Capivalivro

Lucas Joviniano

Version 1.0, 07/10/2022

Table of Contents

Introdução

Troubleshoot

Antes de Enviar

- Escreva testes simples se os fornecidos não são suficientes
- O tempo limite é muito curto? Se sim, escreva um caso de teste de tamanho máximo.
- O uso de memória está no limite?
- Algo pode dar overflow?
- Tenha certeza de que está enviando o arquivo certo

Wrong Answer

- Imprima sua solução! Imprima também os resultados intermediários.
- Limpou todas as estruturas de dados entre os casos de teste?
- Seu algoritmo trata toda a entrada?
- Leia o problema inteiro de novo.
- Todos os casos especiais são tratados?
- Você entendeu o problema corretamente?
- Alguma variável não inicializada?
- Algum overflow?
- Não está confundindo M e N, i e j, etc.?
- Tem certeza que seu algoritmo funciona?
- Quais os casos especiais que você pode não ter pensado?
- Tem certeza que as funções da STL funcionam como você espera?
- Adicione alguns asserts, talvez tente enviar novamente.
- Crie alguns casos de teste para passar pelo algoritmo.
- Explique seu algoritmo para um colega de time.
- Peça para o colega de time olhar seu código.
- Vai dar uma volta
- Está imprimindo corretamente?
- Reescreva toda sua solução ou peça para alguém fazer isso.

Runtime Error

- Testou todos os casos especiais localmente?
- Alguma variável não inicializada?
- Escrevendo ou lendo fora do tamanho de algum vetor?
- Algum assert que possa falhar?

- Alguma possível divisão por 0? (Ou mod 0)
- Alguma possível recursão infinita
- Ponteiros ou iteradores invalidados?
- Usando muita memória?
- Debug enviando novamente

Time Limit Exceeded

- Algum possível loop infinito?
- Qual a complexidade do seu algoritmo?
- Copiando muitos dados desnecessariamente? (Use referências)
- A entrada ou a saída não são muito grandes? (Considere usar scanf)
- Evite vector e map
- O que seus colegas acham do algoritmo?

Memory Limit Exceeded

- Qual o máximo de memória que seu algoritmo deveria precisar?
- Limpou todas as estruturas de dados entre casos de teste?

C++ e STL

template.cpp

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

using namespace std;
#define MOD 1000000007
#define EPS 1e-9
#define pb push_back
#define pf push_front
#define fi first
#define se second
#define mp make_pair
#define all(x) x.begin(), x.end()

template<typename A, typename B>
ostream &operator<<(ostream &s, const pair <A, B> &p) {
    return s << "(" << p.fi << ", " << p.se << ")";
}

template<typename T>
ostream &operator<<(ostream &s, const vector <T> &v) {
    s << "[";
    for (auto it: c) s << it << " ";
    s << "]";
    return s;
}

int main() {
    cin.tie(0)->sync_with_stdio(0);
    cin.exceptions(cin.failbit);
}
```

Estruturas de Dados

Fenwick Tree

Descrição: Calcula soma de prefixos em um array.

```
const int neutral = 0;
#define comp(a, b) ((a)+(b))

// ATENÇÃO: Indexa do 1
class FenwickTree {
private:
    vector<int> ft;
public:
    FenwickTree(int n) {
        ft.assign(n + 1, 0);
    }

    int rsq(int i) { // Retorna RSQ(1, i)
        int sum = neutral;
        for (; i; i -= (i & -i)) sum = comp(sum, ft[i]);
        return sum;
    }

    int rsq(int i, int j) {
        return rsq(j) - rsq(i - 1);
    }

    void update(int i, int v) {
        for (; i < (int)ft.size(); i += (i & -i))
            ft[i] = comp(v, ft[i]);
    }
};
```

Segment Tree

Descrição: Permite responder queries em intervalos de um array de forma eficiente, incluindo:

- Achar a soma de elementos consecutivos
- Achar o menor elemento em um range

```
const int neutral = 0;
#define comp(a, b) ((a)+(b))

class SegmentTree {
    vector<int> a;
    int n;
```

```
public:
    SegmentTree(int *st, int *en) {
        int sz = int(en - st);

        for (n = 1; n < sz; n <<= 1);
        a.assign(n << 1, neutral);
        for (int i = 0; i < sz; i++) a[i + n] = st[i];
        for (int i = n + sz - 1; i > 1; i--) {
            a[i >> 1] = comp(a[i >> 1], a[i]);
        }
    }

    void update(int i, int x) {
        a[i += n] = x;
        for (i >>= 1; i; i >>= 1) {
            a[i] = comp(a[i << 1], a[1 + (i << 1)]);
        }
    }

    int query(int l, int r) {
        int ans = neutral;
        for (l += n, r += n + 1; l < r; l >>= 1, r >>= 1) {
            if (l & 1) ans = comp(ans, a[l++]);
            if (r & 1) ans = comp(ans, a[--r]);
        }
        return ans;
    }
};
```