Juízes Eletrônicos

Prof. Edson Alves

Faculdade UnB Gama

Sumário

Juízes Eletrônicos

Juízes Eletrônicos

- Juízes eletrônicos são programas que fornecem mecanismos de correção automática para problemas de programação competitiva
- A correção é feita através de testes unitários e contempla desde a compilação e execução da solução proposta até a validação dos resultados de cada teste unitário
- Uma solução só é considerada correta se passar, de forma bem sucedida, pelo processo de compilação e por todos os testes unitários
- Um juiz online é uma plataforma ou site que agrega um juiz eletrônico, uma base de problemas e mecanismos de autenticação e gerência de usuários

Juízes Online

- O Beecrowd[1] é o maior juiz online do Brasil
- Conta com mais de 2.400 problemas (2025)
- O Codeforces[2] é um juiz online russo que hospeda uma série *contests* semanalmente acertados
- O Codeforces disponibiliza editoriais e códigos de soluções corretas para análise e estudo, e conta com mais de 1.000 *rounds* (2025)
- O AtCoder[3] é um juiz online japonês semelhante ao Codeforces
- Seus eventos para iniciantes (*AtCoder Beginner Contest*) já contam com mais de 400 edições (2025)

[1] https://www.beecrowd.com.br/
[2] http://codeforces.com/
[3] atcoder.jp

Feedback dos juízes eletrônicos

- A cada solução submetida por parte do usuário, o juiz retornará um *feedback* sobre a solução
- Caso a solução esteja correta, a resposta o juiz será *Accepted* (AC)
- Caso a solução esteja incorreta, será retornada uma dentre várias respostas de erro possíveis, a depender da característica do erro
- Importante ressaltar que o juiz não informa exatamente qual foi o erro, mas uma categorização possível do erro
- Cabe ao usuário interpretar este retorno e tentar localizar e corrigir o erro antes de sua próxima submissão

Respostas para soluções incorretas

Código	Erro	Descrição
WA	*Wrong Answer*	Uma ou mais saídas geradas estão incorretas. O juiz não informa as entradas que geraram o erro nem a resposta correta para tais entradas
PE	*Presentation Error*	As saídas do programa estão corretas, mas a apresentação (formatação, espaçamento, etc) está diferente do que foi especificado
CE	*Compilation Error*	O programa não compila corretamente. Em geral, os juízes listam os parâmetros de compilação utilizados na correção

Respostas para soluções incorretas

Código	Erro	Descrição
RE	*Runtime Error*	O programa trava durante a execução, geralmente por conta de falhas de segmentação, divisão por zero, etc
TLE	*Time Limit Exceeded*	Os programas devem gerar as saídas válidas dentro de um limite de tempo especificado. Caso o programa exceda este tempo, esta será a resposta do juiz
MLE	*Memory Limit Exceeded*	O programa requer mais memória em sua execução do que o juiz permite

Respostas para soluções incorretas

Código	Erro	Descrição
RF	*Restricted Functions*	O programa faz uma chamada a uma função considerada ilegal (por exemplo, `fork()` e `fopen()`)
SE	*Submission Error*	O formulário de envio da submissão tem campos vazios ou incorretos
OLE	*Output Limit Exceeded*	O programa tentou imprimir mais informações do que o permitido. Geralmente causado por laços infinitos

Linguagens Permitidas

- Cada juiz tem um conjunto de linguagens aceitas para a resolução dos problemas
- Em geral, as linguagens aceitas são C, C++, Java e Pascal, embora alguns juízes aceitem centenas de linguagens diferentes
- Na Maratona de Programação da SBC são aceitos: C, C++, Java, Python e Kotlin
- Nos juízes onlines são listados os processos de compilação, de escrita e leitura em console e outras peculiaridades de cada linguagem
- C++ é a linguagem mais utilizada pelos competidores

Entrada e Saída

Entrada e Saída em Console

- Os problemas de programação competitiva requerem que as soluções leiam suas entradas e escrevam suas saídas em arquivos específicos
- Na maioria dos casos, estes arquivos são a entrada (`stdin`) e a saída (`stdout`) padrão do sistema
- Cada linguagem tem mecanismos para ler a entrada e escrever nestes arquivos
- As entradas se encaixam em quatro categorias:
  1. Uma única instância do problema
  2. T instâncias do problema (o valor de T é dado na primeira linha)
  3. N instâncias do problema, a entrada termina com um valor sentinela
  4. N instâncias do problema, a entrada termina com fim de arquivo (EOF)

Exemplo das 4 Categorias de Entrada

Problema: dados dois inteiros positivos X e Y, determine sua soma.

