Pontifícia Universidade Católica de Minas Gerais
Curso de Ciência da Computação
Disciplina: Algoritmos e Estruturas de Dados II

Matheus Henrique Gonçalves

Laboratório 06 - Quicksort e seu pivô

Código:

Quicksort First Pivot:

```java
public static void QuickSortFirstPivot (int[] array ,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(array, i, j);
            i++;
            j--;
        }
    }
    if (esq < j)  QuickSortFirstPivot(array, esq, j);
    if (i < dir)  QuickSortFirstPivot(array, i, dir);
}
```

Quicksort Last Pivot:

```java
public static void QuickSortLastPivot (int[] array ,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(array, i, j);
            i++;
            j--;
        }
    }
    if (esq < j)  QuickSortLastPivot(array, esq, j);
    if (i < dir)  QuickSortLastPivot(array, i, dir);
}
```

Quicksort Random Pivot:

```java
public static void QuickSortRandomPivot (int[] array ,int esq ,int dir ){
    int i = esq, j = dir;
    int pivo = array[rand.nextInt(esq,dir)]; //random ente esq e dir
    while (i <= j) {
        while (array[i] < pivo) i++;
        while (array[j] > pivo) j--;
        if (i <= j) {
            swap(array, i, j);
            i++;
            j--;
        }
    }
    if (esq < j)  QuickSortRandomPivot(array, esq, j);
    if (i < dir)  QuickSortRandomPivot(array, i, dir);
}
```

Quicksort Median Of Three:

```java
public static int getMedian(int a, int b, int c) {
    int median = 0;
    if(a<b){
        if(b<c){
            median = b;
        }else{
            if(a<c){
                median = c;
            }else{
                median = a;
            }
        }
    }else{
        if(c<b){
            median = b;
        }else{
            if(c<a){
                median = c;
            }else{
                median = a;
            }
        }
    }
    return median;
}

public static void QuickSortMedianOfThree (int[] array ,int esq ,int dir ){
    int i = esq, j = dir;
    //medianOfThree
    int pivo = array[getMedian(esq, (esq+dir)/2, dir)];
    while (i <= j) {
        while (array[i] < pivo) i++;
        while (array[j] > pivo) j--;
        if (i <= j) {
            swap(array, i, j);
            i++;
            j--;
        }
    }
    if (esq < j)  QuickSortMedianOfThree(array, esq, j);
    if (i < dir)  QuickSortMedianOfThree(array, i, dir);
}
```

Execução do código:

Saída:

```
QUICKSORT FIRST PIVOT:
    ORDERED ARRAY:
        100 elements: 102800ns
        1000 elements: 1754400ns
        10.000 elements: 38853700ns
    ALMOST ORDERED ARRAY:
        100 elements: 86501ns
        1000 elements: 276300ns
        10.000 elements: 18848400ns
    RANDOM ARRAY:
        100 elements: 73100ns
        1000 elements: 672599ns
        10.000 elements: 1551201ns
```

```
QUICKSORT LAST PIVOT:
    ORDERED ARRAY:
        100 elements: 162200ns
        1000 elements: 1971599ns
        10.000 elements: 54453800ns
    ALMOST ORDERED ARRAY:
        100 elements: 5100ns
        1000 elements: 190400ns
        10.000 elements: 17812700ns
    RANDOM ARRAY:
        100 elements: 111999ns
        1000 elements: 2448800ns
        10.000 elements: 25094900ns
```

```
QUICKSORT RANDOM PIVOT:
    ORDERED ARRAY:
        100 elements: 201300ns
        1000 elements: 530899ns
        10.000 elements: 1071900ns
    ALMOST ORDERED ARRAY:
        100 elements: 10199ns
        1000 elements: 164000ns
        10.000 elements: 851300ns
    RANDOM ARRAY:
        100 elements: 9100ns
        1000 elements: 181300ns
        10.000 elements: 957800ns
```

```
QUICKSORT MEDIAN OF THREE PIVOT:
    ORDERED ARRAY:
        100 elements: 30999ns
        1000 elements: 228700ns
        10.000 elements: 676400ns
    ALMOST ORDERED ARRAY:
        100 elements: 5200ns
        1000 elements: 127100ns
        10.000 elements: 466599ns
    RANDOM ARRAY:
        100 elements: 7300ns
        1000 elements: 55300ns
        10.000 elements: 558100ns
```

Em uma planilha:

