UNIVERSIDADE FEDERAL DE VIÇOSA - CAMPUS FLORESTAL
INSTITUTO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
CIÊNCIA DA COMPUTAÇÃO
ALGORITMOS E ESTRUTURAS DE DADOS I

# IMPLEMENTAÇÃO E AVALIAÇÃO DE DESEMPENHO DE DIFERENTES ALGORITMOS DE ORDENAÇÃO

LARISSA ISABELLE 3871

OTÁVIO SANTOS 3890

PEDRO CARDOSO 3877

FLORESTAL

2019

# SUMÁRIO

1. INTRODUÇÃO 3
2. CÓDIGO 4
3. TEMPOS DE EXECUÇÃO 6
4. CONSIDERAÇÕES FINAIS 14

# 1. INTRODUÇÃO

Este trabalho consistiu na implementação das estruturas de teste Biblioteca, Texto e Palavra em duas implementações: Array e Lista Encadeada. Além disso, foram também elaborados os algoritmos de ordenação SelectSort e QuickSort para ordenação de cada uma dessas estruturas. Por fim foi feita a análise do desempenho desses algoritmos em situações específicas, sendo analisados o número de comparações, o número de movimentações e o tempo total gasto.

# 2. CÓDIGO

Para implantação de todos os casos requisitados pelo trabalho, foram necessários 21 arquivos diferentes, como pode ser visto abaixo na *Figura 1*.

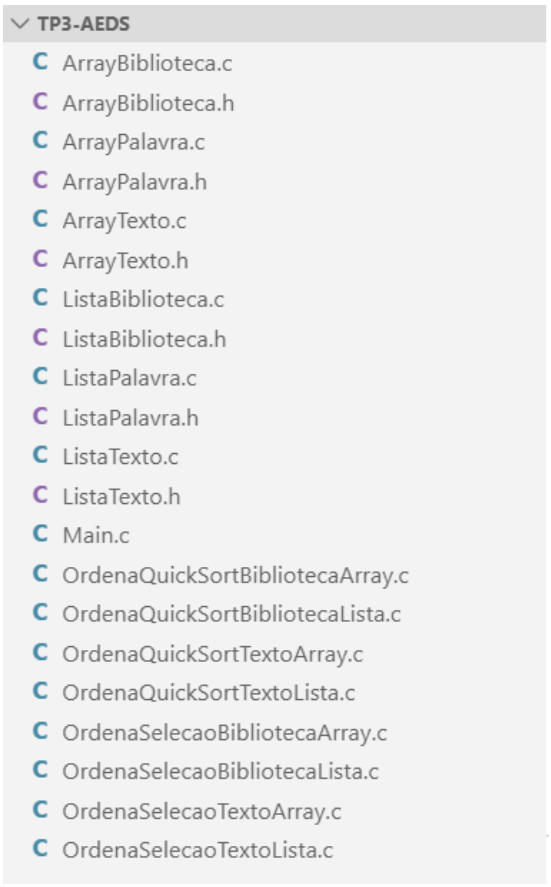


*Figura 1 Arquivos necessários para implementação do trabalho.*

Foram implementados todos os TAD's tanto em Array quanto em Lista Encadeada e suas respectivas ordenações tanto com o algoritmo de ordenação SelectSort, quanto o algoritmo de ordenação QuickSort.

Para a implementação dos TAD's Texto, tanto com Lista encadeada quanto com Array, foram utilizadas funções oferecidas pelos TAD's Palavra. Além disso, para implementação dos TAD's Biblioteca, foram utilizados os TAD's Texto. Dessa forma, uma estrutura do tipo biblioteca contém vários textos que por sua vez contém diversas palavras. Assim, a menor parte dessa cadeia são os caracteres que formam as palavras.

Para a implementação dos TAD's para ordenação, foi necessário a implantação de uma função extra nos TAD's Texto e Biblioteca. Essa função, chamada copiaBiblioteca/copiaTexto, precisou ser desenvolvida para que fosse possível fazer uma cópia da estrutura, uma vez que somente assim seria possível ordenar uma cópia de uma

estrutura e não a estrutura original.

Com isso foi possível gerar uma estrutura aleatória e ordená-la usando como chaves as primeiras letras das palavras no caso dos textos e o tamanho dos textos no caso da ordenação das bibliotecas. Por fim, é importante ressaltar que ao ordenar as Listas encadeadas foram movidos apenas os valores contidos em cada célula, e não os ponteiros de cada uma.

Após o desenvolvimento do trabalho, partimos para a implementação e análise dos casos de teste solicitados.

# 3. TEMPOS DE EXECUÇÃO

Para a realização dos testes, foi utilizado um notebook com as seguintes especificações de software e hardware:

**Versão Kernel :** 5.2.21-1-MANJARO

**Processador** : 4 x Intel Core i7-7500U CPU @ 2.70GHz

**Memória** : 7,7 GiB de RAM

Como foi estabelecido na especificação do trabalho, os testes feitos para o TAD TEXTO foram de tamanho 100, 1000, 10.000, 100.000. Seguem os resultados das medições de comparações, movimentações e o tempo gasto para ordenar com cada algoritmo de ordenação especificado (SelectSort e QuickSort). Vale ressaltar que, para medir o tempo foi utilizada a biblioteca *time.h* e o comando *time* no terminal, avaliando o resultado dado pelo campo *user*, que consiste no tempo gasto pelo usuário no processador.

