CÓDIGO:

```
import java.util.*;

public class Quicksort {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        double inicio,fim;

        int array100Ordenados[] = new int[100];
        crescente(array100Ordenados);
        inicio = now();
        quicksortPrimeiro(array100Ordenados,0, 100-1);
        fim = now();
        System.out.println("Tempo para ordenar: " + (fim-inicio)/1000.0 + "
s.");

        crescente(array100Ordenados);
        inicio = now();
        quicksortUltimo(array100Ordenados,0, 100-1);
        fim = now();
        System.out.println("Tempo para ordenar: " + (fim-inicio)/1000.0 + "
s.");

        crescente(array100Ordenados);
        inicio = now();
        quicksortRandom(array100Ordenados,0, 100-1);
        fim = now();
        System.out.println("Tempo para ordenar: " + (fim-inicio)/1000.0 + "
s.");

        crescente(array100Ordenados);
        inicio = now();
        quicksortMediana(array100Ordenados,0, 100-1);
        fim = now();
        System.out.println("Tempo para ordenar: " + (fim-inicio)/1000.0 + "
s.");

//------------------------------------------------------------------------------
-------------------------

        int array1000Ordenados[] = new int[1000];
        crescente(array1000Ordenados);
        inicio = now();
        quicksortPrimeiro(array1000Ordenados, 0, 1000-1);
        fim = now();
```

```
        System.out.println("Tempo para ordenar: " + (fim-inicio)/1000.0 + "
s.");

        crescente(array1000Ordenados);
        inicio = now();
        quicksortUltimo(array1000Ordenados,0, 1000-1);
        fim = now();
        System.out.println("Tempo para ordenar: " + (fim-inicio)/1000.0 + "
s.");

        crescente(array1000Ordenados);
        inicio = now();
        quicksortRandom(array1000Ordenados,0, 1000-1);
        fim = now();
        System.out.println("Tempo para ordenar: " + (fim-inicio)/1000.0 + "
s.");

        crescente(array1000Ordenados);
        inicio = now();
        quicksortMediana(array1000Ordenados,0, 1000-1);
        fim = now();
        System.out.println("Tempo para ordenar: " + (fim-inicio)/1000.0 + "
s.");

//-------------------------------------------------------------------------
-------------------------

        int array10000Ordenados[] = new int[10000];
        crescente(array10000Ordenados);
        inicio = now();
        quicksortPrimeiro(array10000Ordenados,0, 10000-1);
        fim = now();
        System.out.println("Tempo para ordenar: " + (fim-inicio)/1000.0 + "
s.");

        crescente(array10000Ordenados);
        inicio = now();
        quicksortUltimo(array10000Ordenados,0, 10000-1);
        fim = now();
        System.out.println("Tempo para ordenar: " + (fim-inicio)/1000.0 + "
s.");

        crescente(array10000Ordenados);
        inicio = now();
        quicksortRandom(array10000Ordenados,0, 10000-1);
        fim = now();
        System.out.println("Tempo para ordenar: " + (fim-inicio)/1000.0 + "
s.");
```

```
        crescente(array10000Ordenados);
        inicio = now();
        quicksortMediana(array10000Ordenados,0, 10000-1);
        fim = now();
        System.out.println("Tempo para ordenar: " + (fim-inicio)/1000.0 + "
s.");

//--------------------------------------------------------------------------
-------------------------

        int array100QuaseOrdenados[] = new int[100];
        crescente(array100QuaseOrdenados);
        mudanca(array100QuaseOrdenados);
        inicio = now();
        quicksortPrimeiro(array100QuaseOrdenados, 0, 100-1);
        fim = now();
