

# **Relatório I LAED II**

AUTOR:

Pedro Henrique Maia Duarte

Professor: THIAGO DE SOUZA RODRIGUES

# Prática 1 – Implementação do TAD

## Árvore Binária de Pesquisa

A Árvore Binária de Pesquisa é uma estrutura de dados de árvore binária baseada em nós. Neste sentido, cada nó possui dois outros filhos nós e todos os nós da subárvore esquerda possuem um valor numérico inferior ao nó raiz e todos os nós da subárvore direita possuem um valor superior ao valor do nó. É uma estrutura de dados muito eficiente para armazenar informação, particularmente adequada quando existe uma necessidade de considerar todos ou alguma combinação de acesso direto e sequencial eficientes, facilidade de inserção e retirada de itens.

Sendo assim, a árvore binária foi implementada em linguagem Java. Desta maneira, para testá-la, elaborou-se dois algoritmos de inserção de itens: um que insere itens de modo ordenado e outro que insere de modo aleatório.

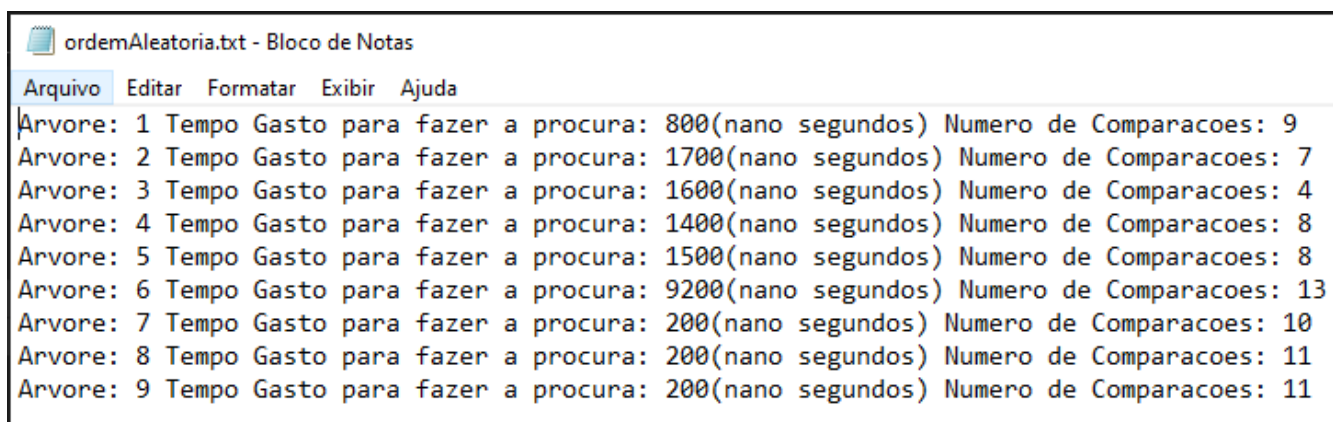
Ademais, na implementação da árvore binária, buscou-se analisar a o tempo gasto para encontrar um item arbitrário na árvore. Por essa perspectiva, com o intuito de analisar a quantidade de comparações realizadas e o tempo gasto nas operações de busca nas árvores, as informações, foram feitas buscas em árvores de 1000 até 9000 elementos (as quais os itens foram inseridos ordenadamente e não ordenadamente) e foram extraídos os tempos gastos nas buscas de cada .