1. **Implementação por array:**

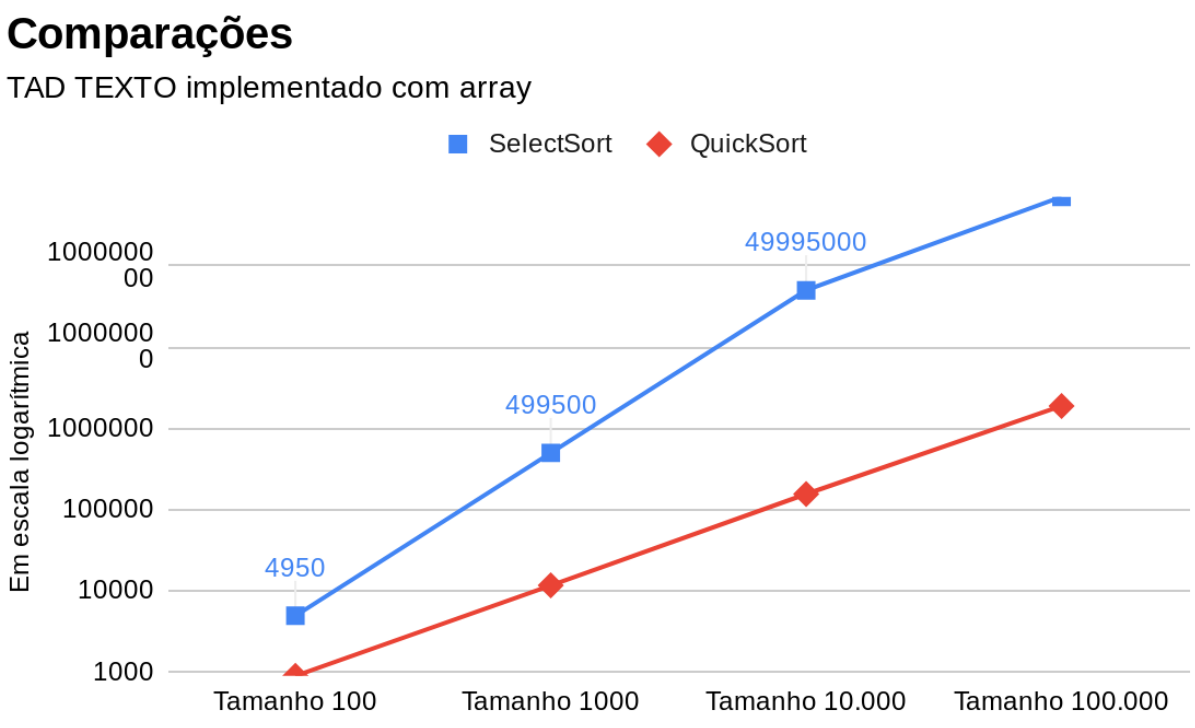


Figura 2. Comparações TAD TEXTO.

## Movimentações

TAD TEXTO implementado com array

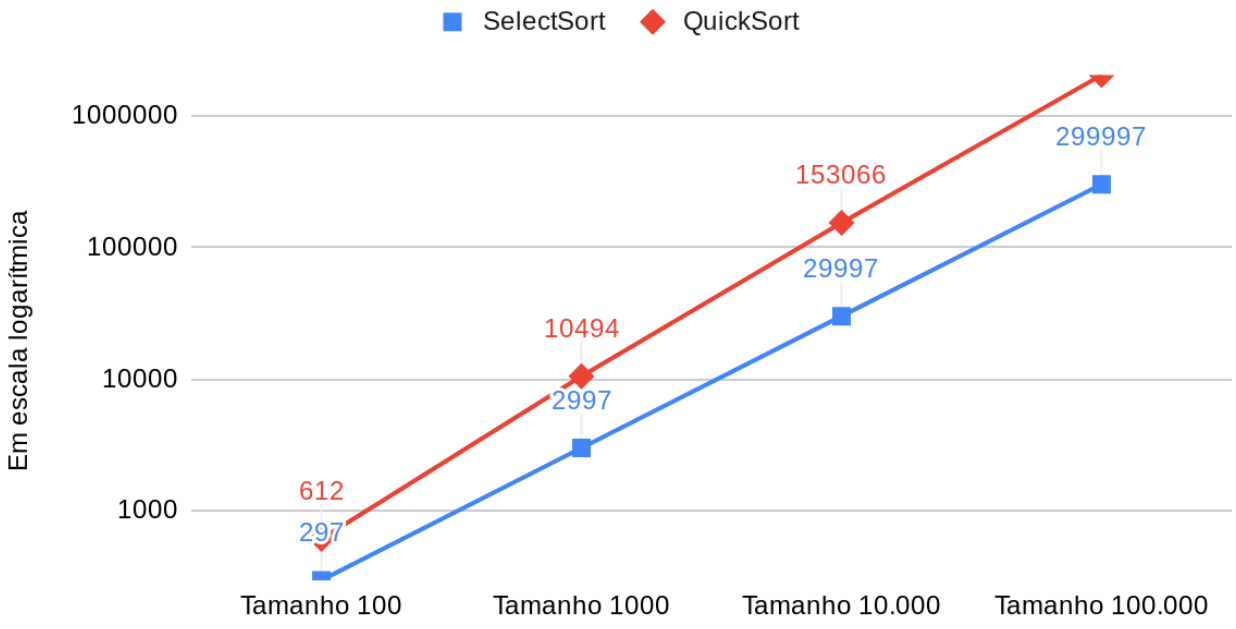


Figura 3. Movimentações TAD TEXTO.

