UTFPR - Universidade Tecnológica Federal do Paraná

Campus Campo Mourão

Aluno: *Jessé P. B. Rocha*

RA: *2149389*

Disciplina: *Algoritmos e Estruturas de Dados 2*

ADNP - SEMANA 12

d. Para N = {1000, 10000, 100000, 500000} ordene um vetor com N inteiros em ordem decrescente usando as funções implementadas nos itens a, b e c. Anote o tempo de execução na Tabela 1. Na Tabela 2 anote a altura da árvore logo após o passo 2 do algoritmo de ordenação descrito acima.

	N			
	1000	**10000**	**100000**	**500000**
ABB	0.013543	0.666482	66.726898	333.63449
ARN	0.000433	0.006268	0.072903	0.600042
AVL	0.000192	0.002400	0.038272	0.197130

Tabela 1: Tempo de Execução (em s) para Ordenar Vetores com N Elementos em Ordem Decrescente

	N			
	1000	**10000**	**100000**	**500000**
ABB	999	9999	99999	499999
ARN	14	17	21	24
AVL	9	13	16	18

Tabela 2: Altura das Árvores Antes do Percurso Em-Ordem

e. Para N = {1000, 10000, 100000, 500000} ordene um vetor com N inteiros gerados aleatoriamente usando as funções implementadas nos itens a, b e c. Para cada N e cada função execute a ordenação com 10 vetores aleatórios.

Anote o tempo de execução médio entre todas as execuções e o desvio padrão na Tabela 3. Na Tabela 4 anote a média da altura das árvores logo após o passo 2 do algoritmo de ordenação descrito acima para todas as execuções, juntamente com o desvio padrão.

	N			
	1000	**10000**	**100000**	**500000**
ABB	0.000603	0.004680	0.079143	0.718259
ARN	0.000435	0.006002	0.097810	0.797810
AVL	0.000238	0.003542	0.058769	0.528201

Tabela 3: Tempo de Execução (em s, média +- desvio) para Ordenar Vetores Aleatórios com N Elementos

	N			
	1000	**10000**	**100000**	**500000**
ABB	20	30	39	46
ARN	11	16	21	24
AVL	10	14	18	21

Tabela 4: Altura das Árvores Antes do Percurso Em-Ordem

f. Houve diferença significativa entre o tempo de execução entre os três métodos avaliados no caso da ordenação dos vetores inicialmente em ordem decrescente? Se sim, qual foi o melhor método? Qual foi o pior? Justifique suas respostas.

Dos três métodos utilizados, o menor tempo foi o que utiliza a árvore *AVL* como estrutura auxiliar. Isso se dá pelo fato de tal árvore conseguir manter a menor altura entre todos os métodos, por ser balanceada. Isso faz com que o tempo para percorrê-la seja, por consequência, menor.

Por outro lado, o método que utiliza a *ABB* teve o maior tempo gasto na ordenação. Isso ocorre porque, como a *Árvore de Busca Binária* não possui uma rotina para balanceá-la, sua altura dependerá da ordem em que os elementos serão

inseridos. Uma vez que os vetores são inseridos em ordem decrescente, a árvore ficará praticamente uma lista encadeada. Logo, o custo para percorrer todos os seus elementos fica muito alto em comparação aos outros dois métodos.

g. Houve diferença significativa entre o tempo de execução entre os três métodos avaliados no caso da ordenação dos vetores aleatórios? Se sim, qual foi o melhor método? Qual foi o pior? Justifique suas respostas.

Novamente, o menor tempo continuou sendo do método que utiliza a *AVL* como estrutura auxiliar. Pelos mesmos motivos explicitados no item *f*, o árvore *AVL* conseguiu mais uma vez manter a menor altura entre todos os métodos. Ela consegue ganhar até mesmo da árvore *Rubro-Negra*, pois, como foi estudado, ela "pende" para a esquerda ou para a direita (dependendo da implementação). Logo, a *AVL* consegue ser mais eficiente também nos casos em que o vetor é aleatório.

Assim como no caso dos vetores decrescentes, o maior tempo gasto foi no método que utiliza a *ABB* como estrutura auxiliar. Ainda que, para o caso analisado, sua altura não foi tão grande quanto no caso anterior, ela continuou sendo a maior, o que faz com que o tempo gasto para ordenar o vetor seja maior.

h. Em cada um dos métodos (usando ABB, AVL, ARN), o tempo de execução foi muito diferente entre a ordenação dos vetores inicialmente em ordem decrescente e dos vetores aleatórios? Em qual método houve a maior variação? E a menor variação?

A maior variação de tempo, ocorreu com o método que utiliza a *ABB*. Isso se deve ao fato de que a altura de uma determinada árvore deste tipo depende da ordem em que os elementos serão inseridos. Quando se tem um vetor aleatório, as chances de se ter uma árvore totalmente degenerada (como no caso do vetor decrescente), são menores. Por isso da variação tão grande de tempo.

Já a menor variação de tempo ficou para o método que utiliza a *ARN*. Uma vez que esta é capaz de reorganizar os elementos após uma inserção para manter-se balanceada, o tempo para ordenar um vetor aleatório e em ordem decrescente fica mais próximo.

i. No item anterior você percebeu que em um dos métodos a variação do tempo de execução entre a ordenação de vetores inicialmente em ordem

decrescente e dos vetores aleatórios é grande. Como você pode diminuir essa diferença?

Já sabemos que o método que teve maior variação de tempo foi o que utiliza a *ABB*. Também sabemos que o tempo para ordenar um vetor decrescente foi maior que o tempo para ordenar o vetor aleatório.

Logo, com tais informações, podemos chegar à conclusão de que uma alternativa para reduzir a diferença é embaralhar o vetor inicialmente em ordem decrescente. Pode-se, para tanto, utilizar um algoritmo de *Fisher-Yates*. Um vetor aleatório continuaria aleatório e o vetor decrescente deixaria de manter esta ordem. Assim, a diferença de tempo poderia ser reduzida. Pode-se gastar um pouco mais de tempo em razão de se introduzir a rotina de embaralhamento no processo, mas a diferença seria reduzida.