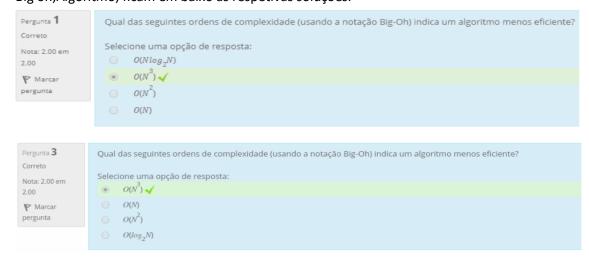
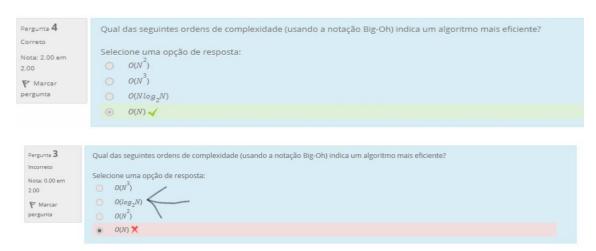
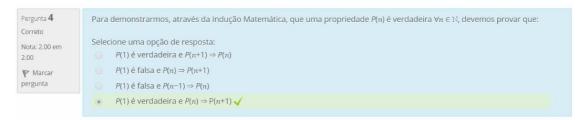
Perguntas de AED no ano de 2019 Big oh, Algoritmo, ficam em baixo as respetivas soluções.



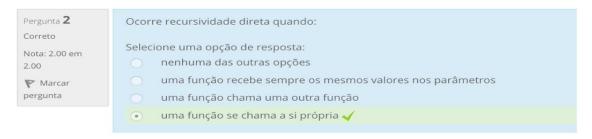
Mais eficiente!



Pergunta da propriedade



Pergunta solta



Big-Oh → Ordem de complexidade!

```
Pergunta 1
                    Usando a notação Big-Oh, qual a ordem de complexidade da execução do seguinte bloco de código?
Correto
                     Nota: 4.00 em
4.00
Marcar
                         }
pergunta
                    Selecione uma opção de resposta:
                          O(N')
                          O(N^5)
                          O(N)
Pergunta 5
                    Usando a notação Big-Oh, qual a ordem de complexidade da execução do seguinte bloco de código?
Correto
                     for (int i = 0; i < n; i++) {
   for (int j = 0; j < i * n; j++) {
      for (int k = 0; k < j * i * n; k++) {</pre>
Nota: 4.00 em
4.00
                                  System.out.println(k);
Marcar
pergunta
                    Selecione uma opção de resposta:
                          O(N^{5})
                          O(N')
                          O(N^8)
                          O(N^6)
```

HARMÓNICOS

```
Pergunta 2
                    Em matemática, a função que calcula números harmónicos é dada por:
                                           \Leftarrow n = 1
                     H(n) = \begin{cases} 1 & \Leftarrow n = 1 \\ \frac{1}{n} + H(n-1) & \Leftarrow n \in \mathbb{N} \land n > 1 \end{cases}
Correto
Nota: 3.00 em
3.00
                    Considere a seguinte implementação:
Marcar
                     private double h(int n) {
pergunta
                          if (n < 1) {
                               throw new IllegalArgumentException("n não pode ser inferior a 1");
                          if (n == 1) {
                               return 0;
                          return 1.0 / n + h(n - 1);
                     }
                   Há algum problema nessa implementação?
                   Selecione uma opção de resposta:

    sim, no tipo de retorno

                          sim, a chamada recursiva não converge
                          sim, o caso base está mal definido 🗸
                          nāo
```

```
Pergunta 3
```

Correto

Nota: 3.00 em

Marcar pergunta Em matemática, a função que calcula números harmónicos é dada por:

$$H(n) = \begin{cases} 1 & \Leftarrow n = 1 \\ \frac{1}{n} + H(n-1) & \Leftarrow n \in \mathbb{N} \land n > 1 \end{cases}$$

Considere a seguinte implementação:

```
private double h(int n) {
    if (n < 1) {
        throw new IllegalArgumentException("n não pode ser inferior a 1");
    }
    if (n == 1) {
        return 1;
    }
    return 1.0 / n + h(n + 1);
}</pre>
```

Há algum problema nessa implementação?

