

## Lista 03

Info.: Os programas das questões abaixo devem ser codificados na linguagem Javascript

1. Faça um programa que leia um número inteiro e informe se ele é perfeito ou não. Um número perfeito é aquele que é igual à soma de seus divisores. Ex.:  $6 = 1 + 2 + 3 = \text{número perfeito}$

Imagine que temos o seguinte problema: Dado um número inteiro  $N$ , queremos encontrar todos os divisores positivos de  $N$ . Uma possível solução seria, novamente, iterar por todos os números de 1 a  $N$  e checar quais deles dividem  $N$ , salvando-os em um vetor de respostas. Dessa forma, obtemos uma complexidade de  $O(N)$ .

Para otimizar este algoritmo, perceba que se  $p$  é um divisor de  $N$  maior que  $\sqrt{N}$ , então o número  $q = \frac{N}{p}$  também é um divisor de  $N$  e  $q < \sqrt{N}$ . Isso sugere que é suficiente iterarmos apenas pelos números de 1 a  $\sqrt{N}$ , e, assim que encontrarmos um divisor  $q$  tal que  $q \leq \sqrt{N}$ , inserimos tanto  $q$  quanto  $\frac{N}{q}$  em nosso vetor de respostas. Dessa forma, encontraremos tanto os divisores "pequenos" (menores ou iguais a  $\sqrt{N}$ ) de  $N$  quanto os divisores "grandes" (maiores que  $\sqrt{N}$ ).

Essa versão otimizada do algoritmo reduz a complexidade de  $O(N)$  para  $O(\sqrt{N})$ , tornando-o muito mais eficiente para números grandes.

```
//src/unidade I/lista03/questao01.js
```

```
function listaDivisores(numero) {
  let divisores = []
  for (let i = 1; i <= numero; i++) {
    if (numero % i === 0) {
      divisores.push(i)
    }
  }
  return divisores
}

function listaDivisores2(numero) {
  let divisores = []
  let raiz = Math.sqrt(numero)
  for (let i = 1; i <= raiz; i++) {
    if (numero % i === 0) {
      divisores.push(i)
      if (i !== raiz) {
        divisores.push(numero / i)
      }
    }
  }
  return divisores.sort((a, b) => a - b)
}

function numeroPerfeito(numero) {
  let soma = listaDivisores2(numero).filter((a) => a < numero).reduce((a, b) => a + b, 0)
  return soma === numero
}

for (let i = 1; i <= 1000000; i++) {
  if (numeroPerfeito(i)) {
    console.log(`${i} é um número perfeito`)
  }
}
```

2. Faça um programa que solicite um número inteiro de até 4 dígitos ao usuário e inverta a ordem de seus algarismos. Ex.: Entrada = 5382 - Saída = 2835

```
//src/unidade I/lista03/questao02.js
```

```
function invert4(num) {
  if (num <= 9999) {
    let numInvertido = 0
    let numString = num.toString()
    for (let i = numString.length - 1; i >= 0; i--) {
      numInvertido += numString[i]
    }
    return numInvertido * 1
  } else return null
}

function testa(num) {
  let inverso = invert4(num)
  if (inverso !== null) {
    console.log(`O inverso de ${num} é ${inverso}`)
  }
}

for (let i = 1; i <= 10000; i = i + 235) {
  testa(i)
}
```

3. Escreva um programa para verificar se um número é palíndromo (Número que é igual ao seu reverso Ex.: 14541)

```
//src/unidade I/lista03/questao03.js
```

```
function invert4(num) {
  if (num <= 9999) {
    let numInvertido = 0
    let numString = num.toString()
    for (let i = numString.length - 1; i >= 0; i--) {
      numInvertido += numString[i]
    }
    return numInvertido * 1
  } else return null
}

function testa(num) {
  let inverso = invert4(num)
  if (inverso !== null) {
    if (inverso === num){
      console.log(`${num} é um número palíndromo`)
    }
  }
}

for (let i = 1; i <= 10000; i = i + 41) {
  testa(i)
}
```

