ICMC
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO

SCC 0202 - Algoritmos e Estruturas de Dados I

Projeto 2 - ALG I

Luís Eduardo Rozante de Freitas Pereira - 10734794

SÃO CARLOS
11 de Dezembro de 2018

UNIVERSIDADE DE SÃO PAULO

1 Observações

Quanto ao uso de funções além da **main**: Nas instruções do projeto estava presente a frase "Deve-se criar uma única função **int main(void){...}** que execute as seguintes operações em cada estrutura para diversos valores de n:". Está frase produziu dúvida em seu significado, se referia-se a ser necessário criar apenas uma função, ou se realmente seria proibido o uso de múltiplas funções. Na implementação, foi optado pelo uso de várias funções: uma função diferente para medir os tempos de cada estrutura e, em seguida, uma para a impressão das tabelas. Essa decisão foi tomada devido a legibilidade extremamente baixa do código antes da quebra em múltiplicas funções e da enorme repetição na parte da impressão.

Quanto a implementação da AVL: Houveram dificuldades na implementação da remoção na AVL, que levaram à busca de ajuda em fontes externas. Esse fato pode ocasionar que o código referente a inserção e remoção na AVL esteja extremamente próximo ao código do site *geeksforgeeks.org*, que foi usado como pesquisa e referência.

2 Descrição

O trabalho visa discutir o uso de diversas estruturas de dados, através de medições no tempo de execução de algumas operações descritas abaixo. As estruturas avaliadas foram: lista sequêncial ordenada e com busca binária, lista encadeada ordenada, lista encadeada ordenada e com sentinela, árvore binária de busca, árvore AVL e lista encadeada ordenada por frequência de busca.

3 Dados e Análise

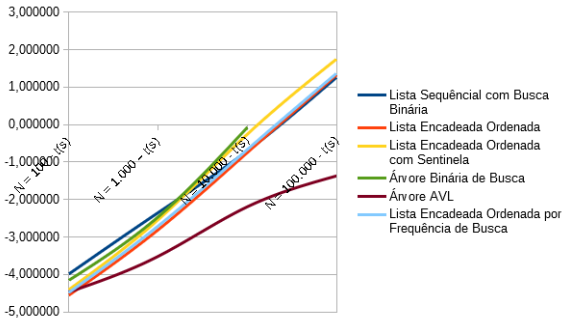
OBS: Todos os gráficos estão em escala **logarítmica**.

3.1 Inserção Crescente

Para a inserção crescente observa-se que, com excessão da árvore AVL, as demais estruturas apresentaram tempos muito longos devido à necessidade de iterar por todos os elementos da lista para encontrar a posição de inserção. No caso da árvore binária de busca em especial, devido à inserção ordenada, a árvore se torna extremamente desbalanceada, degenerando em uma lista

encadeada, porém com constantes muito mais altas devido a sua natureza mais complexa. Assim, claramente a AVL se mostra vantajosa nesse caso.

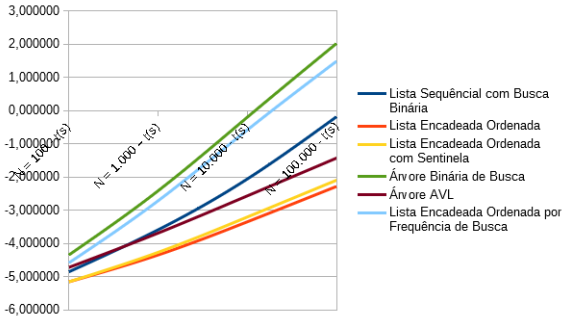
Tabela 1: Tempo Médio de Inserção Crescente				
Estrutura	**N = 100 - t(s)**	**N = 1.000 – t(s)**	**N = 10.000 - t(s)**	**N = 100.000 - t(s)**
Lista Sequêncial com Busca Binária	0.000103	0.004464	0.197268	18.082357
Lista Encadeada Ordenada	0.000028	0.001536	0.178049	21.040663
Lista Encadeada Ordenada com Sentinela	0.000039	0.002876	0.548164	55.525395
Árvore Binária de Busca	0.000070	0.003419	0.832819	143.139443
Árvore AVL	0.000034	0.000304	0.006383	0.042512
Lista Encadeada Ordenada por Frequência de Busca	0.000033	0.001906	0.214355	23.136690



3.2 Inserção Decrescente

Já para a inserção decrescente, observa-se algumas mudanças, as listas ordenadas, com e sem sentinela, superaram até mesmo a AVL, apesar de essa ainda se manter com tempos muito bons, devido ao fato de a inserção ser feita no começo, permitindo que essa aconteça com apenas a troca de alguns ponteiros. Nota-se que essa melhoria não se propagou para a lista ordenada por frequência, pois nessa ainda é necessário checar todos os elementos. Um resultado inesperado foi o bom desempenho da lista sequêncial com busca binária, pois devido a necessidade de *shiftar* elementos ao inserir no início, se esperaria uma grande demora, talvez possa-se atribuir esse bom resultado à otimizações realizadas pelo compilador.

