



Prática 01 – Implementação em JAVA do TAD Árvore Binária de Pesquisa

- **Data de Entrega: 22/03/2019**
- **Deve ser entregue um relatório no moodle contendo:**
 - **Nome, Data de entrega, Número da Prática e Título da Prática;**
 - **Gráfico dos tópicos C e D do item 4;**
 - **Item 5;**
- **Deve ser entregue, via moodle, projeto contendo o código fonte comentado;**

1) Utilizando o Netbeans, crie um projeto chamado *Prática01*;

2) Implemente a classe **Item**, como especificada abaixo para ser utilizada no T.A.D.;

```
package Item;
public class Item {
    private int chave;
    public Item(int chave) {
        this.chave = chave;
    }
    public int compara(Item it) {
        Item item = it;
        if (this.chave < item.chave)
            return -1;
        else if (this.chave > item.chave)
            return 1;
        return 0;
    }
    public int getChave() {
        return chave;
    }
}
```

3) Implemente uma classe chamada **ArvoreBinaria** para manipular uma árvore binária de pesquisa onde os nós da árvore são objetos da classe **No**, especificada abaixo;

```
private static class No {
    Item reg;
    No esq, dir;
}
```

A classe ArvoreBinaria deve conter os seguintes métodos:

- **public ArvoreBinaria():** para inicializar o nó raiz;
- **private No insere(Item reg, No p):** para inserir o elemento **reg** passado por parâmetro;
- **private Item pesquisa(Item reg, No p):** para realizar a busca do elemento **reg** passado por parâmetro;

Cada método deve estar comentado;

4) Realizar os seguintes experimentos:

a) gerar árvores a partir de **n** elementos **ORDENADOS**, com **n** variando de 1.000 até 9.000, com intervalo de 1.000.

Em cada árvore gerada pesquisar por um elemento **não existente** e verificar o número de comparações realizadas e o tempo gasto (dica: função **System.nanoTime()**) na pesquisa em cada árvore;

b) gerar árvores a partir de **n** elementos **ALEATÓRIOS** (**long j = obj.nextInt()**), com **n** variando de 1.000 até 9.000, com intervalo de 1.000.

Em cada árvore gerada pesquisar por um elemento **não existente** e verificar o número de comparações realizadas e o tempo gasto na pesquisa em cada árvore;

- c) Fazer um único gráfico de **n x número de comparações** levando em consideração as árvores geradas com inserções ordenadas e aleatórias;
- d) Fazer um único gráfico de **n x tempo gasto** levando em consideração as árvores geradas com inserções ordenadas e aleatórias;

5) Explique o comportamento dos gráficos gerados