PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Instituto de Ciências Exatas e de Informática

Implementação do QuickSort com Variação na Escolha do Pivô*

Relatório referente a atividade lab05

1543726 - Pedro Henrique Cardoso Maia[1]

*Artigo apresentado ao Instituto de Ciências Exatas e Informática da Pontifícia Universidade Católica de Minas Gerais como pré-requisito para obtenção do título de Bacharel em Ciência da Computação.

[1]Aluno do Programa de Graduação em Ciência da Computação, Brasil – pedro.maia.1543726@sga.pucminas.br.

1 CÓDIGO DA APLICAÇÃO

```java
import java.util.concurrent.ThreadLocalRandom;

class Quicksort extends Geracao {

    public Quicksort(){ super(); }
    public Quicksort(int tamanho){ super(tamanho); }

    @Override
    public void sort(int i) {
        if(i==1) QuickSortFirstPivot(0,n-1);
        if(i==2) QuickSortLastPivot(0,n-1);
        if(i==3) QuickSortRandomPivot(0,n-1);
        if(i==4) QuickSortMedianOfThree(0,n-1);
    }

    private void QuickSortFirstPivot(int esq, int dir) {
        int i=esq, j=dir, pivo=array[i];
        while(i<=j){
            while(array[i]<pivo){ cmp++; i++; }
            while(array[j]>pivo){ cmp++; j--; }
            if(i<=j){ swap(i,j); mov+=3; 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, pivo=array[dir];
        while(i<=j){
            while(array[i]<pivo){ cmp++; i++; }
            while(array[j]>pivo){ cmp++; j--; }
            if(i<=j){ swap(i,j); mov+=3; i++; j--; }
        }
        if(esq<j) QuickSortLastPivot(esq,j);
        if(i<dir) QuickSortLastPivot(i,dir);
    }

    private void QuickSortRandomPivot(int esq, int dir) {
        int randomPivot=ThreadLocalRandom.current().nextInt(esq,dir+1);
        int i=esq, j=dir, pivo=array[randomPivot];
        while(i<=j){
            while(array[i]<pivo){ cmp++; i++; }
            while(array[j]>pivo){ cmp++; j--; }
            if(i<=j){ swap(i,j); mov+=3; i++; j--; }
        }
        if(esq<j) QuickSortRandomPivot(esq,j);
        if(i<dir) QuickSortRandomPivot(i,dir);
    }

    private void QuickSortMedianOfThree(int esq, int dir) {
        int i=esq, j=dir, pivo=array[(dir+esq)/2];
        while(i<=j){
            while(array[i]<pivo){ cmp++; i++; }
            while(array[j]>pivo){ cmp++; j--; }
            if(i<=j){ swap(i,j); mov+=3; i++; j--; }
        }
        if(esq<j) QuickSortMedianOfThree(esq,j);
        if(i<dir) QuickSortMedianOfThree(i,dir);
    }
}
```

Listing 1 – Código da aplicação

2 PIVÔ COMO PRIMEIRO ELEMENTO

Tabela 1 – Comparação dos cenários com pivô sendo o primeiro elemento, considerando $n = 1000$.

Cenário	Ordem de Complexidade	Fórmula Comparações	Fórmula Movimentações	Nº Comparações	Nº Movimentações
Array Crescente	$O(n^2)$	$C(n) = \frac{n(n-1)}{2}$	$M(n) = 3 \cdot (n-1)$	499,500	2,997
Parcialmente Ordenado	Entre $O(n \log n)$ e $O(n^2)$	***	***	8,040	20,751
Array Aleatório	$O(n \log n)$	$C(n) = n \cdot \log_2 n$	$M(n) = 3 \cdot n \cdot \log_2 n$	7,902	9,190

Figura 1 – Execução do Array Crescente

```
loona@DESKTOP-JL0DOVQ:~/Projects/AEDS2/lab/lab08$ java Principal 1000
Tempo para ordenar: 0.002 s.
isOrdenado: true
Número de movimentações realizadas: 2997
Número de comparações realizadas: 499500
loona@DESKTOP-JL0DOVQ:~/Projects/AEDS2/lab/lab08$ java Principal 100
Tempo para ordenar: 0.0 s.
isOrdenado: true
Número de movimentações realizadas: 297
Número de comparações realizadas: 4950
loona@DESKTOP-JL0DOVQ:~/Projects/AEDS2/lab/lab08$ java Principal 10000
Tempo para ordenar: 0.033 s.
isOrdenado: true
Número de movimentações realizadas: 29997
Número de comparações realizadas: 49995000
```

