Educação Profissional Paulista

Técnico em
Ciência de Dados

Secretaria da Educação

SÃO PAULO
GOVERNO DO ESTADO

Lógica de programação e algoritmos

Busca e ordenação

Aula 1

Código da aula: [DADOS]ANO1C3B3S23A1

Mapa da Unidade 1 Componente 4

Lógica de programação e algoritmos

semana 21

Recursividade

semana 23

Você está aqui!

Busca e ordenação

semana 25

Prática de busca e ordenação

semana 27

Algoritmos de contagem e acumulação

semana 28

Pilhas e filas

Lógica de programação e algoritmos

Mapa da Unidade 1 Componente 4

Você está aqui!

Busca e ordenação

23

Aula 1

Código da aula: [DADOS]ANO1C3B3S23A1

Objetivos da aula

- Apresentar o conceito de complexidade de algoritmo.

Recursos Didáticos

- Recurso audiovisual para a exibição de vídeos e imagens.
- Acesso ao laboratório de informática e/ou à internet.

Duração da Aula

50 minutos.

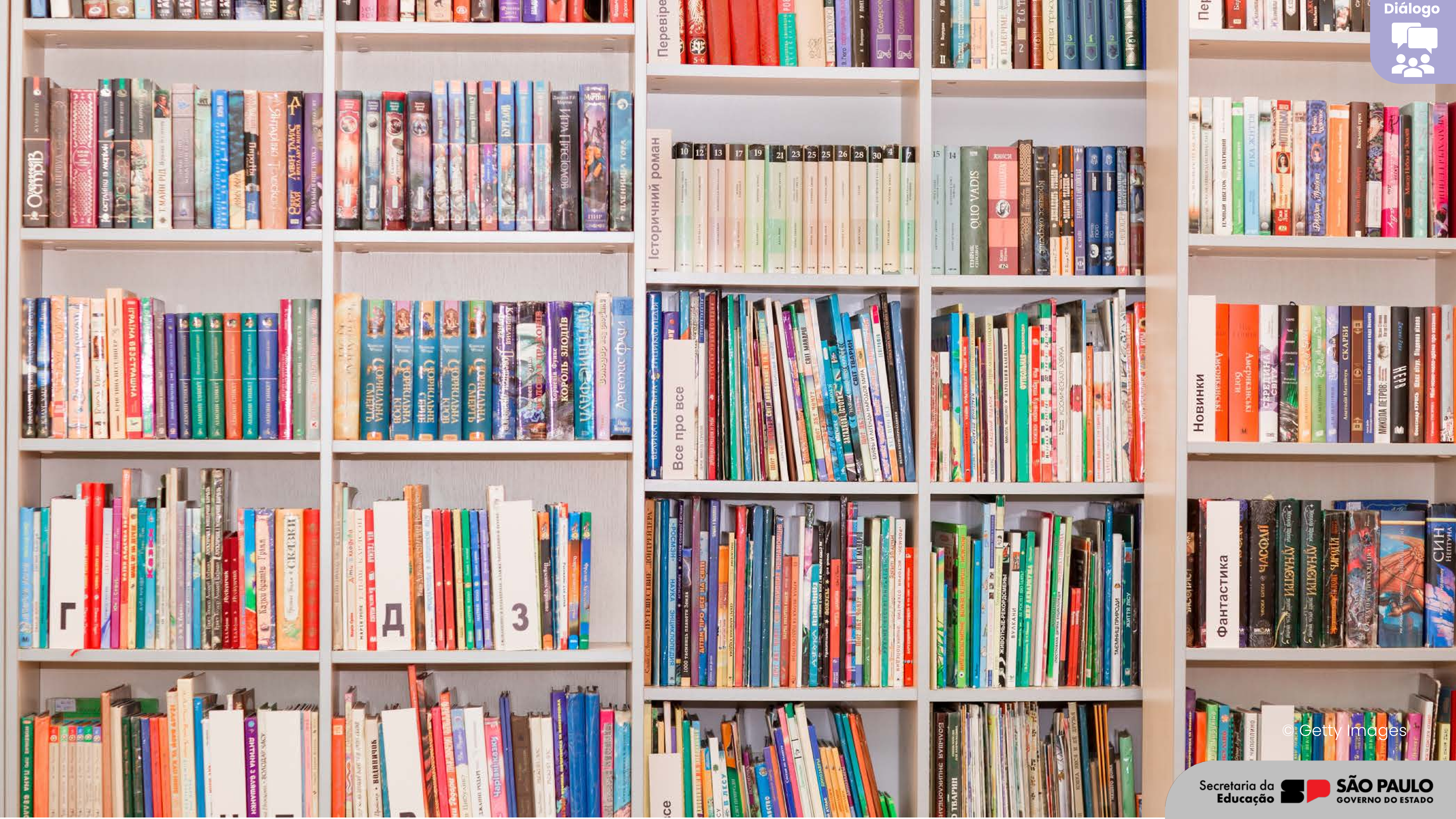
Competência Técnica

- Ser proficiente em linguagens de programação para manipular e analisar grandes conjuntos de dados.

