Relatório comparação de árvores binárias:

Alunos:

Franklin Jerônimo Belasque Bauch Vieira.
Cauã Gregorio.
Guilherme Schwarz.
Julia Cristina Moreira

Código:

https://github.com/xarss/Binary-Trees

1. **Introdução:**
   As árvores binárias e as árvores AVL são estruturas de dados fundamentais em ciência da computação, utilizadas para armazenar e organizar informações de maneira hierárquica. Enquanto as árvores binárias são mais simples e flexíveis, as árvores AVL são uma extensão delas, garantindo um balanceamento que otimiza operações de busca. Este relatório visa comparar ambas as estruturas, destacando suas características, vantagens e desvantagens.

2. **Descrição do código:**
   O código tem 4 classes: App, Node, Binary Tree e AVLBinaryTree.

   App sendo onde se localiza nosso main e métodos que disparam os inserts, removes e finds aleatórios pedidos.

   Node sendo nossa classe de nós, nela se localiza todos os métodos de insert, remove, find e o balanceamento da AVL.

   BinaryTree é a classe que age como nossa árvore binária contêm nosso root, nela é onde realizamos algumas verificações antes de realizar os métodos no Node.

   AVLBinaryTree realiza praticamente a mesma função que a BinaryTree com a adição de verificações para o balanceamento.

3. **Critério para os testes:**
   Foram realizados teste criando árvores de 100, 500, 1000, 10.000 e 20.000 elementos gerados aleatoriamente, medindo o tempo para todos os métodos inserir, remover e achar(find).

4. **Resultado dos Testes**

Num de elementos	Ação	Tempo(ms) Árvore Binária	Tempo(ms) AVL
100	inserir	1	1
100	remover	1	0
100	find	0	0
500	inserir	1	7
500	remover	0	10
500	find	1	2
1000	inserir	101	241
1000	remover	37	452
1000	find	4	6
10000	inserir	3507	8649
10000	remover	3007	9575
10000	find	2	2
20000	inserir	5327	12653
20000	remover	7345	17542
20000	find	2	2

5.Qual funciona melhor?Em qual situação? Porque?

A tabela mostra os resultados dos testes de tempo de execução para duas estruturas de dados: Árvore Binária e AVL, com diferentes quantidades de elementos e operações (inserir, remover e encontrar). Ao analisar a tabela, podemos observar o seguinte:

Para um número pequeno de elementos (100 e 500), os tempos de execução para as operações nas duas estruturas são bastante semelhantes, com diferenças mínimas. Isso sugere que, para pequenas quantidades de dados, ambas as estruturas são eficientes.

No entanto, à medida que o número de elementos aumenta (1000, 10000 e 20000), começamos a ver diferenças mais significativas nos tempos de execução. Especificamente, a Árvore AVL tende a ter tempos de execução mais longos, especialmente para as operações de inserção e remoção.

Isso pode ser devido ao fato de que a Árvore AVL realiza rotações para manter-se balanceada após cada inserção ou remoção. Embora isso garanta que a árvore esteja sempre balanceada (o que é benéfico para operações de busca), pode adicionar um overhead nas operações de inserção e remoção.