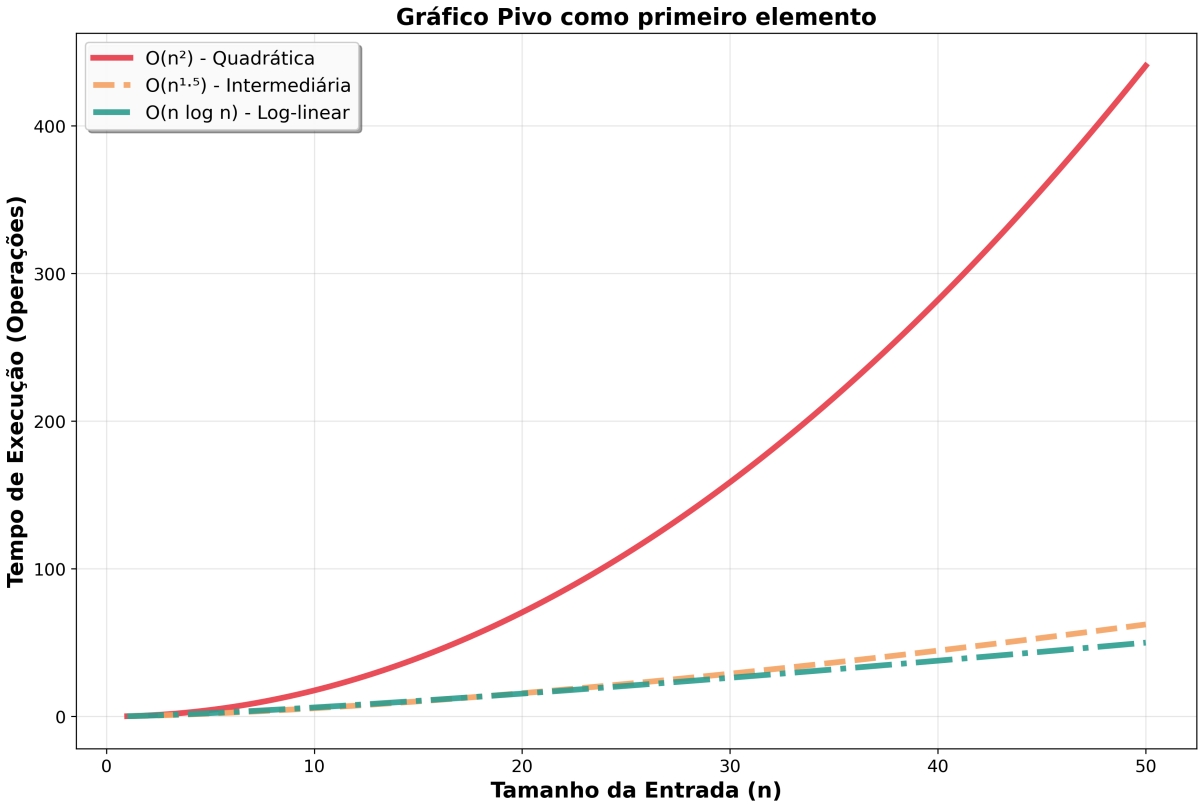
Figura 2 – Execução do Array Parcialmente Ordenado

```
loona@DESKTOP-JL0DOVQ:~/Projects/AEDS2/lab/lab08$ java Principal 100
Tempo para ordenar: 0.0 s.
isOrdenado: true
Número de movimentações realizadas: 609
Número de comparações realizadas: 799
loona@DESKTOP-JL0DOVQ:~/Projects/AEDS2/lab/lab08$ java Principal 1000
Tempo para ordenar: 0.0 s.
isOrdenado: true
Número de movimentações realizadas: 8040
Número de comparações realizadas: 20751
loona@DESKTOP-JL0DOVQ:~/Projects/AEDS2/lab/lab08$ java Principal 10000
Tempo para ordenar: 0.002 s.
isOrdenado: true
Número de movimentações realizadas: 103692
Número de comparações realizadas: 389124
```

