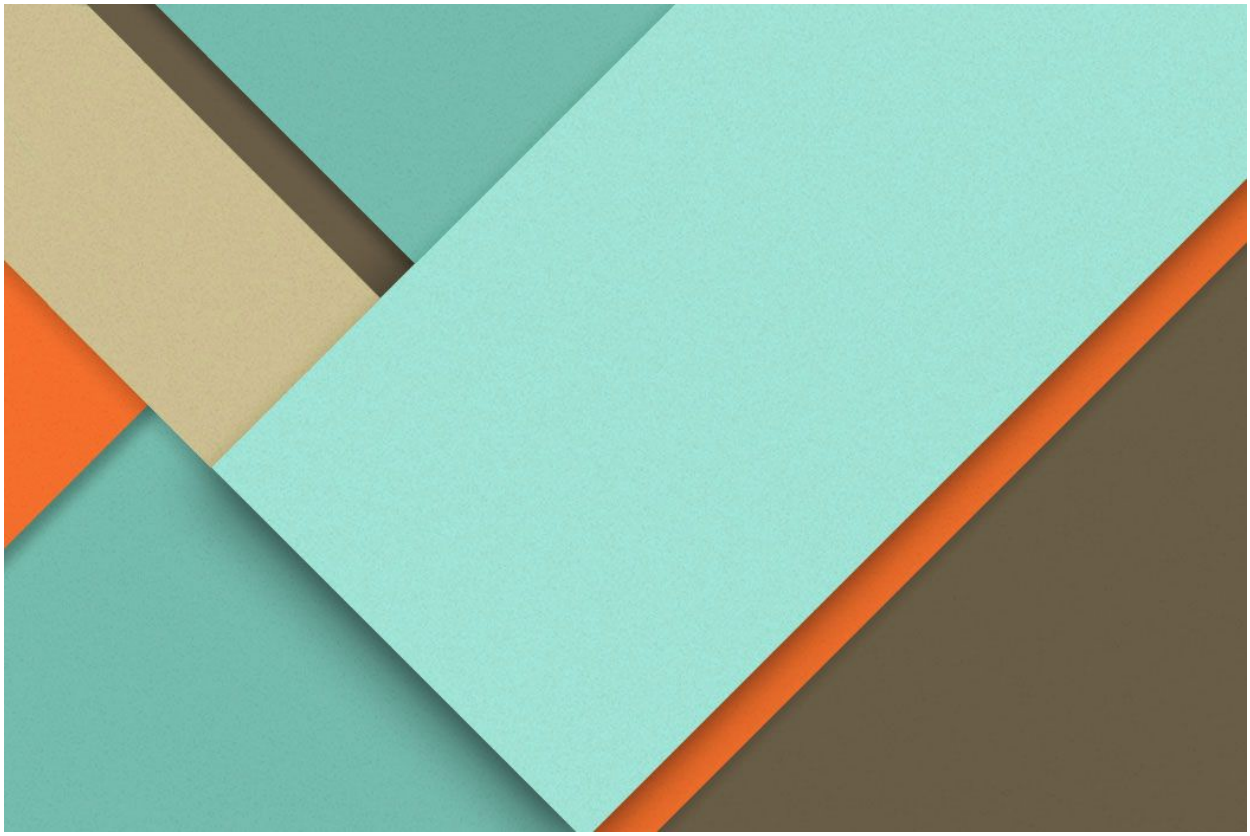


Relatório da Prática

22.08.2018

Daniel Santana Santos

Centro Federal de Educação Tecnológica de Minas Gerais

Introdução

Atualmente existe uma grande demanda por recuperação de informações de maneira cada vez mais rápida. Neste contexto de grande fluxo de informações sendo geradas a cada segundo, a busca e manipulação de dados não pode ser feita através de um array linear. Com isto surgiram as árvores de pesquisa, que possibilitam a busca, bem como outras operações sobre dados em tempo até *log (n).* É essencial a um estudante de engenharia de computação que tenha contato com estas estruturas, entendendo o seu comportamento, funcionamento, e pontos positivos e negativos de cada modelo de árvore de pesquisa.

Objetivos

O presente trabalho tem como o objetivo descrever os resultados obtidos através da prática 1 da disciplina Laboratório de Algoritmos e Estrutura de Dados 2. Esta prática, por sua vez, propôs aos estudantes um contato com a árvore de pesquisa binária sem balanceamento.

O objetivo da prática era que os discentes criassem árvores binárias de pesquisa sem balanceamento na linguagem Java, com um número *n* de elementos cada vez maior, e para cada valor de *n,* observassem o tempo gasto, e o número de comparações realizadas pelo algoritmo, ao buscar por um elemento inexistente na estrutura.

