Implementação de Tabela de Símbolos

Departamento de Computação- UFPI

Estruturas de Dados – 2019.2

Com o intuito de analisar o desempenho de uma Tabela de Símbolos nas buscas binária e sequencial, realizou-se experimentos com as classes “BinarySearchST” e “SequentialSearchST” por meio de uma classe “Principal”, que além dessas duas classes, utiliza da classe “Stopwatch” para análise da estrutura de dados.

Para começar, as classes de busca, por meio de dois “arrays”, realizam a relação de Chave-Valor inserindo um elemento no mesmo índice de ambos os “arrays”. Para realizar a busca binária por meio da classe “BinarySearchST”, as seguintes linhas de código estão implementadas:

```
public int rank(Key key) {
    if (key == null) throw new IllegalArgumentException("argument to rank() is null");

    int lo = 0, hi = n-1;
    while (lo <= hi) {
        int mid = lo + (hi - lo) / 2;
        int cmp = key.compareTo(keys[mid]);
        if      (cmp < 0) hi = mid - 1;
        else if (cmp > 0) lo = mid + 1;
        else return mid;
    }
    return lo;
}
```

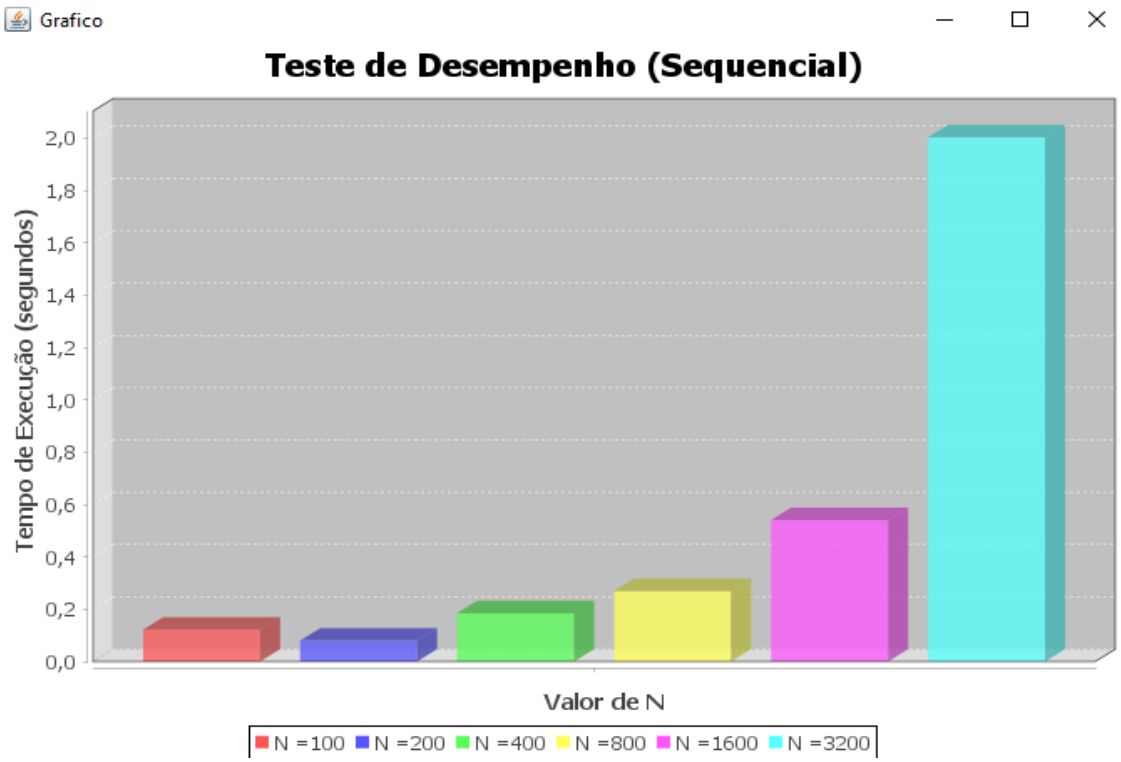
Já a busca sequencial implementada pela classe “SequentialSearchST” consiste em apenas um for percorrendo todos os elementos do vetor, da seguinte maneira:

```
public Value get(Key key) {
    if (key == null) throw new IllegalArgumentException("argument to get() is null")
    for (Node x = first; x != null; x = x.next) {
        if (key.equals(x.key))
            return x.val;
    }
    return null;
}
```

