CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
CAMPUS TIMÓTEO

Raphael Gomes Wagner e José Geraldo Duarte Junior e Pedro Arthur Diniz

ARVORES SBB, TRIE E PATRICIA: EXCLUSÃO E BUSCA

Timóteo

2022

Raphael Gomes Wagner e José Geraldo Duarte Junior e Pedro Arthur Diniz

ARVORES SBB, TRIE E PATRICIA: EXCLUSÃO E BUSCA

Trabalho 6 de Algoritmos e Estrutura de Dados sobre exclusão em Arvores Binarias 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, José Geraldo e Pedro Arthur, no intuito de executar exclusão em Arvores SBB e Patricia(ZIVIANI, 2006) e na arvore Trie(FERINO, 2018), porém editadas, adicionando métodos que permitissem a execução do trabalho. Essas implementações foram utilizadas para medir tempos de execução sobre a exclusão e busca de elementos de cada arvore.

Com o intuito de executar a atividade em diferentes ambientes, foram utilizadas 3 tipos de arvores diferentes para exclusão de 5% dos elementos, e elas são: Arvore SBB com elementos sequenciais, Arvore SBB com elementos randômicos, e Arvore SBB balanceada com elementos sequenciais. Em teoria, deveriam ser utilizados arvores com os seguintes tamanhos:

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

Já para a Trie e Patricia, as arvores geradas devem ser utilizadas para busca de 1% dos elementos e medidos os tempos dessas buscas. Os tamanhos para a arvore são:

