56
57     escolha = "in"
58     #escolha = "num"
59
60     match escolha:
61         case "in":
62             # @1 infinitas linhas agrupadas de 2 em 2
63             # pra infinitos valores em 1 linha pode armazenar em uma lista
64             while True:
65                 global a, b
66                 try: a, b = input().split()
67                     except (EOFError): break #permite ler todas as linhas
68             dentro do .txt
69                 except (ValueError): pass # consegue ler éat linhas em
70             branco
71                 else:
72                     a, b = int(a), int(b)
73                     solve()
74
75         case "num":
76             global lista
77             # int l; cin >> l; while(l--){for(i=0; i<vpl; i++){
78             # retorna listas com inputs de cada linha
79             # leia l linhas com vpl valores em cada uma delas
80             # caseo seja mais de uma linha, retorna lista com listas
81
82     de inputs
83         lista = parsin(l=2, vpl=5)
84         solve()
85
86     sys.stderr.write(f"Run Time : {(clk() - st):.6f} seconds\n")
87
88 main()

```

2 Strings

2.1 Ocorrencias

```

1 /**
2  * @brief str.find() aprimorado
3  * @param str string to be analysed
4  * @param sub substring to be searched
5  * @return vector<int> com indices de todas as @ocorrncias de uma
   substring em uma string
6  */
7 vector<int> ocorrencias(string str,string sub){
8
9     vector<int> ret;
10    int index = str.find(sub);
11    while(index!=-1){
12        ret.push_back(index);
13        index = str.find(sub,index+1);
14    }
15
16    return ret;
17 }

```

2.2 Chaves Colchetes Parenteses

```

1 def balanced(string) -> bool:
2     stack = []
3
4     for i in string:
5         if i in '([{': stack.append(i)
6
7         elif i in ')]}':
8             if (not stack) or ((stack[-1],i) not in [('(',')'), ('[',']'),
9                ('{','}')]):
10                 return False
11             else:
12                 stack.pop()
13
14     return not stack

```

2.3 Split

```

1 //split a string with a delimiter
2 //eg.: split("â0l, tudo bem?", " ") -> ["â0l,", "tudo", "bem?"]
3
4 vector<string> split(string in, string delimiter){
5     vector<string> numbers;
6     string token = "";
7     int pos;
8     while(true){
9         pos = in.find(delimiter);
10        if(pos == -1) break;
11        token = in.substr(0, pos);
12        numbers.push_back(token);
13        in = in.erase(0, pos + delimiter.length());
14    }
15    numbers.push_back(in);
16    return numbers;
17 }

```

2.4 Uppercase

```

1 string to_upper(string a) {
2     for (int i=0;i<(int)a.size();++i)
3         if (a[i]>='a' && a[i]<='z')
4             a[i]-='a'-'A';
5     return a;
6 }

```

2.5 Ispalindrome

```

1 bool isPalindrome(string S){
2     string P = S;
3     reverse(P.begin(), P.end()); // Reverte P
4     return (S == P); //retorna true se verdadeiro, false se falso
5 }

```

2.6 Lowercase

```

1 string to_lower(string a) {
2     for (int i=0;i<(int)a.size();++i)
3         if (a[i]>='A' && a[i]<='Z')
4             a[i]+='a'-'A';
5     return a;
6 }

```

3 Matematica

3.1 Mdc Multiplo

```

1 int mdc_many(vector<int> arr) {
2     int result = arr[0];
3     for (size_t i = 1; i < arr.size(); i++) {
4         result = mdc(arr[i], result);
5
6         if(result == 1)
7             return 1;
8     }
9     return result;
10 }

```

3.2 Mmc Multiplo

```

1 int mmc(vector<int> arr) {
2     int result = arr[0];
3     for(size_t i = 1; i < arr.size(); i++)
4         result = (arr[i] * result / mmc_util(arr[i], result ));
5     return ans;
6 }

```

3.3 Fast Exponentiation

```

1 const int mod = 1e9+7;
2 int fexp(int a, int b)
3 {
4     int ans = 1;

```

```

5 while (b)
6 {
7     if (b & 1)
8         ans = ans * a % mod;
9     a = a * a % mod;
10    b >>= 1;
11 }
12 return ans;
13 }

```

3.4 Sieve

```

1 // Crivo de Eratstenes para gerar primos até um limite 'lim'
2 // Complexidade: O(n log log n), onde n é o limite
3 const int ms = 1e6 + 5;
4 bool notPrime[ms]; // notPrime[i] é verdadeiro se i não é um número
    primo
5 int primes[ms], qnt; // primes[] armazena os números primos e qnt é a
    quantidade de primos encontrados
6
7 void sieve(int lim)
8 {
9     primes[qnt++] = 1; // adiciona 1 como um número primo se ele for válido
        no problema
10    for (int i = 2; i <= lim; i++)
11    {
12        if (notPrime[i])
13            continue; // se i não é primo, pula
14        primes[qnt++] = i; // i é primo, adiciona em primes
15        for (int j = i + i; j <= lim; j += i) // marca todos os múltiplos de i
            notPrime[j] = true; // como não primos
16    }
17 }
18 }

```

3.5 Miller-rabin