        System.out.println("Tempo para ordenar: " + (fim-inicio)/1000.0 + "
s.");

        crescente(array100QuaseOrdenados);
        mudanca(array100QuaseOrdenados);
        inicio = now();
        quicksortUltimo(array100QuaseOrdenados, 0, 100-1);
        fim = now();
        System.out.println("Tempo para ordenar: " + (fim-inicio)/1000.0 + "
s.");

        crescente(array100QuaseOrdenados);
        mudanca(array100QuaseOrdenados);
        inicio = now();
        quicksortRandom(array100QuaseOrdenados, 0, 100-1);
        fim = now();
        System.out.println("Tempo para ordenar: " + (fim-inicio)/1000.0 + "
s.");

        crescente(array100QuaseOrdenados);
        mudanca(array100QuaseOrdenados);
        inicio = now();
        quicksortMediana(array100QuaseOrdenados, 0, 100-1);
        fim = now();
        System.out.println("Tempo para ordenar: " + (fim-inicio)/1000.0 + "
s.");

//--------------------------------------------------------------------------
-------------------------

int array1000QuaseOrdenados[] = new int[1000];
```

```
crescente(array1000QuaseOrdenados);
mudanca(array1000QuaseOrdenados);
inicio = now();
quicksortPrimeiro(array1000QuaseOrdenados, 0, 1000-1);
fim = now();
System.out.println("Tempo para ordenar: " + (fim-inicio)/1000.0 + " s.");

crescente(array1000QuaseOrdenados);
mudanca(array1000QuaseOrdenados);
inicio = now();
quicksortUltimo(array1000QuaseOrdenados, 0, 1000-1);
fim = now();
System.out.println("Tempo para ordenar: " + (fim-inicio)/1000.0 + " s.");

crescente(array1000QuaseOrdenados);
mudanca(array1000QuaseOrdenados);
inicio = now();
quicksortRandom(array1000QuaseOrdenados, 0, 1000-1);
fim = now();
System.out.println("Tempo para ordenar: " + (fim-inicio)/1000.0 + " s.");

crescente(array1000QuaseOrdenados);
mudanca(array1000QuaseOrdenados);
inicio = now();
quicksortMediana(array1000QuaseOrdenados, 0, 1000-1);
fim = now();
System.out.println("Tempo para ordenar: " + (fim-inicio)/1000.0 + " s.");

//-------------------------------------------------------------------------
-------------------------

int array10000QuaseOrdenados[] = new int[10000];
        crescente(array10000QuaseOrdenados);
        mudanca(array10000QuaseOrdenados);
        inicio = now();
        quicksortPrimeiro(array10000QuaseOrdenados, 0, 10000-1);
        fim = now();
        System.out.println("Tempo para ordenar: " + (fim-inicio)/1000.0 + "
s.");

        crescente(array10000QuaseOrdenados);
        mudanca(array10000QuaseOrdenados);
        inicio = now();
        quicksortUltimo(array10000QuaseOrdenados, 0, 10000-1);
        fim = now();
        System.out.println("Tempo para ordenar: " + (fim-inicio)/1000.0 + "
s.");
```

```
        crescente(array10000QuaseOrdenados);
        mudanca(array10000QuaseOrdenados);
        inicio = now();
        quicksortRandom(array10000QuaseOrdenados, 0, 10000-1);
        fim = now();
        System.out.println("Tempo para ordenar: " + (fim-inicio)/1000.0 + "
s.");

        crescente(array10000QuaseOrdenados);
        mudanca(array10000QuaseOrdenados);
        inicio = now();
        quicksortMediana(array10000QuaseOrdenados, 0, 10000-1);
        fim = now();
        System.out.println("Tempo para ordenar: " + (fim-inicio)/1000.0 + "
s.");

//-------------------------------------------------------------------------
-------------------------

        int array100NaoOrdenados[] = new int[100];
        aleatorio(array100NaoOrdenados);
        inicio = now();
        quicksortPrimeiro(array100NaoOrdenados, 0, 100-1);
        fim = now();
        System.out.println("Tempo para ordenar: " + (fim-inicio)/1000.0 + "