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

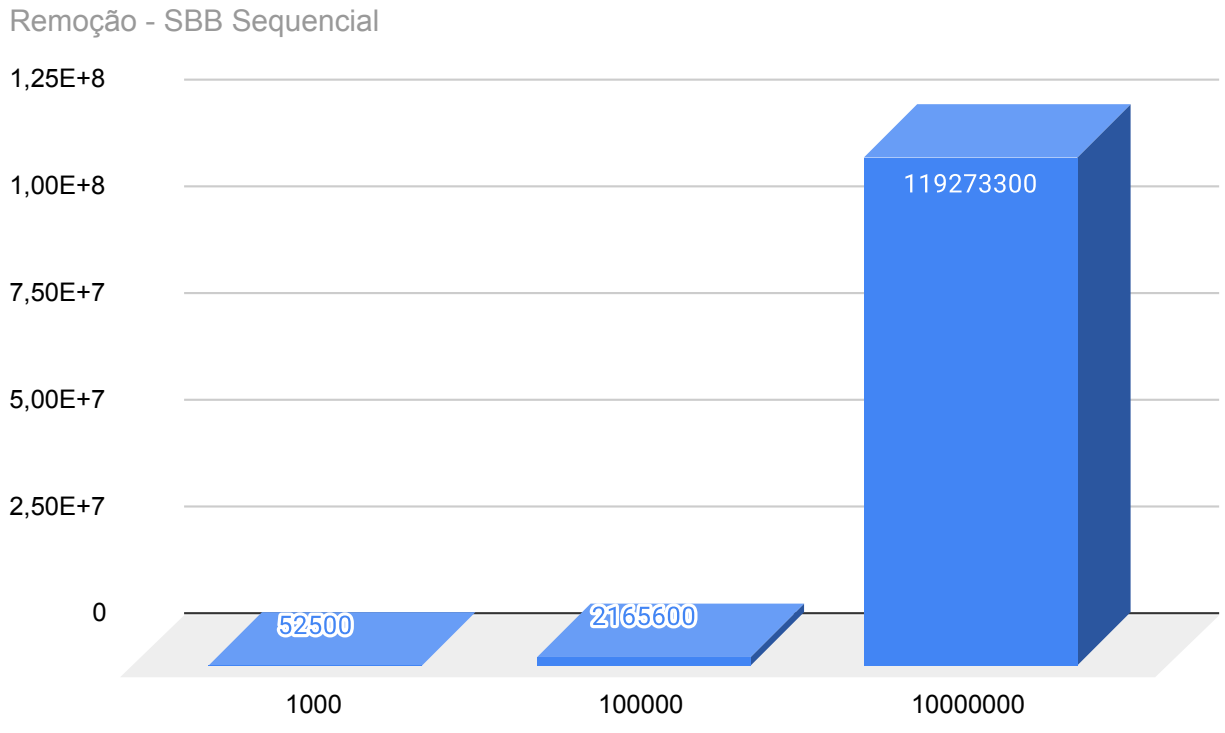
2 Desenvolvimento

2.1 Remoção na SBB 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, foram utilizados métodos de inserção de dados da Pratica 3, desenvolvida anterior a esta, já que este trabalho trata-se de uma continuação direta com estudos sobre remoção.

Aos dados selecionados para a remoção foram gerados utilizando a biblioteca interna do Java "Random", que permitiu gerar valores aleatórios entre 0 e n (n = quantidade de elementos), aumentando a taxa de sucesso para gerar valores validos para a remoção. Os