USP - Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Alunos: Witor Matheus Alves de Oliveira - 10692190 e Victor Machado Gonzaga - 10692033
Professor: Robson Cordeiro
Disciplinas: Algoritmos e estrutura de dados I

Relatório do Projeto 2

08 de Dezembro de 2018
São Carlos - SP

Sumário

Introdução

Neste trabalho foi realizado a comparação do tempo de inserção, busca e remoção dos algoritmos para manipulação de dados vetor com Busca Binária, Lista Encadeada Ordenada, Lista Encadeada Ordenada com Sentinela, Árvore Binária de Busca e Árvore AVL. Não foi realizado os testes de tempo para Lista Circular ordenada por Frequência, pois os alunos não conseguiram implementá-la à tempo, além disso também não foram contados os tempos para n = 100000, pois houve grande demora na geração da tabela, o que impediu que isso fosse realizado.

O objetivo deste projeto é fazer com que os alunos desenvolvam a capacidade de optar de forma correta por uma das estruturas de dados vistas para uma possível aplicação, de forma a deixá-la a mais eficiente possível.

Resultados

- Tempo de inserção crescente

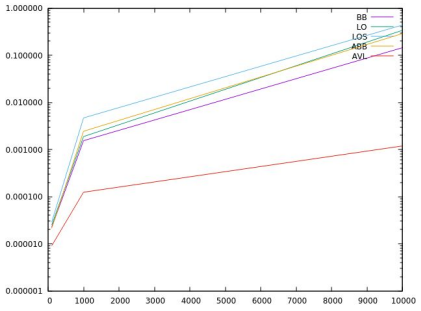

Tabela 1: Tempo de insercao crescente

	n=100	n=1.000	n=10.000	n=100.000
BB	0.00	0.00	0.15	0.0000
LO	0.00	0.00	0.33	0.0000
LOS	0.00	0.00	0.40	0.0000
ABB	0.00	0.00	0.29	0.0000
AVL	0.00	0.00	0.00	0.0000
LFREQ	0.00	0.00	0.00	0.00

Busca binária - Para n = 100 e n = 1000, o custo da operação de ordenação ainda não é tão grande, por isso não indica vantagens ou desvantagens com relação às outras estruturas. Para n = 10000 essa operação começa a se tornar relevante, pois o shift é realizado em um vetor muito grande e a busca pela posição correta de inserção ser feita sequencialmente, mas comparada às outras estruturas, excluindo-se AVL, ela se mostra mais vantajosa.

Lista Ordenada - Para n = 100 e n = 1000, o custo da operação de buscar a posição correta de inserção para manter a lista ordenada ainda não é tão grande, por isso não indica vantagens ou desvantagens com relação às outras estruturas. Para n = 10000 essa operação começa a se tornar relevante, se mostrando desvantajosa se comparada à BB, ABB e AVL.

Lista Ordenada com sentinela - Para n = 100 e n = 1000, o custo da operação de manter o nó sentinela e buscar pela posição correta de inserção ainda não é tão grande, por isso não indica vantagens ou desvantagens com relação às outras estruturas. Para n = 10000 essa operação começa a se tornar relevante, a menos eficiente dentre todas.

Árvore de Busca Binária - Para n = 100 e n = 1000, o custo da inserção na ABB ainda não é tão grande, por isso não indica vantagens ou desvantagens com relação às outras

estruturas. Para n = 10000 essa operação começa a se tornar relevante, se mostrando melhor que a LO a LOS, e pior em relação à BB e à AVL.
Adelson-Veslky & Landis (AVL) - Para n = 100 e n = 1000, o custo das rotações na inserção da AVL ainda não é tão grande, por isso não indica vantagens ou desvantagens com relação às outras estruturas. Para n = 10000 essa operação começa a se tornar relevante, mas muito pouco a ponto de ainda assim ser mais eficiente que todas as outras estruturas utilizadas.

