Exercícios

Leia um número inteiro positivo n do teclado e preencha um vetor de tamanho n com valores aleatórios. Para cada valor do vetor, encontre a maior potência de 10 que seja menor ou igual a esse valor.

Exemplo de entrada e saída:

- **Entrada:** $n = 5$
  Vetor gerado: [78, 4500, 3, 120, 9999]
- **Saída:**

  10
  1000
  1
  100
  1000

Leia um número inteiro positivo n do teclado e preencha um vetor de tamanho n com valores aleatórios. Para cada valor do vetor, verifique se ele é um número primo. Não é permitido o uso de métodos de manipulação de strings. Após a implementação, determine a complexidade do algoritmo e expresse-a em notação Θ.

Exemplo de entrada e saída:

- **Entrada:** $n = 6$ **Vetor gerado:** [15, 7, 21, 31, 40, 97]
- **Saída:**

  Não
  Sim
  Não
  Sim
  Não
  Sim

Professor Rodrigo Bossini <professorbossini@gmail.com>
Produção: 2025

Visite as documentações oficiais das classes `LinkedList` e `ArrayList` da linguagem Java. Escolha cinco métodos que ambas as classes possuem em comum e compare suas complexidades de tempo de execução. Após a análise, determine e escreva a complexidade de cada método em notação Θ.

Exemplo de métodos que podem ser analisados:

- `add(E element)`
- `add(int index, E element)`
- `remove(int index)`
- `get(int index)`
- `contains(Object o)`

Após a comparação, explique as diferenças observadas entre as duas estruturas e justifique os tempos de execução com base na estrutura interna de cada implementação.

https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/LinkedList.html

https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/ArrayList.html

Leia um número inteiro positivo n do teclado e preencha uma matriz quadrada de tamanho $n \times n$ com valores aleatórios. Para cada elemento da matriz, substitua-o pela soma de todos os elementos da mesma linha. Após a implementação, determine a complexidade do algoritmo e expresse-a em notação Θ.

Exemplo de entrada e saída:

- **Entrada:** $n = 3$ **Matriz gerada:**

$$\begin{bmatrix} 2 & 4 & 6 \\ 1 & 5 & 3 \\ 7 & 8 & 2 \end{bmatrix}$$

- **Saída:**

$$\begin{bmatrix} 12 & 12 & 12 \\ 9 & 9 & 9 \\ 17 & 17 & 17 \end{bmatrix}$$

Leia um número inteiro positivo n do teclado e preencha duas matrizes quadradas de tamanho $n \times n$ com valores aleatórios. Em seguida, calcule o produto das duas matrizes. Após a implementação, determine a complexidade do algoritmo e expresse-a em notação Θ.

Exemplo de entrada e saída:

- **Entrada:** $n = 2$ **Matriz A:**

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

Matriz B:

$$\begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

- **Saída: Matriz Resultado:**

$$\begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

Leia um número inteiro positivo n do teclado e preencha um vetor de tamanho n com valores aleatórios. Em seguida, calcule a variância e o desvio padrão dos valores no vetor. Após a implementação, determine a complexidade do algoritmo e expresse-a em notação Θ.

Fórmulas utilizadas:

- Média:

$$\mu = \frac{1}{n} \sum_{i=1}^{n} x_i$$

- Variância:

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \mu)^2$$

- Desvio Padrão:

$$\sigma = \sqrt{\sigma^2}$$

Leia um número inteiro positivo n do teclado e preencha um `HashSet` com n objetos do tipo `Pessoa`, onde cada pessoa possui apenas um nome como propriedade. O `Set` não permite duplicatas, então pessoas com o mesmo nome não devem ser inseridas. Após a inserção, exiba todos os elementos do conjunto. Em seguida, ajuste a implementação para garantir que a ordem de inserção dos elementos seja preservada, substituindo o `HashSet` por outra implementação de `Set` que mantenha essa característica.

Para garantir a correta funcionalidade do `Set`, implemente os métodos `equals()` e `hashCode()` na classe `Pessoa`. Duas instâncias de `Pessoa` devem ser consideradas iguais se possuírem o mesmo nome.

Além disso, estude e explique a importância da implementação correta do método `hashCode()` para evitar colisões e garantir eficiência na estrutura de dados. Analise como a comparação de `String` no método `equals()` afeta a complexidade assintótica da busca e inserção dos elementos no `Set`.

Exemplo de entrada e saída:

- **Entrada:** $n = 2$ **Nomes gerados:** ["Ana", "Carlos"]
- **Saída com HashSet:**

  Carlos
  Ana

- **Saída após ajuste para preservar a ordem de inserção:**

  Ana
  Carlos

Referências

CORMEN, Thomas H. et al. **Introduction to Algorithms**. 3. ed. Cambridge: MIT Press, 2009.

Feofiloff, Paulo**. Anotações sobre Algoritmos: Slides.** São Paulo: Instituto de Matemática e Estatística – USP, [s.d.]. Disponível em: https://www.ime.usp.br/~pf/livrinho-AA/downloads/AA-SLIDES.pdf. Acesso em: março de 2025.

KLEINBERG, Jon; TARDOS, Éva. **Algorithm Design**. Boston: Pearson, 2006.