ordemOrdenada.txt - Bloco de Notas

Arquivo Editar Formatar Exibir Ajuda

```
Arvore: 1 Tempo Gasto para fazer a procura: 86300(nano segundos) Numero de Comparacoes: 1000
Arvore: 2 Tempo Gasto para fazer a procura: 47500(nano segundos) Numero de Comparacoes: 2000
Arvore: 3 Tempo Gasto para fazer a procura: 63500(nano segundos) Numero de Comparacoes: 3000
Arvore: 4 Tempo Gasto para fazer a procura: 32800(nano segundos) Numero de Comparacoes: 4000
Arvore: 5 Tempo Gasto para fazer a procura: 24700(nano segundos) Numero de Comparacoes: 5000
Arvore: 6 Tempo Gasto para fazer a procura: 29600(nano segundos) Numero de Comparacoes: 6000
Arvore: 7 Tempo Gasto para fazer a procura: 34700(nano segundos) Numero de Comparacoes: 7000
Arvore: 8 Tempo Gasto para fazer a procura: 39400(nano segundos) Numero de Comparacoes: 8000
Arvore: 9 Tempo Gasto para fazer a procura: 44700(nano segundos) Numero de Comparacoes: 9000
```

*Dados obtidos por inserção em ordem Ordenada - Imagem 01*



Arquivo	Editar	Formatar	Exibir	Ajuda
Arvore: 1	Tempo	Gasto para fazer a procura:	800(nano segundos)	Numero de Comparacoes: 9
Arvore: 2	Tempo	Gasto para fazer a procura:	1700(nano segundos)	Numero de Comparacoes: 7
Arvore: 3	Tempo	Gasto para fazer a procura:	1600(nano segundos)	Numero de Comparacoes: 4
Arvore: 4	Tempo	Gasto para fazer a procura:	1400(nano segundos)	Numero de Comparacoes: 8
Arvore: 5	Tempo	Gasto para fazer a procura:	1500(nano segundos)	Numero de Comparacoes: 8
Arvore: 6	Tempo	Gasto para fazer a procura:	9200(nano segundos)	Numero de Comparacoes: 13
Arvore: 7	Tempo	Gasto para fazer a procura:	200(nano segundos)	Numero de Comparacoes: 10
Arvore: 8	Tempo	Gasto para fazer a procura:	200(nano segundos)	Numero de Comparacoes: 11
Arvore: 9	Tempo	Gasto para fazer a procura:	200(nano segundos)	Numero de Comparacoes: 11

*Dados obtidos por inserção em ordem Aleatória – imagem 02*

Nas árvores em que os elementos foram inseridos ordenadamente, o elemento que contém o maior valor será o último componente inserido na árvore 9, onde terá como conteúdo a chave 9000. Ademais, quando inserimos na árvore elementos aleatoriamente, o elemento que pode conter a chave 9000, estará presente na extremidade à direita das sub-árvores da árvore 9. Sendo assim, para testar o tempo de busca, iremos procurar em todas as árvores criadas, um elemento cujo valor é 20000, para garantir que uma extremidade da árvore será percorrida totalmente.

Sendo assim, pretendemos analisar quantas comparações de elementos são realizadas em cada árvore, e esperamos que nas árvores de inserção ordenada, o número de comparações seja linear, pois estaremos sempre buscando até o final da árvore, realizando a comparação com todos os elementos existentes. Enquanto isso, esperamos que o número de comparações em árvores com inserção aleatória seja inferior, pois a inserção desses elementos, preencherá as sub-árvores a esquerda, e não somente a direita, como verificado em inserção ordenada. Sendo assim, temos:

