Algoritmos de ordenação

Introdução

Existem diversos algoritmos de ordenação, e dependendo da aplicação cada algoritmo possui alguma vantagem sobre os outros. O objetivo é compreender que existem várias formas de resolver uma mesma tarefa com um desses algoritmos, sendo que alguns são mais eficientes em certos casos. A finalidade deste trabalho é apresentar qual a escolha do método de ordenação mais eficiente para cada caso, quando se tem diferentes tipos de conjuntos de dados a serem ordenados.

Método aplicado

5 tamanhos de vetor para comparar 8 algoritmos (Insertion Sort, Selection Sort, Comb Sort, Bubble Sort, Merge Sort, Quick Sort, Merge Sort Externo e HeapSort)

- Vetores com elementos distintos e distribuídos de forma aleatória
- Vetores com elementos distintos já ordenados
- Vetores com elementos distintos ordenados de forma reversa
- Vetores onde os elementos estão distribuídos de forma aleatória, porém, cada elemento aparece ao menos em quantidade igual ou superior a 5% do tamanho do vetor

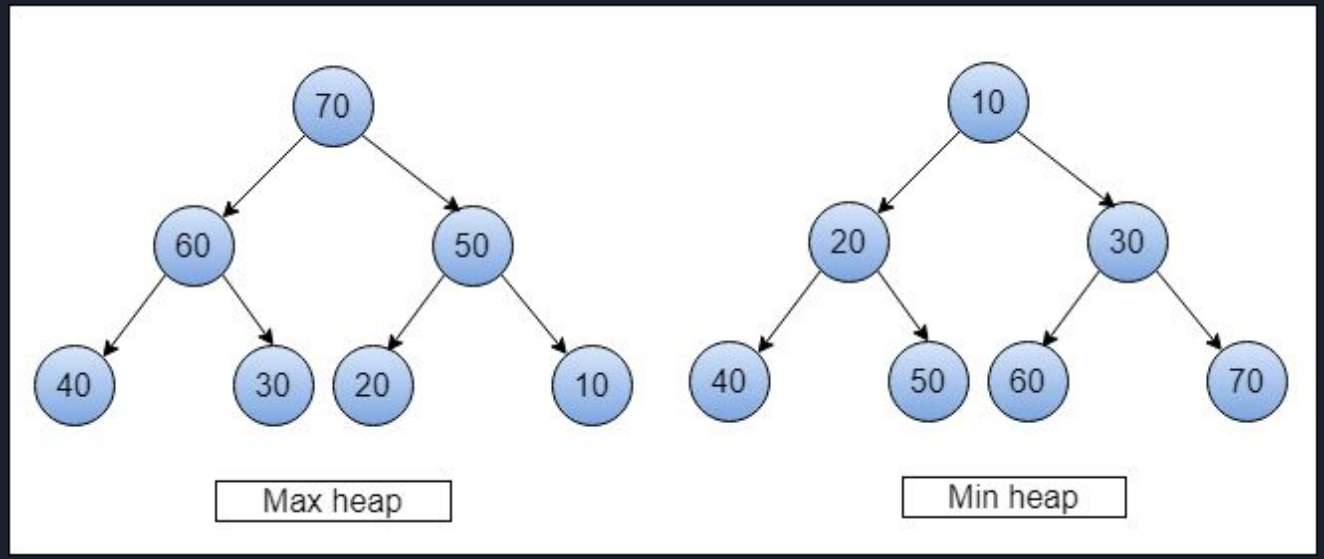
Os tamanhos escolhidos para o vetor foram: 50 mil, 80 mil, 120 mil, 180 mil e 270 mil elementos.

Índice

1. Heapsort e Merge Sort Externo (funcionamento, complexidade e estabilidade)
2. Descrição do ambiente de teste
3. Resultados numéricos dos experimentos (gráficos das tabelas)
4. Discussão
5. Conclusão