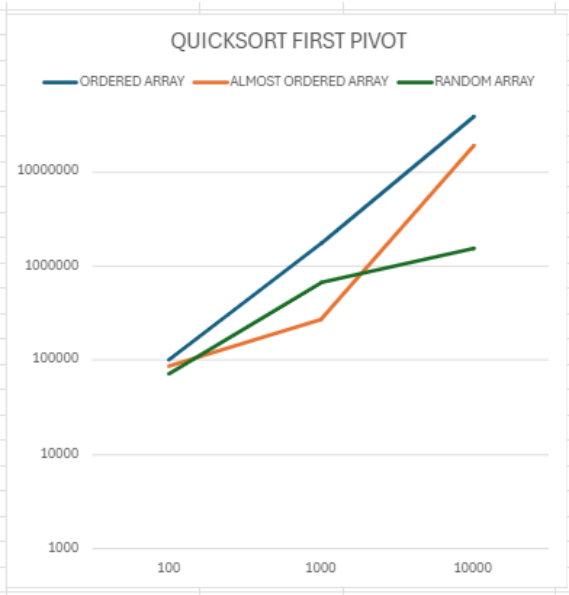
QUICKSORT AND PIVOT

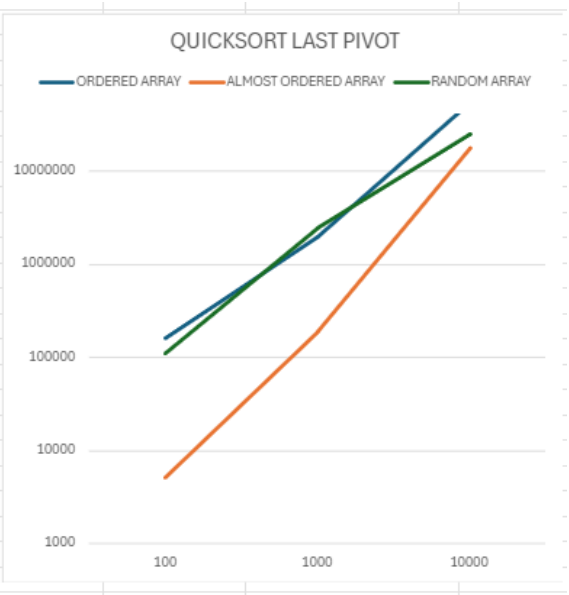
	QUICKSORT FIRST PIVOT			QUICKSORT LAST PIVOT		
NUMBER OF ELEMENTS	ORDERED ARRAY	ALMOST ORDERED ARRAY	RANDOM ARRAY	ORDERED ARRAY	ALMOST ORDERED ARRAY	RANDOM ARRAY
100	102800ns	86501ns	73100ns	162200ns	5100ns	111999ns
1000	1754400ns	276300ns	672599ns	1971599ns	190400ns	2448800ns
10000	38853700ns	18848400ns	1551201ns	54453800ns	17812700ns	25094900ns

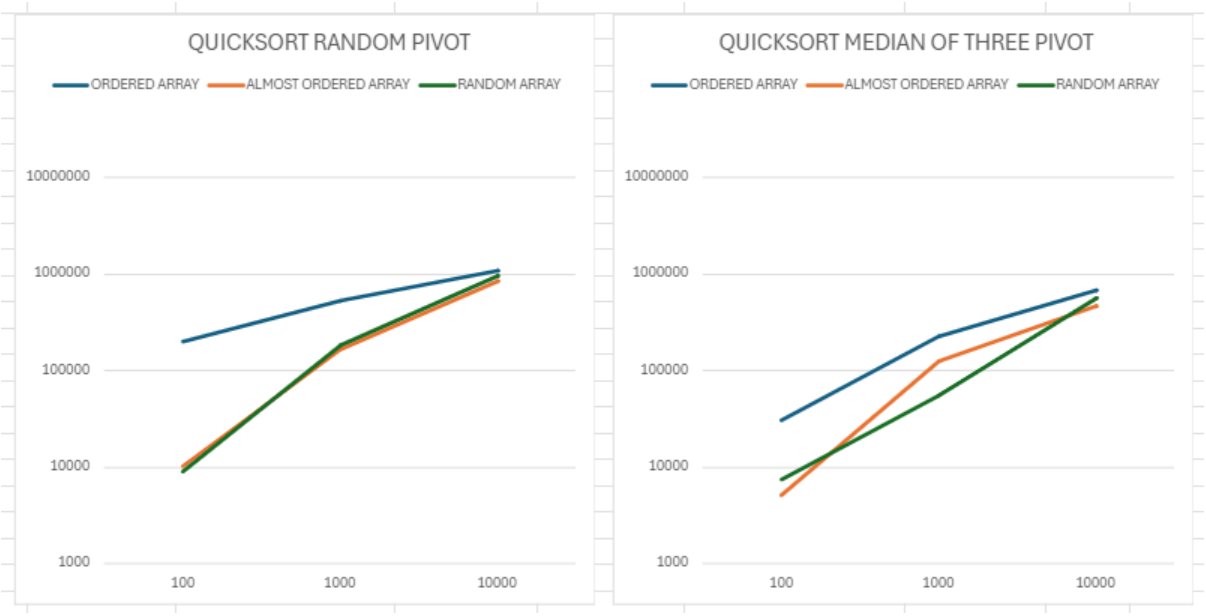
	QUICKSORT RANDOM PIVOT			QUICKSORT MEDIAN OF THREE PIVOT		
NUMBER OF ELEMENTS	ORDERED ARRAY	ALMOST ORDERED ARRAY	RANDOM ARRAY	ORDERED ARRAY	ALMOST ORDERED ARRAY	RANDOM ARRAY
100	201300ns	10199ns	9100ns	30999ns	5200ns	7500ns
1000	530899ns	164000ns	181300ns	228700ns	127100ns	55100ns
10000	1071900ns	851300ns	957800ns	676400ns	466599ns	565800ns

Comparando:

Gráficos:







Discussão:

O quicksort que utiliza a mediana de três elementos como pivô teve o melhor desempenho para todos os tipos de array. Os que utilizam o primeiro e último elemento obtiveram os piores resultados, sendo que o que utiliza o último como pivô foi um pouco melhor, considerando os arrays ordenados e quase ordenados, mas a situação se inverte se o array estiver em ordem decrescente, entrando no pior caso do quick. O pivô aleatório obteve um resultado na média, mas não tão discrepante como os de primeiro e último.