Laboratório 06 – Quicksort e seu pivô

Mateus Resende Ottoni

Link para o repositório do código: AEDS2/Laboratório/LAB06/Quicksort.java at main · Mateus-Resende-Ottoni/AEDS2 (github.com)

Código:

```java
import java.util.*;

public class Quicksort {
    protected int[] array;
    protected int n;

    /* Construtores */

    public Quicksort() {
        array = new int[100];
        n = array.length;
    }

    public Quicksort(int tamanho) {
        array = new int[tamanho];
        n = array.length;
    }

    public Quicksort(Quicksort clone) {
        array = new int[clone.array.length];
        n = array.length;

        for (int x = 0; x < n; x++) {
            array[x] = clone.array[x];
        }
    }

    /* */

    /* Alteração de valores */
    public void crescente() {
        for (int i = 0; i < n; i++) {
            array[i] = i;
        }
    }

    public void decrescente() {
```

```
        for (int i = 0; i < n; i++) {
            array[i] = n - 1 - i;
        }
    }

    public void semialeatorio() {
        this.crescente();

        Random rand = new Random();
        rand.setSeed(System.currentTimeMillis());
        for (int i = 0; i < (n / 20); i++) {
            int random = Math.abs(rand.nextInt()) % n;
            if (random == n - 1) {
                swap(random, random - 1);
            } else {
                swap(random, random + 1);
            }
        }
    }

    public void aleatorio() {
        Random rand = new Random();
        rand.setSeed(System.currentTimeMillis());
        crescente();
        for (int i = 0; i < n; i++) {
            swap(i, Math.abs(rand.nextInt()) % n);
        }
    }

    /* */

    /* Algoritmos de Quicksort */
    private void QuicksortFirstPivot(int esq, int dir) {
        int i = esq, j = dir;
        int pivo = array[esq];
        while (i <= j) {
            while (array[i] < pivo)
                i++;
            while (array[j] > pivo)
                j--;
            if (i <= j) {
                swap(i, j);
                i++;
                j--;
            }
        }
        if (esq < j)
```

```
            QuicksortFirstPivot(esq, j);
        if (i < dir)
            QuicksortFirstPivot(i, dir);
    }

    private void QuicksortLastPivot(int esq, int dir) {
        int i = esq, j = dir;
        int pivo = array[dir];
        while (i <= j) {
            while (array[i] < pivo)
                i++;
            while (array[j] > pivo)
                j--;
            if (i <= j) {
                swap(i, j);
                i++;
                j--;
            }
        }
        if (esq < j)
            QuicksortLastPivot(esq, j);
        if (i < dir)
            QuicksortLastPivot(i, dir);
    }

    private void QuicksortRandomPivot(int esq, int dir) {
        int i = esq, j = dir;
        Random rand = new Random();
        rand.setSeed(System.currentTimeMillis());
        int n_rand = (Math.abs(rand.nextInt()) % (dir - esq)) + esq;
int pivo = array[n_rand];
        while (i <= j) {
            while (array[i] < pivo)
                i++;
            while (array[j] > pivo)
                j--;
            if (i <= j) {
                swap(i, j);
                i++;
                j--;
            }
        }
        if (esq < j)
            QuicksortRandomPivot(esq, j);
        if (i < dir)
            QuicksortRandomPivot(i, dir);
    }
```

```java
private void QuicksortMedianOfThreePivot(int esq, int dir) {
    int i = esq, j = dir;

    int numeros[] = new int[3];

    /* Organizacao para achar a Mediana dos 3 escolhidos */
    int first = array[esq];
    int mid = array[(esq + dir) / 2];
    int end = array[dir];
    numeros[0] = first;
    numeros[1] = mid;
    numeros[2] = end;

    int x = 0;
    boolean organized = false;
    while (x < 3 && organized == false) {
        organized = true;

        int y = 0;
        while (y < 2) {
            if (numeros[y] > numeros[y + 1]) {
                int temp = numeros[y];
                numeros[y] = numeros[y + 1];
                numeros[y + 1] = temp;

                organized = false;
            }
            y++;
        }

        x++;
    }
    /* */

    int pivo = array[numeros[1]];
    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j) {
            swap(i, j);
            i++;
            j--;
        }
```

```
        }
        if (esq < j)
            QuicksortMedianOfThreePivot(esq, j);
        if (i < dir)
            QuicksortMedianOfThreePivot(i, dir);
    }

    /* */

    /* Outros */

    public void mostrar() {
        System.out.print("[");

        for (int i = 0; i < n; i++) {
            System.out.print(" (" + i + ")" + array[i]);
        }

        System.out.println("]");
    }

    public void swap(int i, int j) {
        int temp = array[i];
        array[i] = array[j];
        array[j] = temp;
    }

    public long now() {
        return new Date().getTime();
    }

    /* */

    public static void main(String[] args) {
        double inicio1 = 0.0000, fim1 = 0.0000;
        double inicio2 = 0.0000, fim2 = 0.0000;
        double inicio3 = 0.0000, fim3 = 0.0000;
        double inicio4 = 0.0000, fim4 = 0.0000;

        /* Array Aleatório - INÍCIO */
        /* Teste com array de 100 elementos */

        Quicksort array1 = new Quicksort(100);
        array1.aleatorio();
        Quicksort array2 = new Quicksort(array1);
```

```
        Quicksort array3 = new Quicksort(array1);
        Quicksort array4 = new Quicksort(array1);

        System.out.println("");
        System.out.println("-------------------------");
        System.out.println("");