s.");

        aleatorio(array100NaoOrdenados);
        inicio = now();
        quicksortUltimo(array100NaoOrdenados, 0, 100-1);
        fim = now();
        System.out.println("Tempo para ordenar: " + (fim-inicio)/1000.0 + "
s.");

        aleatorio(array100NaoOrdenados);
        inicio = now();
        quicksortRandom(array100NaoOrdenados, 0, 100-1);
        fim = now();
        System.out.println("Tempo para ordenar: " + (fim-inicio)/1000.0 + "
s.");

        aleatorio(array100NaoOrdenados);
        inicio = now();
        quicksortMediana(array100NaoOrdenados, 0, 100-1);
        fim = now();
        System.out.println("Tempo para ordenar: " + (fim-inicio)/1000.0 + "
s.");
```

```
//--------------------------------------------------------------------------
-------------------------

        int array1000NaoOrdenados[] = new int[1000];
        aleatorio(array1000NaoOrdenados);
        inicio = now();
        quicksortPrimeiro(array1000NaoOrdenados, 0, 1000-1);
        fim = now();
        System.out.println("Tempo para ordenar: " + (fim-inicio)/1000.0 + "
s.");

        aleatorio(array1000NaoOrdenados);
        inicio = now();
        quicksortUltimo(array1000NaoOrdenados, 0, 1000-1);
        fim = now();
        System.out.println("Tempo para ordenar: " + (fim-inicio)/1000.0 + "
s.");

        aleatorio(array1000NaoOrdenados);
        inicio = now();
        quicksortRandom(array1000NaoOrdenados, 0, 1000-1);
        fim = now();
        System.out.println("Tempo para ordenar: " + (fim-inicio)/1000.0 + "
s.");

        aleatorio(array1000NaoOrdenados);
        inicio = now();
        quicksortMediana(array1000NaoOrdenados, 0, 1000-1);
        fim = now();
        System.out.println("Tempo para ordenar: " + (fim-inicio)/1000.0 + "
s.");

//--------------------------------------------------------------------------
-------------------------

        int array10000NaoOrdenados[] = new int[10000];
        aleatorio(array10000NaoOrdenados);
        inicio = now();
        quicksortPrimeiro(array10000NaoOrdenados, 0, 10000-1);
        fim = now();
        System.out.println("Tempo para ordenar: " + (fim-inicio)/1000.0 + "
s.");

        aleatorio(array10000NaoOrdenados);
        inicio = now();
        quicksortUltimo(array10000NaoOrdenados, 0, 10000-1);
        fim = now();
```

```
        System.out.println("Tempo para ordenar: " + (fim-inicio)/1000.0 + "
s.");

        aleatorio(array10000NaoOrdenados);
        inicio = now();
        quicksortRandom(array10000NaoOrdenados, 0, 10000-1);
        fim = now();
        System.out.println("Tempo para ordenar: " + (fim-inicio)/1000.0 + "
s.");

        aleatorio(array10000NaoOrdenados);
        inicio = now();
        quicksortMediana(array10000NaoOrdenados, 0, 10000-1);
        fim = now();
        System.out.println("Tempo para ordenar: " + (fim-inicio)/1000.0 + "
s.");



        scanner.close();
    }

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

    public static void crescente(int array[]) {
        for (int i = 0; i < array.length; i++) {
            array[i] = i;
        }
    }

    public static void mudanca(int array[]) {
        int n = array.length/4;

        for(int i=0;i<n-1;i++) {
            for(int j=1;j<n;j++) {
                swap(array,i,j);
            }
        }
    }

    public static void aleatorio(int array[]) {
        Random rand = new Random();
        crescente(array);
        for (int i = 0; i < array.length; i++) {
            swap(array,i, Math.abs(rand.nextInt()) % array.length);
        }
    }
```

