

# Desafios de Programação

## Introdução

**Wladimir Araújo Tavares**<sup>1</sup>

<sup>1</sup>Universidade Federal do Ceará - Campus de Quixadá

13 de março de 2017

1 Por que fazer Desafios de Programação?

2 Tópicos

3 Bibliografia

4 Competições

5 Como praticar

6 Competições semanais

7 Problema 1

8 Problema 2

9 Dicas para ser Competitivo

- Dica 5: Domine a arte de testar

# Por que fazer Desafios de Programação?

- Você pode aprender
  - ▶ Muitos algoritmos úteis e vários insights matemáticos
  - ▶ Como programar/debugar rapidamente
  - ▶ Como trabalhar em equipe
- Aumentar a capacidade de avaliação de tempo e memória de algoritmos.
- Um treinamento para entrevistas de empregos.

# Pré-requisitos

- QXD0010 - Estrutura de Dados
- QXD0041 - Projeto e Análise de Algoritmos
- Bom conhecimento matemático
- Avidéz por conhecimento

1 Por que fazer Desafios de Programação?

2 Tópicos

3 Bibliografia

4 Competições

5 Como praticar

6 Competições semanais

7 Problema 1

8 Problema 2

9 Dicas para ser Competitivo

- Dica 5: Domine a arte de testar

# Tópicos

- 1 Estrutura de Dados
- 2 Matemáticos
- 3 Backtracking
- 4 Programação Dinâmica
- 5 Algoritmos em Grafos
- 6 Fluxos em redes
- 7 Algoritmos para IA
- 8 Algoritmos em cadeias de caracteres
- 9 Algoritmos Geométricos
- 10 Jogos Combinatórios

- 1 Por que fazer Desafios de Programação?
- 2 Tópicos
- 3 Bibliografia**
- 4 Competições
- 5 Como praticar
- 6 Competições semanais
- 7 Problema 1
- 8 Problema 2
- 9 Dicas para ser Competitivo
  - Dica 5: Domine a arte de testar





- 1 Por que fazer Desafios de Programação?
- 2 Tópicos
- 3 Bibliografia
- 4 Competições**
- 5 Como praticar
- 6 Competições semanais
- 7 Problema 1
- 8 Problema 2
- 9 Dicas para ser Competitivo
  - Dica 5: Domine a arte de testar

# Competições

- Maratona de Programação
- TopCoder
- Codeforces
- Google Code Jam

1 Por que fazer Desafios de Programação?

2 Tópicos

3 Bibliografia

4 Competições

5 Como praticar

6 Competições semanais

7 Problema 1

8 Problema 2

9 Dicas para ser Competitivo

- Dica 5: Domine a arte de testar

# Como praticar

- URI Online Judge
- SPOJ
- Project Euler
- Code Chef
- Competições semanais

1 Por que fazer Desafios de Programação?

2 Tópicos

3 Bibliografia

4 Competições

5 Como praticar

6 Competições semanais

7 Problema 1

8 Problema 2

9 Dicas para ser Competitivo

- Dica 5: Domine a arte de testar

# Competições semanais

- Toda às quarta-feira
- Aberto para qualquer pessoa
- Pelo URI Online Judge

1 Por que fazer Desafios de Programação?

2 Tópicos

3 Bibliografia

4 Competições

5 Como praticar

6 Competições semanais

**7 Problema 1**

8 Problema 2

9 Dicas para ser Competitivo

- Dica 5: Domine a arte de testar

# Problema 1

Dado  $N-1$  números distintos entre 1 e  $N$ , descubra o número que está faltando?

Restrições:

$$N \leq 1000000$$



# Solução 1

```
#include <stdio.h>
#include <string.h>
#define MAX 1000000
char table[MAX]; //~1MB
int N,x;
int main(){
    scanf("%d", &N);
    memset(table, sizeof(table), 0);
    for(int i = 0; i < N-1; i++){
        scanf("%d", &x);
        table[--x] = 1;
    }
    for(int i = 0; i < N; i++)
        if(table[i] == 0)
            printf("%d\n", i+1);
}
```

## Solução 2

```
#include <stdio.h>
#include <string.h>
int S,N,x;
int main(){
    scanf("%lld", &N);
    S = ((N+1)*N)/2LL;
    //printf("%lld\n", S);
    for(int i = 0; i < N-1;i++)
    {
        scanf("%lld", &x);
        S = S - x;
    }
    printf("%lld\n", S);
}
```

## Solução 2

- A variável **int** é capaz de representar o intervalo  $[-2.147.483.647, +2.147.483.647]$
- Para  $N=1000000$ , o valor de  $S$  será  $-363189984$ .
- Precisamos um tipo de dado com maior poder de representação.
- **long long int** capaz de representar o intervalo  $[-9.223.372.036.854.775.807, +9.223.372.036.854.775.807]$