Figura 3 – Execução do Array Aleatório

```
loona@DESKTOP-JL0DOVQ:~/Projects/AEDS2/lab/lab08$ java Principal 100
Tempo para ordenar: 0.0 s.
isOrdenado: true
Número de movimentações realizadas: 564
Número de comparações realizadas: 498
loona@DESKTOP-JL0DOVQ:~/Projects/AEDS2/lab/lab08$ java Principal 1000
Tempo para ordenar: 0.001 s.
isOrdenado: true
Número de movimentações realizadas: 7902
Número de comparações realizadas: 9190
loona@DESKTOP-JL0DOVQ:~/Projects/AEDS2/lab/lab08$ java Principal 10000
Tempo para ordenar: 0.002 s.
isOrdenado: true
Número de movimentações realizadas: 101007
Número de comparações realizadas: 135378
loona@DESKTOP-JL0DOVQ:~/Projects/AEDS2/lab/lab08$
```

Figura 4 – Gráfico geral do cenário com pivô primeiro elemento



3 PIVÔ COMO ÚLTIMO ELEMENTO

Tabela 2 – Comparação dos cenários com pivô sendo o ultimo elemento, considerando $n = 1000$.

Cenário	Ordem de Complexidade	Fórmula Comparações	Fórmula Movimentações	Nº Comparações	Nº Movimentações
Array Crescente	$O(n^2)$ (Pior Caso)	$C(n) = \frac{n(n-1)}{2}$	$M(n) = 3 \cdot (n-1)$	499,500	2,997
Parcialmente Ordenado	Entre $O(n \log n)$ e $O(n^2)$	***	***	8,040	20,751
Array Aleatório	$O(n \log n)$ (Caso Médio)	***	***	7,902	9,190

Figura 5 – Execução do Array Crescente

```
loona@DESKTOP-JL0DOVQ:~/Projects/AEDS2/lab/lab05$ java Principal 100
Tempo para ordenar: 0.0 s.
isOrdenado: true
Número de movimentações realizadas: 297
Número de comparações realizadas: 4950
loona@DESKTOP-JL0DOVQ:~/Projects/AEDS2/lab/lab05$ java Principal 2000
Tempo para ordenar: 0.004 s.
isOrdenado: true
Número de movimentações realizadas: 5997
Número de comparações realizadas: 1999000
loona@DESKTOP-JL0DOVQ:~/Projects/AEDS2/lab/lab05$ java Principal 9000
Tempo para ordenar: 0.03 s.
isOrdenado: true
Número de movimentações realizadas: 26997
Número de comparações realizadas: 40495500
loona@DESKTOP-JL0DOVQ:~/Projects/AEDS2/lab/lab05$
```