### Número de comparações entre as árvores

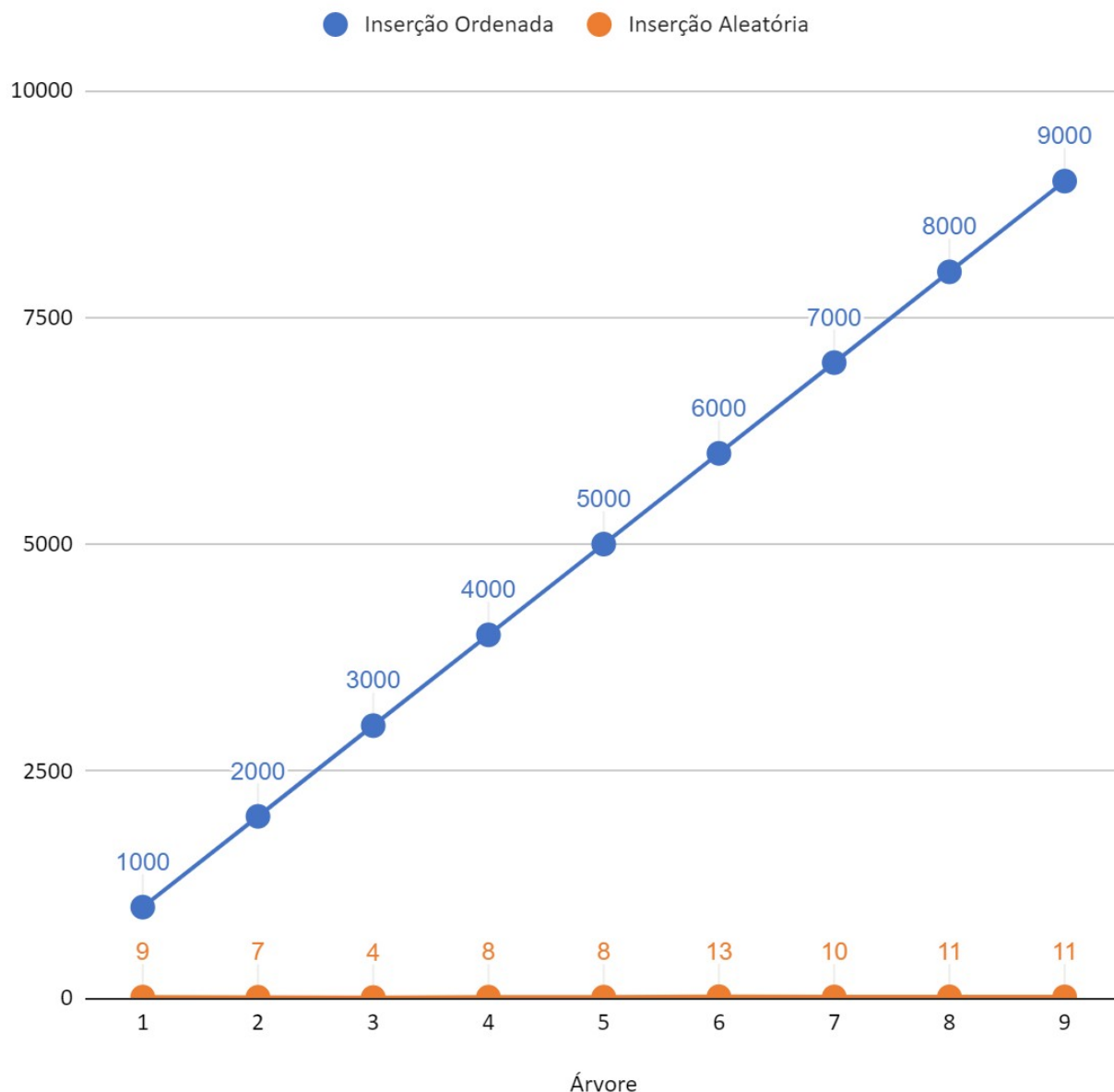


Gráfico 01

O gráfico apresentou os resultados esperados, garantindo que todos os elementos foram inseridos nas árvores, e consequentemente, em ambos os casos, foram comparados com o valor de 9000, e não foi encontrado a chave procurada. Sendo assim, temos um gráfico linear crescente para a inserção ordenada. Enquanto isso, verificamos que por inserção aleatória, os valores de comparação são inferiores do que a inserção ordenada, devido ao fato que a sub-árvores à direita são menores, pois durante a inserção, as sub-árvores a esquerda da árvore também foi preenchida, diferentemente da árvore por inserção ordenada. Isso comprova que a inserção por elementos aleatórios é superior do que

## CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

uma lista linear, devido ao fato que existe menos comparações para se encontrar um elemento existente, ou não existente.

Logo, o gasto em procura de elementos por inserção ordenada é  $O(n)$  e o gasto por inserção aleatória é  $O(\log(n))$ .