Resultados

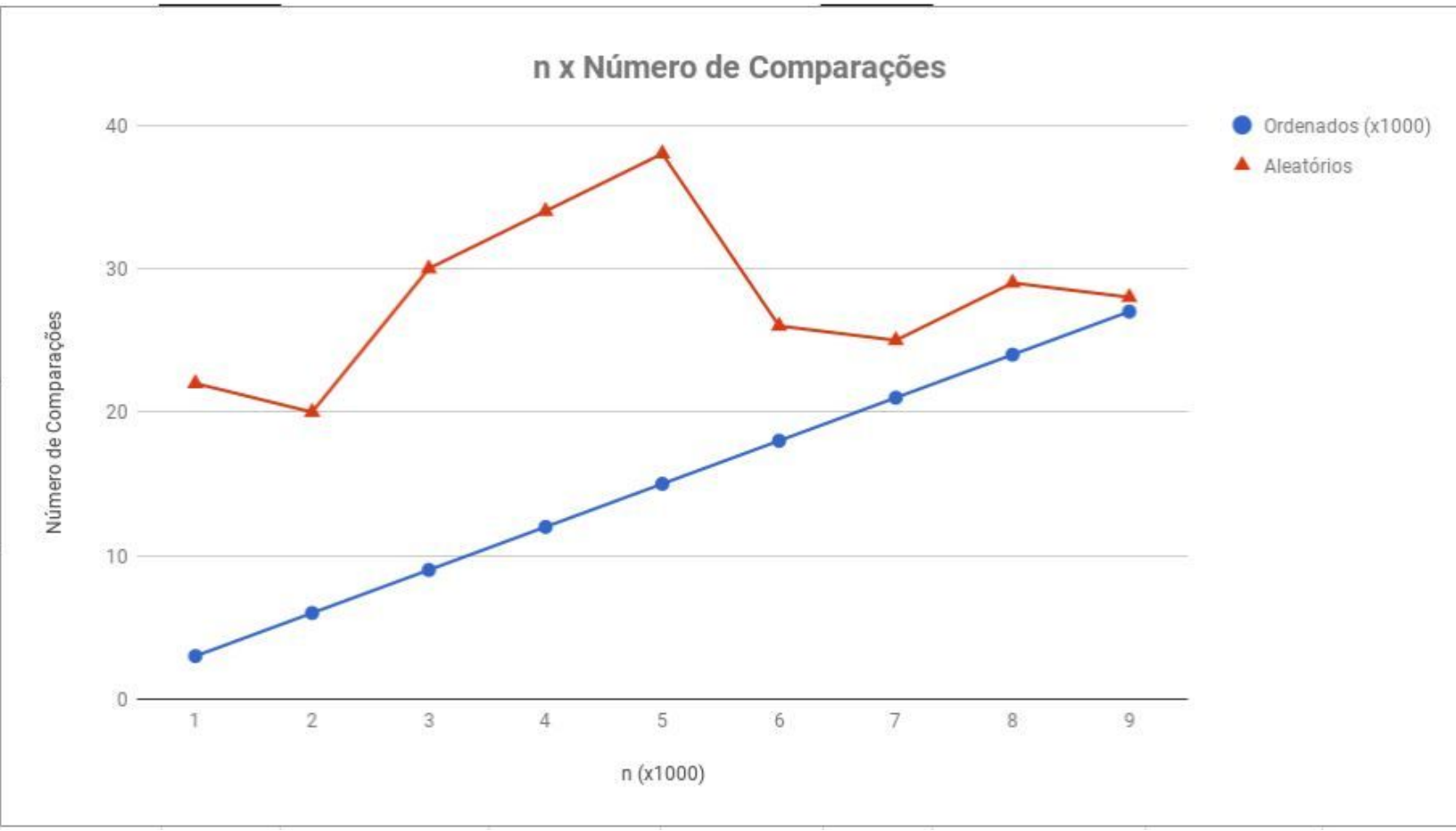
Para cada árvore, e cada valor de *n* nestas, o código foi executado um total de 5 vezes, e a média entre os valores obtidos de tempos e comparações realizadas, foi inserida em uma tabela, que posteriormente foi usada para gerar dois gráficos: n x Número de Comparações, e n x Tempo Gasto.

