CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
CAMPUS TIMÓTEO

Raphael Gomes Wagner e José Geraldo Duarte Junior

ARVORES BINÁRIA: INSERÇÃO E BUSCA

Timóteo

2022

Raphael Gomes Wagner e José Geraldo Duarte Junior

ARVORES BINÁRIA: INSERÇÃO E BUSCA

Trabalho 3 de Algoritmos e Estrutura de Dados sobre Polias, Roldanas e Atrito no curso de Engenharia de Computação no Centro Federal de Educação Tecnológica de Minas Gerais

Orientador: Gustavo Martins

Timóteo

2022

.

.

.

Dedico a
Leandro Santana, Arthur Gomes, José Geraldo Duarte Junior
Felipe Reggiane, Daniel Vidal, Guilherme Heringer, Nathan Teixeira e Pedro Arthur.

Agradecimentos

Agradeço a todos da dedicação por terem passado em AED I comigo, e aos dois primeiros por terem ajudado no estudo da matéria, me ensinando.

1 Introdução

Esse trabalho foi desenvolvido em conjunto, feito por Raphael Gomes e José Geraldo, no intuito de executar processos com uma Arvore Binária, desenvolvida pela Professora Divani Barbosa Gavinier(GAVINIER, 2020), porém editada por ambos, cada um fazendo as devidas edições para entendimento individual do código. Essa implementação foi utilizada para medir tempos de execução sobre a inserção e busca, e assim fazendo a média e desvio padrão desses tempos.

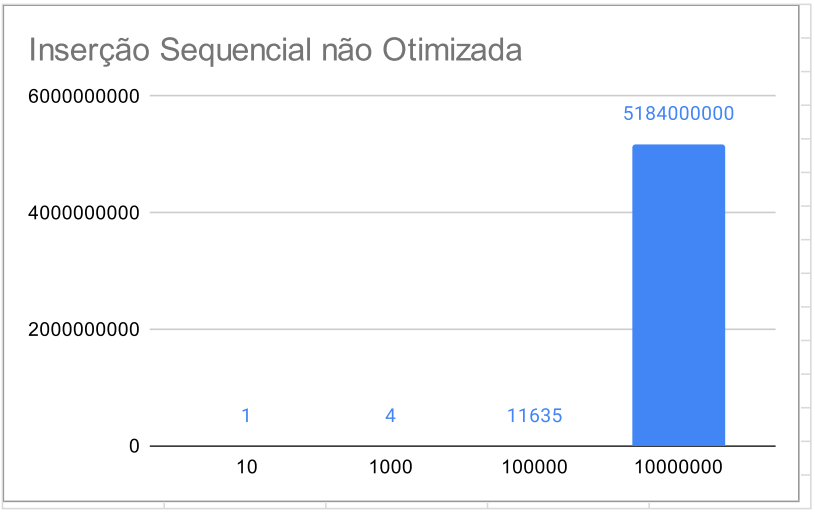
Com o intuito de executar a atividade em diferentes ambientes, foram utilizadas 3 tipos de arvores diferentes, e elas são: Arvore com elementos sequenciais, Arvore com elementos randômicos, e Arvore balanceada com elementos sequenciais. Em teoria, deveriam ser utilizados arvores com os seguintes tamanhos:

1. 10^1 elementos
2. 10^3 elementos
3. 10^5 elementos
4. 10^7 elementos
4. 10^9 elementos