```

1 // Miller-Rabin
2 //
3 // Testa se n eh primo, n <= 3 * 10^18
4 //
5 // O(log(n)), considerando multiplicacao
6 // e exponenciacao constantes
7
8 int mul(int a, int b, int m) {
9     int ret = a*b - int((long double)1/m*a*b+0.5)*m;
10    return ret < 0 ? ret+m : ret;
11 }
12
13 int pow(int x, int y, int m) {
14     if (!y) return 1;
15     int ans = pow(mul(x, x, m), y/2, m);
16     return y%2 ? mul(x, ans, m) : ans;
17 }
18 }

```

```

19 bool prime(int n) {
20     if (n < 2) return 0;
21     if (n <= 3) return 1;
22     if (n % 2 == 0) return 0;
23     int r = __builtin_ctzint(n - 1), d = n >> r;
24
25     // com esses primos, o teste funciona garantido para n <= 2^64
26     // funciona para n <= 3*10^24 com os primos ate 41
27     for (int a : {2, 325, 9375, 28178, 450775, 9780504, 795265022}) {
28         int x = pow(a, d, n);
29         if (x == 1 or x == n - 1 or a % n == 0) continue;
30
31         for (int j = 0; j < r - 1; j++) {
32             x = mul(x, x, n);
33             if (x == n - 1) break;
34         }
35         if (x != n - 1) return 0;
36     }
37     return 1;
38 }

```

3.6 Mdc

```

1 int mdc(int x, int y) {
2     return y ? mdc(y, x % y) : abs(x);
3 }

```

3.7 Fatorial Grande

```

1 void multiply(vector<int>& num, int x) {
2     int carry = 0;
3     for (int i = 0; i < num.size(); i++) {
4         int prod = num[i] * x + carry;
5         num[i] = prod % 10;
6         carry = prod / 10;
7     }
8     while (carry != 0) {
9         num.push_back(carry % 10);
10        carry /= 10;
11    }
12 }
13
14 vector<int> factorial(int n) {
15     vector<int> result;
16     result.push_back(1);
17     for (int i = 2; i <= n; i++) {
18         multiply(result, i);
19     }
20     return result;
21 }

```

3.8 Sieve Linear

```

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

```

```

4
5 const int N = 10000000;
6 vector<int> lp(N + 1); // lp[i] = menor fator primo de i
7 vector<int> pr;        // vetor de primos
8
9 for (int i = 2; i <= N; ++i)
10 {
11     if (lp[i] == 0)
12     {
13         lp[i] = i;
14         pr.push_back(i);
15     }
16     for (int j = 0; i * pr[j] <= N; ++j)
17     {
18         lp[i * pr[j]] = pr[j];
19         if (pr[j] == lp[i])
20         {
21             break;
22         }
23     }
24 }

```

3.9 Mmc

```

1 int mmc(int x, int y) {
2     return (x && y ? (return abs(x) / mdc(x, y) * abs(y)) : abs(x | y));
3 }

```

4 Grafos

4.1 Bfs

```

1 // BFS com informacoes adicionais sobre a distancia e o pai de cada
  vertice
2 // Complexidade: O(V + E), onde V eh o numero de vertices e E o numero de
  areqas
3 vector<vector<int>>> adj; // liqa de adjacencia
4 int n, s; // n = numero de vertices, s = vertice inicial
5
6 vector<bool> used(n);
7 vector<int> d(n), p(n);
8
9 void bfs(int s) {
10     queue<int> q;
11     q.push(s);
12     used[s] = true;
13     d[s] = 0;
14     p[s] = -1;
15
16     while (!q.empty()) {
17         int v = q.front();
18         q.pop();
19         for (int u : adj[v]) {
20             if (!used[u]) {
21                 used[u] = true;

```

```

22         q.push(u);
23         d[u] = d[v] + 1;
24         p[u] = v;
25     }
26 }
27 }
28 }
29
30 //pra uma bfs que n guarda o backtracking:
31 void bfs(int p) {
32     memset(visited, 0, sizeof visited);
33     queue<int> q;
34     q.push(p);
35
36     while (!q.empty()) {
37         int curr = q.top();
38         q.pop();
39         if (visited[curr]==1) continue;
40         visited[curr]=1;
41         // process current node here
42
43         for (auto i : adj[curr]) {
44             q.push(i);
45         }
46     }
47 }
48 }

```

4.2 Dijkstra