Selecione uma opção de resposta:

- não
- sim, no tipo de retorno
- sim, a chamada recursiva não converge
- sim, o caso base está mal definido

```
Pergunta 5
Correto
```

Nota: 3.00 em

Marcar pergunta Em matemática, a função que calcula números harmónicos é dada por:

$$H(n) = \begin{cases} 1 & \Leftarrow n = 1 \\ \frac{1}{n} + H(n-1) & \Leftarrow n \in \mathbb{N} \land n > 1 \end{cases}$$

Considere a seguinte implementação:

```
private double h(int n) {
    if (n < 1) {
        throw new IllegalArgumentException("n n\u00e3o pode ser inferior a 1");
    }
    if (n == 1) {
        return 1;
    }
    return 1.0 / n + h(n - 1);
}</pre>
```

Há algum problema nessa implementação?

- o sim, no tipo de retorno
- ⊚ não √
- sim, a chamada recursiva não converge
- o sim, o caso base está mal definido

```
Pergunta 5
                   Em matemática, a função que calcula números harmónicos é dada por:
                    H(n) = \begin{cases} 1 & \Leftarrow n = 1 \\ \frac{1}{n} + H(n-1) & \Leftarrow n \in \mathbb{N} \land n > 1 \end{cases}
Por responder
Nota de 3.00
P Marcar
                   Considere a seguinte implementação:
pergunta
                   private long h(int n) {
                         if (n < 1) {
                              throw new IllegalArgumentException("n não pode ser inferior a 1");
                         if (n == 1) {
                              return 1;
                         return 1.0 / n + h(n - 1);
                    }
                   Há algum problema nessa implementação?
                   Selecione uma opção de resposta:
                         sim, o caso base está mal definido
                          não
                    9
                          sim, no tipo de retorno
                          sim, a chamada recursiva não converge
```

Pergunta de pesquisa

```
Considere uma função de pesquisa que devolva a lista de índices (indicativos das posições) de um dado elemento numa sequência ordenada de elementos.
Indique qual das seguintes implementações resolve o problema de forma mais eficiente.
Selecione uma opção de resposta:
           public LinkedList<Integer> pesquisar(EstatisticaDeComparacoes estatistica,
                indices.add(i);
                          return indices;
                return indices;
            public LinkedList<Integer> pesquisar(EstatisticaDeComparacoes estatistica,
                LinkedList<Integer> indices = new LinkedList<>();
for (int i = 0; i < elementos.length; i++) {
    estatistica.incrementarComparacoes();
    if (criterio.comparar(elemento, elementos[i]) == 0) {
        indices.add(i);
    }
                return indices;
               LinkedList<Integer> indices = new LinkedList<();
for (int i = 0; i < elementos.length; i++) {
    estatistica.incrementarComparacoes();
    if (criterio.compara(elemento, elementos[i]) == 0) {
        indices.add(i);
        while (++i < elementos.length && criterio.comparar(elemento, elementos[i]) == 0) {
        indices.add(i);
    }
           public LinkedList<Integer> pesquisar(EstatisticaDeComparacoes estatistica,
                         return indices:
                return indices;
           indices.add(i):
                          return indices;
                return indices;
```

Perguma **2** Correto Nota: 6.00 em

Considere uma função de pesquisa que devolva a lista de índices (indicativos das posições) de um dado elemento numa sequência ordenada de elementos.

Indique qual das seguintes implementações resolve o problema de forma eficiente.