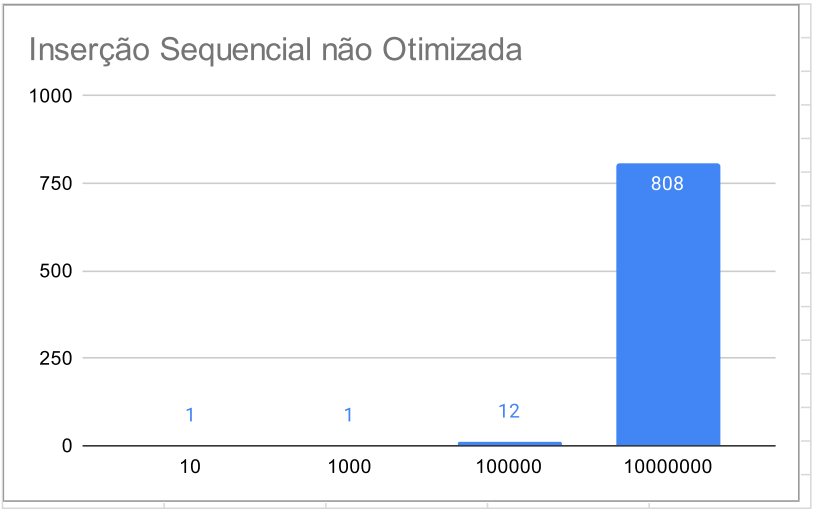
2 Desenvolvimento

2.1 Inserção Sequencial

Iniciando o desenvolvimento da prática, as arvores foram criadas na IDE assim como os vetores de tamanhos diversos para armazenar os valores que serão inseridos na arvore. Porém, o Java não permite a criação de vetores em 10^9 elementos, então todas as arvores criadas, foram geradas até 10^7 elementos. A partir disso, foi utilizada a função de inserção da classe arvore para inserir os elementos, mas como a inserção sequencial gastava muito tempo de execução, foi necessário criar uma função exclusiva de inserção para a arvore sequencial, o que reduziu bastante o tempo de execução, fazendo assim que fosse possível fazer a inserção sequencial em até 10^7 elementos. Abaixo se tem uma comparação dos dois tipos de métodos para a inserção sequencial:
Tempo de execução para inserção não otimizada:
O tempo de execução de uma arvore com 10 elementos em milissegundos é: 1
O tempo de execução de uma arvore com 1000 elementos em milissegundos é: 4
O tempo de execução de uma arvore com 100000 elementos em milissegundos é: 11635
O tempo de execução de uma arvore com 10000000 elementos em milissegundos é: 5184000000 (Tempo estimado)



Tempo de execução para inserção otimizada:
O tempo de execução de uma arvore com 10 elementos em milissegundos é: 1
O tempo de execução de uma arvore com 1000 elementos em milissegundos é: 1
O tempo de execução de uma arvore com 100000 elementos em milissegundos é: 12
O tempo de execução de uma arvore com 10000000 elementos em milissegundos é: 808

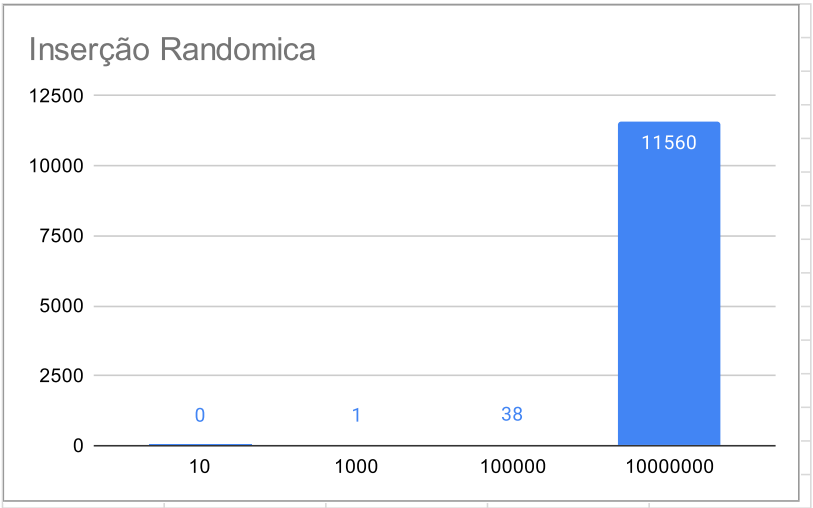


2.2 Inserção Randômica

Para a inserção randômica, foi utilizado o gerador congruencial linear, no qual foi utilizado os seguintes parâmetros:

