Instituto de Ciência e Tecnologia - São José dos Campos, SP
UC: Algoritmos e Estrutura de Dados 2 - 2º semestre/2020
Prof.: Dra. Lilian Berton
Aluno: Davi Juliano Ferreira Alves - 133595

Atividade 6 - Prática

Entrega: 12/01/2020

Análise de complexidade do código para Árvore Binária de Busca, vetores ordenados e desordenados

A estrutura de Árvore Binária é dada por nós que contém informações, que no código foram ditas como “info”. Para a inserção, utilizamos as funções rand e as funções do pacote time.h. Para a inserção do vetor aleatório, foi utilizado apenas a rand, entretanto, para o vetor ordenado, utilizamos o Insertion Sort para ordenar o vetor. Ainda na mesma lógica, foi utilizado o vetor randômico para gerar a ABB1 e o vetor ordenado para gerar a ABB2 e foi computado ambos valores. Como a ABB1 e a ABB2 possuem a mesma informação, mas alturas e inserções distintas, continuamos os próximos passos com a ABB1. A tabela do tempo pode ser observada abaixo:

	Inserir 10.000 elementos	Pesquisa elemento 50	Pesquisa elemento 5000
Vetor Randômico	0.00026 segundos	0.000024 segundos	0.000016 segundos
Vetor Ordenado (Pelo Insertion Sort)	0.11883 segundos	0.000001 segundos	0.000033 segundos
Árvore Binária de Busca 1	0.02905 segundos	0.000003 segundos	0.000004 segundos
Árvore Binária de Busca 2	0.232789 segundos	X	X

Figura 1 - Tabela dos Tempos de cada uma das ações pedidas.

```
A geração de valores randômicos demorou 0.000260 segundos
O algoritmo do Insertion Sort demorou 0.118883 segundos para ordenar o vetor randômico
A inserção da ABB demorou 0.002905 segundos para inserir o vetor randômico
A inserção da ABB demorou 0.232789 segundos para inserir o vetor ordenado
```

Figura 2 - O Output dos tempos de inserção.

```
O Busca Vetor demorou 0.000001 segundos para buscar 50 no vetor ordenado
O Busca Vetor demorou 0.000024 segundos para buscar 50 no vetor randômico
O Busca ABB demorou 0.000003 segundos para buscar 50 na Árvore Binária
```

Figura 3 - O Output dos tempos de busca no valor 50 nas estruturas de dados

```
O Busca Vetor demorou 0.000033 segundos para buscar 5000 no vetor ordenado
O Busca Vetor demorou 0.000016 segundos para buscar 5000 no vetor randômico
O Busca ABB demorou 0.000004 segundos para buscar 5000 na Árvore Binária
```

Figura 4 - O Output dos tempos de busca no valor 5000 nas estruturas de dados

Analisando a complexidade de cada uma das ações, temos que a inserção do vetor randômico é dado por um fator linear de complexidade O(n), enquanto a inserção do vetor ordenado por meio do Insertion Sort é dado pela complexidade $O(n^2)$ no pior caso de ordenação. A inserção da árvore binária de busca também segue um fator O(n) caso seja ordenado decrescentemente, entretanto, se o vetor for ordenado crescentemente, a complexidade chega a $O(n^2)$ (Visto que a própria estrutura de dados não permite um valor pequeno demais como raíz, ou seja, a árvore será apenas uma sub-árvore à direita (Suponha n como raíz, no próximo vetor (n+m) > n, e n+m entrará à direita de n e assim sucessivamente)).

Para as buscas, temos um fator linear O(n) para os vetores, sendo o vetor ordenado melhor para menores valores e o vetor randômico uma incógnita, pois os valores são aleatórios e nunca se sabe onde está um certo n. O ponto que a busca em vetores foi dada de maneira linear O(n), enquanto a busca da ABB é dada por uma O(h), pois ela sempre percorre metade dos valores que os vetores ordenam, ou seja, se n é maior que a raiz, ele vai pra direita, caso contrário, vai pra esquerda, e assim sucessivamente.

Como utilizamos duas Árvores Binárias logo acima, tomamos a dos vetores randômicos para fazer a análise da altura das sub-árvores e para imprimir as folhas. Pois temos o resultado como:

```
A Altura da Sub-ABB1 na direita é 17
A Altura da Sub-ABB1 na esquerda é 25
```

Figura 4 - O Output da altura das sub-árvores.

O código da altura das árvores é dado por uma complexidade O(h), sendo h a altura da árvore, pois é uma recursão que faz todas as averiguações da altura da árvore, ou seja, passa por todos os nós da árvore até chegar nas folhas. A função de mostrar as folhas das árvores possuem a mesma complexidade O(h), pois precisa mostrar toda e qualquer folha que exista nas ramificações da árvore.