Tabela 2: Tempo Médio de Inserção Decrescente				
Estrutura	**N = 100 - t(s)**	**N = 1.000 – t(s)**	**N = 10.000 - t(s)**	**N = 100.000 - t(s)**
Lista Sequêncial com Busca Binária	0.000014	0.000257	0.008642	0.654151
Lista Encadeada Ordenada	0.000007	0.000045	0.000447	0.005182
Lista Encadeada Ordenada com Sentinela	0.000007	0.000054	0.000623	0.007996
Árvore Binária de Busca	0.000045	0.003835	0.616849	104.680534
Árvore AVL	0.000019	0.000205	0.002699	0.037160
Lista Encadeada Ordenada por Frequência de Busca	0.000026	0.001920	0.259924	30.394944

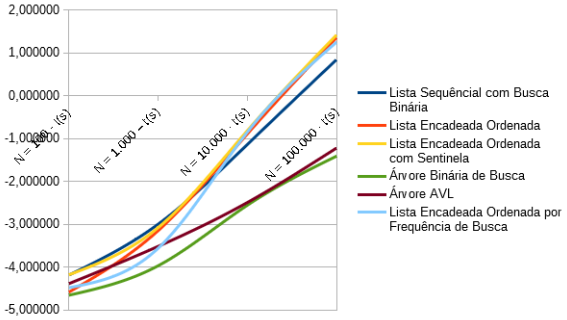


3.3 Inserção Aleatória

Na inserção aleatória pode-se observar o poder da árvore binária de busca, com a aleatoriedade facilitando a formação de uma árvore balanceada a ABB supera até mesmo a AVL por não executar rotações para corrigir o balanceamento. Outra estrutura que se saiu melhor do que nos casos anteriores, foi a lista ordenada por frequência, devido a aleatoriedade, alguns elementos podem ter sido buscados mais de uma vez, o que reduziria o tempo gasto, pois a cada busca esses estariam mais próximos do início da árvore.

Tabela 3: Tempo Médio de Inserção Aleatória

Estrutura	N = 100 - t(s)	N = 1.000 – t(s)	N = 10.000 - t(s)	N = 100.000 - t(s)
Lista Sequêncial com Busca Binária	0.000065	0.000998	0.072493	6.942316
Lista Encadeada Ordenada	0.000026	0.000706	0.127742	22.739257
Lista Encadeada Ordenada com Sentinela	0.000066	0.000830	0.152741	26.169720
Árvore Binária de Busca	0.000022	0.000107	0.002770	0.038897
Árvore AVL	0.000041	0.000304	0.003212	0.060709
Lista Encadeada Ordenada por Frequência de Busca	0.000033	0.000274	0.140604	17.804379

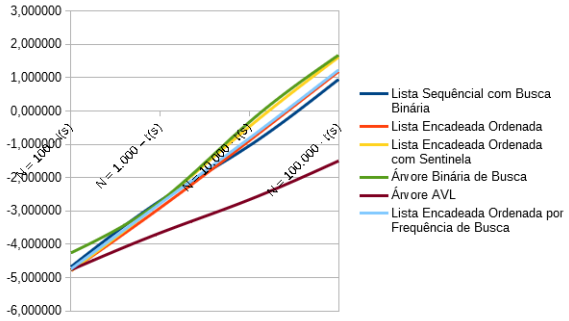


3.4 Remoção Crescente (após inserção crescente)

A remoção crescente se apresenta similar à inserção crescente, porém com tempos em geral menores, muito provavelmente devido a não ser necessária

a criação de novos elementos, principalmente nos casos onde ocorre alocação de memória.

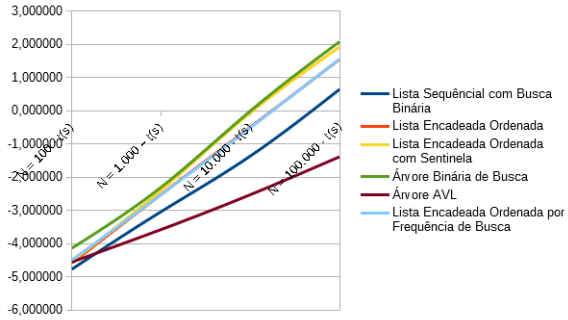
Tabela 4: Tempo Médio de Remoção Crescente (Após inserção crescente)				
Estrutura	**N = 100 - t(s)**	**N = 1.000 – t(s)**	**N = 10.000 - t(s)**	**N = 100.000 - t(s)**
Lista Sequêncial com Busca Binária	0.000021	0.002016	0.090355	8.832162
Lista Encadeada Ordenada	0.000016	0.001235	0.128573	15.189937
Lista Encadeada Ordenada com Sentinela	0.000016	0.001798	0.368078	41.192631
Árvore Binária de Busca	0.000055	0.001932	0.462588	47.527427
Árvore AVL	0.000017	0.000219	0.002179	0.031108
Lista Encadeada Ordenada por Frequência de Busca	0.000018	0.001659	0.151602	16.980517