Competência Socioemocional

- Demonstrar resiliência para lidar com pressões e enfrentar novos desafios, bem como com frustrações quando um projeto de Ciência de Dados falhar.

Diálogo



Secretaria da Educação
SÃO PAULO
GOVERNO DO ESTADO

Primeiras **ideias**

Estante bagunçada

- Imagine que essa seja sua estante de livros. Qual seria o seu método para encontrar um livro específico?
- Quão custoso é organizar a estante?
- É possível organizar a estante de mais de uma forma?

Ponto de
partida

Como organizar uma lista de números

9 1 6 7 5 4 8 2 3

Elaborado especialmente para este curso.

Construindo
o **conceito**

Complexidade de algoritmos

- A complexidade de um algoritmo nos ajuda a entender **"quanto" o algoritmo "custa"** em termos de recursos necessários (**tempo de execução** e **memória usada**).
- Como exemplo, podemos pensar em como uma receita pode variar em complexidade: uma salada é rápida e simples de preparar, enquanto um assado pode ser demorado e complexo.

Construindo o **conceito**

Por que saber a complexidade é importante?

- Compreender a complexidade permite prever como o algoritmo escalará à medida que o tamanho da entrada aumenta.
- Isso é crucial para o desempenho em aplicações reais, em que escolher o algoritmo errado pode significar a diferença entre uma resposta instantânea e uma que nunca termina.

Dica

Na área de dados, em particular, lidamos com volumes de dados muito grandes. Sendo assim, é muito importante estarmos antenados nas tendências sobre a eficiência de algoritmos.

Construindo o **conceito**

Tipos de complexidade: tempo e espaço

- Quando avaliamos a complexidade de um algoritmo, podemos considerar tanto o tempo quanto o espaço relacionados à sua execução:
- **Complexidade de tempo**: quanto tempo um algoritmo leva para completar com base no tamanho da entrada. Por exemplo, ler um livro com n páginas, página por página, consome um certo tempo por página vezes n páginas.
- **Complexidade de espaço**: quanta memória um algoritmo precisa para ser executado. Em um exemplo, ler um e-book consome n kilobytes.

Construindo o **conceito**

Medindo a complexidade: notação Big O

- Há diversas formas de avaliar o desempenho de um algoritmo. Nesta aula, veremos apenas o que chamamos de Big O.
- Big O descreve o pior cenário de crescimento do tempo de execução ou espaço de um algoritmo à medida que o tamanho da entrada aumenta.
- Por exemplo, O(n) significa que o tempo ou o espaço cresce linearmente com o aumento do tamanho da entrada. Por exemplo, em uma lista de n elementos, o algoritmo custa n operações.

Construindo
o **conceito**

Considere que queremos encontrar o número 4 na lista a seguir

9 1 6 7 5 4 8 2 3

Elaborado especialmente para este curso.

Construindo o **conceito**

Analisando a complexidade de uma busca linear

- A busca linear **verifica cada elemento** de uma lista sequencialmente **até encontrar o elemento desejado** ou chegar ao final da lista.

- Se tivermos uma lista de n elementos, no pior caso, precisamos verificar todos os n elementos. Portanto, a complexidade de tempo é O(n).

Construindo
o **conceito**

Estratégias para reduzir a complexidade

Para reduzir a complexidade, podemos usar algumas estratégias:

- **usar uma estrutura de dados mais adequada:** por exemplo, podemos usar uma lista ordenada previamente em vez de uma lista aleatória para buscas mais rápidas;
- **Dividir o problema:** usar técnicas como divisão e conquista para quebrar o problema em partes menores e mais gerenciáveis.

Construindo
o **conceito**

Limitações da análise de complexidade

Quando analisamos complexidade, há algumas limitações:

- a análise de complexidade se concentra em **como o algoritmo escala**, não necessariamente em como ele se comporta com entradas pequenas;
- constantes e fatores de baixa ordem são ignorados, o que pode ser relevante em aplicações do mundo real.