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



public static void quicksortPrimeiro(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[esq];

    while (i <= j) {
        while (array[i] < pivo)
            i++;
        while (array[j] > pivo)
            j--;
        if (i <= j) {
            int temp = array[i];
            array[i] = array[j];
            array[j] = temp;
            i++;
            j--;
        }
    }
    if (esq < j)
        quicksortPrimeiro(array, esq, j);
    if (i < dir)
        quicksortPrimeiro(array, i, dir);
}



public static void quicksortUltimo(int[] array, int esq, int dir) {
    int i = esq, j = dir, pivo = array[dir];

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

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

    public static void quicksortRandom(int[] array, int esq, int dir) {
        Random rand = new Random();
        int pivotIndex = esq + rand.nextInt(dir - esq + 1);
        int pivo = array[pivotIndex];


        int i = esq, j = dir;

        while (i <= j) {
            while (array[i] < pivo)
                i++;
            while (array[j] > pivo)
                j--;
            if (i <= j) {
                int temp = array[i];
                array[i] = array[j];
                array[j] = temp;
                i++;
                j--;
            }
        }
        if (esq < j)
            quicksortRandom
    (array, esq, j);
        if (i < dir)
            quicksortRandom
    (array, i, dir);
    }


    public static void quicksortMediana(int array[], int left, int right) {
        int i = left, j = right;
        int pivo = array[calculaMediana(array, i , j)];
        while (i <= j) {
            while (array[i] < pivo) i++;
            while (array[j] > pivo) j--;
            if (i <= j) {
                swap(array, i, j);
                i++;
                j--;
            }
        }
```

```
        if (left < j)  quicksortMediana(array, left, j);
        if (i < right)  quicksortMediana(array, i, right);
    }