4. Dona Lesma é esportista e aventureira e definiu como objetivo deste verão alcançar o topo do muro do jardim em que vive. A cada dia, valente e metodicamente ela sobe exatamente uma certa distância (sempre a mesma a cada dia). Mas a cada noite enquanto dorme Dona Lesma escorrega para baixo uma outra distância (sempre a mesma a cada noite) ... Dadas a altura do muro, a distância que ela sobe a cada dia e a distância que ela desce

a cada noite, ajude Dona Lesma a calcular quantos dias ela levará para chegar ao topo do muro. altura = 10000  
subida = 100 descida = 50

dias	início do dia	fim do dia
1º	0	100
2º	50	150
3º	100	200
...	...	...
$n$	$(n - 1) \times (\text{subida} - \text{descida})$	$(n - 1) \times (\text{subida} - \text{descida}) + \text{subida}$

$$\text{distancia} = (n - 1) \times (\text{subida} - \text{descida}) + \text{subida}$$

$$(n - 1) \times (\text{subida} - \text{descida}) = \text{distancia} - \text{subida}$$

$$n - 1 = \frac{\text{distancia} - \text{subida}}{\text{subida} - \text{descida}}$$

$$n = \frac{\text{distancia} - \text{subida}}{\text{subida} - \text{descida}} + 1$$

```
//src/unidade I/lista03/questao04.js
```

```
function quantidadesDias(distancia, subida, descida) {
  dias = ((distancia - subida) / (subida - descida)) + 1
  return dias
}
```

```
distancia = 10000
subida = 100
descida = 50
```

```
dias = quantidadesDias(distancia, subida, descida)
```

```
console.log(`A quantidade de dias necessários para a dona lesma subir ${distancia} metros é ${dias}`)
```

5. Pedrinho está implementando o sistema de controle de pagamentos parcelados de uma grande empresa de cartão de crédito digital. Os clientes podem parcelar as compras sem juros no cartão, em até 18 vezes. Quando o valor  $V$  da compra é divisível pelo número  $P$  de parcelas que o cliente escolhe, todas as parcelas terão o mesmo valor. Por exemplo, se o cliente comprar um livro de  $V=30$  reais em  $P=6$  vezes, então as parcelas terão valores: 5, 5, 5, 5, 5 e 5. Mas se o valor da compra não for divisível pelo número de parcelas será preciso fazer um ajuste, pois a empresa quer que todas as parcelas tenham sempre um valor inteiro e somem no total, claro, o valor exato da compra. O que Pedrinho decidiu foi distribuir o resto da divisão de  $V$  por  $P$  igualmente entre as parcelas iniciais. Por exemplo, se a compra for de  $V=45$  e o número de parcelas for  $P=7$ , então as parcelas terão valores: 7, 7, 7, 6, 6, 6 e 6. Quer dizer, como o resto da divisão de 45 por 7 é 3, então as 3 parcelas iniciais devem ter valor um real maior do que as 4 parcelas finais. Você precisa ajudar Pedrinho e escrever um programa que, dado o valor da compra e o número de parcelas, imprima os valores de cada parcela. O programa deve receber como entrada o valor de  $V$ , representando o valor da compra e o valor de  $P$ , indicando o número de parcelas. A saída deve ser as parcelas

```
//src/unidade I/lista03/questao05.js
```

```
function divideParcelas(valor, numeroParcela) {
  if (valor < 0 || numeroParcela < 0)
    throw new Error('Valor e número de parcelas devem ser maiores que zero')

  if(numeroParcela > 18)
    throw new Error('Número de parcelas deve ser menor ou igual a 18')

  let parcelas = []

  let resto = valor % numeroParcela

  let valorParcela = (valor - resto) / numeroParcela
```

```
    for (let i = 0; i < numeroParcela; i++) {  
      if(i < resto)  
        parcelas.push(valorParcela + 1.00)  
      else  
        parcelas.push(valorParcela)  
    }  
  
    return parcelas  
  }  
  
  try{  
    parcelas = divideParcelas(100, 19)  
    console.log(parcelas)  
  }catch(e){  
    console.log(e.message)  
  }  
}
```