Registro

Vamos fazer um **quiz**

O que é essencial para evitar uma recursão infinita?

- **Uma condição de parada.**
- **Uma função auxiliar.**
- **Um Loop For.**
- **Uma chamada de sistema.**

Vamos fazer um **quiz**

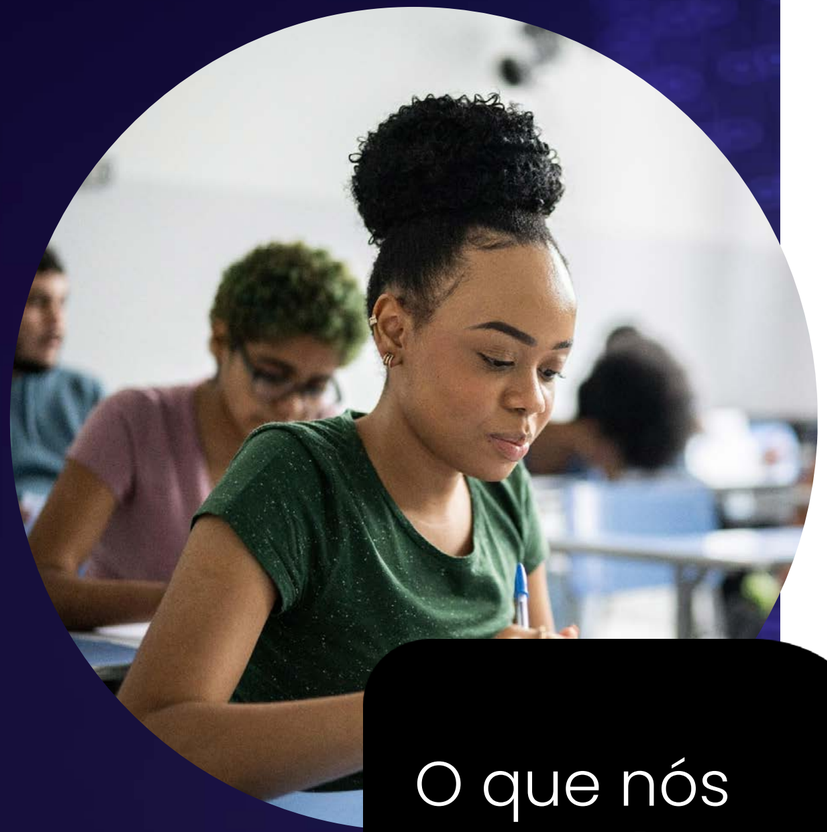
Qual das seguintes afirmações sobre recursão é verdadeira?

- Recursão sempre usa menos memória que iteração.
- Todas as funções recursivas podem ser escritas como iteração de forma simples.
- Recursão não pode ser usada em linguagens modernas.
- Recursão pode resolver problemas com subproblemas menores.

Vamos fazer um **quiz**

Quando a iteração é preferível à recursão?

- Quando o problema não tem uma condição base clara.
- Quando estamos trabalhando com problemas de divisão e conquista.
- Quando a eficiência de memória e tempo é uma prioridade.
- Quando a linguagem de programação não suporta loops.



© Getty Images

O que nós **aprendemos hoje?**

Então ficamos assim...

1. Aprendemos que a complexidade de algoritmos pode ser medida de diversas formas. Uma delas, é a notação Big O.

2. A notação Big O serve como ferramenta para medir o comportamento de um algoritmo em relação ao tamanho da entrada.

3. Discutimos que, apesar de ser um guia valioso, a análise pode não captar todos os aspectos do desempenho do algoritmo e deve ser complementada por testes empíricos.

Saiba mais

Descubra a importância da análise da complexidade dos algoritmos e como ela pode transformar seu desenvolvimento de software. Leia mais no artigo da Alura disponível no link abaixo:

MATOS, B. R. S. Análise de complexidade de algoritmos: qual é a importância?. *Alura*, 23 maio 2022. Disponível em: https://www.alura.com.br/artigos/analise-complexidade-algoritmos-qual-importancia/. Acesso em: 25 jun. 2024.

Referências da aula

Identidade visual: Imagens © Getty Images

FORBELLONE, A. L. V.; EBERSPÄCHER, H. F. *Lógica de programação*: a construção de algoritmos e estruturas de dados com aplicações em Python. Porto Alegre: Bookman, 2022.

Educação Profissional Paulista

Técnico em
Ciência de Dados

Secretaria da Educação

SÃO PAULO
GOVERNO DO ESTADO