```

1 vector<vector<pair<int, int>>> adj;
2 int n, s;
3
4 vector<int> d(n, LLINF);
5 vector<int> p(n, -1);
6 vector<bool> used(n);
7
8 //Complexidade: O((V + E)logV)
9 void dijkstra(int s) {
10     d[s] = 0;
11     priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<
12     int, int>>> q;
13     q.push({0, s});
14     while (!q.empty()) {
15         int v = q.top().second;
16         q.pop();
17         if (used[v]) continue;
18         used[v] = true;
19         for (auto edge : adj[v]) {
20             int to = edge.first, len = edge.second;
21             if (d[v] + len < d[to]) {
22                 d[to] = d[v] + len;
23                 p[to] = v;
24                 q.push({d[to], to});
25             }
26         }
27     }
28 }

```

```

27 }
28
29 //Complexidade: O(V)
30 vector<int> restorePath(int v) {
31     vector<int> path;
32     for (int u = v; u != -1; u = p[u])
33         path.push_back(u);
34     reverse(path.begin(), path.end());
35     return path;
36 }

```

4.3 Kruskal

```

1 //vector<pair<int,int>> arestas[MAXN] em que cada aresta[i] contem o peso
  e o vertice adjacente
2 //vector<peso,conexao>
3 vector<pair<int,int>> adj[MAXN];
4 vector<pair<int,int>> adjtree[MAXN];
5 vector<pair<int, pair<int, int>>> kruskadj;
6 int cost;
7 void kruskal(){
8     for(int i = 1;i<MAXN;i++){
9         for(auto j:adj[i]){
10             kruskadj.push_back({j.first,{i,j.second}});
11         }
12     }
13     sort(kruskadj.begin(),kruskadj.end());
14     cost=0;
15     int r = kruskadj.size();
16     vector<int> id(r);
17     for (int i = 0; i < r; i++) id[i] = i;
18     for (auto p : kruskadj){
19         int x = p.second.first;
20         int y = p.second.second;
21         int w = p.first;
22         if (id[x] != id[y]){
23             cost += w;
24             adjtree[x].push_back({w,y});
25             int old_id = id[x], new_id = id[y];
26             for (int i = 0; i < r; i++)
27                 if (id[i] == old_id) id[i] = new_id;
28         }
29     }
30 }
31 }

```

4.4 Dfs

```

1
2 vector<int> adj[MAXN];
3 int visited[MAXN];
4
5 void dfs(int p) {
6     memset(visited, 0, sizeof visited);
7     stack<int> st;
8     st.push(p);

```

```

9
10 while (!st.empty()) {
11     int curr = st.top();
12     st.pop();
13     if (visited[curr]==1) continue;
14     visited[curr]=1;
15     // process current node here
16
17     for (auto i : adj[curr]) {
18         st.push(i);
19     }
20 }
21 }
22 }

```

5 Outros

5.1 Binarysearch

```

1 int BinarySearch(<vector>int arr, int x){
2     int k = 0;
3     int n = arr.size();
4
5     for (int b = n/2; b >= 1; b /= 2) {
6         while (k+b < n && arr[k+b] <= x) k += b;
7     }
8     if (arr[k] == x) {
9         return k;
10    }
11 }

```

5.2 Hoursconvert

```

1 int cts(int h, int m, int s) {
2     int total = (h * 3600) + (m * 60) + s;
3     return total;
4 }
5
6 tuple<int, int, int> cth(int total_seconds) {
7     int h = total_seconds / 3600;
8     int m = (total_seconds % 3600) / 60;
9     int s = total_seconds % 60;
10    return make_tuple(h, m, s);
11 }

```

5.3 Maxsubarraysum

```

1 int maxSubarraySum(vector<int> x){
2
3     int best = 0, sum = 0;
4     for (int k = 0; k < n; k++) {
5         sum = max(x[k],sum+x[k]);
6         best = max(best,sum);
7     }
8     return best;
9 }

```

5.4 Fibonacci

```
1 int fib(int n){
2     if(n <= 1){
3         return n;
4     }
5     return fib(n - 1) + fib(n - 2);
6 }
```

5.5 Binaryconvert

```
1 string decimal_to_binary(int dec) {
2     string binary = "";
3     while (dec > 0) {
4         int bit = dec % 2;
5         binary = to_string(bit) + binary;
```

```
6         dec /= 2;
7     }
8     return binary;
9 }
10
11 int binary_to_decimal(string binary) {
12     int dec = 0;
13     int power = 0;
14     for (int i = binary.length() - 1; i >= 0; i--) {
15         int bit = binary[i] - '0';
16         dec += bit * pow(2, power);
17         power++;
18     }
19     return dec;
20 }
```