dados apresentados a seguir inicialmente trazem os resultados do tempo de execução da da remoção de 5% de elementos aleatórios de arvores de diferentes tamanhos que receberam seus valores de forma ordenada:

Quantidade de elementos	Tempo em Nanossegundos
10^3	820800
10^5	2165600
10^7	119273300



2.2 Remoção na SBB Randômica

Para a remoção de elementos aleatórios desta arvore foi gerado um vetor com valores aleatórios utilizando um gerador congruencial pseudo aleatório com os seguintes parâmetros:

Semente = 0;

Modulo = 1073741824;

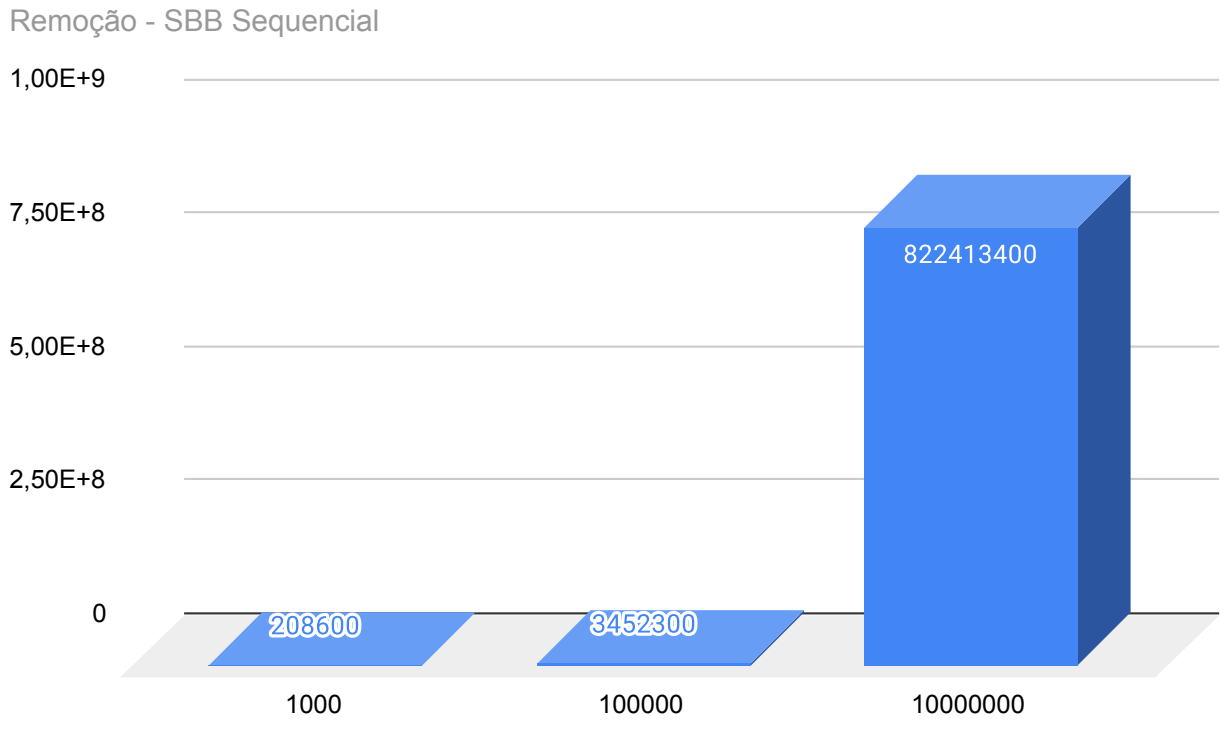
Multiplicador = 843314861;