- Tempo de inserção decrescente

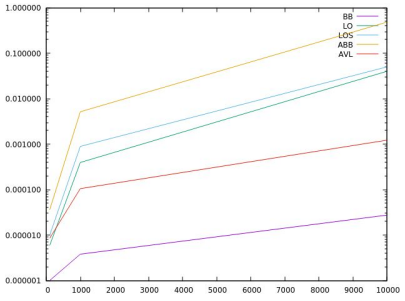

Tabela 2: Tempo de insercao decrescente

	n=100	n=1.000	n=10.000	n=100.000
BB	0.00	0.00	0.00	0.00
LO	0.00	0.00	0.04	0.00
LOS	0.00	0.00	0.04	0.00
ABB	0.00	0.00	0.48	0.00
AVL	0.00	0.00	0.00	0.00
LFREQ	0.00	0.00	0.00	0.00

Busca binária - Para todos n, o custo da operação de buscar a posição correta de inserção é pequeno, por isso indicando ser mais vantajosa com relação às outras estruturas. Este é o melhor caso para inserção ordenada no vetor, pois a posição a ser inserida é sempre a primeira do vetor ainda que tenha que ser realizado o shift de vários elementos.
Lista Ordenada - Para n = 100 e n = 1000, o custo da operação de buscar a posição correta de inserção para manter a lista ordenada ainda não é tão grande, por isso não indica vantagens ou desvantagens com relação às outras estruturas, que não a BB. Para n = 10000 essa operação começa a se tornar relevante, mas ainda assim é tem bom rendimento, pois a posição de inserção calculada nesse caso é sempre a primeira da lista.
Lista Ordenada com sentinela - Para n = 100 e n = 1000, o custo da operação de manter o nó sentinela e buscar pela posição correta de inserção ainda não é tão grande, por isso não indica vantagens ou desvantagens com relação às outras estruturas. Para n = 10000 essa operação começa a se tornar relevante, mas ainda é boa pelo mesmo motivo da LO.
Árvore de Busca Binária - Para n = 100 e n = 1000, o custo da inserção na ABB já é maior que o das outras estruturas. Para n = 10000 essa operação começa a se tornar ainda mais relevante, se mostrando pior que todas a outras estruturas testadas..
Adelson-Veslky & Landis (AVL) - Para n = 100 e n = 1000, o custo das rotações na inserção da AVL ainda não é tão grande, por isso não indica vantagens ou desvantagens com relação às outras estruturas. Para n = 10000 essa operação começa a se tornar relevante, mas muito pouco a ponto de ainda assim ser mais eficiente que todas as outras estruturas utilizadas, excluindo-se a BB.

-Tempo de inserção aleatória

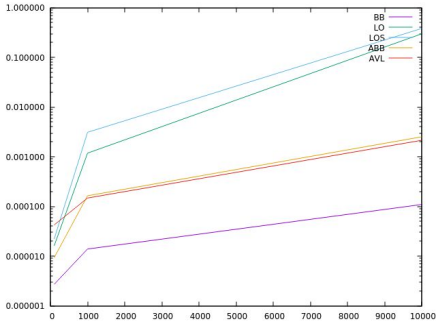

Tabela 3: Tempo de insercao aleatoria

	n=100	n=1.000	n=10.000	n=100.000
BB	0.00	0.00	0.00	0.00
LO	0.00	0.00	0.29	0.00
LOS	0.00	0.00	0.35	0.00
ABB	0.00	0.00	0.00	0.00
AVL	0.00	0.00	0.00	0.00
LFREQ	0.00	0.00	0.00	0.00

Busca binária - Para todos n, o custo da operação de ordenação é pequeno e mostra que a BB tem vantagens com relação às outras estruturas.
Lista Ordenada - Para todos n o custo da operação de buscar a posição correta de inserção para manter a lista ordenada só não é pior que o da LOS, deixando ela atrás de todo o resto.
Lista Ordenada com sentinela - Para todos n, o custo da operação de manter o nó sentinela e buscar pela posição correta de inserção é o pior dentre todas as outras estruturas.
Árvore de Busca Binária - Para n = 100 e n = 1000, o custo da inserção na ABB ainda não é tão grande, se mostrando pior apenas que a BB. Para n = 10000 essa operação começa a se tornar relevante, mostrando-se melhor que LO e LOS..
Adelson-Veslky & Landis (AVL) - Para n = 100 e n = 1000, o custo das rotações na inserção da AVL é maior apenas que o da ABB e da BB. Para n = 10000 essa operação começa a se tornar relevante, mas muito pouco a ponto de ainda assim ser mais eficiente que todas as outras estruturas utilizadas, excluindo-se a BB.