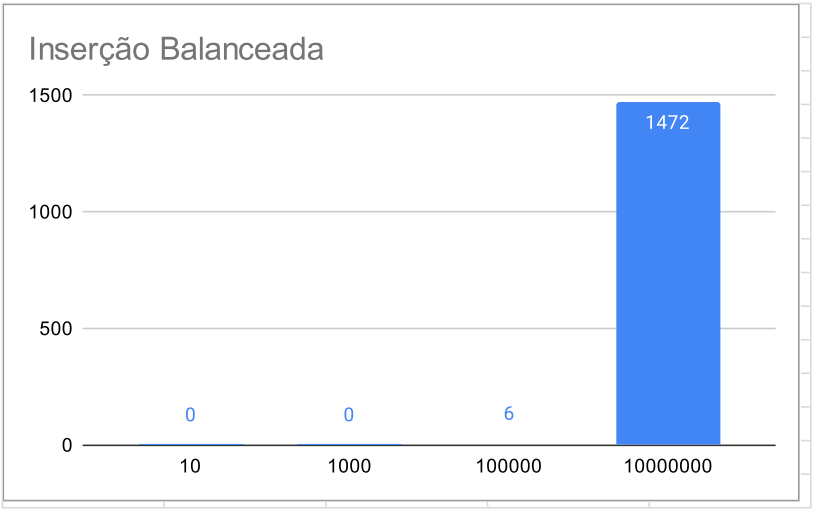
Semente = 0;
Modulo = 1073741824;
Multiplicador = 843314861;
Incremento = 453816693;

Dessa forma, utilizando a própria classe do gerador, um vetor era criado pra cada arvore, e assim que criado, era utilizado para inserir os valores na sua respectiva arvore, agora utilizando o método padrão de inserção da arvore, que dessa vez conseguia lidar com os valores e executar a tarefa completamente. Abaixo tem-se os tempos de execução para as inserções randômicas:
Tempo de execução para inserção randomizada:
O tempo de execução de uma arvore com 10 elementos em milissegundos é: 0
O tempo de execução de uma arvore com 1000 elementos em milissegundos é: 1
O tempo de execução de uma arvore com 100000 elementos em milissegundos é: 38
O tempo de execução de uma arvore com 10000000 elementos em milissegundos é: 11560



2.3 Inserção Balanceada

Para a inserção balanceada, foi necessário a criação de um método para que os valores fossem colocados de forma balanceada em um vetor. Assim que todos valores são inseridos de forma balanceada no vetor, o método de inserção na arvore é chamado, e então é utilizado o método de inserção padrão da arvore. Abaixo tem se os tempos de execução para a inserção na arvore balanceada:
Tempo de execução para inserção balanceada:
O tempo de execução de uma arvore com 10 elementos em milissegundos é: 0
O tempo de execução de uma arvore com 1000 elementos em milissegundos é: 0
O tempo de execução de uma arvore com 100000 elementos em milissegundos é: 6
O tempo de execução de uma arvore com 10000000 elementos em milissegundos é: 1472