## Solução 3

```
#include <stdio.h>
#include <string.h>
long long int S,N,x;
int main(){
    scanf("%lld", &N);
    S = ((N+1)*N)/2LL;
    printf("%lld\n", S);
    for(int i = 0; i < N-1;i++)
    {
        scanf("%lld", &x);
        S = S - x;
    }
    printf("%lld\n", S);
}
```

# Solução 4 XOR Ring

Propriedades  $\oplus$ :

- $A \oplus A = 0$  (Elemento Inverso)
- $A \oplus 0 = A$  (Elemento Neutro)
- $A \oplus B = B \oplus A$  (Comutatividade)

Algoritmo:

- 1 Faça  $S \leftarrow 1 \oplus 2 \oplus \dots \oplus N$ .
- 2 Para cada valor dado  $x$ , faça  $S \leftarrow S \oplus x$ .
- 3 Imprima  $S$ .

## Solução 4

```
#include <stdio.h>
int S,N,x;
int main(){
    scanf("%d", &N);
    S = 0;
    for(int i = 1; i <= N; i++){
        S = S ^ i;
    }
    for(int i = 1; i <= N-1; i++){
        scanf("%d", &x);
        S = S ^ x;
    }
    printf("%d\n", S);
}
```

- 1 Por que fazer Desafios de Programação?
- 2 Tópicos
- 3 Bibliografia
- 4 Competições
- 5 Como praticar
- 6 Competições semanais
- 7 Problema 1
- 8 Problema 2**
- 9 Dicas para ser Competitivo
  - Dica 5: Domine a arte de testar

## Problema 2

Dado  $N+1$  números no intervalo  $[1..N]$ , somente um valor está duplicado, descubra qual é o valor duplicado.

**Exemplo de Entrada**  $N = 5$ , vetor de valores = 1, 3, 4, 3, 5, 2.

**Exemplo de Saída** duplicado = 3

Restrições:

$N \leq 1000000$



- 1 Por que fazer Desafios de Programação?
- 2 Tópicos
- 3 Bibliografia
- 4 Competições
- 5 Como praticar
- 6 Competições semanais
- 7 Problema 1
- 8 Problema 2
- 9 Dicas para ser Competitivo**
  - Dica 5: Domine a arte de testar

# Dica 1: Digite rápido

- Typing test: <http://www.typingtest.com/>
- Typing Speed: 44 WPM
- Errors: 5 mistyped words
- Adjusted Speed: 39 WPM

## Dica 2: Identifique rapidamente o tipo de problema

### Categorias

---

Ad Hoc

Força Bruta

Divisão e conquista

Guloso

Programação Dinâmica

Grafos

Matemática

Processamento de String

Geometria Computacional

---

[Tabela](#) Tipos de Problemas

## Dica 3: Domine Análise do algoritmo

- Computadores modernos realizam em torno de  $\approx 100M(10^8)$  por segundos.
- Se o tamanho máximo de sua entrada é  $\approx 100K(10^5)$  e seu algoritmo tem complexidade  $O(n^2)$  então seu algoritmo realiza  $10^{10}$  operações.
- Isso significa que seu algoritmo requer na ordem de centenas de segundos para resolver o problema.
- Se seu algoritmo tem complexidade  $O(n \log_2 n)$  então seu algoritmo realiza  $1.7 \times 10^6$  então seu algoritmo consegue rodar em menos de 1 segundo.

Tamanho da Entrada	Complexidade do Algoritmo
$\leq 10$	$O(n!)$
$\leq 20$	$O(2^n)$
$\leq 50$	$O(n^4)$
$\leq 100$	$O(n^3)$
$\leq 10^3$	$O(n^2)$
$\leq 10^5$	$O(n \log_2 n)$
$\leq 10^6$	$O(n), O(\log_2 n)$

Tabela Análise do Algoritmo

# Exercício 1

Existem  $n$  páginas na internet ( $1 \leq n \leq 10^7$ ). Cada página  $i$  tem um page rank diferente  $r_i$ . Encontre as 10 páginas com os maiores page rank. Qual método é mais viável?

- 1 Leia  $n$  páginas e ordene e escolha 10 maiores.
- 2 Use uma fila de prioridade (heap).

## Exercício 2

Dada uma lista  $L$  com até  $10^4$  inteiros, você quer saber freqüentemente o valor  $sum(i, j) = L[i] + \dots + L[j]$ . Qual é a estrutura de dados viável?