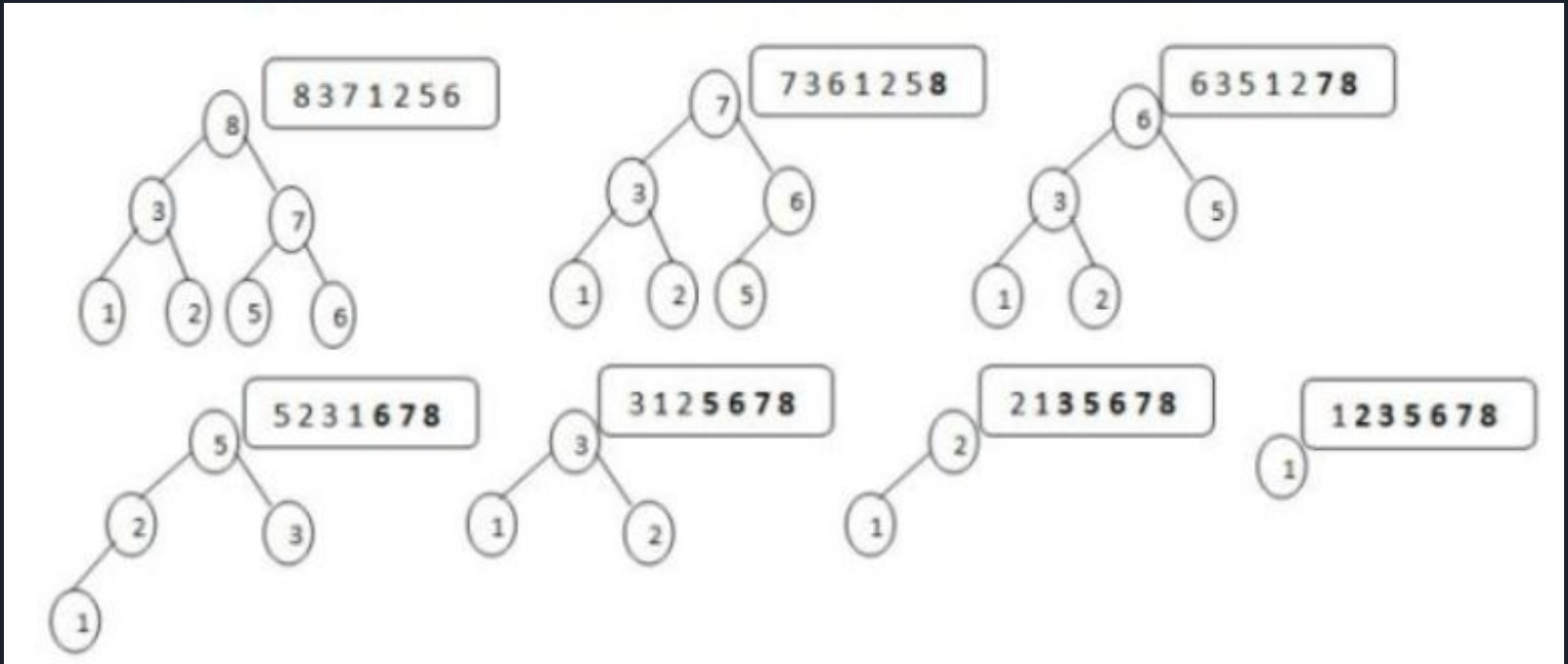
Heapsort

Técnica baseada em comparação com base na estrutura de dados Binary Heap. É semelhante ao Selection Sort, onde primeiro encontramos o elemento mínimo e colocamos o elemento mínimo no início. Repetimos o mesmo processo para os elementos restantes.



Heap Sort

Vamos ordenar o seguinte vetor em ordem crescente: **2 3 7 1 8 5 6**



Heap Sort

Complexidade: O (nLogn) para melhor, pior e médio caso.

Estabilidade: Algoritmo instável.

Porém, adaptando sua estrutura pode se tornar estável. Cada elemento da estrutura adaptada deve ficar no formato de um par (elemento original, índice original). Assim, caso dois elementos sejam iguais, o desempate ocorrerá pelo índice na estrutura original.

Merge Sort Externo

O que é o Merge Sort?