        System.out.println("");

        inicio1 = array1.now();
        array1.QuicksortFirstPivot(0, (array1.n - 1));
        fim1 = array1.now();
        inicio2 = array1.now();
        array2.QuicksortLastPivot(0, (array2.n - 1));
        fim2 = array1.now();
        inicio3 = array1.now();
        array3.QuicksortRandomPivot(0, (array3.n - 1));
        fim3 = array1.now();
        inicio4 = array1.now();
        array4.QuicksortMedianOfThreePivot(0, (array4.n - 1));
        fim4 = array1.now();

        System.out.println("Array organizado: Aleatório _ 100 elementos");
        System.out.println("Tempos para ordenar: ");
        System.out.printf("FirstPivot: %.4fs.\n", (fim1 - inicio1) / 1000);
        System.out.printf("LastPivot: %.4fs.\n", (fim2 - inicio2) / 1000);
        System.out.printf("RandomPivot: %.4fs.\n", (fim3 - inicio3) / 1000);
        System.out.printf("MedianOfThreePivot: %.4fs.\n", (fim4 - inicio4) /
1000);
        System.out.println("");
        System.out.println("-------------------------");

        /* */

        /* Teste com array de 1000 elementos */

        array1 = new Quicksort(1000);
        array1.aleatorio();
        array2 = new Quicksort(array1);
        array3 = new Quicksort(array1);
        array4 = new Quicksort(array1);

        System.out.println("");
        System.out.println("-------------------------");
        System.out.println("");
        System.out.println("");
```

```
        inicio1 = array1.now();
        array1.QuicksortFirstPivot(0, (array1.n - 1));
        fim1 = array1.now();
        inicio2 = array1.now();
        array2.QuicksortLastPivot(0, (array2.n - 1));
        fim2 = array1.now();
        inicio3 = array1.now();
        array3.QuicksortRandomPivot(0, (array3.n - 1));
        fim3 = array1.now();
        inicio4 = array1.now();
        array4.QuicksortMedianOfThreePivot(0, (array4.n - 1));
        fim4 = array1.now();

        System.out.println("Array organizado: Aleatório _ 1000 elementos");
        System.out.println("Tempos para ordenar: ");
        System.out.printf("FirstPivot: %.4fs.\n", (fim1 - inicio1) / 1000);
        System.out.printf("LastPivot: %.4fs.\n", (fim2 - inicio2) / 1000);
        System.out.printf("RandomPivot: %.4fs.\n", (fim3 - inicio3) / 1000);
        System.out.printf("MedianOfThreePivot: %.4fs.\n", (fim4 - inicio4) /
1000);
        System.out.println("");
        System.out.println("-------------------------");

        /* */

        /* Teste com array de 10000 elementos */

        array1 = new Quicksort(10000);
        array1.aleatorio();
        array2 = new Quicksort(array1);
        array3 = new Quicksort(array1);
        array4 = new Quicksort(array1);

        System.out.println("");
        System.out.println("-------------------------");
        System.out.println("");
        System.out.println("");

        inicio1 = array1.now();
        array1.QuicksortFirstPivot(0, (array1.n - 1));
        fim1 = array1.now();
        inicio2 = array1.now();
        array2.QuicksortLastPivot(0, (array2.n - 1));
        fim2 = array1.now();
        inicio3 = array1.now();
        array3.QuicksortRandomPivot(0, (array3.n - 1));
```

```
        fim3 = array1.now();
        inicio4 = array1.now();
        array4.QuicksortMedianOfThreePivot(0, (array4.n - 1));
        fim4 = array1.now();

        System.out.println("Array organizado: Aleatório _ 10000 elementos");
        System.out.println("Tempos para ordenar: ");
        System.out.printf("FirstPivot: %.4fs.\n", (fim1 - inicio1) / 1000);
        System.out.printf("LastPivot: %.4fs.\n", (fim2 - inicio2) / 1000);
        System.out.printf("RandomPivot: %.4fs.\n", (fim3 - inicio3) / 1000);
        System.out.printf("MedianOfThreePivot: %.4fs.\n", (fim4 - inicio4) /
1000);
        System.out.println("");
        System.out.println("-------------------------");

        /* */

        /* Teste com array de 17500 elementos */

        array1 = new Quicksort(17500);
        array1.aleatorio();
        array2 = new Quicksort(array1);
        array3 = new Quicksort(array1);
        array4 = new Quicksort(array1);

        System.out.println("");
        System.out.println("-------------------------");
        System.out.println("");
        System.out.println("");

        inicio1 = array1.now();
        array1.QuicksortFirstPivot(0, (array1.n - 1));
        fim1 = array1.now();
        inicio2 = array1.now();
        array2.QuicksortLastPivot(0, (array2.n - 1));
        fim2 = array1.now();
        inicio3 = array1.now();
        array3.QuicksortRandomPivot(0, (array3.n - 1));
        fim3 = array1.now();
        inicio4 = array1.now();
        array4.QuicksortMedianOfThreePivot(0, (array4.n - 1));
        fim4 = array1.now();

        System.out.println("Array organizado: Aleatório _ 17500 elementos");
        System.out.println("Tempos para ordenar: ");
        System.out.printf("FirstPivot: %.4fs.\n", (fim1 - inicio1) / 1000);
```

```
        System.out.printf("LastPivot: %.4fs.\n", (fim2 - inicio2) / 1000);
        System.out.printf("RandomPivot: %.4fs.\n", (fim3 - inicio3) / 1000);
        System.out.printf("MedianOfThreePivot: %.4fs.\n", (fim4 - inicio4) /
1000);
        System.out.println("");
        System.out.println("-------------------------");

        /* */

        /* Array Aleatório - FIM */

        /* Array Organizado - INÍCIO */
        /* Teste com array de 100 elementos */

        array1 = new Quicksort(100);
        array1.crescente();
        array2 = new Quicksort(array1);
        array3 = new Quicksort(array1);
        array4 = new Quicksort(array1);

        System.out.println("");