3.5 Remoção Decrescente (após inserção crescente)

Na remoção decrescente - após inserção crescente - percebe-se que só a AVL se manteve veloz, pois nas demais estruturas foi necessário percorrer grandes porções das mesmas para encontrar o elemento a se remover.

Tabela 5: Tempo Médio de Remoção Decrescente (Após inserção crescente)				
Estrutura	**N = 100 - t(s)**	**N = 1.000 – t(s)**	**N = 10.000 - t(s)**	**N = 100.000 - t(s)**
Lista Sequêncial com Busca Binária	0.000017	0.000915	0.043143	4.377211
Lista Encadeada Ordenada	0.000027	0.003012	0.319679	35.077279
Lista Encadeada Ordenada com Sentinela	0.000029	0.004045	0.861347	82.773268
Árvore Binária de Busca	0.000073	0.004993	0.982507	118.839609
Árvore AVL	0.000028	0.000265	0.002962	0.041033
Lista Encadeada Ordenada por Frequência de Busca	0.000032	0.002964	0.306846	36.077974

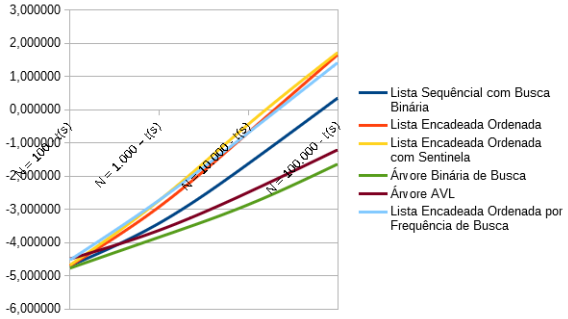


3.6 Remoção Aleatória (após inserção aleatória)

Nota-se novamente uma vantagem na árvore binária de busca na remoção aleatória, já nas demais estruturas não nota-se tanta diferença.

Tabela 6: Tempo Médio de Remoção Aleatória (Após inserção aleatória)

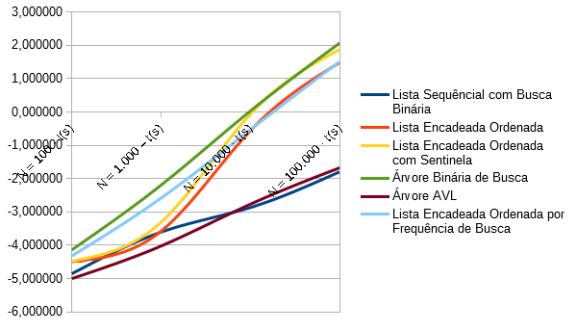
Estrutura	N = 100 - t(s)	N = 1.000 – t(s)	N = 10.000 - t(s)	N = 100.000 - t(s)
Lista Sequêncial com Busca Binária	0.000020	0.000383	0.023729	2.225570
Lista Encadeada Ordenada	0.000020	0.001193	0.206430	44.967314
Lista Encadeada Ordenada com Sentinela	0.000021	0.001852	0.364738	52.117188
Árvore Binária de Busca	0.000017	0.000145	0.001389	0.029662
Árvore AVL	0.000032	0.000234	0.003263	0.061776
Lista Encadeada Ordenada por Frequência de Busca	0.000030	0.001893	0.202530	25.910539



3.7 Busca (após inserção crescente)

A busca permite ver as vantagens da lista sequêncial com busca binária, sendo a mais rápida de todas no processo. A AVL, por também se utilizar de uma forma de busca binária também se saiu muito bem. As demais listas foram muito mais lentas por se tratarem de processos de busca sequêncial. Nota-se também que, apesar do nome, na ABB, devido a sua degeneração pela inserção sequêncial, acaba ocorrendo uma busca sequencial.