## Tempo gasto em segundos

TAD TEXTO implementado com array

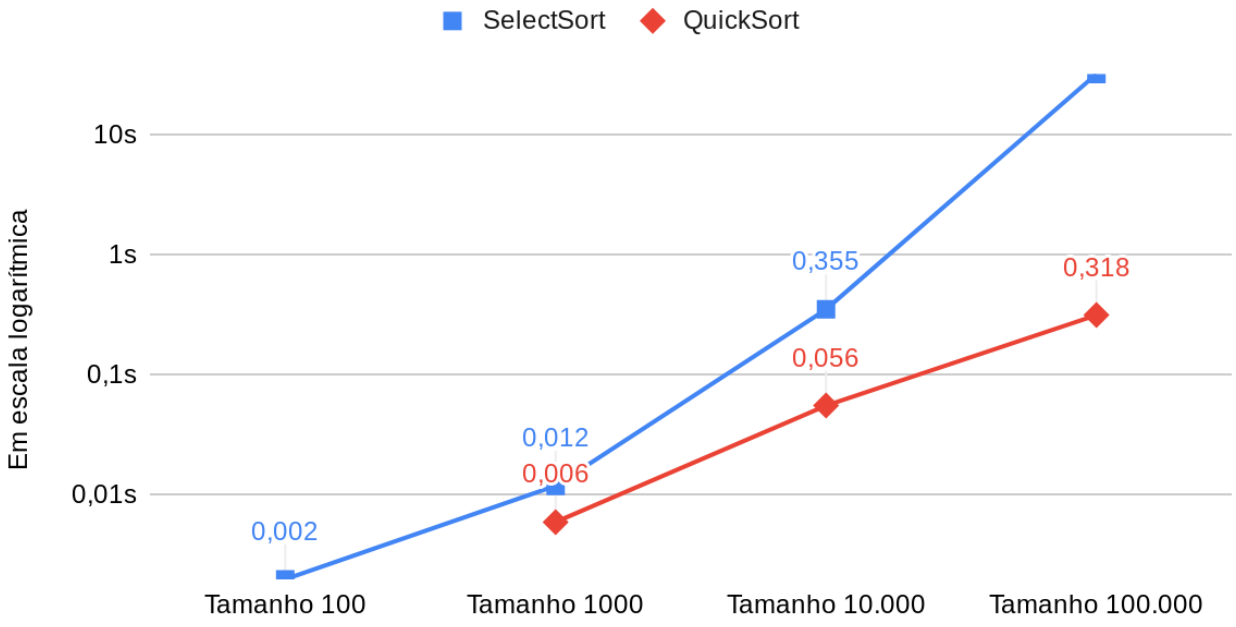


Figura 4. Tempo gasto TAD TEXTO.

2. **Implementação por lista encadeada:**

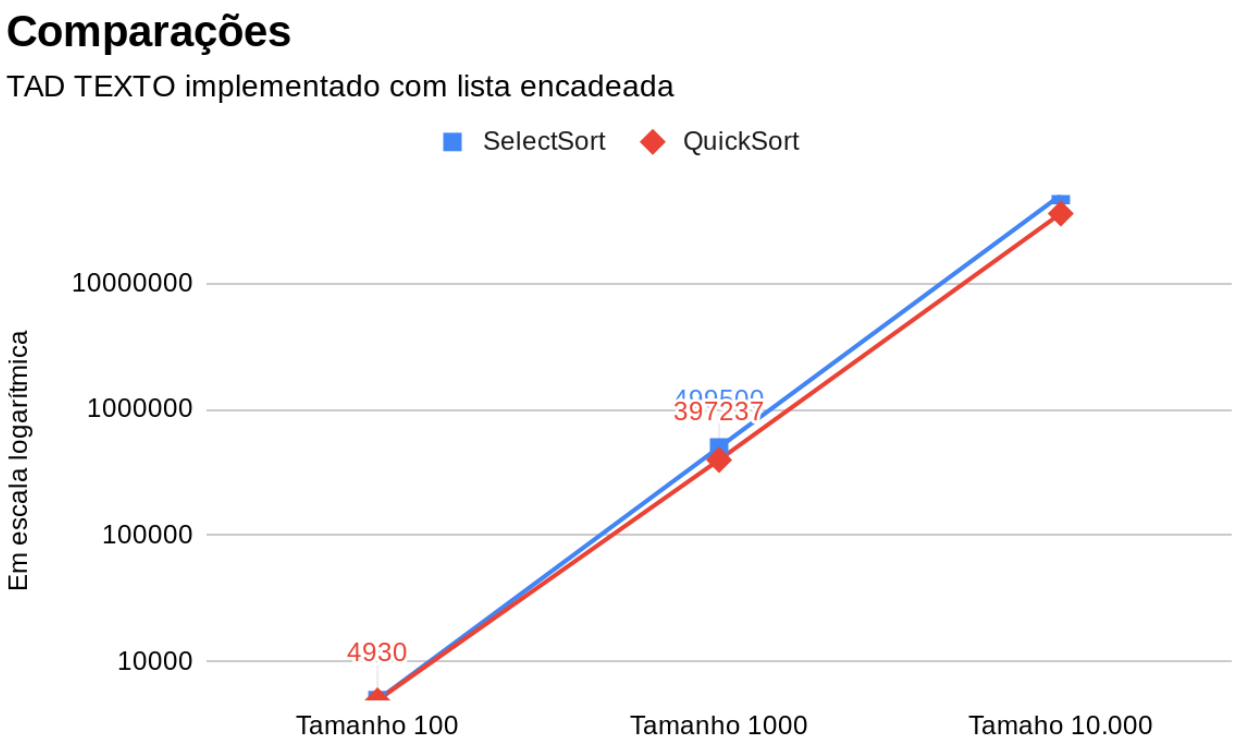


Figura 5. Comparações TAD TEXTO.