    public static int calculaMediana(int[] array, int i, int j){
        int mediana;
        int meio = (i + j) / 2;
        if((array[i] >= array[j] && array[j] >= array[meio]) || (array[meio] >=
array[j] && array[j] >= array[i])){
            mediana = j;
        }
        else if((array[j] >= array[i] && array[i] >= array[meio]) ||
(array[meio] >= array[i] && array[i] >= array[j])){
            mediana = i;
        }
        else mediana = meio;
        return mediana;
    }
}
```

FUNCIONAMENTO DE CADA PIVOT

PRIMEIRO ELEMENTO:
Técnica simples de escolha de pivô, onde o primeiro elemento é escolhido como pivô.

ÚLTIMO ELEMENTO:
Técnica simples de escolha de pivô, onde o último elemento é escolhido como pivô.

ALEATÓRIO:
Escolhe de forma aleatória o pivô que será usado.

MEDIANA:
O pivô é escolhido com o cálculo de 3 elementos dividido por 3 (início, meio e fim).

TEMPOS DE EXECUÇÃO PARA QUICKSORT COM PIVOTS DIFERENTES

PIVOT NO PRIMEIRO ELEMENTO

Ordenados

array[100] = 0.0 seg;
array[1000] = 0.001 seg;
array[10000] = 0.033 seg;

Quase Ordenados

array[100] = 0.0 seg;
array[1000] = 0.001 seg;
array[10000] = 0.019 seg;

Não Ordenados

array[100] = 0.0 seg;
array[1000] = 0.0 seg;
array[10000] = 0.001 seg;

PIVOT NO ÚLTIMO ELEMENTO

Ordenados

array[100] = 0.0 seg;
array[1000] = 0.003 seg;
array[10000] = 0.029 seg;

Quase Ordenados

array[100] = 0.0 seg;
array[1000] = 0.002 seg;
array[10000] = 0.031 seg;

Não Ordenados

array[100] = 0.0 seg;
array[1000] = 0.0 seg;
array[10000] = 0.002 seg;

PIVOT ALEATÓRIO

Ordenados

array[100] = 0.001 seg;
array[1000] = 0.001 seg;
array[10000] = 0.001 seg;

Quase Ordenados

array[100] = 0.0 seg;
array[1000] = 0.0 seg;
array[10000] = 0.002 seg;

Não Ordenados

array[100] = 0.0 seg;

array[1000] = 0.001 seg;
array[10000] = 0.001 seg;

PIVOT MEDIANA

Ordenados

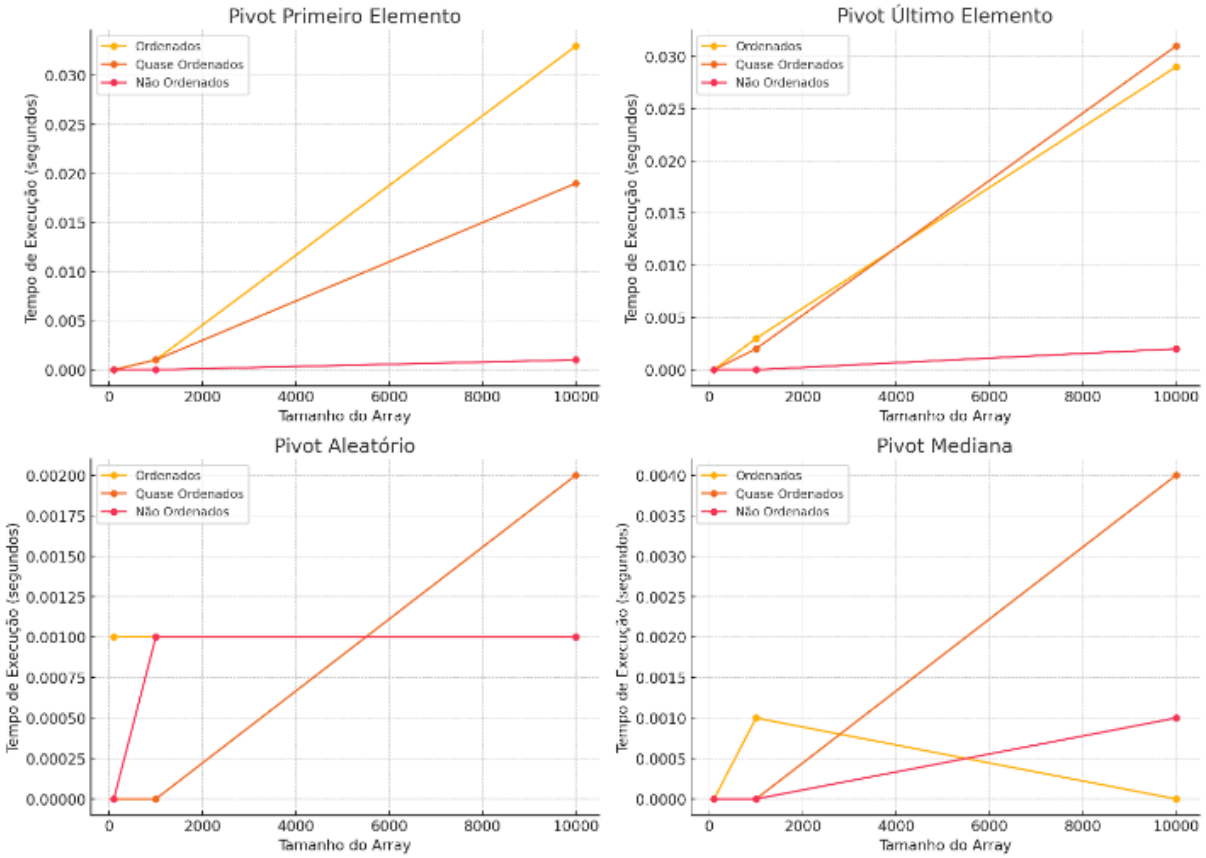
array[100] = 0.0 seg;
array[1000] = 0.001 seg;
array[10000] = 0.0 seg;

Quase Ordenados

array[100] = 0.0 seg;
array[1000] = 0.0 seg;
array[10000] = 0.004 seg;

Não Ordenados

array[100] = 0.0 seg;
array[1000] = 0.0 seg;
array[10000] = 0.001 seg;



ARRAYS ORDENADOS: O melhor pivô para arrays ordenados é a mediana ou, alternativamente, o pivô aleatório. Ambos evitam o pior caso e proporcionam melhor balanceamento.

ARRAYS QUASE ORDENADOS: Novamente, o pivô mediana é a escolha ideal, seguido pelo pivô aleatório. Eles garantem um bom balanceamento e previnem o pior caso.

ARRAYS NÃO ORDENADOS: O pivô aleatório é uma das melhores escolhas para arrays não ordenados, já que mantém a eficiência sem a

necessidade de calcular a mediana. O pivô no primeiro ou último elemento também é razoável, mas o aleatório é mais flexível.