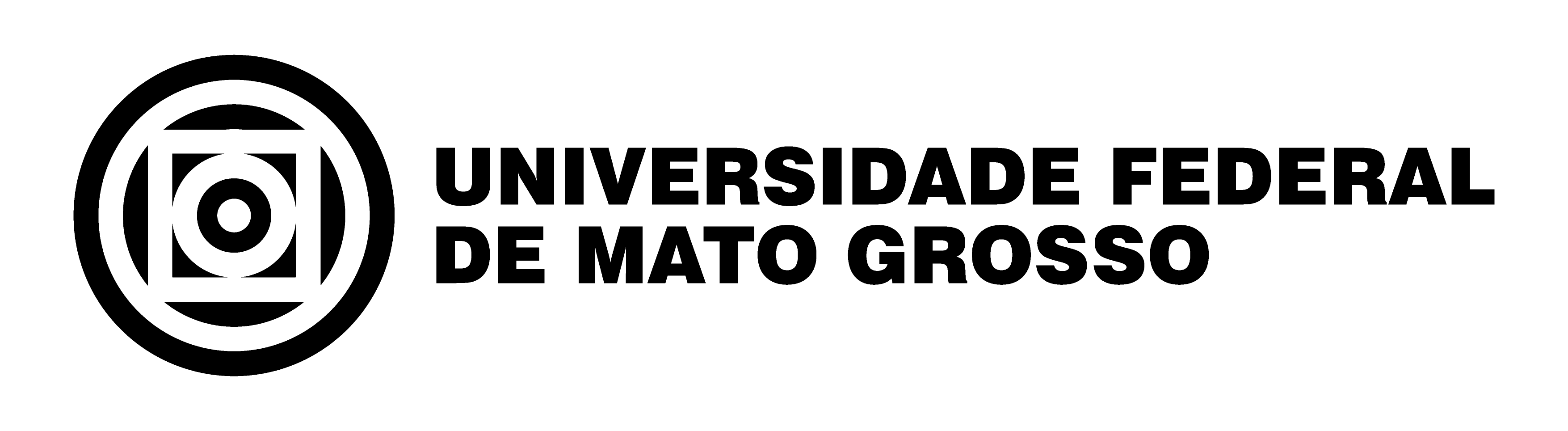
****

**Relatório de Algoritmos e estrutura de dados:**

**Algoritmos de ordenação**

**Discentes:**

**Cesar Augusto de Araújo Filho**

**Leandro Steffens de Oliveira**

**Docente:**

**Frederico Santos de Oliveira**

**Cuiabá**

**2022**

Sumário

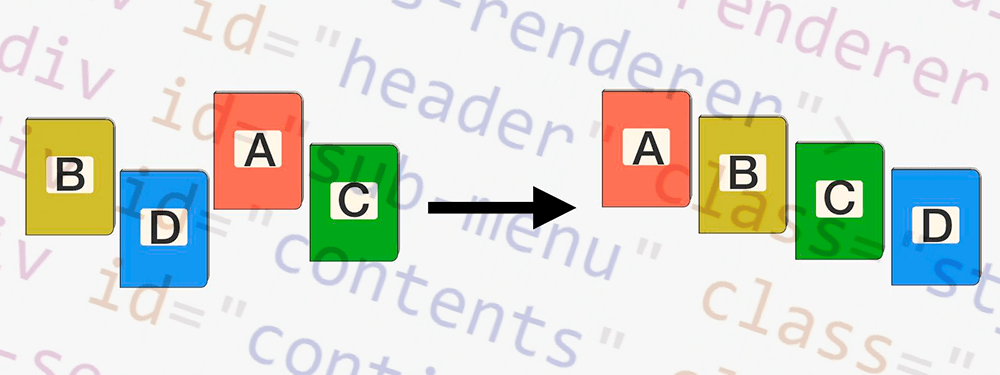
[Introdução 3](#_Toc107160701)

[Procedimentos experimentais 4](#_Toc107160702)

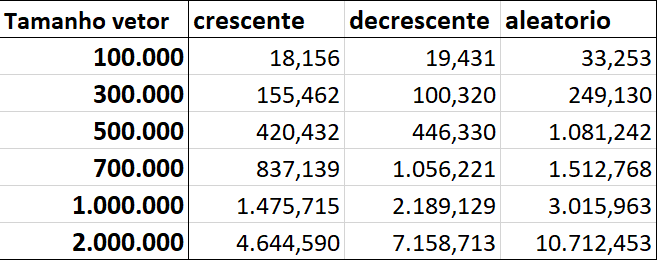
[Análise dos resultados 12](#_Toc107160703)

# Introdução

O objetivo deste trabalho é comparar os algoritmos de ordenação em relação ao tempo de processamento e complexidade. Executando e comparando os algoritmos Bubblesort, Insertionsort, Selectionsort, Shellsort, Mergesort, Heapsort, Quicksort e Binary Insertionsort (escolhido por nós). Após realizarmos as analises propostas pelo professor, obtemos os seguintes resultados.