        System.out.println("-------------------------");
        System.out.println("");
        System.out.println("");

        inicio1 = array1.now();
        array1.QuicksortFirstPivot(0, (array1.n - 1));
        fim1 = array1.now();
        inicio2 = array1.now();
        array2.QuicksortLastPivot(0, (array2.n - 1));
        fim2 = array1.now();
        inicio3 = array1.now();
        array3.QuicksortRandomPivot(0, (array3.n - 1));
        fim3 = array1.now();
        inicio4 = array1.now();
        array4.QuicksortMedianOfThreePivot(0, (array4.n - 1));
        fim4 = array1.now();

        System.out.println("Array organizado: Organizado _ 100 elementos");
        System.out.println("Tempos para ordenar: ");
        System.out.printf("FirstPivot: %.4fs.\n", (fim1 - inicio1) / 1000);
        System.out.printf("LastPivot: %.4fs.\n", (fim2 - inicio2) / 1000);
        System.out.printf("RandomPivot: %.4fs.\n", (fim3 - inicio3) / 1000);
        System.out.printf("MedianOfThreePivot: %.4fs.\n", (fim4 - inicio4) /
1000);
        System.out.println("");
```

```
        System.out.println("-------------------------");

        /* */

        /* Teste com array de 1000 elementos */

        array1 = new Quicksort(1000);
        array1.crescente();
        array2 = new Quicksort(array1);
        array3 = new Quicksort(array1);
        array4 = new Quicksort(array1);

        System.out.println("");
        System.out.println("-------------------------");
        System.out.println("");
        System.out.println("");

        inicio1 = array1.now();
        array1.QuicksortFirstPivot(0, (array1.n - 1));
        fim1 = array1.now();
        inicio2 = array1.now();
        array2.QuicksortLastPivot(0, (array2.n - 1));
        fim2 = array1.now();
        inicio3 = array1.now();
        array3.QuicksortRandomPivot(0, (array3.n - 1));
        fim3 = array1.now();
        inicio4 = array1.now();
        array4.QuicksortMedianOfThreePivot(0, (array4.n - 1));
        fim4 = array1.now();

        System.out.println("Array organizado: Organizado _ 1000 elementos");
        System.out.println("Tempos para ordenar: ");
        System.out.printf("FirstPivot: %.4fs.\n", (fim1 - inicio1) / 1000);
        System.out.printf("LastPivot: %.4fs.\n", (fim2 - inicio2) / 1000);
        System.out.printf("RandomPivot: %.4fs.\n", (fim3 - inicio3) / 1000);
        System.out.printf("MedianOfThreePivot: %.4fs.\n", (fim4 - inicio4) /
1000);
        System.out.println("");
        System.out.println("-------------------------");

        /* */

        /* Teste com array de 10000 elementos */

        array1 = new Quicksort(10000);
        array1.crescente();
```

```
        array2 = new Quicksort(array1);
        array3 = new Quicksort(array1);
        array4 = new Quicksort(array1);

        System.out.println("");
        System.out.println("-------------------------");
        System.out.println("");
        System.out.println("");

        inicio1 = array1.now();
        array1.QuicksortFirstPivot(0, (array1.n - 1));
        fim1 = array1.now();
        inicio2 = array1.now();
        array2.QuicksortLastPivot(0, (array2.n - 1));
        fim2 = array1.now();
        inicio3 = array1.now();
        array3.QuicksortRandomPivot(0, (array3.n - 1));
        fim3 = array1.now();
        inicio4 = array1.now();
        array4.QuicksortMedianOfThreePivot(0, (array4.n - 1));
        fim4 = array1.now();

        System.out.println("Array organizado: Organizado _ 10000 elementos");
        System.out.println("Tempos para ordenar: ");
        System.out.printf("FirstPivot: %.4fs.\n", (fim1 - inicio1) / 1000);
        System.out.printf("LastPivot: %.4fs.\n", (fim2 - inicio2) / 1000);
        System.out.printf("RandomPivot: %.4fs.\n", (fim3 - inicio3) / 1000);
        System.out.printf("MedianOfThreePivot: %.4fs.\n", (fim4 - inicio4) /
1000);
        System.out.println("");
        System.out.println("-------------------------");

        /* */

        /* Teste com array de 17500 elementos */

        array1 = new Quicksort(17500);
        array1.crescente();
        array2 = new Quicksort(array1);
        array3 = new Quicksort(array1);
        array4 = new Quicksort(array1);

        System.out.println("");
        System.out.println("-------------------------");
        // array1.mostrar();
        System.out.println("");
```

```
        System.out.println("");

        inicio1 = array1.now();
        array1.QuicksortFirstPivot(0, (array1.n - 1));
        fim1 = array1.now();
        inicio2 = array1.now();
        array2.QuicksortLastPivot(0, (array2.n - 1));
        fim2 = array1.now();
        inicio3 = array1.now();
        array3.QuicksortRandomPivot(0, (array3.n - 1));
        fim3 = array1.now();
        inicio4 = array1.now();
        array4.QuicksortMedianOfThreePivot(0, (array4.n - 1));
        fim4 = array1.now();

        System.out.println("Array organizado: Organizado _ 17500 elementos");
        System.out.println("Tempos para ordenar: ");
        System.out.printf("FirstPivot: %.4fs.\n", (fim1 - inicio1) / 1000);
        System.out.printf("LastPivot: %.4fs.\n", (fim2 - inicio2) / 1000);
        System.out.printf("RandomPivot: %.4fs.\n", (fim3 - inicio3) / 1000);
        System.out.printf("MedianOfThreePivot: %.4fs.\n", (fim4 - inicio4) /
1000);
        System.out.println("");
        System.out.println("-------------------------");

        /* */

        /* Array Organizado - FIM */

        /* Array Semi-organizado - INÍCIO */
        /* Teste com array de 100 elementos */

        array1 = new Quicksort(100);
        array1.semialeatorio();