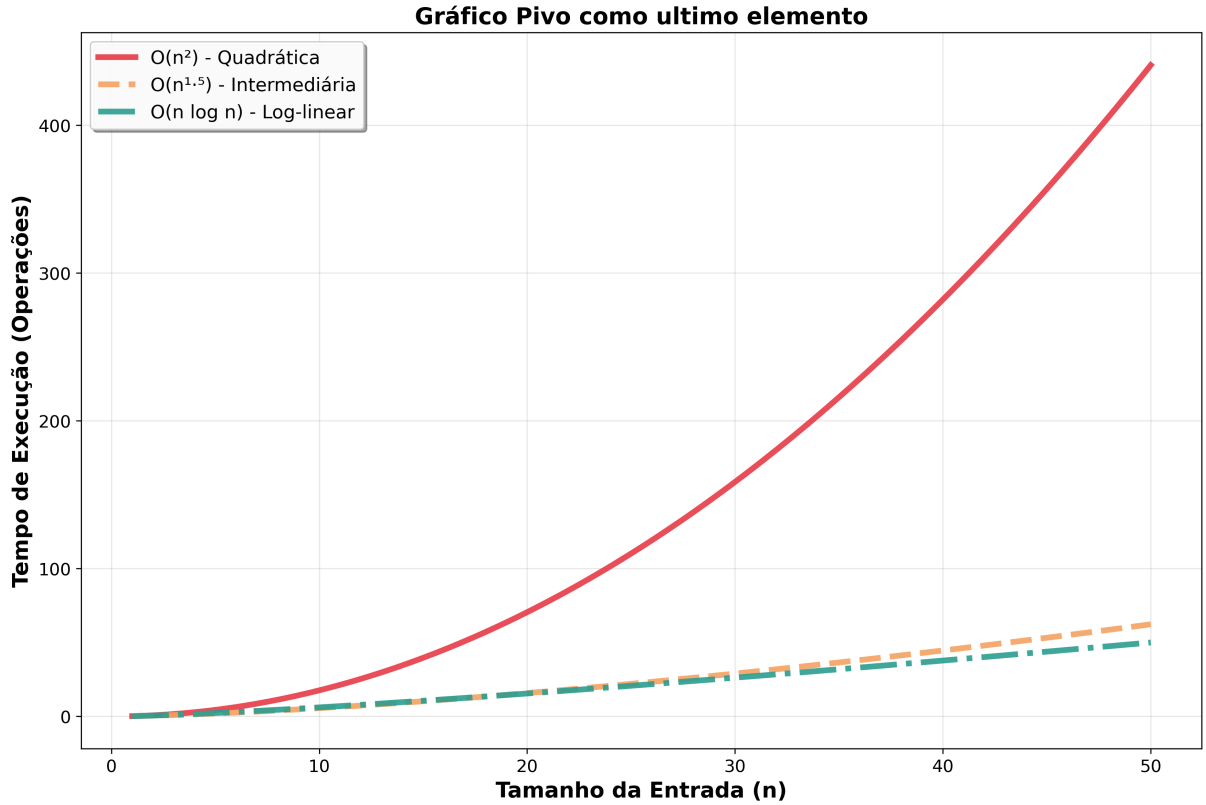
Figura 6 – Execução do Array Parcialmente Ordenado

```
loona@DESKTOP-JL0DOVQ:~/Projects/AEDS2/lab/lab08$ java Principal 100
Tempo para ordenar: 0.0 s.
isOrdenado: true
Número de movimentações realizadas: 552
Número de comparações realizadas: 1022
loona@DESKTOP-JL0DOVQ:~/Projects/AEDS2/lab/lab08$ java Principal 1000
Tempo para ordenar: 0.0 s.
isOrdenado: true
Número de movimentações realizadas: 7740
Número de comparações realizadas: 23682
loona@DESKTOP-JL0DOVQ:~/Projects/AEDS2/lab/lab08$ java Principal 10000
Tempo para ordenar: 0.002 s.
isOrdenado: true
Número de movimentações realizadas: 99375
Número de comparações realizadas: 441539
loona@DESKTOP-JL0DOVQ:~/Projects/AEDS2/lab/lab08$
```

Figura 7 – Execução do Array Aleatório

```
loona@DESKTOP-JL0DOVQ:~/Projects/AEDS2/lab/lab08$ java Principal 100
Tempo para ordenar: 0.0 s.
isOrdenado: true
Número de movimentações realizadas: 552
Número de comparações realizadas: 521
loona@DESKTOP-JL0DOVQ:~/Projects/AEDS2/lab/lab08$ java Principal 1000
Tempo para ordenar: 0.001 s.
isOrdenado: true
Número de movimentações realizadas: 7917
Número de comparações realizadas: 7914
loona@DESKTOP-JL0DOVQ:~/Projects/AEDS2/lab/lab08$ java Principal 5000
Tempo para ordenar: 0.001 s.
isOrdenado: true
Número de movimentações realizadas: 46833
Número de comparações realizadas: 59677
```

Figura 8 – Gráfico geral do cenário com pivô último elemento



4 PIVÔ ALEATÓRIO

Tabela 3 – Comparação dos cenários com pivô aleatório, considerando $n = 1000$.

Cenário	Ordem de Complexidade	Fórmula Comparações	Fórmula Movimentações	Nº Comparações	Nº Movimentações
Array Crescente	$O(n \log n)$ (Caso médio)	$C(n) = n \cdot \log_2 n$	$M(n) = 3 \cdot n \cdot \log_2 n$	9,966	29,897
Parcialmente Ordenado	$O(n \log n)$ (Caso médio)	$C(n) = n \cdot \log_2 n$	$M(n) = 3 \cdot n \cdot \log_2 n$	9,966	29,897
Array Aleatório	$O(n \log n)$ (Caso médio)	$C(n) = n \cdot \log_2 n$	$M(n) = 3 \cdot n \cdot \log_2 n$	9,966	29,897

