



EXERCÍCOS DE FIXAÇÃO

ESTRUTURA DE DADOS

CST em Desenvolvimento de Software Multiplataforma



PROF. Me. TIAGO A. SILVA



ÁRVORES BINÁRIAS E ÁRVORES AVL

Exercícios usando as classes implementadas

EXERCÍCIO DE FIXAÇÃO: ÁRVORES BINÁRIAS E ÁRVORES AVL

- i) Construindo e Percorrendo uma Árvore Binária de Busca Utilizando a classe `BinaryTree` do arquivo `BinaryTree.js`, instancie uma nova árvore e insira os seguintes valores na ordem apresentada: 15, 10, 20, 5, 12, 18, 25. Depois de inserir todos os valores, chame os três métodos de percurso (`inOrder`, `preOrder` e `postOrder`) e anote a sequência de números impressa por cada um deles.
- ii) Remoção em Árvore Binária de Busca Com a árvore criada no exercício anterior, utilize o método `remove` para remover os seguintes nós: 6, 15, 20. Desenhe a árvore resultante após cada remoção para visualizar a nova estrutura.
- iii) Buscando Elementos Ainda com a árvore do exercício i), utilize o método `search` para verificar se os valores 12 e 30 existem na árvore. Anote o retorno (`true` ou `false`) para cada busca.

EXERCÍCIO DE FIXAÇÃO: ÁRVORES BINÁRIAS E ÁRVORES AVL

- 4) Diferenças entre BinaryTree e AVLTree. Crie duas instâncias, uma de BinaryTree e outra de AVLTree (AVLTree.js). Insira os seguintes valores em ambas as árvores, na mesma ordem: 10, 20, 30, 40, 50. Desenhe a estrutura final de cada uma das árvores. Qual a principal diferença que você observa entre elas e por que ela ocorre?
- 5) Balanceamento da AVL. Utilizando a classe AVLTree, insira os valores 30, 20, 10. Qual tipo de rotação é realizada para manter a árvore balanceada? Agora, insira o valor 25. Qual o novo tipo de rotação que ocorre?



GRAFOS

Por que seu código demora para executar?

EXERCÍCIO DE FIXAÇÃO: GRAFOS

- na) Modelando uma Rede Social Utilizando a classe Grafo do arquivo Grafo.js, modele uma pequena rede de amigos
- a) Adicione os seguintes vértices
- b) Adicione as seguintes arestas para representar as amizades
- Ana e Zeto
 - Ana e Caio
 - Zeto e Dani
 - Caio e Dani
 - Edu e Ana
- c) Utilize o método **imprimirGrafo()** para visualizar a lista de adjacências
- d) Simule que 'Caio' deixou a rede social removendo o vértice 'Caio' e todas as suas conexões
- e) Imprima o grafo novamente

EXERCÍCIO DE FIXAÇÃO: GRAFOS

7) Grafos Ponderados (Conceitual) A classe Grafo fornecida representa um grafo não dirigido e não ponderado. Como você modificaria a estrutura **adjacencia** e o método **adicionarAresta** para que o grafo pudesse armazenar pesos nas arestas (grafo ponderado)? Por exemplo, para representar a distância entre cidades? Descreva a mudança necessária no código.



BUSCA E ORDENAÇÃO

Por que seu código demora para executar?

EXERCÍCIO DE FIXAÇÃO: ALGORITMOS DE BUSCA

- pa) Exercício 6 Busca Sequencial vs Binária Considere o seguinte array desordenado
[45, 23, 11, 89, 77, 98, 4, 28, 65, 43]
- a) Utilizando o método de busca sequencial (do arquivo buscasf.ad) quantos passos (comparações) são necessários para encontrar o valor 10?
- ba) Agora ordene o array Utilizando o método de busca binária, quantos passos são necessários para encontrar o mesmo valor?
- qa) Quando a Interpolação Falha A busca por interpolação (buscasf.interpolacao) é eficiente para arrays com valores uniformemente distribuídos Crie um array ordenado onde a busca por interpolação teria um desempenho ruim (próximo ao da busca sequencial) Por exemplo **[1, 2, 3, 4, 5, 1000]** Explique por que a fórmula de interpolação não consegue estimar bem a posição do elemento nesse caso

EXERCÍCIO DE FIXAÇÃO: ALGORITMOS DE ORDENAÇÃO

- i) Ordenando com Bubble Sort Utilize a classe Sorter (Sorter.js) para ordenar o array **[5, 1, 4, 2, 8]** com o método bubbleSort. Anote o estado do array após cada "passada" completa do algoritmo (ou seja, cada vez que o loop externo termina) para entender como os elementos se movem.
- ii) Ordenando com Quick Sort Utilize o método Sorter.quickSort para ordenar o array **[7, 2, 1, 6, 8, 5, 3, 4]**. Descreva como o primeiro pivô (o último elemento) divide o array em duas sub-listas (menores e maiores).
- iii) Ordenando com Merge Sort Utilize o método Sorter.mergeSort para ordenar o mesmo array do exercício anterior **[7, 2, 1, 6, 8, 5, 3, 4]**. Mostre como o array é dividido recursivamente até chegar em arrays de um único elemento e como eles são mesclados (merge) de volta, já ordenados.

NOTAÇÃO ASSINTÓTICA

Por que seu código demora para executar?

EXERCÍCIO DE FIXAÇÃO: NOTAÇÃO BIG O

1) Complexidade dos Métodos de Busca. Analise os métodos na classe `Busca`. Qual é a complexidade de tempo (O) no pior caso para

- a) `sequencial()`
- b) `binaria()`
- c) `interpolacao()`

Justifique sua resposta para cada um.

2) Complexidade dos Métodos de Ordenação. Analise os métodos na classe `Sorter`. Qual é a complexidade de tempo (O) no pior caso para

- a) `bubbleSort()`
- b) `quickSort()`
- c) `mergeSort()`

Explique por que o **quickSort** pode ter um pior caso diferente de seu caso médio.

3) Análise de Complexidade na **BinaryTree**. Analisando o arquivo `BinaryTree.js`, qual a complexidade de tempo (O) no caso médio e no pior caso para os métodos **insert**, **search** e **remove**? Explique o que causa a diferença entre o caso médio e o pior caso.

OBRIGADO!

- Encontre este **material on-line** emr
– Slidesr Plataforma Microsoft Teams
- Em caso de **dúvidas**, entre em contator
– **Prof. Tiago** tiago.silva23@fatecsp.gov.br