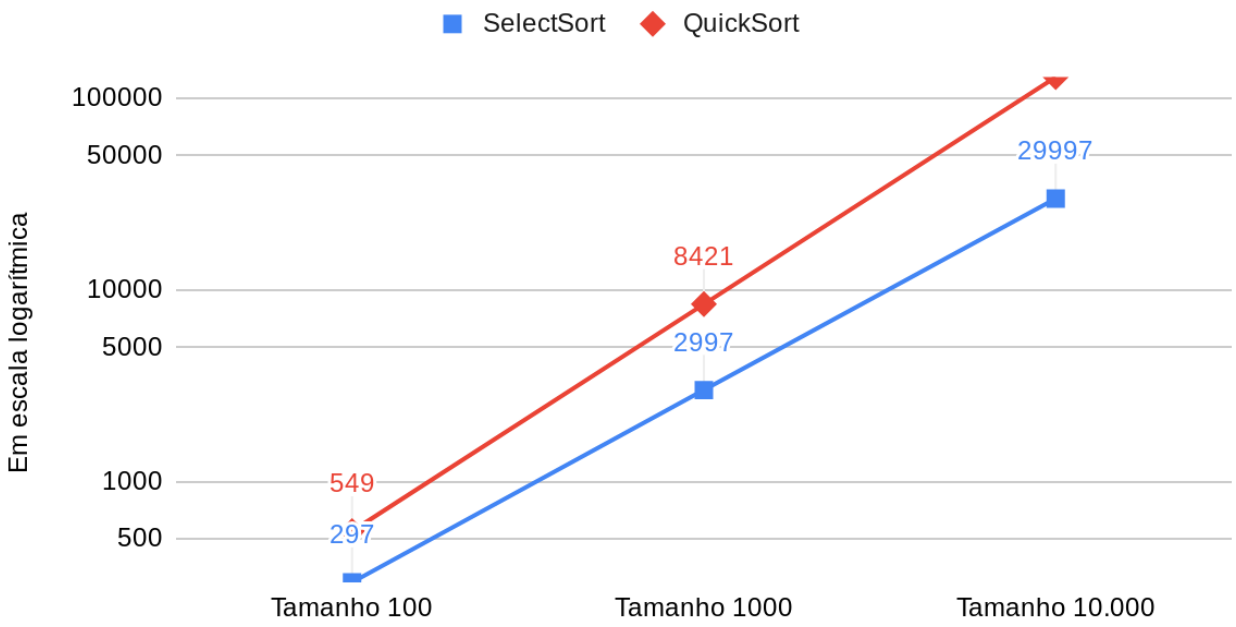


Figura 6. Movimentações TAD TEXTO.