Figura 9 – Execução do Array Crescente

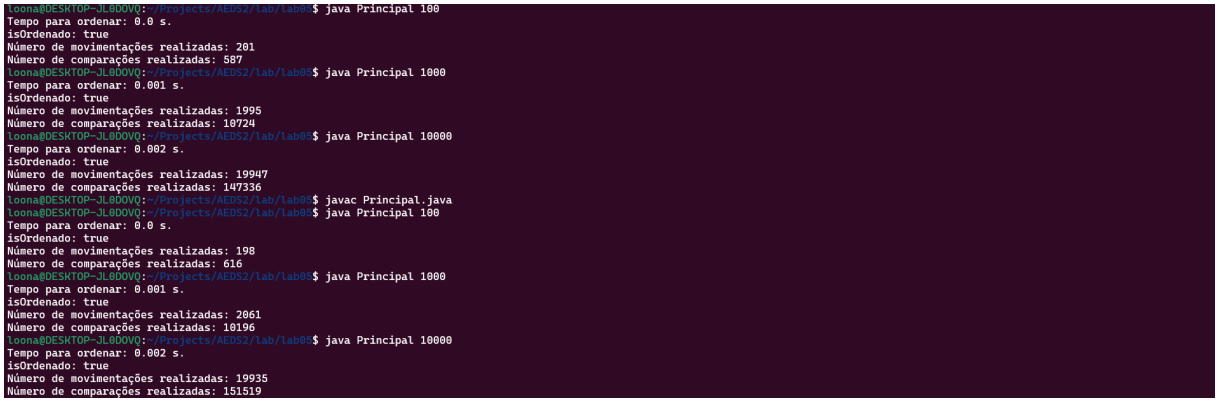
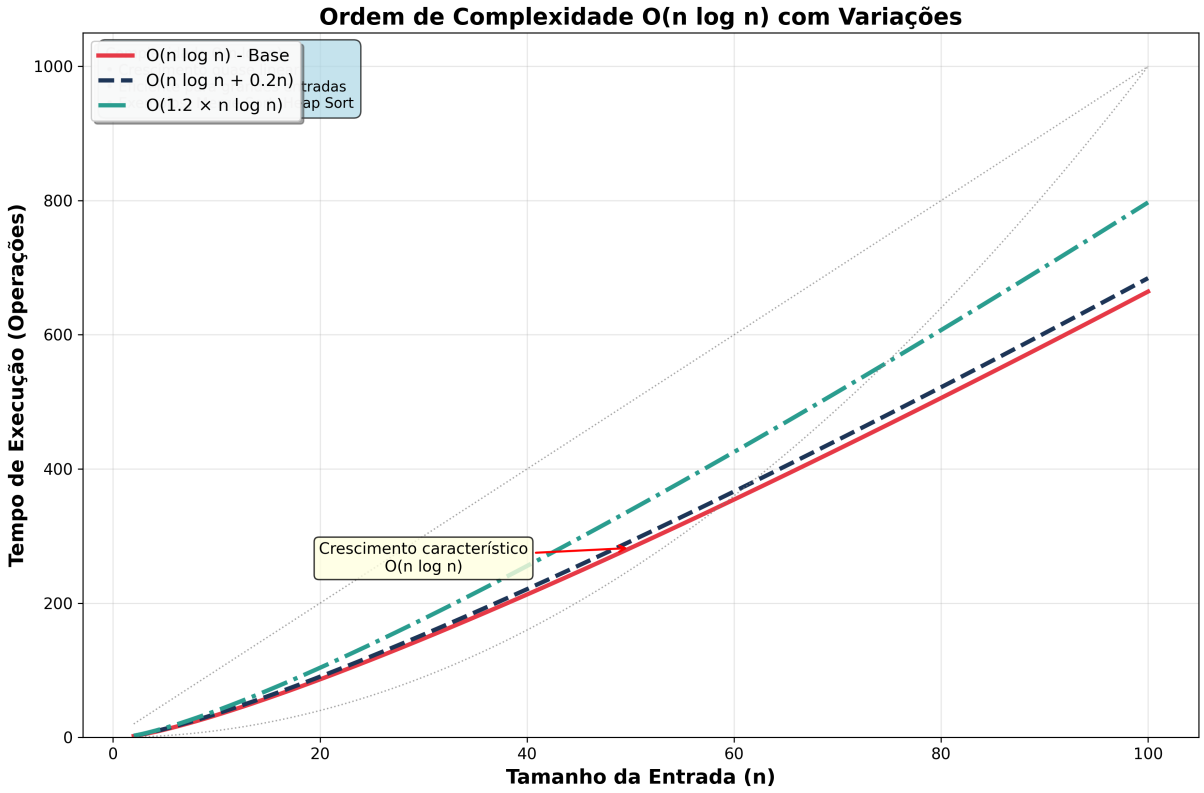