2.4 Busca

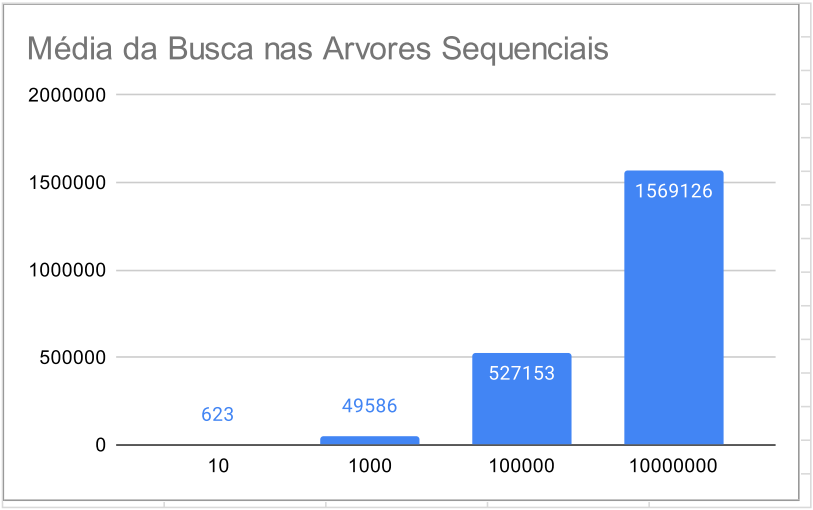
Para fazer a média de 30 elementos na busca, foi utilizada o calculo em Nanosegundos, já que os valores em milissegundos seriam muito pequenos então dariam 0 em vários casos. Para cada arvore, foram feitas 30 buscas, e cada tempo de execução de cada busca era guardada como um elemento em um vetor de tamanho 30, para que depois fosse tirada a média. Essa média em breve será também utilizada para o calculo de Desvio Padrão. A formula da média seria:

$$Média = (\frac{X_1 + X_2 + X_3 + ... + X_n}{n}) \tag{2.1}$$

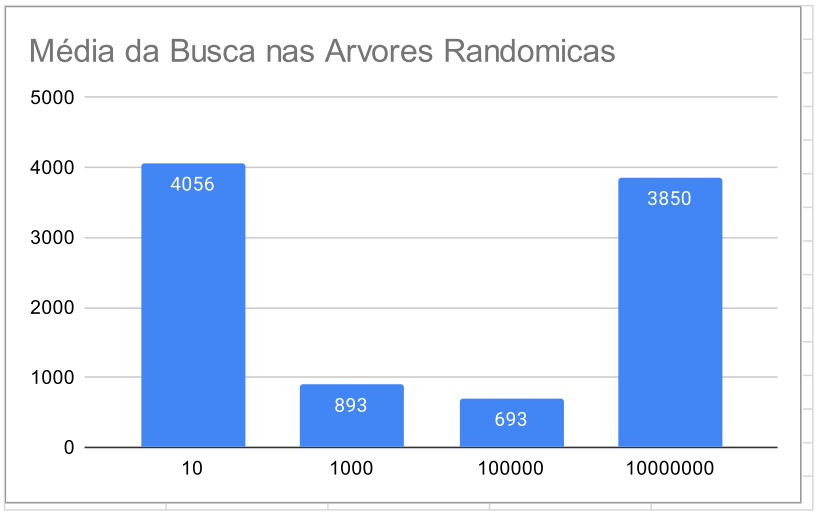
No qual:

O numerador é a soma dos termos numéricos
O denominador é o total de termos;

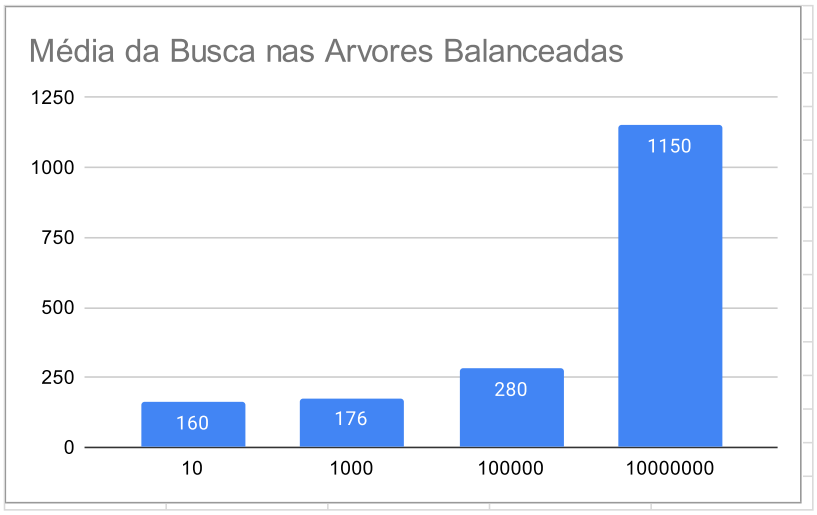
Abaixo tem os tempos de execução após o calculo de Média para cada arvore:
Média de busca na Arvore Sequencial
A media de busca da arvore 1 em nanosegundos foi de: 623
A media de busca da arvore 2 em nanosegundos foi de: 49586
A media de busca da arvore 3 em nanosegundos foi de: 527153
A media de busca da arvore 4 em nanosegundos foi de: 1569126