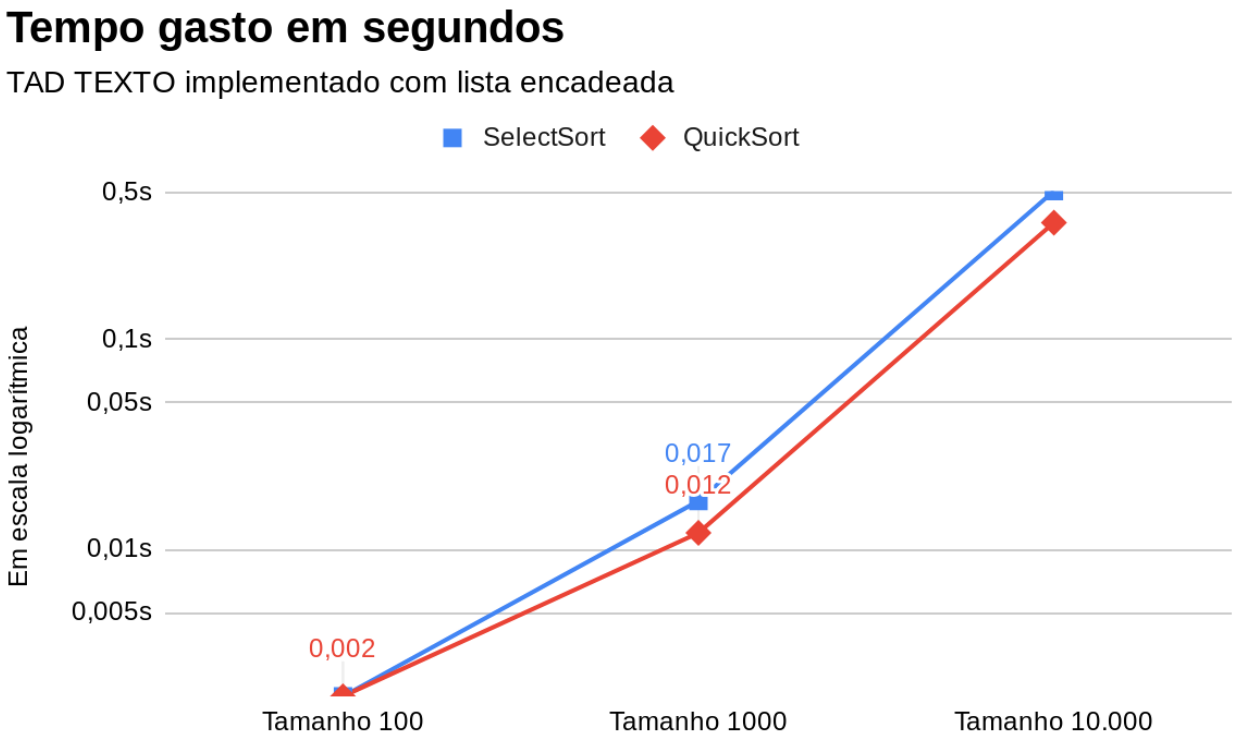


Figura 7. Tempo gasto TAD TEXTO.

- Texto de tamanho 100.000:

Este caso de teste apresentou como resultado final o estado *"killed"* no terminal. Como a lista encadeada é uma estrutura que consome mais espaço, o espaço de memória RAM disponível na máquina utilizada para realizar os testes não foi suficiente para alocar todo o espaço necessário.

Como foi estabelecido na especificação do trabalho, os testes feitos para o TAD BIBLIOTECA foram: biblioteca com 100 textos, cada um com tamanhos entre 1 e 100; biblioteca com 100 textos, cada um com tamanhos entre 50.000 e 100.000; biblioteca com 100.000 textos, cada um com tamanhos entre 1 e 100 e por fim, biblioteca com 100.000 textos, cada um com tamanhos entre 50.000 e 100.000. Seguem os resultados das medições de comparações, movimentações e o tempo gasto para ordenar com cada algoritmo de ordenação especificado (SelectSort e QuickSort).

1. **Implementação por array:**

## Comparações

TAD BIBLIOTECA implementado com array

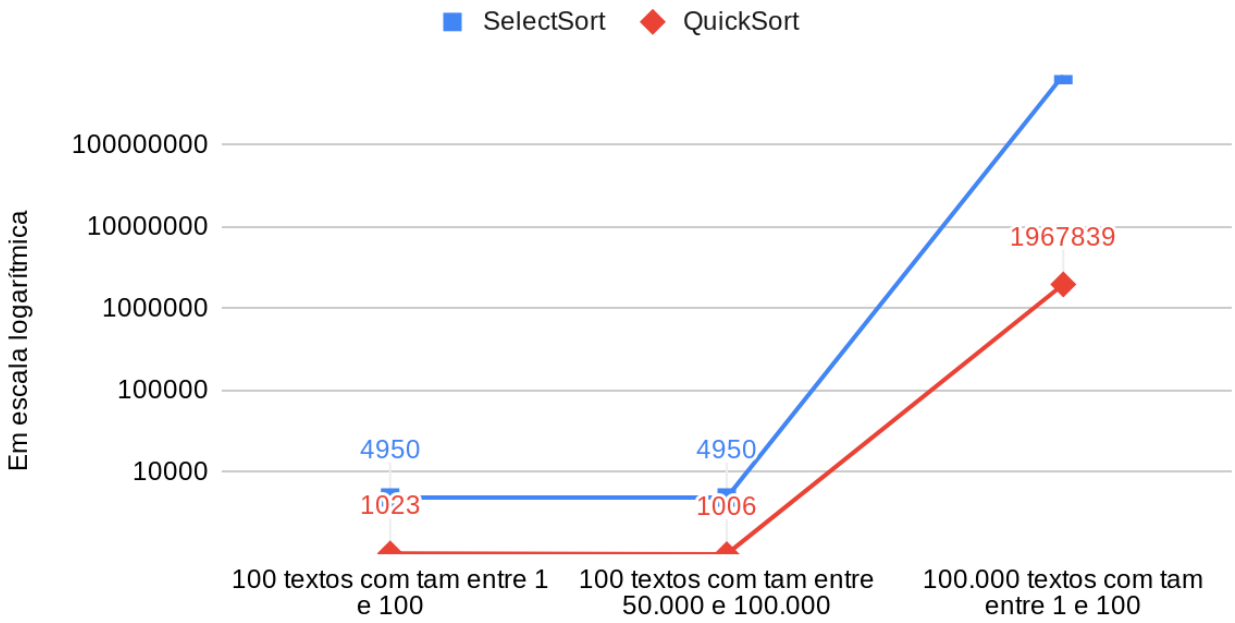


Figura 8. Comparações TAD BIBLIOTECA.