Figura 10 – Execução do Array Parcialmente Ordenado

Figura 11 – Execução do Array Aleatório

Figura 12 – Gráfico geral do cenário com pivô último elemento



5 PIVÔ COMO MEDIANA DOS ELEMENTOS DO ARRAY

Tabela 4 – Comparação dos cenários com pivô sendo a mediana, considerando $n = 1000$.

Cenário	Ordem de Complexidade	Fórmula Comparações	Fórmula Movimentações	Nº Comparações	Nº Movimentações
Array Crescente	$O(n \log n)$ (Caso médio)	$C(n) = n \cdot \log_2 n$	$M(n) = 3 \cdot n \cdot \log_2 n$	9,966	29,897
Parcialmente Ordenado	$O(n \log n)$ (Caso médio)	$C(n) = n \cdot \log_2 n$	$M(n) = 3 \cdot n \cdot \log_2 n$	9,966	29,897
Array Aleatório	$O(n \log n)$ (Caso médio)	$C(n) = n \cdot \log_2 n$	$M(n) = 3 \cdot n \cdot \log_2 n$	9,966	29,897

Figura 13 – Execução do Array Crescente

```
loona@DESKTOP-JL0DOVQ:~/Projects/AEDS2/lab/lab08$ java Principal 100
Tempo para ordenar: 0.0 s.
isOrdenado: true
Número de movimentações realizadas: 189
Número de comparações realizadas: 480
loona@DESKTOP-JL0DOVQ:~/Projects/AEDS2/lab/lab08$ java Principal 1000
Tempo para ordenar: 0.0 s.
isOrdenado: true
Número de movimentações realizadas: 1533
Número de comparações realizadas: 7987
loona@DESKTOP-JL0DOVQ:~/Projects/AEDS2/lab/lab08$ java Principal 10000
Tempo para ordenar: 0.001 s.
isOrdenado: true
Número de movimentações realizadas: 17712
Número de comparações realizadas: 113631
loona@DESKTOP-JL0DOVQ:~/Projects/AEDS2/lab/lab08$
```