- ❶ Um vetor simples.
- ❷ Um vetor simples com pré-processamento.
- ❸ Um segtree

## Exercício 3

Receba um vetor  $A[0..N - 1]$  e devolva os elementos de  $A[0..N - 1]$  em ordem crescente. Sabendo que cada  $A[i]$  está em  $\{0, \dots, K\}$ . Restrições:  $1 \leq K \leq N \leq 10^8$



# Counting Sort

```
#include <stdio.h>
#include <algoritmo>
using namespace std;
int main()
{
    int A[] = {3, 1, 4, 5, 7, 2};
    int K = max_element(myints, myints+7);
    int N = 6;
    int C[K+1];
    int B[N];
    for(int i = 0; i <= K; i++) C[i] = 0;
    for(int i = 0; i < N; i++) C[A[i]]++;
    for(int i = 1; i <= K; i++) C[i] = C[i] + C[i-1];
    for(int j = N-1; j >= 0; j--)
    {
        B[C[A[j]]] = A[j];
        C[A[j]]--;
    }
}
```

## Dica 4: Domine várias linguagens de programação

- Conheça STL do C++.
- Conheça as bibliotecas BigInteger/BigDecimal, GregorianCalendar, Regex do Java
- Conheça os recursos de sua linguagem de programação

## Dica 4: Domine várias linguagens de programação

- Temos N linhas, cada linha começa com '0' seguido de '.' uma quantidade de dígitos x terminada com "...

2

0.1227...

0.517611738...

```
#include <iostream> // or <cstdio>
using namespace std;
// using global variables in contests can be a good st
char digits[100];
int main() {
    scanf("%d", &N);
    while (N--) { // we simply loop from N, N-1, N-2, ...
        scanf("0.%[0-9]...", &digits); // surprised?
        printf("the_digits_are_0.%s\n", digits);
    }
}
```

## Dica 4: Domine várias linguagens de programação

- 1 Desenvolva um código mais conciso possível para a seguinte tarefa:  
Dado uma lista de inteiros ordenada  $L$  de tamanho  $1M$  de itens, determine se o valor  $v$  existe na lista  $L$  com no mais de 20 comparações?

```

#include <iostream>
#include <algorithm>
#include <vector>
#define all(c) (c).begin(), (c).end()
#define tr(c,i)\
for(typeof((c).begin()) i =(c).begin(); i != (c).end(); i++)
using namespace std;
bool compare(int i, int j){ return (i<j);}
int main(){
    ios::sync_with_stdio(false);
    int vetint[] = {4,1,4,7,6,4};
    vector<int> v(vetint, vetint+6);
    vector<int>::iterator it;
    sort(all(v), compare);
    tr(v,it) cout << *it << endl;
    binary_search(all(v), 6, compare) ?
    cout << "encontrado" << endl : cout << "nao_encontrado" << endl;
}

```

```

#include <iostream>
#include <algorithm>
#include <vector>
#define all(c) (c).begin(), (c).end()
#define tr(c,it)\
for(auto it = (c).begin(); it != (c).end(); it++)
using namespace std;
bool compare(int i, int j){ return (i<j);}
int main(){
    ios::sync_with_stdio(false);
    int vetint[] = {4,1,4,7,6,4};
    vector<int> v(vetint, vetint+6);
    sort(all(v), compare);
    tr(v,it) cout << *it << endl;
    binary_search(all(v), 6, compare) ?
    cout << "encontrado" << endl : cout << "nao_encontrado" << endl;
}

```

## Dica 4: Domine várias linguagens de programação

Método	Time(s)
cin	2.70
scanf	0.84

Tabela Performance na leitura  $10^7$  inteiros

## Dica 4: Domine várias linguagens de programação

Adicione `ios::sync_with_stdio(false)` no início do seu programa para melhorar a performance da entrada.

Método	Time(s)
cin	2.70
cin sync false	0.78

Tabela Performance na leitura  $10^7$  inteiros



# Domine a arte de testar

- Pense nos casos de borda.
- Repita o mesmo caso de teste para checar a inicialização das variáveis.
- Crie testes de casos problemáticos.
- Não assuma nada de especial da entrada.

# 10 mandamentos

- 1 Não dividirás por zero.
- 2 Não alocarás dinamicamente a menos que seja necessário.
- 3 Compararás números de ponto flutuante usando `cmp()`
- 4 Verificarás se o grafo pode ser desconexo.
- 5 Verificarás se as arestas do grafo podem ter peso negativo.
- 6 Verificarás se pode haver mais de uma aresta ligando dois vértices.
- 7 Conferirás todos os índices de uma programação dinâmica.
- 8 Reduzirás o branching factor da DFS.
- 9 Farás todos os cortes possíveis em uma DFS.
- 10 Tomarás cuidado com pontos coincidentes e com pontos colineares.

Extraído do caderno de código da PUC RIO 2006.