## Movimentações

TAD BIBLIOTECA implementado com array

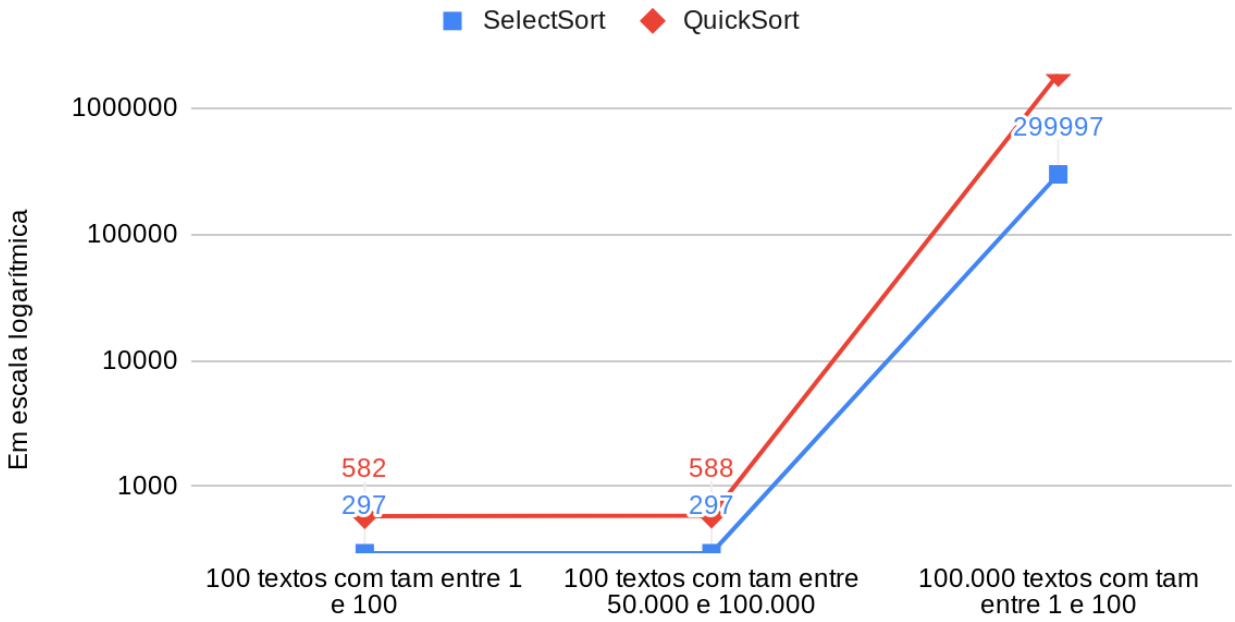


Figura 9. Movimentações TAD BIBLIOTECA.

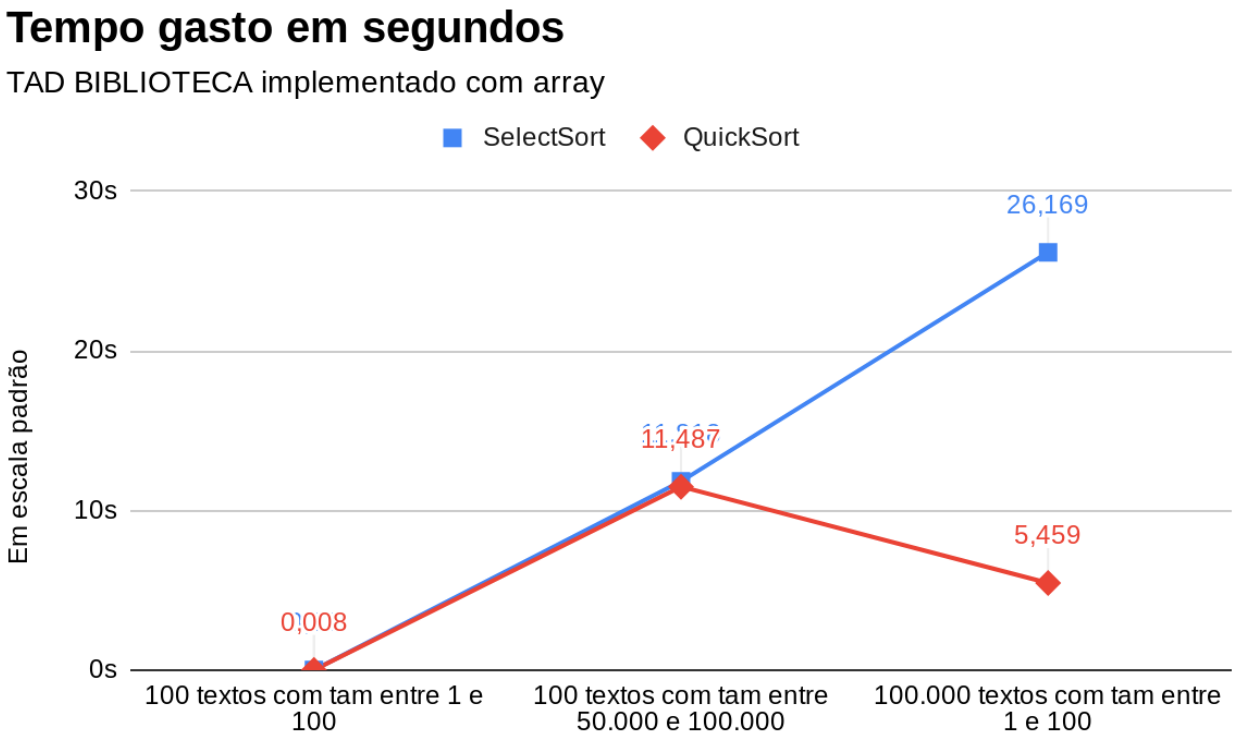


Figura 10. Tempo gasto TAD BIBLIOTECA.