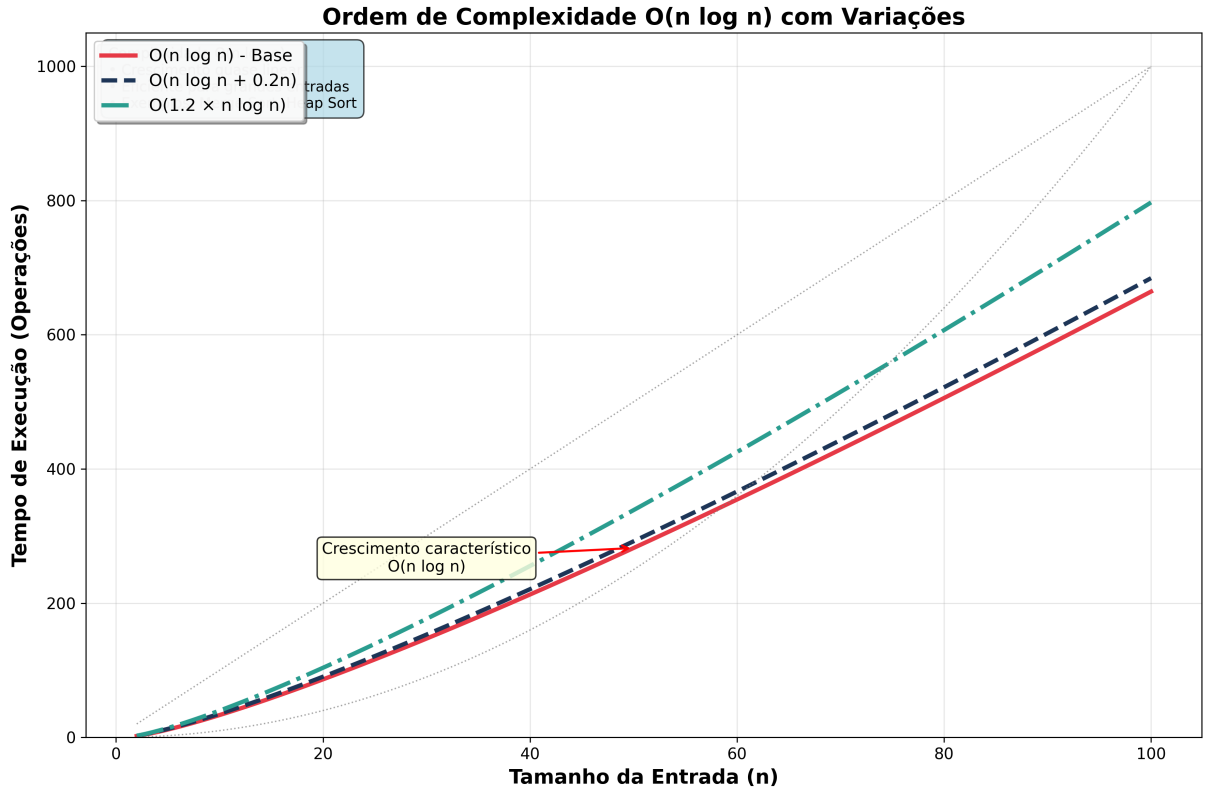
Figura 14 – Execução do Array Parcialmente Ordenado

```
loona@DESKTOP-JL0DOVQ:~/Projects/AEDS2/lab/lab05$ java Principal 100
Tempo para ordenar: 0.0 s.
isOrdenado: true
Número de movimentações realizadas: 432
Número de comparações realizadas: 2368
loona@DESKTOP-JL0DOVQ:~/Projects/AEDS2/lab/lab05$ java Principal 1000
Tempo para ordenar: 0.002 s.
isOrdenado: true
Número de movimentações realizadas: 4473
Número de comparações realizadas: 248509
loona@DESKTOP-JL0DOVQ:~/Projects/AEDS2/lab/lab05$ java Principal 10000
Tempo para ordenar: 0.03 s.
isOrdenado: true
Número de movimentações realizadas: 44910
Número de comparações realizadas: 24987742
```

Figura 15 – Execução do Array Aleatório

```
loona@DESKTOP-JL0DOVQ:~/Projects/AEDS2/lab/lab05$ java Principal 100
Tempo para ordenar: 0.001 s.
isOrdenado: true
Número de movimentações realizadas: 522
Número de comparações realizadas: 580
loona@DESKTOP-JL0DOVQ:~/Projects/AEDS2/lab/lab05$ java Principal 1000
Tempo para ordenar: 0.0 s.
isOrdenado: true
Número de movimentações realizadas: 7761
Número de comparações realizadas: 7487
loona@DESKTOP-JL0DOVQ:~/Projects/AEDS2/lab/lab05$ java Principal 10000
Tempo para ordenar: 0.002 s.
isOrdenado: true
Número de movimentações realizadas: 101337
Número de comparações realizadas: 98953
```

Figura 16 – Gráfico geral do cenário com pivô último elemento



6 CONSIDERAÇÕES E CONCLUSÃO

O funcionamento do QuickSort ocorre dividindo o array em duas partes, escolhendo um pivô para partição. Dependendo da escolha do pivô o algoritmo terá complexidade $O(n \log n)$, o que seria o ideal ou complexidade $O(n^2)$, o que piora em muito o algoritmo. Escolher o pivô como primeiro ou último elemento do array não é uma boa escolha pois em casos do array estar em ordem crescente ou decrescente, a divisão do array irá ocorrer de forma desequilibrada e a complexidade do algoritmo será de $O(n^2)$. Escolher um pivô aleatório minimiza o risco de acontecer o pior caso e a complexidade do algoritmo será de quase sempre $O(n \log n)$, o que configura uma boa escolha do pivô. Escolher o pivô como mediana do array é a estratégia mais eficiente pois evita totalmente o pior caso e a complexidade será sempre de $O(n \log n)$.