Média de busca na Arvore Randômica
A media de busca da arvore 5 em nanosegundos foi de: 4056
A media de busca da arvore 6 em nanosegundos foi de: 893
A media de busca da arvore 7 em nanosegundos foi de: 693
A media de busca da arvore 8 em nanosegundos foi de: 3850



Média de busca na Arvore Balanceada
A media de busca da arvore 9 em nanosegundos foi de: 160
A media de busca da arvore 10 em nanosegundos foi de: 176
A media de busca da arvore 11 em nanosegundos foi de: 280
A media de busca da arvore 12 em nanosegundos foi de: 1150



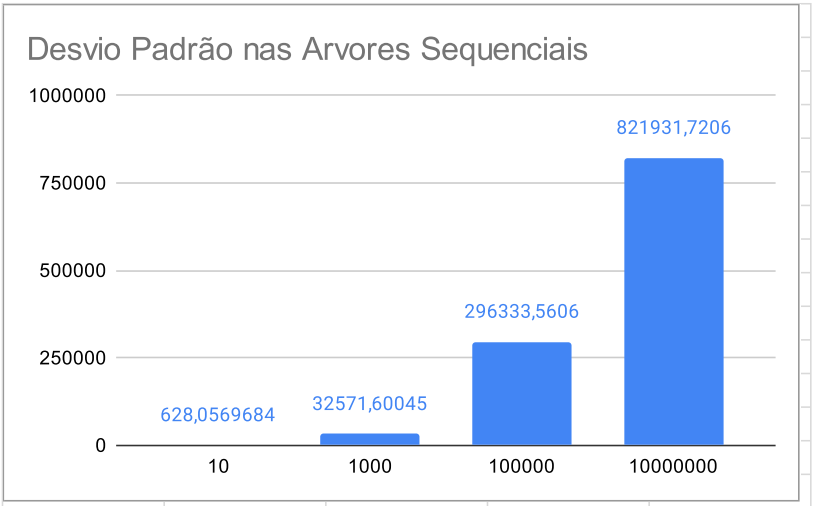
2.5 Desvio Padrão

Para calcular o Desvio Padrão da media dos tempos de busca da arvores, foi utilizado o vetor que armazenava cada um dos tempos de execução de busca de cada uma das arvores. A formula do desvio padrão consiste em:

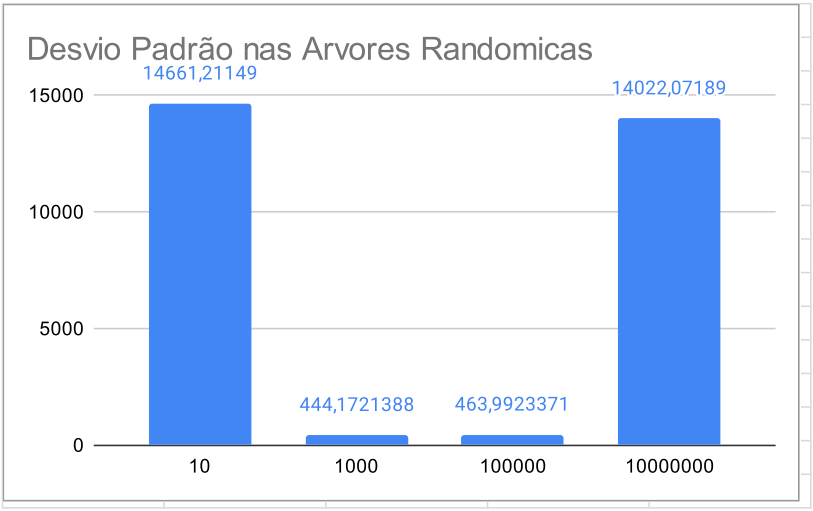
$$s = \sqrt{\frac{\sum_1^N (x_i - m)^2}{N}} \tag{2.2}$$