Incremento = 453816693;

Estes parâmetros também foram utilizados na inserção, então é certo que os mesmos estarão na arvore, mas não foram removidos na mesma ordem que foram inseridos, e sim de forma aleatória também.

Logo os dados apresentados a seguir trazem os resultados do tempo de execução da da remoção de 5% de elementos aleatórios de arvores de diferentes tamanhos que receberam seus valores de forma aleatória:

Quantidade de elementos	Tempo em Nanossegundos
10^3	208600
10^5	3452300
10^7	822413400

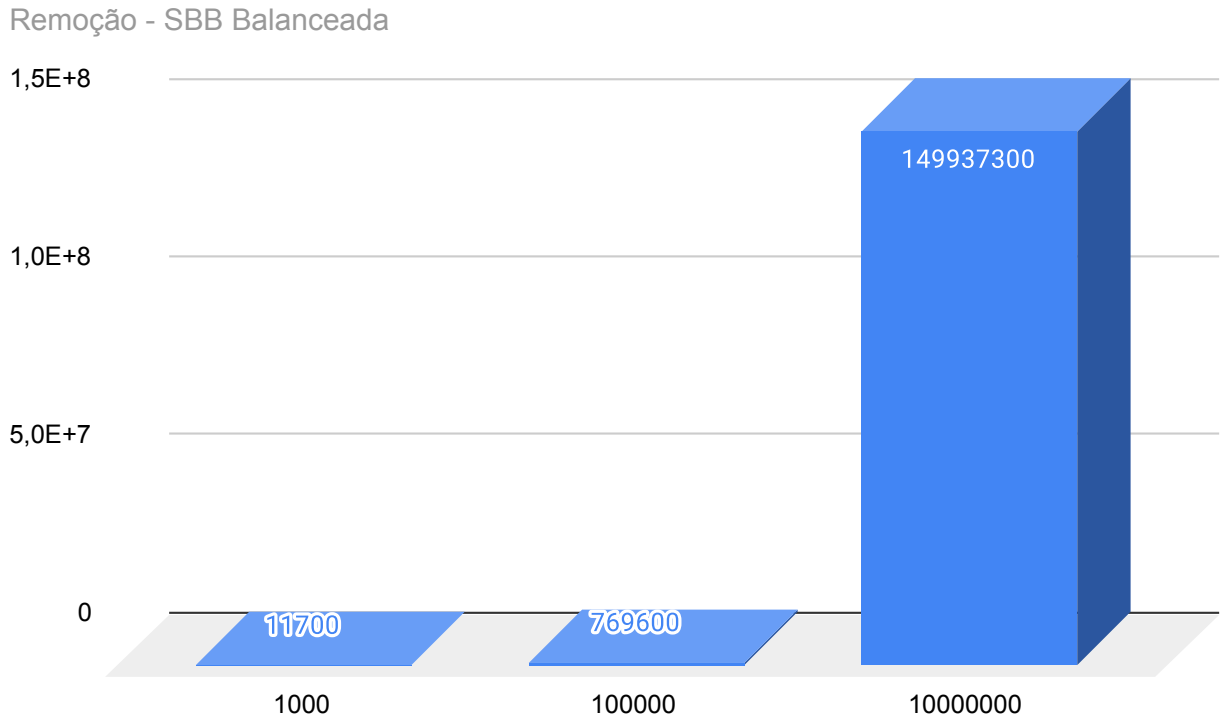


2.3 Remoção na SBB Balanceada

Para a remoção de elementos de dentro da arvore balanceada a logica utilizada segue a mesma da arvore sequencial, já que os valores inseridos na mesma estão entre 0 e n (n = quantidade de elementos), porem foram inseridos em uma condição especial que força a arvore a ser formada de já balanceada, como foi visto na atividade pratica 3.

