Projeto 3: Coleção

Henrique Hiram Libutti Núñez - 11275300 e Filipe Oliveira Costa - 11219161

Dezembro de 2019

1 Tempos encontrados

Tempos de inserção.

Estrutura	Exec. 1	Exec. 2	Exec. 3
LISTA ORDENADO	8.820660	8.826297	8.675530
LISTA ULTIMO	3.949594	3.943230	3.939000
LISTA PRIMEIRO	0.002154	0.002145	0.002146
ARVORE BINARIA	0.019238	0.019259	0.019167
ARVORE AVL	0.028355	0.028288	0.028143

Tempos de busca.

Estrutura	Exec. 1	Exec. 2	Exec. 3
LISTA ORDENADO	24.601939	24.970982	24.325650
LISTA ULTIMO	5.363966	5.352065	5.387579
LISTA PRIMEIRO	6.085266	6.078754	6.085131
ARVORE BINARIA	0.011489	0.011473	0.011579
ARVORE AVL	0.011395	0.011360	0.011468

2 Avaliação do tempo da Estrutura

1. O tempo de inserção seria similar ao da lista com inserção no início, uma vez não seria necessário percorrer todos os elementos, e sim apenas alterar os ponteiros nessa última posição.

2. A lista com inserção no início tem o menor tempo de inserção uma vez que é uma simples organização de ponteiros o seu processo de inserção. A lista ordenada tem o maior tempo de inserção pois ela tem que percorrer, no pior caso, n elementos, e no caso médio, n/2 elementos. Além disso provavelmente alguma otimização não pode ser feita pelo compilador, uma vez que o valor no próximo nó da lista não é previsível (o que muda o fluxo da execução), em contraste com a inserção no final, onde incondicionalmente o ponteiro temporário deve percorrer todas as posições até o final (NULL).

3. Sobre os valores nas listas:

   - A ordem dos valores de entrada importa apenas no caso da lista ordenada. Se os valores forem inseridos de forma decrescente, teríamos o menor tempo de execução nessa estrutura em particular.
   - A quantidade de valores é proporcional ao tempo de busca nas listas, no caso médio, onde uma busca linear tem complexidade $O(n)$.
   - Algo muito interessante ocorreu nas listas, uma vez que a de inserção no final teve seus ponteiros alocados próximos uns aos outros (aproximadamente consecutivos na mémória), o que em processadores que utilizam cache apresenta um grande benefício, uma vez que a memória próxima à "atual" será carregada nessa região de rápido acesso. No caso da lista de inserção no início, algumas regiões puderam ser carregadas no cache, enquanto outras não (pois foi alocada de forma reversa), o que apresentou uma performance intermediária entre a de inserção no final e a ordenada. Já o caso da lista ordenada, como deve haver uma reatribuição de ponteiros em grande parte dos casos, dificilmente um ponteiro está apontando para um endereço próximo, o que atrapalha o processador, ao ter que requsitar um novo acesso à RAM, o que resulta na lentidão da estrutura.
   - O tamanho dos valores não deve fazer diferença nessas estruturas.

4. Houve uma diferença da ordem de 1% na busca das estruturas de árvore. Provavelmente a ordem de inserção dos dados de entrada não desbalacearam significativamente a estrutura de árvore binária.

5. Não, se a entrada apresentar um padrão similar ao da entrada dada, a estrutura AVL não apresenta uma melhora sensível. É importante ressaltar que para alguns casos, a árvore binária de busca pode se degenerar e tornar o tempo de busca similar ao de uma lista, o que favorece o uso da AVL.

6. Com essa entrada em específico, a estrutura com melhor performance foi, de acordo com os testes realizados, a árvore binária de busca. Mas para uma entrada geral, a melhor estrutura é a AVL, que independentemente da entrada será balanceada, e assim terá uma complexidade de busca exatamente de $O(n \log n)$.

7. Algo aprendido no desenvolvimento desse projeto foram as questões relacionadas a cache, que atrapalharam muito o desempenho da estrutura de lista ordenada e de lista com inserção no início (em menor escala).