- Biblioteca com 100.000 textos, cada um com tamanhos entre 50.000 e 100.000:

Este caso de teste apresentou como resultado final o estado *"killed"* no terminal. Tendo em vista o tamanho do teste, o espaço de memória RAM disponível na máquina utilizada para realizar os testes não foi suficiente para alocar todo o espaço necessário.

**2. Implementação por lista encadeada:**

## Comparações

TAD BIBLIOTECA implementado com lista encadeada

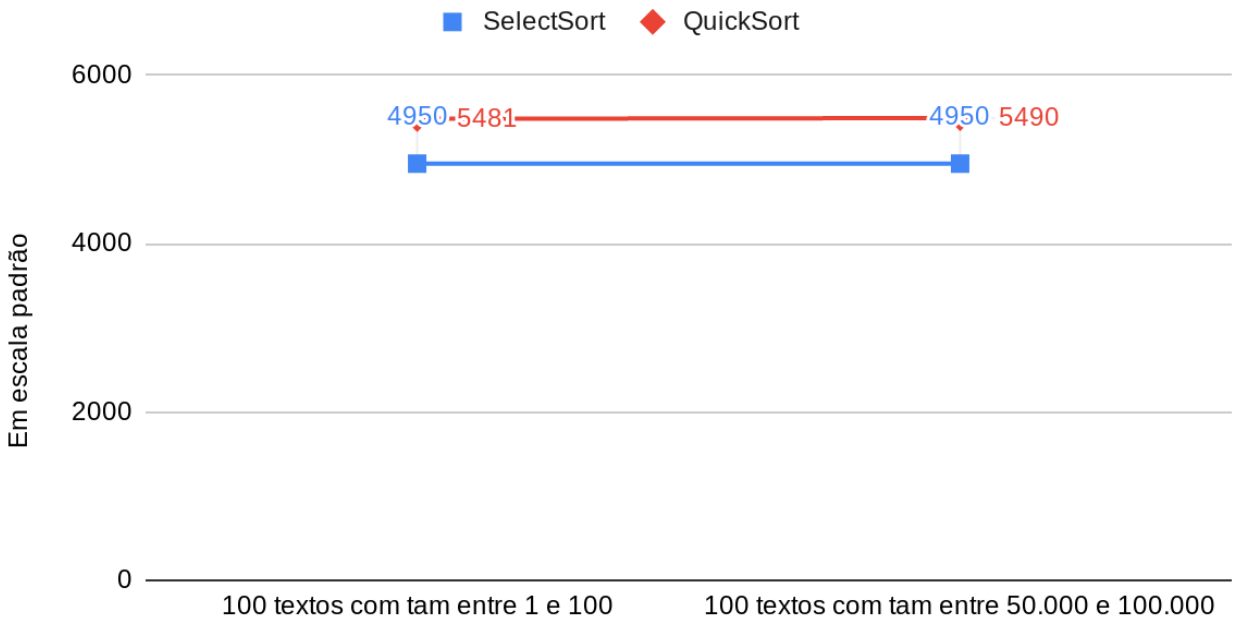


Figura 11. Comparações TAD BIBLIOTECA.

## Movimentações

TAD BIBLIOTECA implementado com lista encadeada

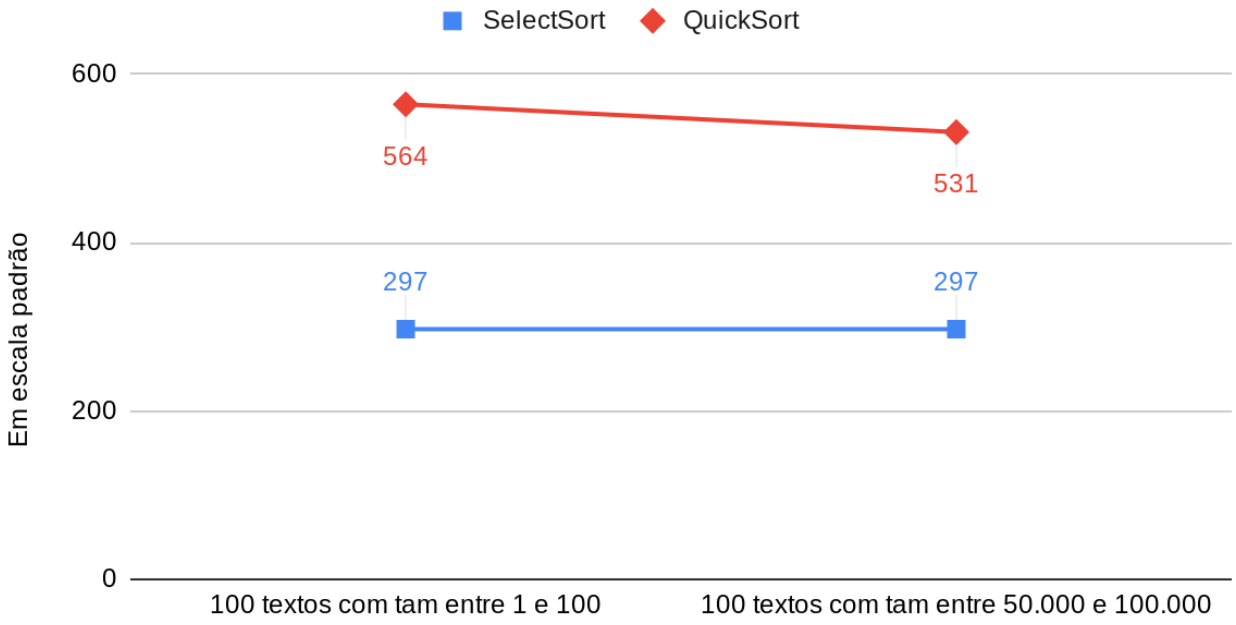


Figura 12. Movimentações TAD BIBLIOTECA.