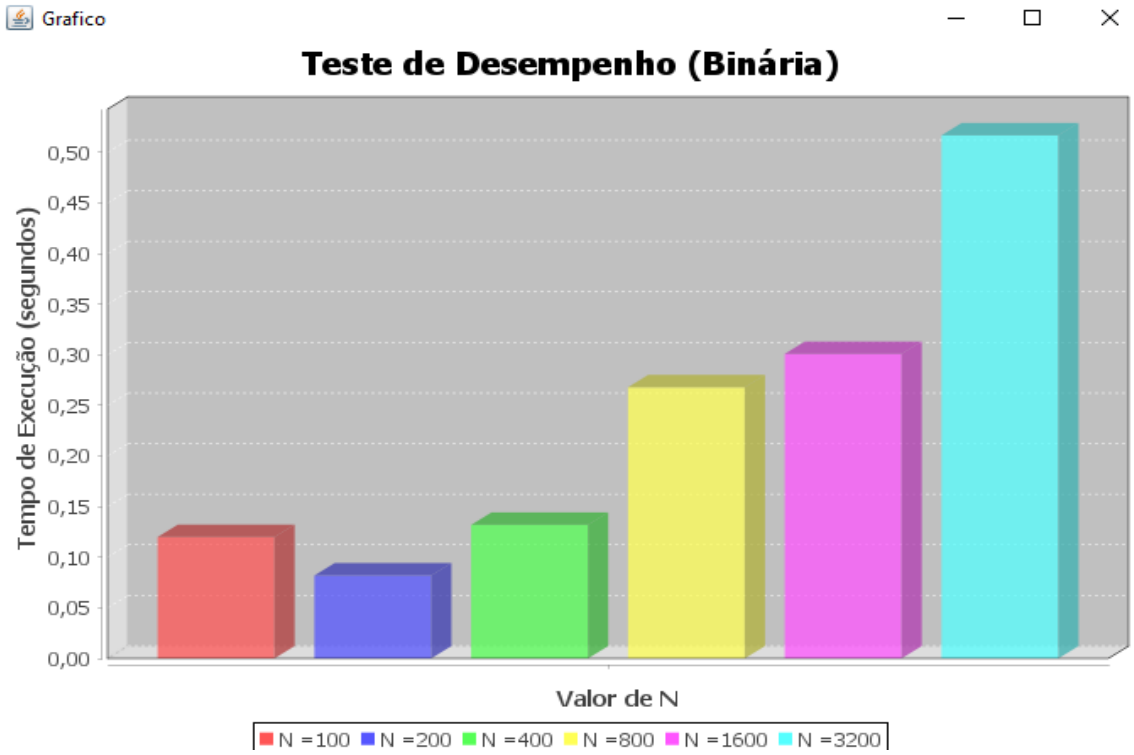
Agora, na classe “Principal”, desenvolveu-se uma aplicação que pergunta ao usuário o número de chaves aleatórias desejadas para realizar o experimento, a quantidade de vezes que ele deve ser repetido e qual método de busca usar. O experimento em questão é para medir o tempo levado para realizar 10 buscas bem-sucedidas e 10 buscas malsucedidas, repetindo essa ação quantas vezes desejada pelo usuário. Ademais, a cada nova interação de buscas, o número de chaves inicialmente digitado é dobrado e um novo vetor de chaves é criado com a nova quantidade de chaves requeridas.

Nesse sentido, para gerar as chaves optou-se pela classe “Random” do java. Os valores gerados são inteiros e guardados em um vetor de elementos únicos para a posterior passagem desses elementos para as classes de busca. Ainda assim, vale pontuar que se o vetor passado para as classes de busca não possuísse elementos únicos, ao inserir a chave repetida, apenas o Valor para aquela chave mudaria, assim, não permitindo chaves duplicadas.

Para finalizar, de acordo com o selecionado pelo usuário, o gráfico é gerado, mostrando o tempo de execução das buscas bem-sucedida e malsucedidas na mesma barra. Cada barra corresponde a um determinado valor de chaves utilizados na interação. Segue imagem da execução da busca binária e sequencial:



No experimento acima o valor escolhido para o número inicial das chaves foi 100. O número de repetições foi 6, portanto, as 6 barras foram geradas.



No experimento acima, de busca binária, o valor inicial de chaves foi 100, sendo realizadas 6 interações também.

Obs.: Para execução do projeto basta, pelo cmd executar a seguinte linha:

java –jar TabelaSimbolos.jar

Componentes:

Gabriel Menezes Moreira

Krisna Calixto de Carvaho Alves da Silva