        array2 = new Quicksort(array1);
        array3 = new Quicksort(array1);
        array4 = new Quicksort(array1);

        System.out.println("");
        System.out.println("-------------------------");
        System.out.println("");
        System.out.println("");

        inicio1 = array1.now();
        array1.QuicksortFirstPivot(0, (array1.n - 1));
        fim1 = array1.now();
```

```
        inicio2 = array1.now();
        array2.QuicksortLastPivot(0, (array2.n - 1));
        fim2 = array1.now();
        inicio3 = array1.now();
        array3.QuicksortRandomPivot(0, (array3.n - 1));
        fim3 = array1.now();
        inicio4 = array1.now();
        array4.QuicksortMedianOfThreePivot(0, (array4.n - 1));
        fim4 = array1.now();

        System.out.println("Array organizado: Semi-organizado _ 100
elementos");
        System.out.println("Tempos para ordenar: ");
        System.out.printf("FirstPivot: %.4fs.\n", (fim1 - inicio1) / 1000);
        System.out.printf("LastPivot: %.4fs.\n", (fim2 - inicio2) / 1000);
        System.out.printf("RandomPivot: %.4fs.\n", (fim3 - inicio3) / 1000);
        System.out.printf("MedianOfThreePivot: %.4fs.\n", (fim4 - inicio4) /
1000);
        System.out.println("");
        System.out.println("-------------------------");

        /* */

        /* Teste com array de 1000 elementos */

        array1 = new Quicksort(1000);
        array1.semialeatorio();
        array2 = new Quicksort(array1);
        array3 = new Quicksort(array1);
        array4 = new Quicksort(array1);

        System.out.println("");
        System.out.println("-------------------------");
        System.out.println("");
        System.out.println("");

        inicio1 = array1.now();
        array1.QuicksortFirstPivot(0, (array1.n - 1));
        fim1 = array1.now();
        inicio2 = array1.now();
        array2.QuicksortLastPivot(0, (array2.n - 1));
        fim2 = array1.now();
        inicio3 = array1.now();
        array3.QuicksortRandomPivot(0, (array3.n - 1));
        fim3 = array1.now();
        inicio4 = array1.now();
```

```
        array4.QuicksortMedianOfThreePivot(0, (array4.n - 1));
        fim4 = array1.now();

        System.out.println("Array organizado: Semi-organizado _ 1000
elementos");
        System.out.println("Tempos para ordenar: ");
        System.out.printf("FirstPivot: %.4fs.\n", (fim1 - inicio1) / 1000);
        System.out.printf("LastPivot: %.4fs.\n", (fim2 - inicio2) / 1000);
        System.out.printf("RandomPivot: %.4fs.\n", (fim3 - inicio3) / 1000);
        System.out.printf("MedianOfThreePivot: %.4fs.\n", (fim4 - inicio4) /
1000);
        System.out.println("");
        System.out.println("-------------------------");

        /* */

        /* Teste com array de 10000 elementos */

        array1 = new Quicksort(10000);
        array1.semialeatorio();
        array2 = new Quicksort(array1);
        array3 = new Quicksort(array1);
        array4 = new Quicksort(array1);

        System.out.println("");
        System.out.println("-------------------------");
        System.out.println("");
        System.out.println("");

        inicio1 = array1.now();
        array1.QuicksortFirstPivot(0, (array1.n - 1));
        fim1 = array1.now();
        inicio2 = array1.now();
        array2.QuicksortLastPivot(0, (array2.n - 1));
        fim2 = array1.now();
        inicio3 = array1.now();
        array3.QuicksortRandomPivot(0, (array3.n - 1));
        fim3 = array1.now();
        inicio4 = array1.now();
        array4.QuicksortMedianOfThreePivot(0, (array4.n - 1));
        fim4 = array1.now();

        System.out.println("Array organizado: Semi-organizado _ 10000
elementos");
        System.out.println("Tempos para ordenar: ");
        System.out.printf("FirstPivot: %.4fs.\n", (fim1 - inicio1) / 1000);
```

```
        System.out.printf("LastPivot: %.4fs.\n", (fim2 - inicio2) / 1000);
        System.out.printf("RandomPivot: %.4fs.\n", (fim3 - inicio3) / 1000);
        System.out.printf("MedianOfThreePivot: %.4fs.\n", (fim4 - inicio4) /
1000);
        System.out.println("");
        System.out.println("-------------------------");

        /* */

        /* Teste com array de 17500 elementos */

        array1 = new Quicksort(17500);
        array1.semialeatorio();
        array2 = new Quicksort(array1);
        array3 = new Quicksort(array1);
        array4 = new Quicksort(array1);

        System.out.println("");
        System.out.println("-------------------------");
        System.out.println("");
        System.out.println("");

        inicio1 = array1.now();
        array1.QuicksortFirstPivot(0, (array1.n - 1));
        fim1 = array1.now();
        inicio2 = array1.now();
        array2.QuicksortLastPivot(0, (array2.n - 1));
        fim2 = array1.now();
        inicio3 = array1.now();
        array3.QuicksortRandomPivot(0, (array3.n - 1));
        fim3 = array1.now();
        inicio4 = array1.now();
        array4.QuicksortMedianOfThreePivot(0, (array4.n - 1));
        fim4 = array1.now();

        System.out.println("Array organizado: Semi-organizado _ 17500
elementos");
        System.out.println("Tempos para ordenar: ");
        System.out.printf("FirstPivot: %.4fs.\n", (fim1 - inicio1) / 1000);
        System.out.printf("LastPivot: %.4fs.\n", (fim2 - inicio2) / 1000);
        System.out.printf("RandomPivot: %.4fs.\n", (fim3 - inicio3) / 1000);
        System.out.printf("MedianOfThreePivot: %.4fs.\n", (fim4 - inicio4) /
1000);
        System.out.println("");
        System.out.println("-------------------------");
```

```
        /* */

        /* Array Semi-organizado - FIM */

    }

}
```

