Pedro Augusto Ulisses Andrade Lucas Andrade

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

Contents

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

clear:
    find . -maxdepth 1 -type f -executable -exec rm {} +

runc:
    g++ -g $(f).cpp $(CPPFLAGS) -o $(f)
    ./$(f)

runci:
    g++ -g $(f).cpp $(CPPFLAGS) -o $(f)
    ./$(f) < $(f).txt

runp:
    python3 $(f).py

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

1.2 Mini Template Cpp

```cpp
#include <bits/stdc++.h>

using namespace std;

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

#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--)
    //while(cin >> a >> b)
        solve();


    return 0;
}
```

1.3 Template Cpp

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

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

#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 print_vp(a)     for(auto x : a) cout << x.F << " " << x.S << endl
#define print2(a,x,y)   for(int i = x; i < y; i++) cout<< a[i]<< " "; 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;
#define bug(...)        __f (#__VA_ARGS__, __VA_ARGS__)

const int INF = 0x7f3f3f3f;
const int MAX = 1e8+10; // 10^6 + 10

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; }
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; }
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; }

template <typename Arg1> void __f (const char* name, Arg1&& arg1) { cout << name << " : " << arg1 << endl; }
template <typename Arg1, typename... Args> void __f (const char* names, Arg1&& arg1, Args&&... args) {
    const char* comma = strchr (names + 1, ',');
    cout.write (names, comma - names) << " : " << arg1 << " | "; __f (comma + 1, args...);
}

int a, b;
vector<vector<int>> graph;
vector<bool> vis;

void solve() {

}

int32_t main() { _

    clock_t z = clock();

    int t = 1; // cin >> t;
    while (t--)
    //while(cin >> a >> b)
        solve();


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

1.4 Template 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 Strings

2.1 Ocorrencias

```
/**
 *  @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;
}
```

2.2 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
```

2.3 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;
}
```

2.4 Uppercase

```
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;
}
```

2.5 Ispalindrome

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

2.6 Lowercase

```
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;
}
```

3 Matematica

3.1 Mdc Multiplo

```
int mdc_many(vector<int> arr) {
   int result = arr[0];
   for (size_t i = 1; i < arr.size(); i++) {
       result = mdc(arr[i], result);

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

3.2 Mmc Multiplo

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

3.3 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;
}
```

3.4 Sieve

```cpp
// 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;
  }
}
```

3.5 Miller-rabin

```cpp
// 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;
}
```

3.6 Mdc

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

3.7 Fatorial Grande

```cpp
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;
}
```

3.8 Sieve Linear

```cpp
// 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;
        }
    }
}
```

3.9 Mmc

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

4 Grafos

4.1 Bfs

```
// 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
    arestas
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);
        }

    }
}
```

4.2 Dijkstra

```
vector<vector<pair<int, int>>> adj;
int n, s;

vector<int> d(n, LLINF);
vector<int> p(n, -1);
vector<bool> used(n);

//Complexidade: O((V + E)logV)
void dijkstra(int s) {
    d[s] = 0;
    priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<
    int, int>>> q;
    q.push({0, s});
    while (!q.empty()) {
        int v = q.top().second;
        q.pop();
        if (used[v]) continue;
        used[v] = true;
        for (auto edge : adj[v]) {
            int to = edge.first, len = edge.second;
            if (d[v] + len < d[to]) {
                d[to] = d[v] + len;
                p[to] = v;
                q.push({d[to], to});
            }
        }
    }
}

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

4.3 Kruskal

```cpp
//vector<pair<int,int>> arestas[MAXN] em que cada aresta[i] contem o peso
    e o vertice adjacente
//vector<peso,conexao>
vector<pair<int,int>> adj[MAXN];
vector<pair<int,int>> adjtree[MAXN];
vector<pair<int, pair<int, int>>> kruskadj;
int cost;
void kruskal(){
    for(int i = 1;i<MAXN;i++){
        for(auto j:adj[i]){
            kruskadj.push_back({j.first,{i,j.second}});
        }
    }
    sort(kruskadj.begin(),kruskadj.end());
    cost=0;
    int r = kruskadj.size();
    vector<int> id(r);
    for (int i = 0; i < r; i++) id[i] = i;
    for (auto p : kruskadj){
        int x = p.second.first;
        int y = p.second.second;
        int w = p.first;
        if (id[x] != id[y]){
            cost += w;
            adjtree[x].push_back({w,y});
            int old_id = id[x], new_id = id[y];
            for (int i = 0; i < r; i++)
                if (id[i] == old_id) id[i] = new_id;
        }
    }

}
```

4.4 Dfs

```cpp

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);
        }

    }
}
```

5 Outros

5.1 Binarysearch

```cpp
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;
    }
}
```

5.2 Hoursconvert

```cpp
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);
}
```

5.3 Maxsubarraysum

```cpp
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;
}
```

5.4 Fibonacci

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

5.5 Binaryconvert

```cpp
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;
}
```