Além disso, podemos analisar o tempo de máquina gasto na inserção por ambos os métodos:

### Tempo de procura(ns) entre as árvores

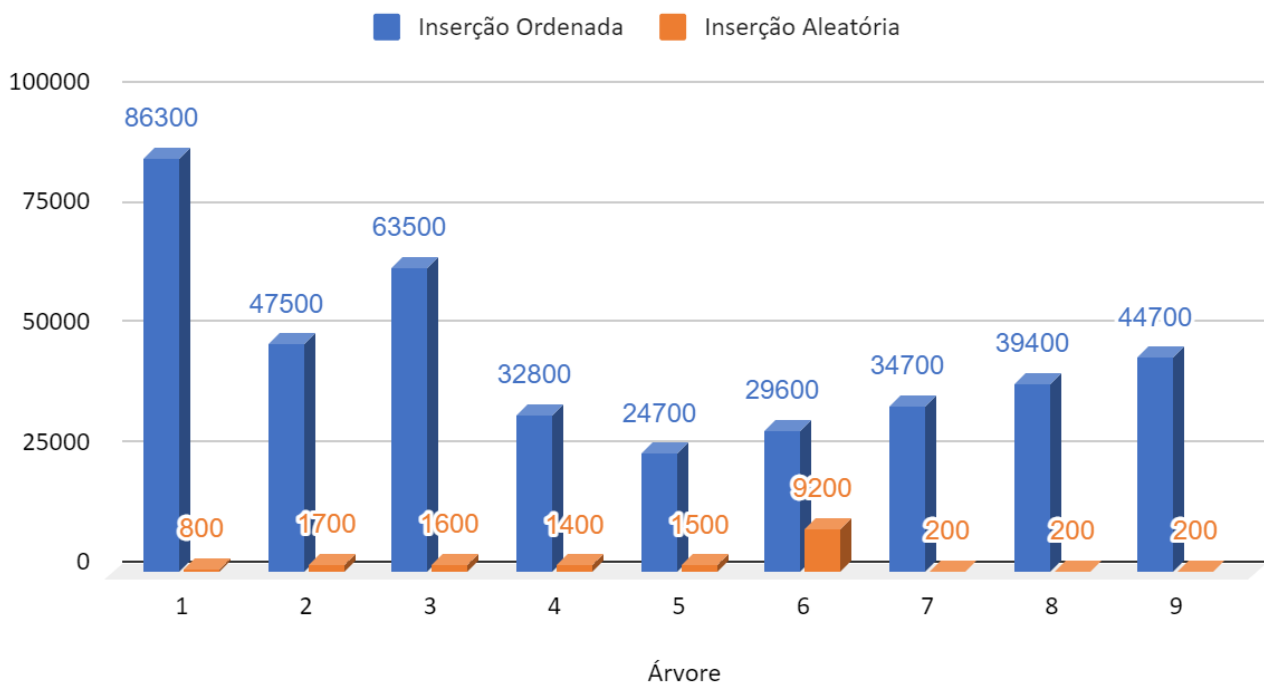


Gráfico 02

Analisando o gráfico, percebemos que a procura em árvores por inserção aleatória levou um tempo menor que a inserção ordenada, garantido que as sub-árvores à direita da árvore por inserção aleatória são menores que as sub-árvores à direita da árvore por inserção ordenada, pois em ambas as árvores, o maior valor contido estará presente na extremidade a direita da árvore. Entretanto, existe uma discrepância grande entre os valores.

Logo, concluímos que a grandeza tempo de relógio não é capaz para inferir em uma conclusão de complexidade de busca, por apresentar resultados imprecisos e discrepantes, atuando como apenas uma informação e sem produzir uma relevância direta. Sendo assim, o número de comparações é mais adequado para analisar a complexidade de busca, onde podemos comparar o número de comparações executadas para cada árvore para obter o mesmo resultado, como nesse caso, é procurar por toda extensão da árvore, um elemento que nunca estará presente.