Selecione uma oncăn de resporta

```
Considere uma função de pesquisa que devolva a lista de índices (indicativos das posições) de um dado elemento numa sequência ordenada de elementos.
                        Indique qual das seguintes implementações resolve o problema de forma eficiente
Nota: 6.00 em
                                      public LinkedList<Integer> pesquisar(EstatisticaDeComparacoes estatistica,
                                            LinkedListCInteger> indices = relemento, T... elemento
LinkedListCInteger> indices = new LinkedList<>();
for (int i = 0; i < elementos.length; i++) {
    estatistica.incrementarComparacoes();
    if (criterio.comparar(elemento, elementos[i]) == 0) {
        indices.add(i);
        return indices;
    }
}</pre>
                                           }
return indices;
                                      public LinkedList<Integer> pesquisar(EstatisticaDeComparacoes estatistica,
                                           LinkedList<Integer> indices = new LinkedList<>();
for (int i = 0; i < elementos.length; i++) {
    estatistica.incrementarComparacoes();
    if (criterio.comparar(elemento, elementos[i]) == 0) {
        indices.add(i);
    }
}</pre>
                                                }
                                           return indices;
                                            return indices;
```

Considere uma função de pesquisa que devolva a lista de índices (indicativos das posições) de um dado elemento numa sequência ordenada de elementos. Indique qual das seguintes implementações resolve o problema de forma eficiente.

Selecione uma opção de resposta:

Pergunta 5

P Marcar

pergunta

```
Linkediat(Integer: indices - new Linkediat(); to elements
for (lat i = 0; i = elementos.length; i++) {
    ratio intiga, increasiful timps corex();
    if (oritoria, compara*(elemento, elementos[i]) -- 0) {
        indices.adu(i);
    }
     return indices;
   b.
reture indices;
    return indices;
 c. Nenhuma
    d.
     Linkedist-Integer) promise (EstatisticaDeCorporacors
T of counts, f... of counts
for (Est 1 = 0, 1 = decents, length, i++)
relatistical incrementaring counts;
if (criterio.compare(elemento, elements[1]) -- 8) {
    return indices.add(1);
    return indices;
}
```

```
Pergunta 1
                          Considere uma função de pesquisa que devolva a lista de índices (indicativos das posições) de um dado elemento numa sequência
Correto
                          não ordenada de elementos.
                           Indique qual das seguintes implementações resolve o problema.
Nota: 6.00 em
                          Selecione uma opcão de resposta:
                                       P Marcar
                                              LinkedList<Integer> indices = new LinkedList<);
for (int i = 0; i < elementos.length; i++) {
    estatistica.incrementarComparacoes();
    if (criterio.comparar(elemento, elementos[i]) == 0) {
        indices.add(i);
        while (++i < elementos.length && criterio.comparar(elemento, elementos[i]) == 0) {
            indices.add(i);
        }
}
pergunta
                                                         return indices:
                                                   }
                                             }
return indices;
                           ( a. }
                                         public LinkedList<Integer> pesquisar(EstatisticaDeComparacoes estatistica,
                                              LinkedList<Integer> indices = new LinkedList<();
for (int i = 0; i < elementos.length; i++)
    estatistica.incrementarComparacoes();
    if (criterio.comparar(elemento, elementos[i]) == 0) {
        indices.add(i);
    }
}</pre>
                                              return indices;
                                        T elemento, T... elemento
LinkedList<Integer> indices = new LinkedList<();
for (int i = 0; i < elementos.Length; i++) {
    estatistica.incrementarComparacoes();
    if (criterio.comparar(elemento, elementos[i]) == 0) {
        indices.add(i);
        return indices;
    }
}</pre>
                                              return indices;
                           O c. }
                                              public LinkedList<Integer> pesquisar(EstatisticaDeComparacoes estatistica,
                                                         return indices;
                                                   }
                                              return indices;
Pergunta 2
```