Exemplo de resultado:

```
------------------------

Array organizado: Aleatório _ 100 elementos
Tempos para ordenar:
FirstPivot: 0,0000s.
LastPivot: 0,0000s.
RandomPivot: 0,0000s.
MedianOfThreePivot: 0,0000s.

-------------------------

------------------------

Array organizado: Aleatório _ 1000 elementos
Tempos para ordenar:
FirstPivot: 0,0000s.
LastPivot: 0,0000s.
RandomPivot: 0,0000s.
MedianOfThreePivot: 0,0000s.

-------------------------

-------------------------

Array organizado: Aleatório _ 10000 elementos
Tempos para ordenar:
FirstPivot: 0,0070s.
LastPivot: 0,0000s.
RandomPivot: 0,0020s.
MedianOfThreePivot: 0,0000s.
```

Array organizado: Aleatório _ 17500 elementos
Tempos para ordenar:
FirstPivot: 0,0050s.
LastPivot: 0,0010s.
RandomPivot: 0,0050s.
MedianOfThreePivot: 0,0060s.

Array organizado: Organizado _ 100 elementos
Tempos para ordenar:
FirstPivot: 0,0000s.
LastPivot: 0,0000s.
RandomPivot: 0,0000s.
MedianOfThreePivot: 0,0000s.

Array organizado: Organizado _ 1000 elementos
Tempos para ordenar:
FirstPivot: 0,0040s.
LastPivot: 0,0010s.
RandomPivot: 0,0000s.
MedianOfThreePivot: 0,0000s.

Array organizado: Organizado _ 10000 elementos
Tempos para ordenar:
FirstPivot: 0,0350s.
LastPivot: 0,0510s.
RandomPivot: 0,0000s.
MedianOfThreePivot: 0,0000s.

Array organizado: Organizado _ 17500 elementos
Tempos para ordenar:
FirstPivot: 0,0850s.
LastPivot: 0,1250s.

RandomPivot: 0,0050s.
MedianOfThreePivot: 0,0000s.

Array organizado: Semi-organizado _ 100 elementos
Tempos para ordenar:
FirstPivot: 0,0000s.
LastPivot: 0,0000s.
RandomPivot: 0,0000s.
MedianOfThreePivot: 0,0000s.

Array organizado: Semi-organizado _ 1000 elementos
Tempos para ordenar:
FirstPivot: 0,0000s.
LastPivot: 0,0000s.
RandomPivot: 0,0000s.
MedianOfThreePivot: 0,0000s.

Array organizado: Semi-organizado _ 10000 elementos
Tempos para ordenar:
FirstPivot: 0,0220s.
LastPivot: 0,0500s.
RandomPivot: 0,0000s.
MedianOfThreePivot: 0,0000s.

Array organizado: Semi-organizado _ 17500 elementos
Tempos para ordenar:
FirstPivot: 0,0880s.
LastPivot: 0,1300s.
RandomPivot: 0,0000s.
MedianOfThreePivot: 0,0000s.

Tabela de resultados dos testes (em milissegundos):

FirstPivot:

_Execução 1

	100	1000	10000	17500
Aleatória	0	0	7	5
Organizado	0	4	35	85
Semi-organizado	0	0	22	88

_Execução 2

	100	1000	10000	17500
Aleatória	0	1	1	1
Organizado	0	0	27	76
Semi-organizado	0	1	25	69

_Execução 3

	100	1000	10000	17500
Aleatória	0	0	1	2
Organizado	0	1	28	71
Semi-organizado	0	0	25	70

_Execução 4

	100	1000	10000	17500
Aleatória	0	0	1	1
Organizado	0	0	24	80
Semi-organizado	0	1	24	68

_Execução 5

	100	1000	10000	17500
Aleatória	0	0	1	1
Organizado	0	0	25	69
Semi-organizado	0	0	22	66

LastPivot:

_Execução 1

	100	1000	10000	17500
Aleatória	0	0	0	1
Organizado	0	1	51	125
Semi-organizado	0	0	50	130

_Execução 2

	100	1000	10000	17500
Aleatória	0	0	1	2
Organizado	1	2	41	116
Semi-organizado	0	0	39	113

_Execução 3

	100	1000	10000	17500
Aleatória	0	1	2	2
Organizado	0	0	39	112
Semi-organizado	0	0	37	117

_Execução 4

	100	1000	10000	17500
Aleatória	0	1	0	2
Organizado	0	1	36	115
Semi-organizado	0	0	36	118

_Execução 5

	100	1000	10000	17500
Aleatória	0	1	1	2
Organizado	0	1	37	110
Semi-organizado	0	0	35	108

RandomPivot:

_Execução 1

	100	1000	10000	17500
Aleatória	0	0	2	5
Organizado	0	0	0	5
Semi-organizado	0	0	0	0

_Execução 2

	100	1000	10000	17500
Aleatória	0	1	2	3
Organizado	0	0	1	1
Semi-organizado	0	0	0	1

_Execução 3

	100	1000	10000	17500
Aleatória	0	0	1	4
Organizado	0	0	0	0
Semi-organizado	0	0	0	1

_Execução 4

	100	1000	10000	17500
Aleatória	0	0	2	3
Organizado	0	0	0	1
Semi-organizado	0	0	1	1

_Execução 5

	100	1000	10000	17500
Aleatória	1	0	3	3
Organizado	0	0	1	0
Semi-organizado	0	0	0	1

MedianOfThreePivot:

_Execução 1

	100	1000	10000	17500
Aleatória	0	0	1	4
Organizado	0	0	0	0
Semi-organizado	0	0	0	0

_Execução 2

	100	1000	10000	17500
Aleatória	0	0	0	0
Organizado	0	0	0	0
Semi-organizado	0	0	0	0

_Execução 3

	100	1000	10000	17500
Aleatória	0	1	2	2
Organizado	0	0	0	1
Semi-organizado	0	0	1	0