- Tempo de remoção crescente (após inserção crescente)

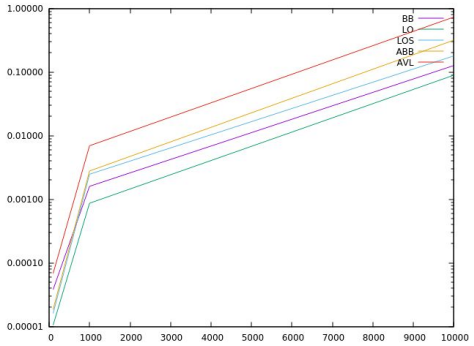

Tabela 4: Tempo de remocao crescente (apos insercao crescente)

	n=100	n=1.000	n=10.000	n=100.000
BB	0.00	0.00	0.13	0.00
LO	0.00	0.00	0.09	0.00
LOS	0.00	0.00	0.17	0.00
ABB	0.00	0.00	0.31	0.00
AVL	0.00	0.01	0.73	0.00
LFREQ	0.00	0.00	0.00	0.00

Busca binária - Para todos n, ela se mostra menos eficiente apenas que a LO, pois ainda que tenha que ser feita a operação de shift a cada remoção, a busca pelos elementos a serem removidos já é a busca binária.

Lista Ordenada - Para todos n, o custo da operação de buscar a posição correta de remoção é pequeno, se mostrando mais vantajosa a todas outras estruturas.
Lista Ordenada com sentinela - Para n = 100 e n = 1000, o custo da operação de manter o nó sentinela e buscar pela posição correta remoção ainda não é tão grande, é mais eficiente apenas que a AVL. Para n = 10000 essa operação começa a se tornar relevante, se mostrando melhor que a AVL e ABB apenas.
Árvore de Busca Binária - Para n = 100 e n = 1000, o custo da remoção na ABB ainda não é tão grande, mas é mais eficiente apenas que a AVL. Para n = 10000 essa operação começa a se tornar relevante, mas ainda assim só é mais eficiente que a AVL.
Adelson-Veslky & Landis (AVL) - Para todos n, as operações de rotação na remoção aparentam são menos eficientes que as das outras estruturas.

- Tempo de remoção decrescente (após inserção crescente)

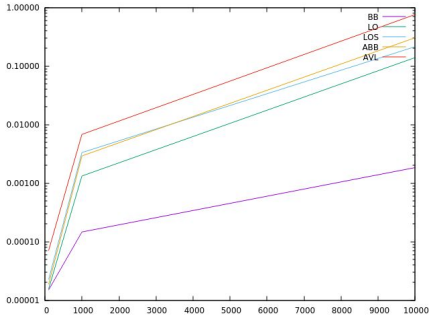

Tabela 5: Tempo de remocao decrescente (apos insercao crescente)

	n=100	n=1.000	n=10.000	n=100.000
BB	0.00	0.00	0.00	0.00
LO	0.00	0.00	0.14	0.00
LOS	0.00	0.00	0.20	0.00
ABB	0.00	0.00	0.30	0.00
AVL	0.00	0.01	0.80	0.00
LFREQ	0.00	0.00	0.00	0.00

Busca binária - Este é o melhor caso para a remoção de um vetor já ordenado, pois além da busca pelo elemento ser binária, não há shift a ser realizado a cada remoção, pois o elemento a ser removido sempre estará no fim do vetor, se mostrando assim a estrutura mais eficiente dentre as outras.
Lista Ordenada - Para todos n, o custo da operação de buscar a posição correta de remoção ainda não é tão grande, é menos eficiente apenas que a BB.
Lista Ordenada com sentinela - Para n = 100 e n = 1000, o custo da operação de manter o nó sentinela e buscar pela posição correta remoção ainda não é tão grande, é mais eficiente apenas que a AVL. Para n = 10000 essa operação começa a se tornar relevante, se mostrando melhor que a AVL e ABB.
Árvore de Busca Binária - Para n = 100 e n = 1000, o custo da remoção na ABB ainda não é tão grande, é mais eficiente apenas que a AVL e LOS. Para n = 10000 essa operação começa a se tornar relevante, mas ainda assim só é mais eficiente que a AVL.
Adelson-Veslky & Landis (AVL) - Para todos n, as operações de rotação na remoção aparentam são menos eficientes que as das outras estruturas.

