**Exercício 2)** Para N = 10^6, com geração de números pseudoaleatórios, foram obtidos os seguintes resultados: [48, 49, 55, 48, 48, 58, 50, 51, 47, 50], em que cada item do vetor representa a altura da árvore resultante em uma execução, tendo sido feitas 10 execuções. A altura média obtida nessas dez execuções foi de 50,4 e o desvio padrão de 3.322649545.

ITER HEIGHT

0 48

1 49

2 55

3 48

4 48

5 58

6 50

7 51

8 47

9 50

AVG 50.4

STD 3.322649545

**1)** A conclusão que se pode obter é que, perto da magnitude da altura da árvore, o desvio padrão é relativamente pequeno, o que indica que a árvore mantém uma altura dentro de um intervalo esperado com variação pequena entre os extremos. Além disso, a altura da árvore ficou pequena em relação ao número de itens inseridos, sendo que cenário parecido foi obtido para N maiores, o que indica que essa estrutura de dados lida bem com entrada de chaves aleatórias em relação a sua altura.

**2)** Se uma entrada de chaves estiver ordenada e for inserida na BST em ordem, todas as chaves ficarão em um único lado da árvores e ela se tornará uma basicamente uma lista encadeada. Assim, a altura máxima que uma árvore pode obter é quando cada nó representa um nível (inserção ordenada) e é de N - 1. Para uma entrada com itens relativamente embaralhados e, considerando que a inserção irá produzir uma árvores balanceada, a altura da árvore só cresce quando N é uma potência de 2 (1 nível para N = 1, 2 níveis para N = 2, 3 níveis para N = 4, etc.). Assim, altura mínima que uma BST pode possuir é |\_lgN\_|, (função floor do logaritmo de N em base 2).

**3)** Para o experimento descrito no início do exercício e os resultados obtidos, a altura da árvore ficou boa para números aleatórios pois ela não cresceu muito, mesmo com N grande. Assim, para entradas aleatórias, a média da altura é boa e fica próxima à lg(N). Entretanto, se as entradas tivessem sido ordenadas antes da inserção na BST, ela se tornaria uma lista encadeada e sua altura seria N - 1, o que é ruim.

**Exercício 6)**

Resultados de tempo de execução (em segundos) para os algoritmos de caminhamento

(versões recursivas e não-recursivas) para N = 10^7.

Type Rec Iter

Pre-order 1.149537 1.125477

In-order 1.186208 2.390708

Post-order 1.034002 3.097419

As versões recursivas obtiveram desempenho superior para N = 10^7 (com exceção do Pre-Order, embora a diferença entre as versões foi mínima). As versões iterativas, apesar de demonstrarem de forma menos abstrata o caminhamento na árvore, exigem outras estruturas de dados complementares e a execução desse caminhamento acaba envolvendo também as complexidades de tempo dessas estruturas auxiliares. Assim, a versão recursiva do caminhamento acaba tornando-se mais rápida e obtém também a vantagem de que o código torna-se mais limpo e organizado, com apenas 3 linhas de código.