Logo os dados apresentados a seguir trazem os resultados do tempo de execução da remoção de 5% de elementos aleatórios de SBBs de diferentes tamanhos:

Quantidade de elementos	Tempo em Nanossegundos
10^3	11700
10^5	769600
10^7	149937300



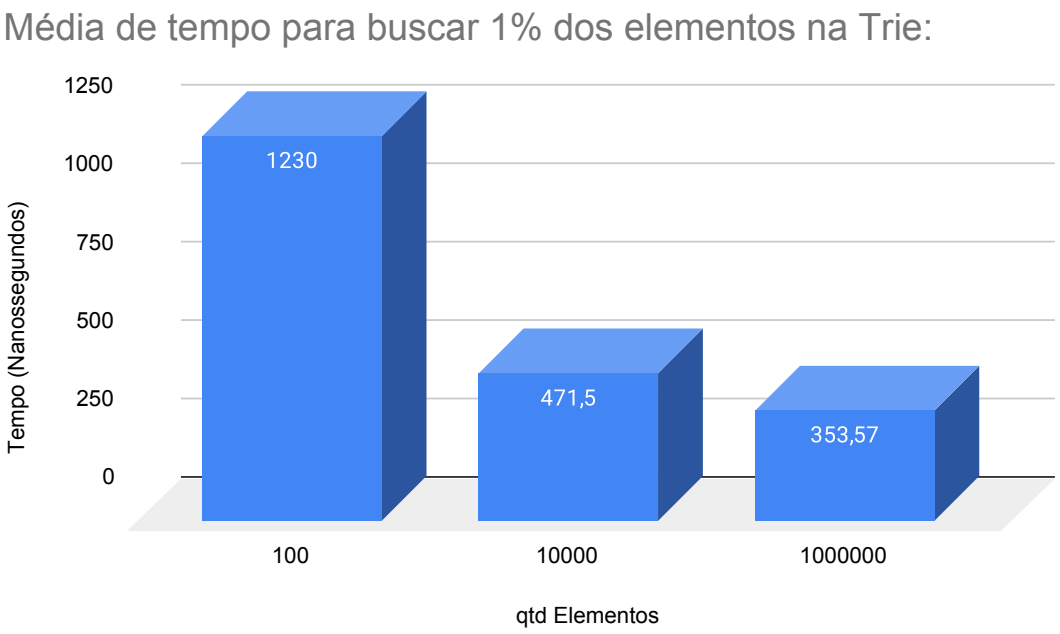
2.4 Busca de elementos na TRIE

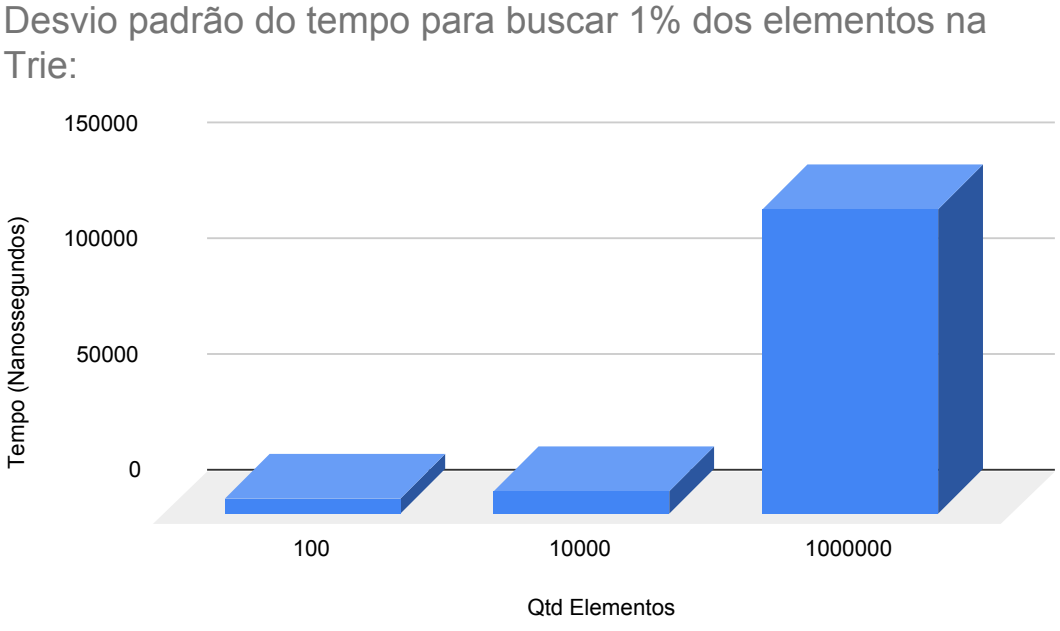
Para a execução dos testes neste modelo de estrutura de dados foi utilizada uma implementação(FERINO, 2018) que recebe palavras no formato de String e as compara de acordo com seu tamanho e prioridade em ordem alfabética e facilitar a implementação de grandes quantidades de palavras foi implementado um gerador de Strings aleatórias, que utiliza de um

vetor com todas as lestras do alfabeto que este método seleciona posições aleatórias do vetor e gera uma String com as letras destas posições(DELFTSTACK, 2021).

Logo os dados apresentados a seguir trazem os resultados do tempo de execução da busca de 1% de elementos aleatórios de TRIEs de diferentes tamanhos:

Quantidade de elementos	Elementos Buscados	Media de Tempo	Desvio Padrão
10^3	100	1230	6845,509
10^5	1000	471,5	9743,600
10^7	100000	353,57	132471,980