_Execução 4

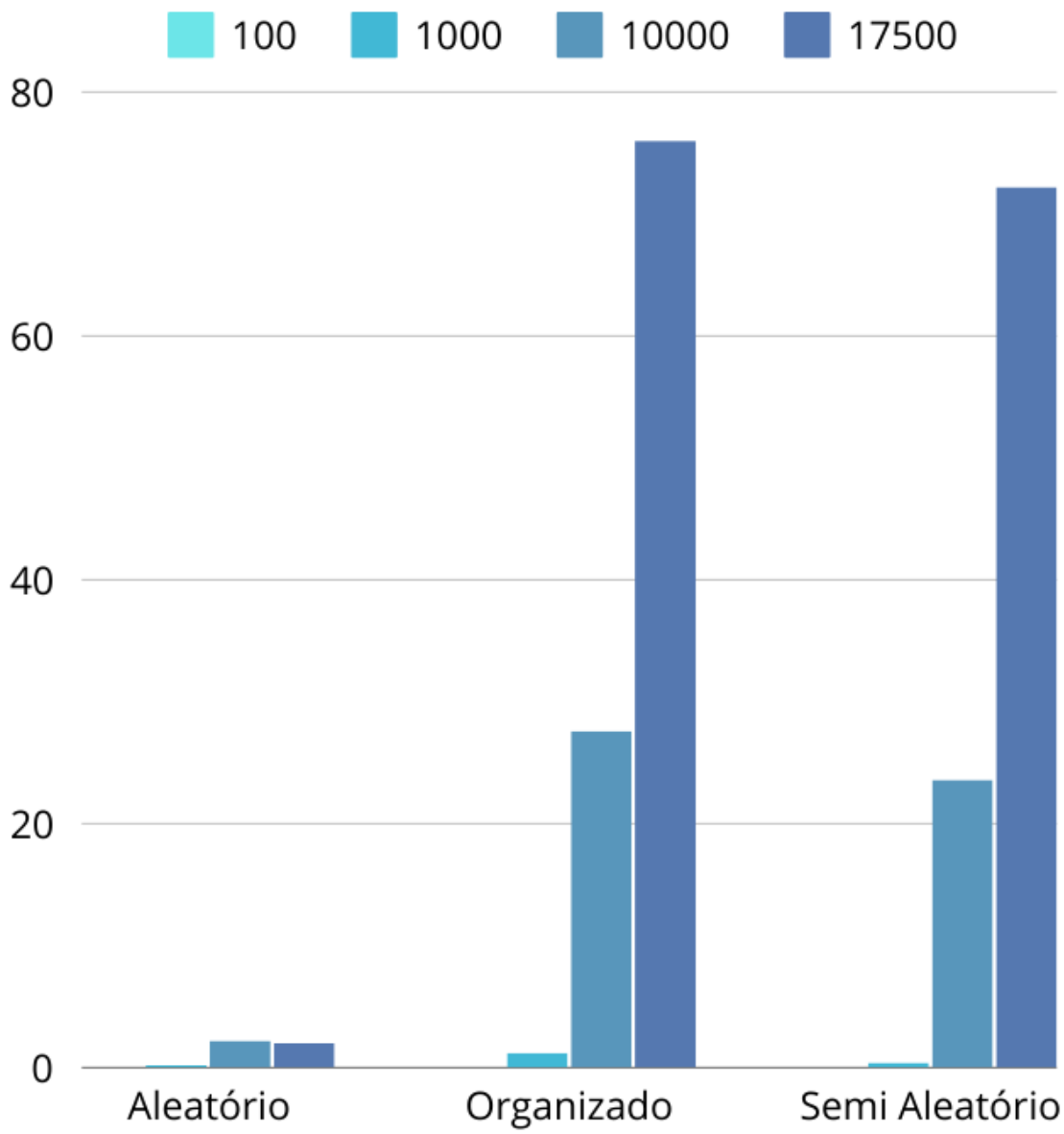
	100	1000	10000	17500
Aleatória	0	1	1	2
Organizado	0	0	1	0
Semi-organizado	0	0	0	0

_Execução 5

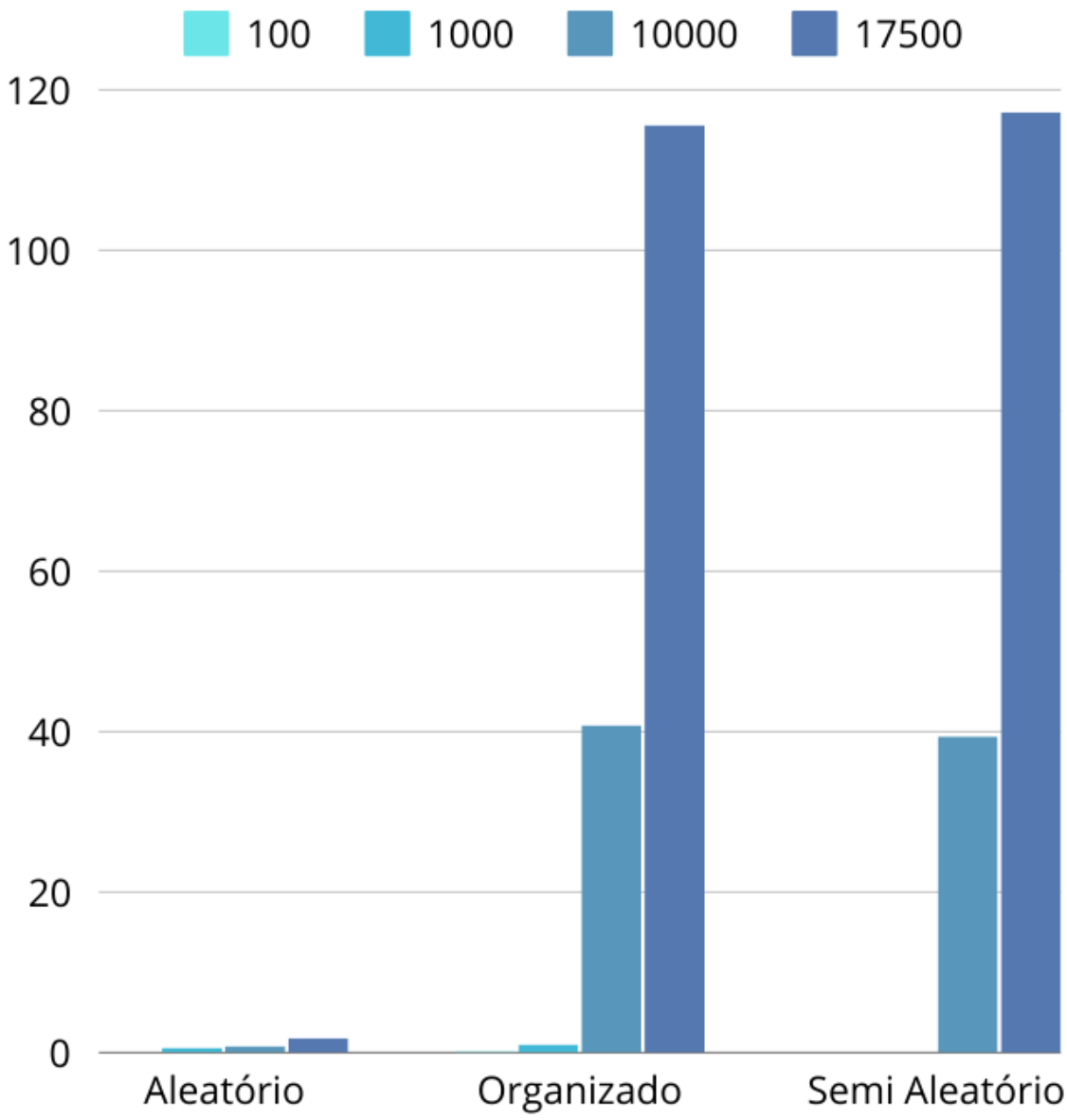
	100	1000	10000	17500
Aleatória	0	1	1	2
Organizado	0	0	0	0
Semi-organizado	0	0	2	0