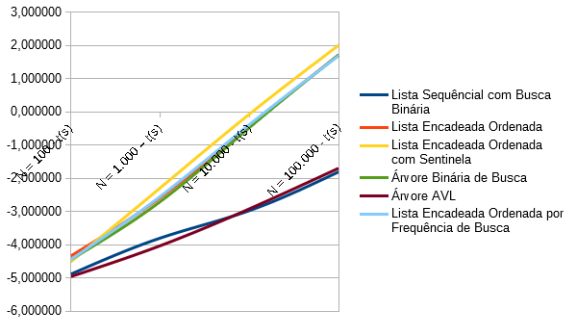
Tabela 7: Tempo Médio de Busca(Após inserção crescente)				
Estrutura	N = 100 - t(s)	N = 1.000 – t(s)	N = 10.000 - t(s)	N = 100.000 - t(s)
Lista Sequêncial com Busca Binária	0.000014	0.000248	0.001353	0.015924
Lista Encadeada Ordenada	0.000032	0.002649	0.272518	30.148608
Lista Encadeada Ordenada com Sentinela	0.000033	0.004922	0.857381	75.189667
Árvore Binária de Busca	0.000073	0.006385	1.067562	115.562601
Árvore AVL	0.000010	0.000095	0.001706	0.021067
Lista Encadeada Ordenada por Frequência de Busca	0.000048	0.002608	0.292530	32.397724



3.8 Busca (após inserção decrescente)

Para a busca após a inserção decrescente não se notam muitas mudanças.

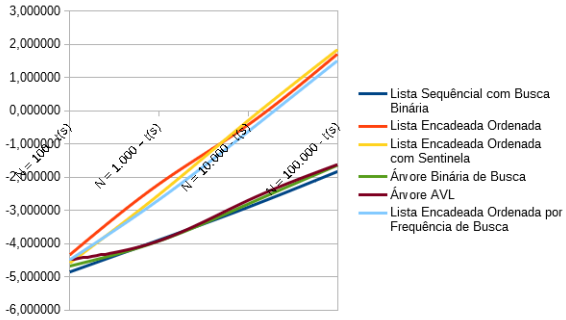
Tabela 8: Tempo Médio de Busca (Após inserção decrescente)				
Estrutura	**N = 100 - t(s)**	**N = 1.000 – t(s)**	**N = 10.000 - t(s)**	**N = 100.000 - t(s)**
Lista Sequêncial com Busca Binária	0.000013	0.000158	0.001061	0.015597
Lista Encadeada Ordenada	0.000046	0.002610	0.377517	50.725486
Lista Encadeada Ordenada com Sentinela	0.000031	0.005107	0.849357	100.199852
Árvore Binária de Busca	0.000036	0.001973	0.333871	51.515550
Árvore AVL	0.000011	0.000093	0.001203	0.023413
Lista Encadeada Ordenada por Frequência de Busca	0.000036	0.002826	0.412400	48.301905



3.9 Busca (após inserção aleatória)

Por fim, com a busca acontecendo após uma inserção aleatória, a ABB mostra novamente seu poder, porém não é o suficiente para vencer a lista sequêncial, que ganha no quesito tempo devido a suas constantes menores, provacadas pelo acesso direto, em contrapartida ao uso de pnteiros na ABB.

Tabela 9: Tempo Médio de Busca (Após inserção aleatória)				
Estrutura	**N = 100 - t(s)**	**N = 1.000 – t(s)**	**N = 10.000 - t(s)**	**N = 100.000 - t(s)**
Lista Sequêncial com Busca Binária	0.000014	0.000128	0.001246	0.014572
Lista Encadeada Ordenada	0.000023	0.001717	0.310899	57.797376
Lista Encadeada Ordenada com Sentinela	0.000026	0.002985	0.508499	68.441932
Árvore Binária de Busca	0.000021	0.000119	0.001556	0.022742
Árvore AVL	0.000033	0.000121	0.002004	0.023843
Lista Encadeada Ordenada por Frequência de Busca	0.000041	0.002106	0.235552	31.988790



4 Indicação de Estrutura para um Caso Geral

Baseando-se nos dados acima é fácil perceber a versatilidade da árvore AVL. Inspirada na árvore binária de busca, mas dotando-se de ferramentas para se manter balanceada, a AVL garante um tempo O(log(n)) para qualquer operação em qualquer caso. Sua maior desvantagem é uma constante relativamente maior que muitas outras estruturas devido aos acessos e operações com ponteiros e as operações de balanceamento. Mas, quando não se tem garantias sobre o conjunto de dados a ser usado, é sem dúvida a melhor aposta, sendo consistente em seus baixos tempos, mesmo para um número elevado de elementos.

5 Conclusão

O trabalho permite perceber os diferentes usos de estruturas de dados, suas vantagens e desvantagens, ensina que algumas estuturas podem ser muito úteis em casos específicos, mas extremamente lentas se usadas de maneira errada. Também mostra que outras apesar de muito boas em quase todos os casos, nunca alcançarão a eficiência dos casos ideais de outras menos versáteis. Em conclusão, um bom domínio das mais diversas estruturas e sua implementações garante a capacidade de escolher a ferramenta certa para o problema apresentado.