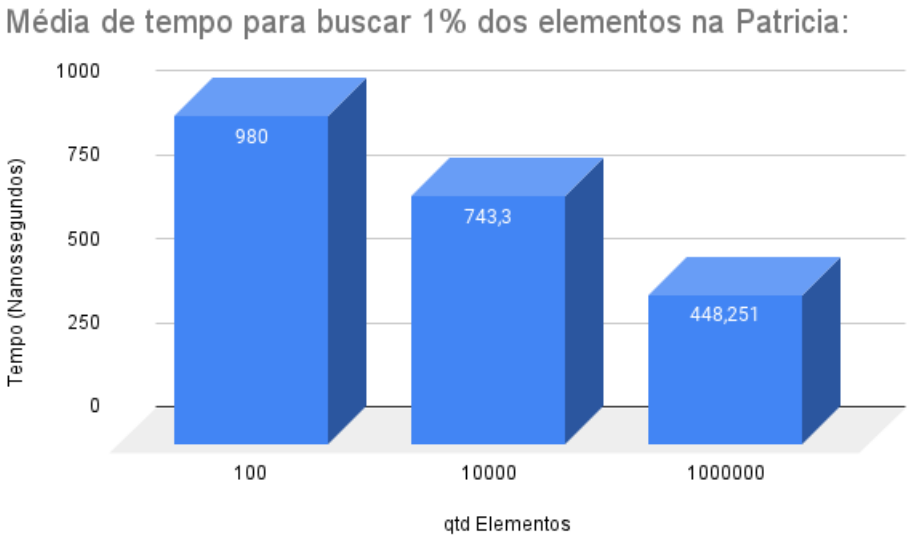


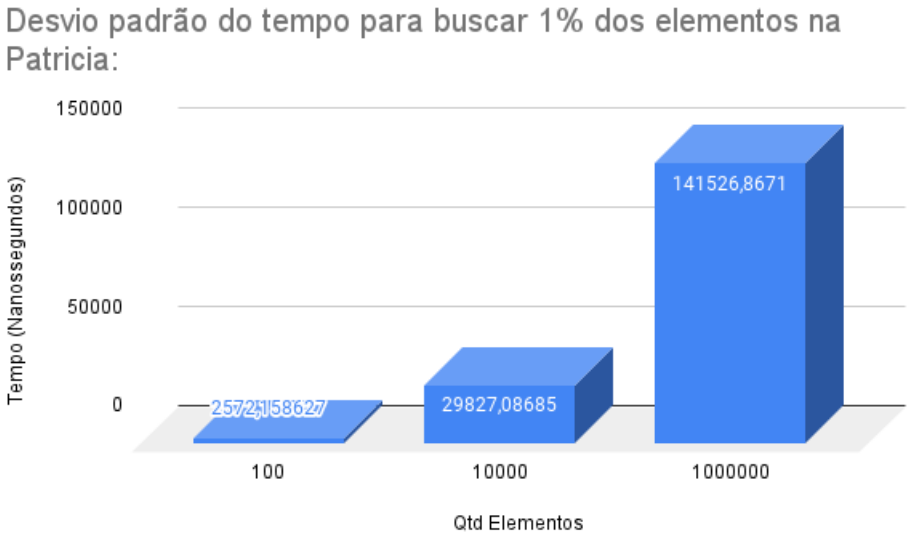
2.5 busca de elementos na PATRICIA

Para execução dos testes neste modelo de dados foi utilizado uma implementação que recebe inteiros no no formato de Interger e os compara byte a byte(ZIVIANI, 2006).

Logo os dados apresentados a seguir trazem os resultados do tempo de execução da busca de 1% de elementos aleatórios de Patricia de diferentes tamanhos:

Quantidade de elementos	Elementos Buscados	Media de Tempo	Desvio Padrão
10^3	100	980,0	2572,158626523644
10^5	1000	743,3	29827,08685071316
10^7	100000	448,251	141526,86706021812





3 Conclusão

Ao fim deste trabalho, foi abortado como foi realizada a remoção de elementos nas arvore SBB com elementos sequenciais, arvore SBB com elementos randômicos, e arvore SBB balanceada com elementos sequenciais e seus diversos tamanhos. Foi observado o tempo de execução da remoção de 5% de elementos aleatórios em cada arvore SBB cujo o tamanho varia de 10^3 elementos a 10^7 elementos, pois como relatado anteriormente o Java não permite a criação de uma arvore de tamanho 10^9.Foi notado que dependendo do tamanho da arvore SBB e seu tipo de remoção seu tempo de execução varia, como pode ser observado no relatório o tempo de execução na remoção na SBB balanceada foi o menor dos três quando o seu tamanho era o menor entre as três arvores, mas em contrapartida seu tempo de execução quando é no maior tamanho é o pior entres as arvores e quem se sobre sai nesse quesito é a remoção na SBB sequencial. Outro tópico abortado foi a realização de busca de elementos nas arvores TRIEs e nas arvores Patricia cujo foi observado os seus tempos de execução e seus desvios padrões

Referências

DELFTSTACK. *Gerar String Aleatória em Java*. 2021. <https://www.delftstack.com/pt/howto/java/random-alphanumeric-string-in-java/>. Citado na página 8.

FERINO, S. L. de M. *Trie*. 2018. Disponível em: <https://github.com/Samuellucas97/Trie>. Citado nas páginas 4 e 7.

ZIVIANI, N. *Projeto de algoritmos com implementação em java e c++*. 2006. Disponível em: <http://www2.dcc.ufmg.br/livros/algoritmos-java/implementacoes-06.php>. Citado nas páginas 4 e 9.