- Tempo de remoção aleatória (após inserção aleatória)

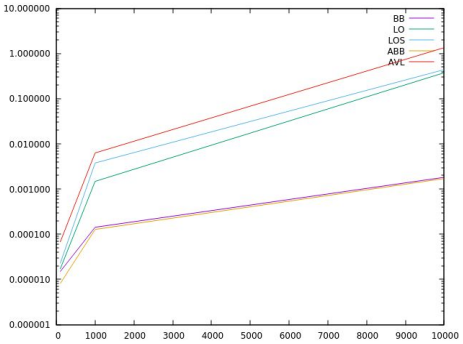

Tabela 6: Tempo de remocao aleatoria (apos insercao aleatoria)

	n=100	n=1.000	n=10.000	n=100.000
BB	0.00	0.00	0.00	0.00
LO	0.00	0.00	0.36	0.00
LOS	0.00	0.00	0.42	0.00
ABB	0.00	0.00	0.00	0.00
AVL	0.00	0.01	1.36	0.00
LFREQ	0.00	0.00	0.00	0.00

Busca binária - Para todos os n, o custo da operação de remoção ainda não é tão grande, devido a utilização da busca binária para encontrar o elemento a ser removido e a operação de shift não se mostrar tão custosa, a BB se torna a estrutura mais eficiente junto à ABB.
Lista Ordenada - Para todos n, o custo da operação de buscar a posição correta de remoção é melhor apenas que o custo da AVL e LOS.
Lista Ordenada com sentinela - Para todos n, o custo da operação de manter o nó sentinela e buscar pela posição correta de remoção só se mostra melhor que o da AVL.
Árvore de Busca Binária - Para todos n, o custo da remoção na ABB é o menor dentre todas as estruturas.
Adelson-Veslky & Landis (AVL) - Para todos n as operações de rotação na remoção da AVL tem alto custo e por isso que AVL é a mais desvantajosa com relação às outras estruturas.

- Tempo de busca (após inserção crescente)

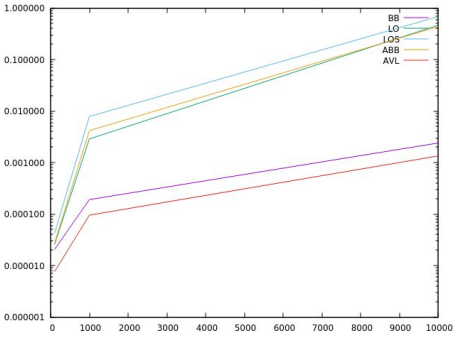

Tabela 7: Tempo de busca (apos insercao crescente)

	n=100	n=1.000	n=10.000	n=100.000
BB	0.00	0.00	0.00	0.00
LO	0.00	0.00	0.46	0.00
LOS	0.00	0.01	0.68	0.00
ABB	0.00	0.00	0.43	0.00
AVL	0.00	0.00	0.00	0.00
LFREQ	0.00	0.00	0.00	0.00

Busca binária - Para todos os n, o custo da busca não é tão grande, devido a utilização da busca binária por isso, a BB se torna a estrutura mais eficiente atrás apenas da AVL.
Lista Ordenada - Para n = 100 e n = 1000, o custo da busca sequencial ainda não é tão grande, mas já é mais custosa que a BB e AVL. Para n = 10000 essa operação começa a se tornar relevante, se mostrando mais vantajosa apenas a LOS.
Lista Ordenada com sentinela - Para todos n, o custo da busca sequencial com o nó sentinela ainda é o maior de todos, mostrando que ela é a pior estrutura nesse caso.

Árvore de Busca Binária - Para n = 100 e n = 1000, o custo da busca na ABB é próximo da LO, pois com a inserção crescente ela se torna basicamente uma lista encadeada, mas ainda assim é mais eficiente que a LOS. Para n = 10000, por esse mesmo motivo ela se mostra melhor apenas que, a LO e LOS.
Adelson-Veslky & Landis (AVL) - Para todos n a busca na AVL tem baixo custo, por serem realizadas poucas comparações, por isso que AVL é a mais vantajosa com relação às outras estruturas.