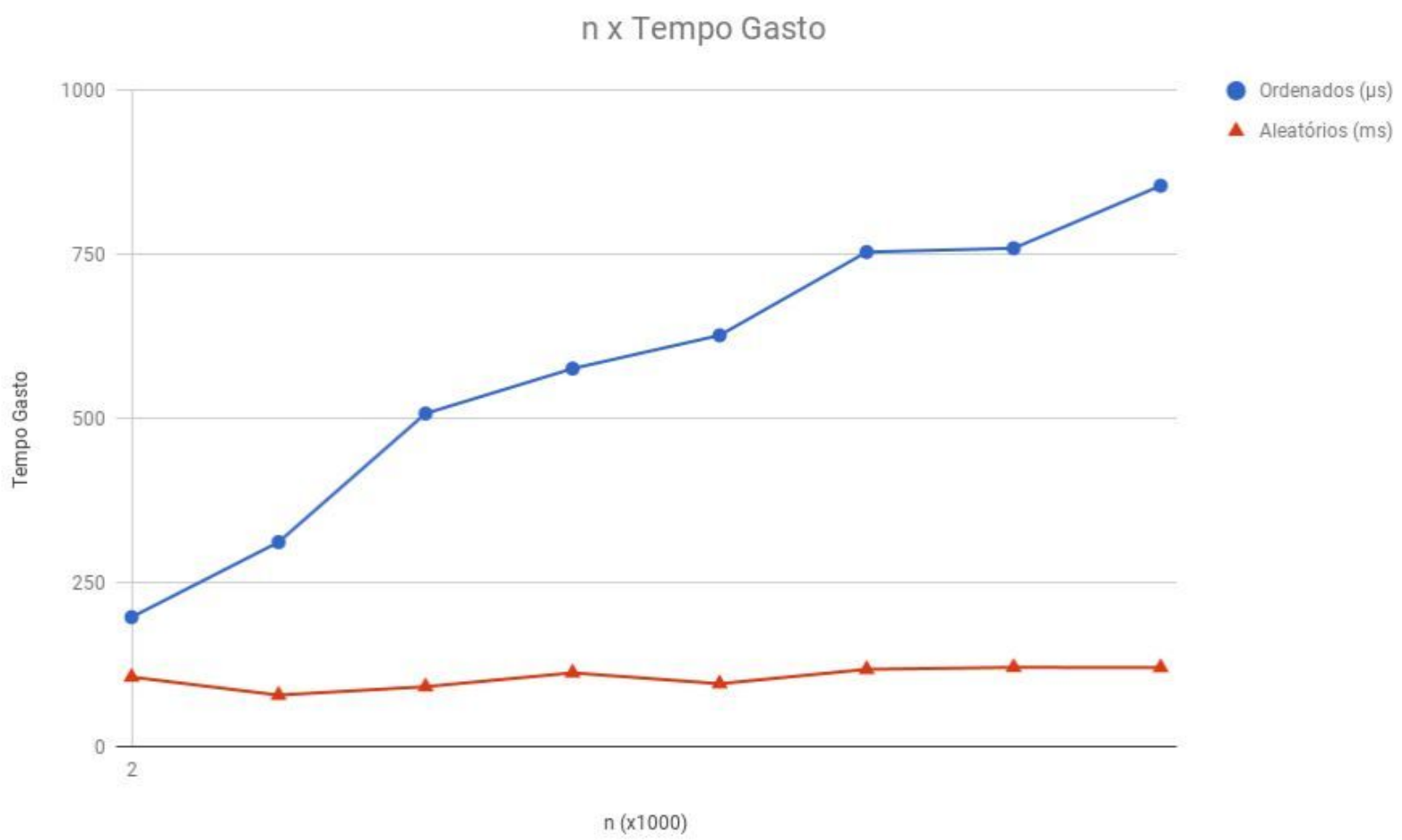
No gráfico n x Número de Comparações, pode-se observar que para elementos inseridos de maneira ordenada e crescentes, ao buscar por um elemento inexistente na árvore, esta executava cada vez mais comparações na busca, de maneira linear. Por outro lado, com elementos inseridos de maneira aleatória, o número de comparações realizadas na busca foi consideravelmente menor, e sem padrão evidente. Vale ressaltar que no caso em que elementos foram adicionados de maneira crescente, a árvore sempre realizou mais de mil comparações, de forma que, no gráfico, foi usada uma escala 1x1000 para representar estes dados.

Já no gráfico n x Tempo Gasto, podemos observar que no caso da árvore com elementos ordenados, o tempo gasto cresceu de acordo com o número de elementos inseridos. Na

árvore com elementos aleatórios, o tempo gasto na busca foi consideravelmente menor (a escala utilizada neste caso teve de ser reduzida), e não houve comportamento padronizado.

Ambos os gráficos podem ser vistos a seguir:




n x Tempo Gasto
1000
750
500
250
0
Tempo Gasto
2
n (x1000)
Ordenados (µs)
Aleatórios (ms)