Sumário em gráficos (medido em milissegundos, por cada tamanho do conjunto). Dados representam média dos resultados demonstrados anteriormente:

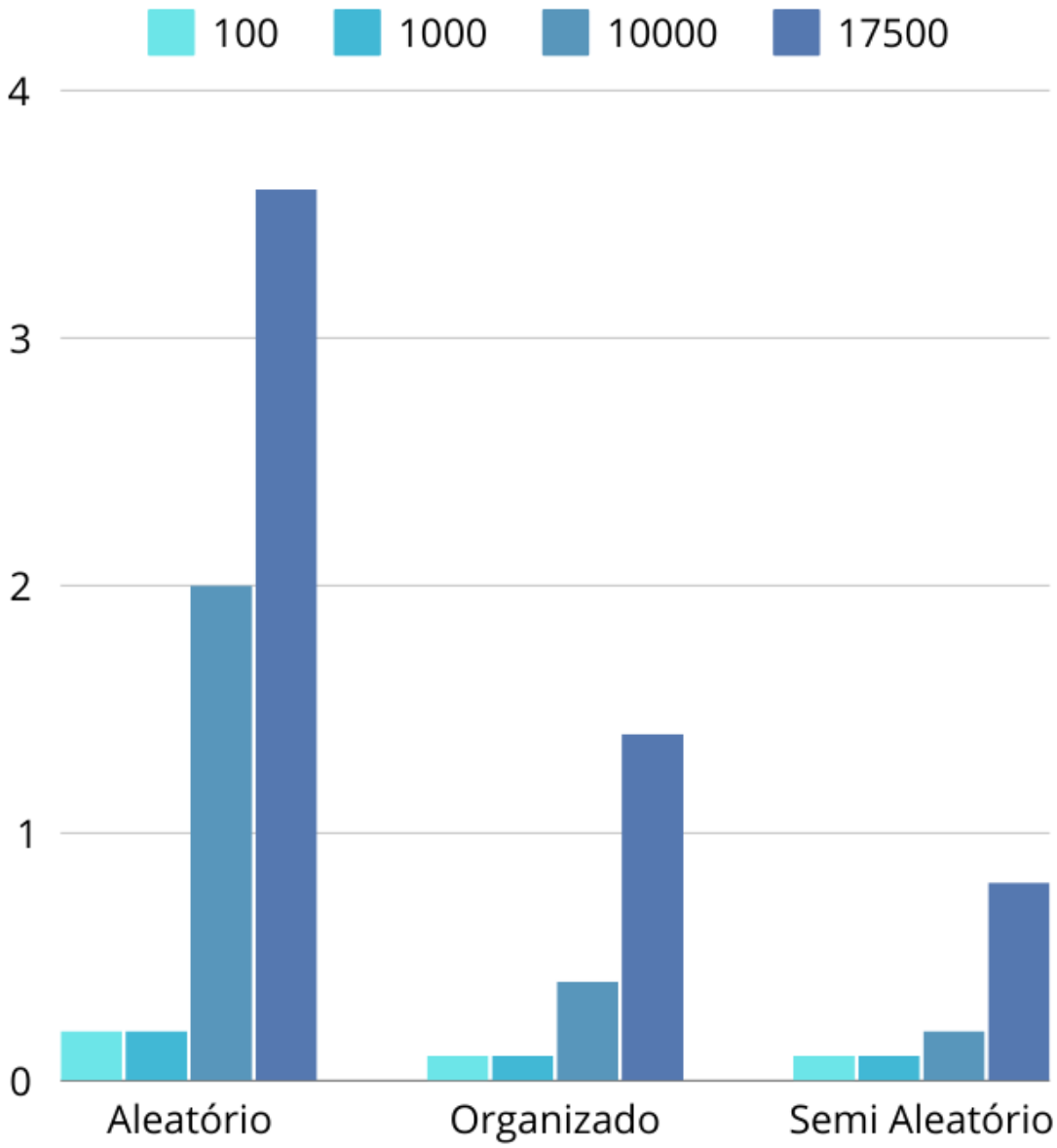
FirstPivot: Esse método utiliza do primeiro elemento do conjunto/subconjunto como sendo o pivô para a organização