1

$X\ Y$

2

T
$X_1\ Y_1$
$\dots$
$X_T\ Y_T$

3

$X_1\ Y_1$
$X_2\ Y_2$
$\dots$
$-1\ -1$

4

$X_1\ Y_1$
$X_2\ Y_2$
$\dots$
$X_N\ Y_N$

Solução C: Categoria 1

```c
#include <stdio.h>

int main()
{
    int X, Y;
    scanf("%d %d", &X, &Y);

    printf("%d\n", X + Y);

    return 0;
}
```

Solução C: Categoria 2

```c
#include <stdio.h>

int main()
{
    int T;
    scanf("%d", &T);

    while (T--)
    {
        int X, Y;
        scanf("%d %d", &X, &Y);

        printf("%d\n", X + Y);
    }

    return 0;
}
```

Solução C: Categoria 3

```c
#include <stdio.h>

int main()
{
    int X, Y;

    while (scanf("%d %d", &X, &Y), X != -1 && Y != -1)
    {
        printf("%d\n", X + Y);
    }

    return 0;
}
```

Solução C: Categoria 4

```c
#include <stdio.h>

int main()
{
    int X, Y;

    while (scanf("%d %d", &X, &Y) == 2)
    {
        printf("%d\n", X + Y);
    }

    return 0;
}
```

Solução C++: Categoria 1

```cpp
#include <iostream>

using namespace std;

int main()
{
    ios::sync_with_stdio(false);

    int X, Y;
    cin >> X >> Y;

    cout << X + Y << endl;

    return 0;
}
```

Solução C++: Categoria 2

```cpp
#include <iostream>

using namespace std;

int main()
{
    ios::sync_with_stdio(false);

    int T;
    cin >> T;

    while (T--)
    {
        int X, Y;
        cin >> X >> Y;
        cout << X + Y << endl;
    }

    return 0;
}
```

Solução C++: Categoria 3

```cpp
#include <iostream>

using namespace std;

int main()
{
    ios::sync_with_stdio(false);

    int X, Y;

    while (cin >> X >> Y, X != -1 and Y != -1)
    {
        cout << X + Y << endl;
    }

    return 0;
}
```

Solução C++: Categoria 4

```cpp
#include <iostream>

using namespace std;

int main()
{
    ios::sync_with_stdio(false);

    int X, Y;

    while (cin >> X >> Y)
    {
        cout << X + Y << endl;
    }

    return 0;
}
```

Solução Java: Categoria 1

```java
import java.util.Scanner;

public class C1 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        int X = scanner.nextInt();
        int Y = scanner.nextInt();

        System.out.println(X + Y);
    }
}
```

Solução Java: Categoria 2

```java
import java.util.Scanner;

public class C2 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        int T = scanner.nextInt();

        for (int i = 0; i < T; ++i) {
            int X = scanner.nextInt();
            int Y = scanner.nextInt();

            System.out.println(X + Y);
        }
    }
}
```

Solução Java: Categoria 3

```java
import java.util.Scanner;

public class C3 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        while (true) {
            int X = scanner.nextInt();
            int Y = scanner.nextInt();

            if (X == -1 && Y == -1)
                break;

            System.out.println(X + Y);
        }
    }
}
```

Solução Java: Categoria 4

```java
import java.util.Scanner;

public class C4 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        while (scanner.hasNext()) {
            int X = scanner.nextInt();
            int Y = scanner.nextInt();

            System.out.println(X + Y);
        }
    }
}
```

Solução Python: Categoria 1

```python
X, Y = map(int, input().split())

print(X + Y)
```

Solução Python: Categoria 2

```python
T = int(input())

for _ in range(T):
    X, Y = map(int, input().split())

    print(X + Y)
```

Solução Python: Categoria 3

```python
while True:
    X, Y = map(int, input().split())

    if X == -1 and Y == -1:
        break

    print(X + Y)
```

Solução Python: Categoria 4

```python
while True:
    try:
        X, Y = map(int, input().split())

        print(X + Y)
    except EOFError:
        break
```

Teste das soluções

Assuma que a entrada do problema esteja no arquivo `in.txt`, e que a solução do juiz para esta entrada esteja no arquivo `out.txt`.

Para gerar o arquivo `ans.txt` referente à saída produzida pela solução proposta, podemos usar um dos três comandos abaixo:

```
$ ./a.out < in.txt > ans.txt        # C/C++
$ java Main < in.txt > ans.txt      # Java
$ python sol.py < in.txt > ans.txt  # Python
```

Para verificar se a solução proposta está correta, basta usar o comando `diff` do Linux:

```
$ diff ans.txt out.txt
```

Referências

1. **HALIM**, Felix; **HALIM**, Steve. *Competitive Programming 3*, 2010.
2. **LAAKSONEN**, Antti. *Competitive Programmer's Handbook*, 2018.
3. **SKIENA**, Steven S.; **REVILLA**, Miguel A. *Programming Challenges*, 2003.
4. AtCoder[4].
5. Beecrowd[5].
6. Codeforces[6].

[4] https://atcoder.jp/
[5] https://www.beecrowd.com.br/
[6] http://codeforces.com/