Incorreto
Note: 0.00 em

P Marcar

pergunta

6.00

```
Considere uma função de pesquisa que devolva a lista de índices (indicativos das posições) de um dado elemento numa sequência não ordenada de elementos.
Selecione uma opção de resposta:
                   public LinkedList<Integer> pesquisar(EstatisticaDeComparacoes estatistica,
T elemento, T... elementos) {
                           LinkedList<Integer> indices = new LinkedList<2();
for (int i = 0; i < elemento.S.length; i++) {
    estatistica.incrementarComparacoes();
    if (criterio.comparar(elemento, elementos[i]) == 0) {
        indices.add(i);
    }
}</pre>
                                  )
                           return indices;
                    public LinkedList<Integer> pesquisar(EstatisticaDeComparacoes estatistica,
                           LinkedList<Integer> indices = new LinkedList<);
for (int i = 0; i < elementos, i++) {
    estatistica.incrementarComparacoes();
    if (criterio.comparar(elemento, elementos[i]) == 0) {
        indices.add(i);
        while (++i < elementos.length && criterio.comparar(elemento, elementos[i]) == 0) {
        indices.add(i);
    }
                                         return indices;
                                  }
                           }
return indices;
  6 b. )
                   LinkedList(Integer) indices = new LinkedList();
for (int i = 0; i < elementos.Length; i++) (
    estatistica.incrementarComparacoes();
    if (criterio.comparar(elemento, elementos[i]) == 0) {
        indices.add(i);
        return indices;
    }
}</pre>
                           return indices;
                           LinkedList<Integer> indices = new LinkedList<();
for (int i = 0; i < elementos. Length; i++) {
    estatistica.incrementarComparacoes();
    if (criterio.comparar(elemento, elementos[i]) == 0) {
        indices.add(i);
        while (criterio.comparar(elemento, elementos[i]) == 0 & & ++i < elementos.length) {
        indices.add(i);
    }
                    public LinkedList<Integer> pesquisar(EstatisticaDeComparacoes estatistica,
                                           }
return indices;
```

Considere uma função de pesquisa que devolva a lista de índices (indicativos das posições) de um dado elemento numa sequência não ordenada de elementos. Nota: 6.00 em Selecione uma opção de resposta: 6.00 P Marcar return indices;) return indices; return indices; public LinkedList<Integer> pesquisar(EstatisticaDeComparacoes estatistica, indices.add();
shile (criterio.compara[elemento, elementos[i]) == 0 88 iii c elementos.length) (
indices.add()); return indices: 1 return indices; return indices;

Pergunta 4
Correto
Nota: 6.00 em
6.00

Marcar
pergunta

```
Pergunta 3
Correto
Nota: 5.00 em
5.00

Marcar
pergunta
```

Considere a classe Automovel e o critério de comparação descritos de seguida.

```
public class Automovel {
   private String marca;
    private int cilindrada;
private float consumo;
     public Automovel(String marca, int cilindrada, float consumo) {
         this.marca = marca;
this.cilindrada = cilindrada;
          this.consumo = consumo;
    public String getMarca() {
         return marca;
    public int getCilindrada() {
    return cilindrada;
     public float getConsumo() {
         return consumo;
public \ enum \ Comparacao Automove is \ implements \ Comparacao < Automove l> \ \{
    CRITERIO:
     public int comparar(Automovel o1, Automovel o2) {
         int comparacao = o1.getMarca().compareTo(o2.getMarca());
if (comparacao != 0) {
               return comparacao;
         comparacao = -Long.compare(o1.getCilindrada(), o2.getCilindrada());
if (comparacao != 0) {
               return comparacao;
          return Float.compare(o2.getConsumo(), o1.getConsumo());
```

O critério de comparação compara automóveis:

- o por marca descendente seguido por cilindrada descendente seguido por consumo ascendente
- por marca ascendente seguido por cilindrada descendente seguido por consumo descendente
- o por marca ascendente seguido por cilindrada ascendente seguido por consumo ascendente
- o por marca ascendente seguido por cilindrada descendente seguido por consumo ascendente

```
Pergunta 2
```