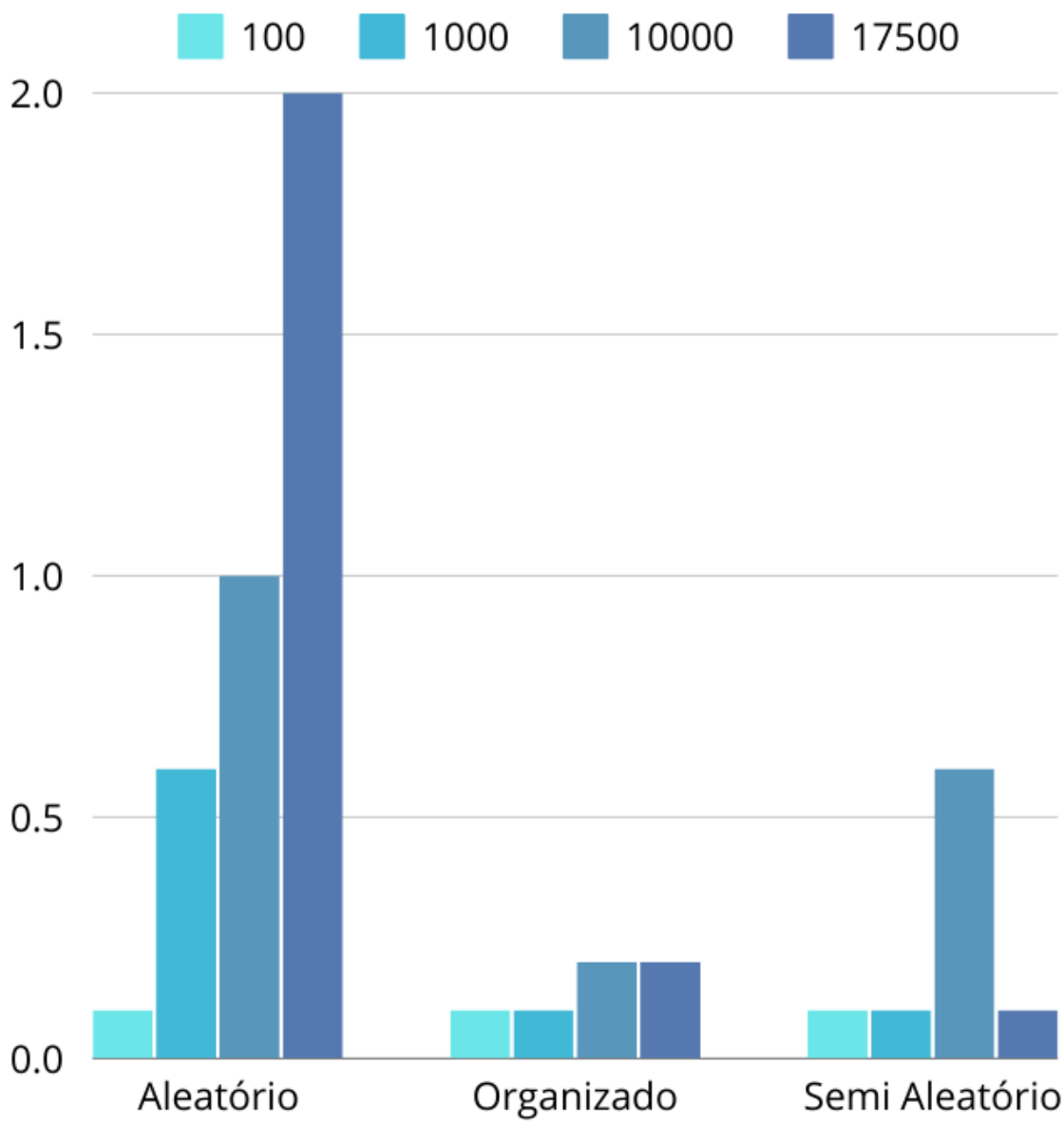
LastPivot: Esse método utiliza do último elemento do conjunto/subconjunto como sendo o pivô para a organização



RandomPivot: Esse método utiliza de um elemento aleatório do conjunto/subconjunto como sendo o pivô para a organização



MedianOfThreePivot: Esse método utiliza do valor mediano entre o valor do início, o valor do meio, e o valor do fim do conjunto/subconjunto como sendo o pivô para a organização



Análise para cada caso:

Conjunto aleatório:

Notavelmente nesse caso, a diferença da execução de cada método de seleção de pivô é mínima, com o método *LastPivot* tendo tido o melhor resultado, com os métodos *FirstPivot* e *MedianOfThreePivot* com resultados extremamente próximos, porém, isso pode ser atribuído puramente a uma questão de "sorte", já que a organização aleatória evita o resultado visto nos demais casos do método em que o pior caso ocorre consistentemente (o que igualmente se aplica ao *FirstPivot*). Atribuindo a lógica dos métodos, o que demonstraria o resultado mais consistente seria *MedianOfThreePivot* já que, pela sua natureza, ele evita a ocorrência do pior caso, apesar que o custo de achar a o elemento mediano entre o primeiro, o último e o do meio deve ser considerado.

Conjunto organizado:

Nesse caso, os que demonstraram melhor desempenho foram *RandomPivot* e *MedianOfThreePivot*, com esse último tendo o melhor caso possível. Para o *RandomPivot*, sua escolha aleatória do pivô torna improvável a seleção dos piores casos (como ocorre com o *FirstPivot* e *LastPivot*, que em um conjunto organizado sempre farão a pior escolha), o tornando eficiente, já o *MedianOfThreePivot* sempre selecionará o elemento do meio do conjunto/subconjunto, que será o elemento mediano (já que está organizado), o que garante a ocorrência do melhor caso, com o menor número possível de trocas.

Conjunto semi-organizado:

Para esse caso, a lógica é quase idêntica à do conjunto organizado, com o *MedianOfThreePivot* sendo o método de maior eficácia pela maior probabilidade de encontrar o elemento mediano do conjunto/subconjunto.