Onde é xi é valor na posição i no conjunto de dados
m é a média
N é a quantidade de dados

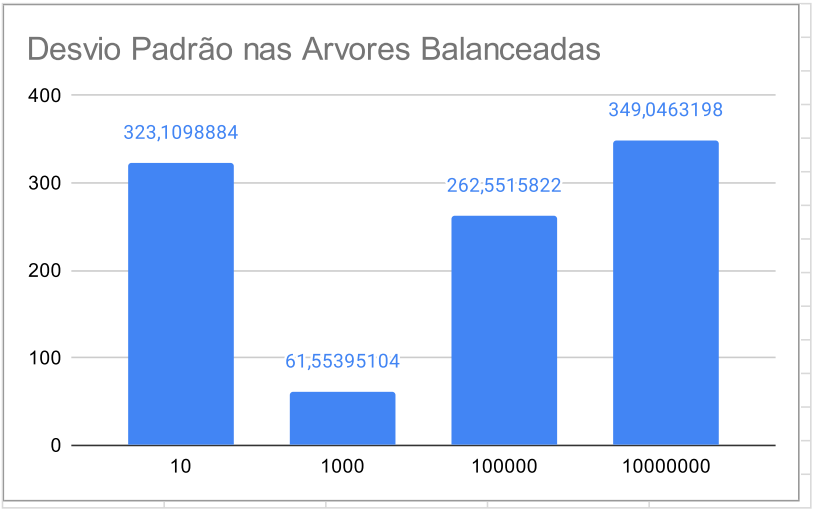
Abaixo tem se os valores dos desvio padrão das arvores:
Desvio Padrão nas Arvores Sequenciais
O desvio padrão da arvore 1 em nanosegundos foi de: 628.056968399807
O desvio padrão da arvore 2 em nanosegundos foi de: 32571.600445104865
O desvio padrão da arvore 3 em nanosegundos foi de: 296333.56062983413
O desvio padrão da arvore 4 em nanosegundos foi de: 821931.0284493803



Desvio Padrão nas Arvore Randômicas
O desvio padrão da arvore 5 em nanosegundos foi de: 14661.211485488573
O desvio padrão da arvore 6 em nanosegundos foi de: 444.17213880306457
O desvio padrão da arvore 7 em nanosegundos foi de: 463.99233710147496
O desvio padrão da arvore 8 em nanosegundos foi de: 14022.071886850388



Desvio Padrão nas Arvores Balanceadas
O desvio padrão da arvore 9 em nanosegundos foi de: 323.10988842807024
O desvio padrão da arvore 10 em nanosegundos foi de: 61.553951042064604
O desvio padrão da arvore 11 em nanosegundos foi de: 262.55158223353624
O desvio padrão da arvore 12 em nanosegundos foi de: 349.0463197533149



3 Conclusão

Após a análise da pratica e dos tempos de execução em todas as arvores, é notável que, se tomado os mesmos métodos para inserir e buscar, a arvore balanceada é a que tem melhor desempenho entre as 3, seguindo por randômica e sequencial (A arvore sequencial só foi capaz de ser medida para inserção quando teve sua implementação da inserção modificada, e assim o seu tempo de execução foi reduzido drasticamente). Esses desempenhos se devem a organização das arvores, que demonstram que, quanto mais organizadas e balanceadas, melhor será o desempenho computacional de seus métodos.

Referências

GAVINIER, D. B. *Arvore Binaria em Java*. 2020. Disponível em: <https://gist.github.com/divanibarbosa/819e7cfcf1b9bae48c4e0f5bd74fb658>. Citado na página 4.