Incorreto

Nota: 0.00 em

Marcar pergunta Considere a classe Automovel e o critério de comparação descritos de seguida.

```
public class Automovel {
   private String marca;
    private int cilindrada;
    private float consumo;
    public Automovel(String marca, int cilindrada, float consumo) {
        this.marca = marca;
        this.cilindrada = cilindrada;
        this.consumo = consumo;
    public String getMarca() {
        return marca;
    public int getCilindrada() {
       return cilindrada;
    public float getConsumo() {
        return consumo;
public enum ComparacaoAutomoveis implements Comparacao<Automovel> {
    CRITERIO;
    @Override
    public int comparar(Automovel 01, Automovel 02) {
       int comparacao = o1.getMarca().compareTo(o2.getMarca());
        if (comparacao != 0) {
            return comparacao;
        comparacao = -Long.compare(o2.getCilindrada(), o1.getCilindrada());
        if (comparacao != 0) {
            return comparacao;
        return Float.compare(o1.getConsumo(), o2.getConsumo()):
```

,

O critério de comparação compara automóveis:

- por marca descendente seguido por cilindrada descendente seguido por consumo ascendente
- por marca ascendente seguido por cilindrada descendente seguido por consumo ascendente
- opor marca ascendente seguido por cilindrada descendente seguido por consumo descendente
- por marca ascendente seguido por cilindrada ascendente seguido por consumo ascendente

Fergurea 2 Corneto Nota: 5.00 em 5.00 P Marcar

pergunta

Considere a classe Automovel e o critério de comparação descritos de seguida.

```
public class Autonovel (
   private String marca;
   private int cilindrada;
   private float consumo;
     public Automovel(String marca, int cilindrada, float consumo) (
           this.marca = marca;
this.cilindrada = cilindrada;
           this.consumo = consumo;
     public String getMarca() (
           return marca;
     public int getCilindrada() {
    return cilindrada;
     public float getConsumo() {
         return consumo;
public enum ComparacaoAutomoveis implements Comparacao<Automovel> {
     CRITERIO;
     #Override
     public int comparar(Automovel o1, Automovel o2) {
   int comparacao = o1.getMarca().compareTo(o2.getMarca());
   if (comparacao != 0) {
                return comparacao;
          comparacao = -Long.compare(o1.getCilindrada(), o2.getCilindrada());
if (comparacao != 0) {
                return comparacao;
           return Float.compare(ol.getConsumo(), o2.getConsumo());
```

O critério de comparação compara automóveis:

- por marca ascendente seguido por cilindrada descendente seguido por consumo descendente
- por marca ascendente seguido por cilindrada descendente seguido por consumo ascendente
 - por marca descendente seguido por cilindrada descendente seguido por consumo ascendente
- por marca ascendente seguido por cilindrada ascendente seguido por consumo ascendente

```
Pergunta 3
                         Considere a classe Automovel e o critério de comparação descritos de seguida.
                          public class Automovel {
   private String marca;
   private int cilindrada;
   private float consumo;
Nota: 0.00 em
5.00
№ Remover
                                public Automovel(String marca, int cilindrada, float consumo) {
    this.marca = marca;
    this.cilindrada = cilindrada;
    this.consumo = consumo;
marcação
                                public String getMarca() {
    return marca;
                                public int getCilindrada() {
    return cilindrada;
                                public float getConsumo() {
   return consumo;
                           public enum ComparacaoAutomoveis implements Comparacao<Automovel> {
    CRITERIO;
                                public int comparar(Automovel 01, Automovel 02) {
  int comparaca = 02.getNarca().compareTo(01.getNarca());
  if (comparaca = 0 0 (
    return comparaca);
                                     }
comparacao = -Long.compare(o1.getCilindrada(), o2.getCilindrada());
if (comparacao | = 0) {
    return comparacao;
                                      return Float.compare(o1.getConsumo(), o2.getConsumo());
                         O critério de comparação compara automóveis:
                         Selecione uma opção de resposta:
                          o por marca ascendente seguido por cilindrada ascendente seguido por consumo ascendente
                          o por marca descendente seguido por cilindrada descendente seguido por consumo ascendente
                          o por marca descendente seguido por cilindrada descendente seguido por consumo descendente

    por marca ascendente seguido por cilindrada descendente seguido por consumo ascendente X
```

A opção correta é a B!