- Tempo de busca (após inserção decrescente)

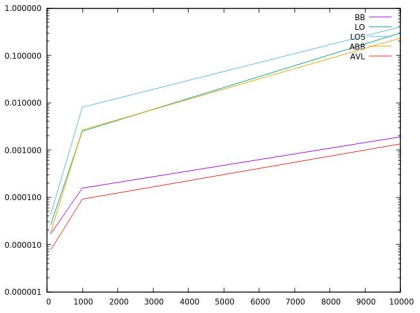

Tabela 8: Tempo de busca (apos insercao decrescente)

	n=100	n=1.000	n=10.000	n=100.000
BB	0.00	0.00	0.00	0.00
LO	0.00	0.00	0.29	0.00
LOS	0.00	0.01	0.36	0.00
ABB	0.00	0.00	0.23	0.00
AVL	0.00	0.00	0.00	0.00
LFREQ	0.00	0.00	0.00	0.00

Busca binária - Para todos os n, o custo da busca não é tão grande, devido a utilização da busca binária por isso, a BB se torna a estrutura mais eficiente junto à AVL.
Lista Ordenada - Para n = 100 e n = 1000, o custo da busca sequencial ainda não é tão grande, mas já é maior que da ABB, BB e AVL. Para n = 10000 essa operação começa a se tornar relevante, se mostrando mais vantajosa apenas a LOS.
Lista Ordenada com sentinela - Para todos n, o custo da busca sequencial com o nó sentinela ainda é o maior de todos, mostrando que ela também é a pior estrutura nesse caso.
Árvore de Busca Binária - Para n = 100 e n = 1000, o custo da busca na ABB é próximo da LO, pois com a inserção crescente ela se torna basicamente uma lista encadeada, mas ainda assim é mais eficiente que a LO e LOS. Para n = 10000, por esse mesmo motivo ela também se mostra melhor apenas que, a LO e LOS.
Adelson-Veslky & Landis (AVL) - Para todos n a busca na AVL tem baixo custo, por serem realizadas poucas comparações, por isso que AVL é a mais vantajosa com relação às outras estruturas.

- Tempo de busca (após inserção aleatória)

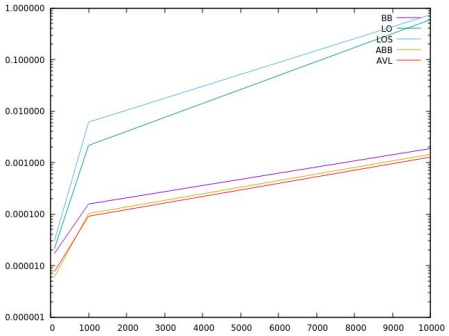

Tabela 9: Tempo de busca (apos insercao aleatoria)

	n=100	n=1.000	n=10.000	n=100.000
BB	0.00	0.00	0.00	0.00
LO	0.00	0.00	0.59	0.00
LOS	0.00	0.01	0.68	0.00
ABB	0.00	0.00	0.00	0.00
AVL	0.00	0.00	0.00	0.00
LFREQ	0.00	0.00	0.00	0.00

Busca binária - Para todos os n, o custo da busca não é tão grande, devido a utilização da busca binária por isso, a BB se torna a estrutura mais eficiente, excluindo-se a ABB e AVL.
Lista Ordenada - Para todos n, o custo da busca sequencial já é grande se comparado ao da BB, ABB, AVL, se mostrando melhor apenas que a LOS.
Lista Ordenada com sentinela - Para todos n, o custo da busca sequencial com o nó sentinela ainda é o maior de todos, mostrando que ela também é a pior estrutura nesse caso.
Árvore de Busca Binária - Para todos n, o custo da busca na ABB é melhor que o de todas estruturas, fora AVL, por este ser o melhor caso para a busca em ABB..
Adelson-Veslky & Landis (AVL) - Para todos n a busca na AVL tem baixo custo, por serem realizadas poucas comparações, por isso que AVL é a mais vantajosa com relação às outras estruturas.

Conclusão

A partir da análise dos dados obtidos, conclui-se que para utilização de uma dessas estruturas em um aplicação mais geral, ou seja, onde não se conhece o tamanho específico, a forma como os dados vão ser inseridos, buscados e removidos, ou qual dessas operações mais vai ser realizada, a estrutura Árvore AVL é a mais indicada, pois ainda que na remoção elas seja bastante custosa, por realizar primeiramente uma busca e logo após rotações simples ou duplas no desempilhamento da chamada recursiva (que é a parte mais custosa dessa operação), as operações de busca e inserção (ainda que também sejam realizadas rotações) são muito rápidas pois realizam poucas comparações, da ordem de log(n).