# Procedimentos experimentais

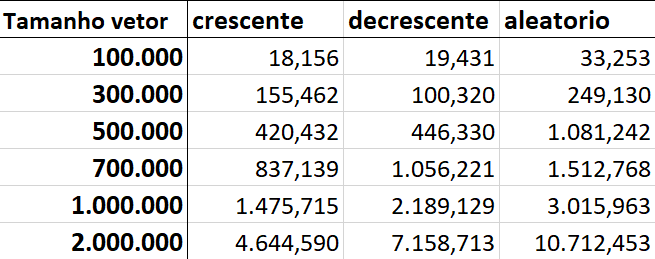
Bubblesort

Casos analisados:

No melhor caso não realiza trocas, no pior caso O(n²) trocas e no caso médio realiza trocas.

Notamos a complexidade do algoritmo de O(n²), pois o gráfico tende a função exponencial.

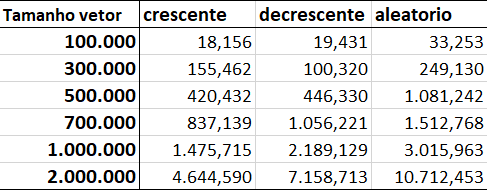
Inseritonsort



Casos analisados:

Em qualquer situação realiza trocas.

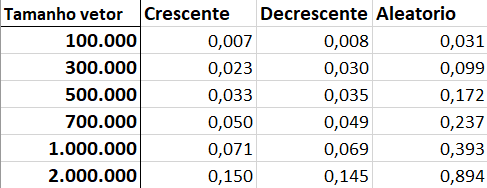
Notamos que o melhor caso dele é muito superior do que seu pior caso ou caso médio, ou seja, ele funciona melhor com vetores crescentes.

Selectionsort

Casos analisados:

No melhor caso realiza trocas, no pior caso realiza trocas e no caso médio realiza trocas.

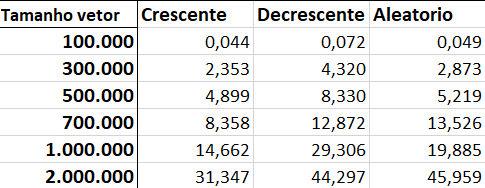
Notamos que esse algoritmo funciona melhor com vetores decrescentes.

Shellsort

Casos analisados:

Não existe complexidade.

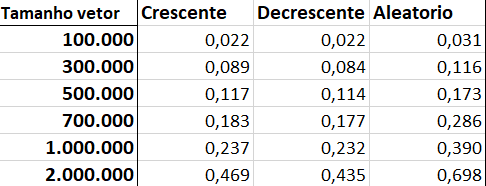
Notamos que ele lida bem com vetores crescentes e decrescente, porém com vetores aleatórios ele tem dificuldade, ele se assimila a uma complexidade O(n log n).

Mergesort

Casos analisados:

Complexidade em todos os casos são O(n log n).

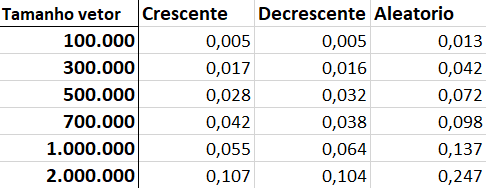
Notamos que ele é bem estável independente do tipo de vetor e é um dos melhores, pois sua complexidade é O(n log n).

Heapsort

Casos analisados:

Complexidade em todos os casos são O(n log n).

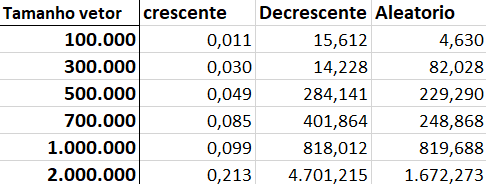
Notamos que ele é bem estável independente do tipo de vetor e é um dos melhores, pois sua complexidade é O(n log n), porém ele executa em menor tempo que a maioria.

Quicksort

Casos analisados:

A complexidade no melhor caso é O(n log n), no pior caso é O(n²) e no caso médio é O(n log n).

Notamos que ele tem certa dificuldade com vetores aleatórios, porém dentre todos ele é o que executa e menor tempo, pois sua complexidade é O(n log n).

Binary Insertionsort

Casos analisados:

A complexidade em todos casos são O(n²).

Notamos que ele é bom com vetores crescentes e pequenos, mas ao todo ele é um dos piores, pois sua complexidade é de O(n²).

# Análise dos resultados

Vemos no gráfico abaixo a comparação de todos os algoritmos pelo tempo de execução:

Notamos que há uma grande diferença entre eles, já que alguns tem complexidade de O(n²) e outros O(n log n). Ao separar os melhores algoritmos “O(n log n)”, obtemos o seguinte gráfico:

Analisando os resultados acima, identificamos que o QuickSort é o algoritmo com o menor tempo de execução, assim sendo, o melhor algoritmo para ordenação de vetores.