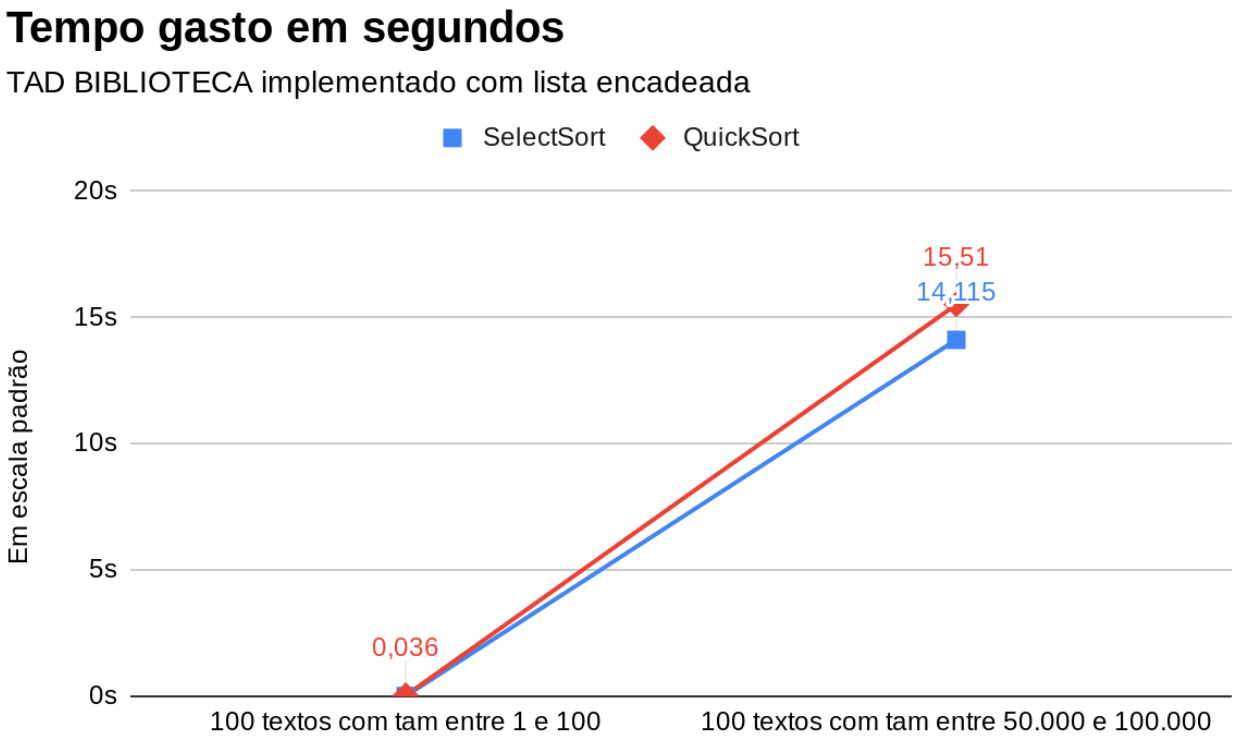


Figura 13. Tempo gasto TAD BIBLIOTECA.

- Biblioteca com 100.000 textos, cada um com tamanhos entre 1 e 100 e biblioteca com 100.000 textos, cada um com tamanhos entre 50.000 e 100.000.

Em ambos os casos teste, o resultado final foi o estado *"killed"* no terminal. Tendo em vista tanto o tamanho dos testes quanto a estrutura utilizada, no caso lista encadeada que consome mais memória, o espaço de memória RAM disponível na máquina utilizada para realizar os testes não foi suficiente para alocar todo o espaço necessário.

# 4. CONSIDERAÇÕES FINAIS

A partir da nálise dos gráficos e do conhecimento prévio acerca dos algoritmos de ordenação SelectSort e QuickSort foi possível perceber uma correlação entre o aumento dos valores de entrada e o tempo gasto para ordenação dela. Além disso, ficou claro que a medida que os valores de entrada aumentam, o algoritmo QuickSort se torna exponencialmente melhor se comparado ao algoritmo SelectSort. No entanto, quando o tamanho dos itens se tornam maiores a vantagem do algoritmo QuickSort com relação ao algoritmo SelectSort se tona menos evidente, uma vez que o algoritmo QuickSort efetua mais trocas do que o algoritmo SelectSort.

Assim, ficou evidente que o algoritmo QuickSort é melhor na grande maioria dos casos, no entanto sua utilização também depende do volume de dados com o qual você lida, além do tamanho dos itens das entradas, o que pode tornar o QuickSort desvantajoso nesses casos. Por fim, não foi possível perceber uma diferença tão grande de tempo quanto ao uso de Array ou Lista Encadeada, no entanto ficou claro que a quantidade de memória gasta pela segunda estrutura de dados é muito maior e, portanto, parece ser mais atraente o uso de Array para execução dos algoritmos de ordenação.