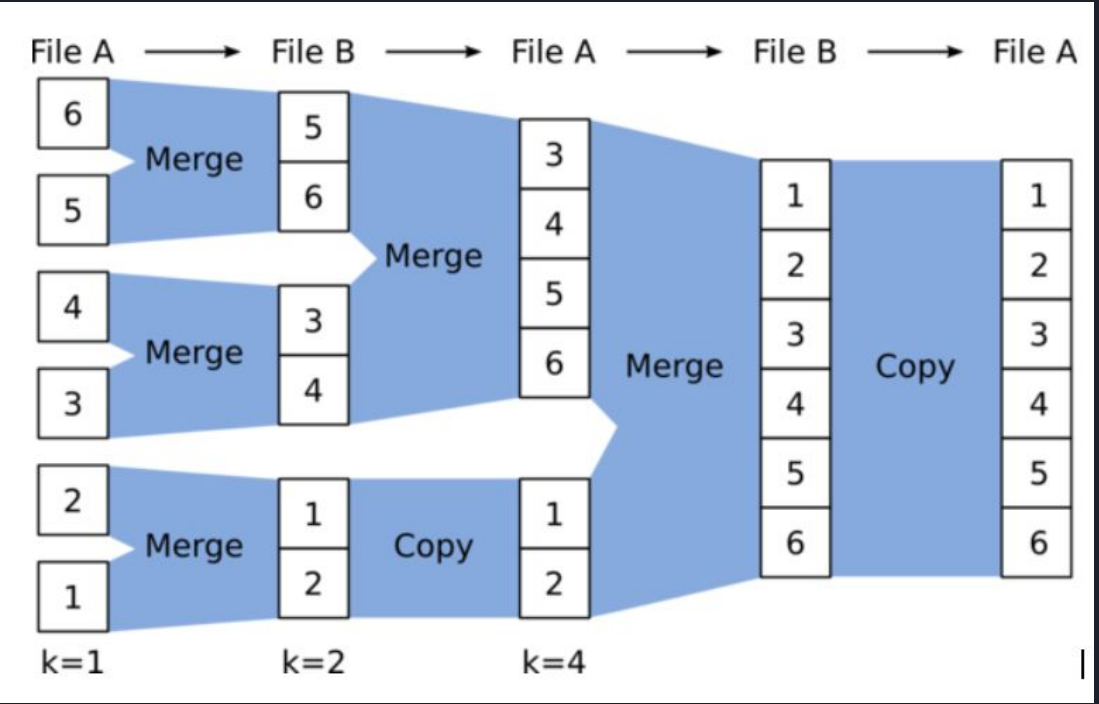
- É um dos algoritmos de classificação que podem lidar com grandes quantidades de dados.

Complexidade:

- Possui complexidade: $O\left(\frac{N}{B}\log_{\frac{M}{B}}\frac{N}{B}\right)$.

Estabilidade: Algoritmo estável.

Merge Sort Externo



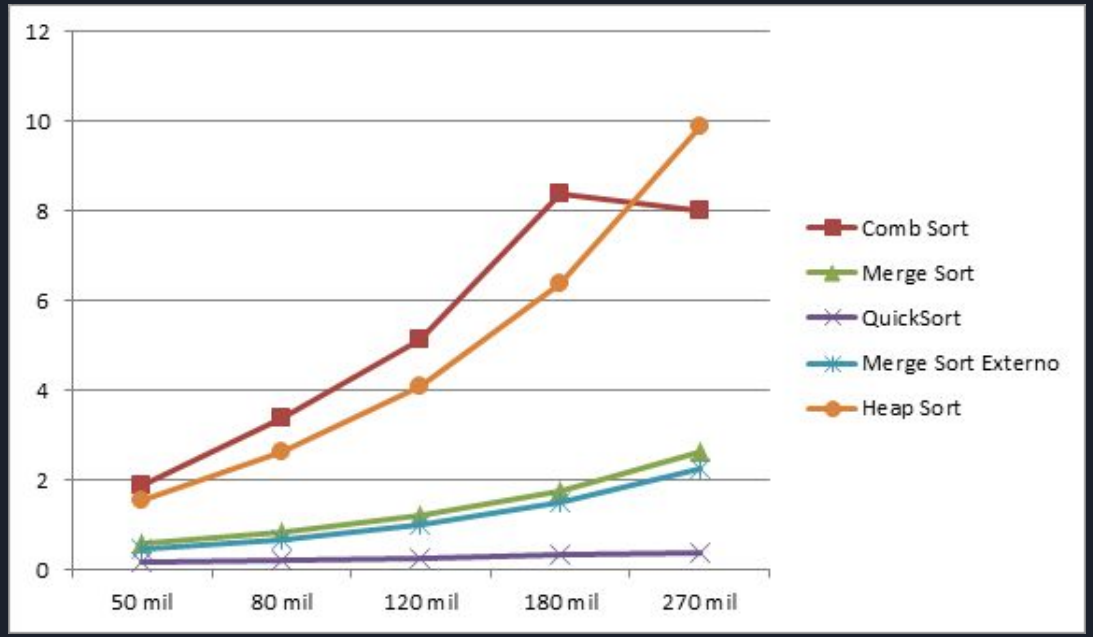
Descrição do ambiente de teste

- Windows 10 Enterprise
- Intel core i7 (7ª geração)
- 16GB RAM
- SSD 256MB
- Compilador GCC (Code Blocks)

Resultados gráficos dos experimentos

Relação entre Comparações x Tamanho do Vetor

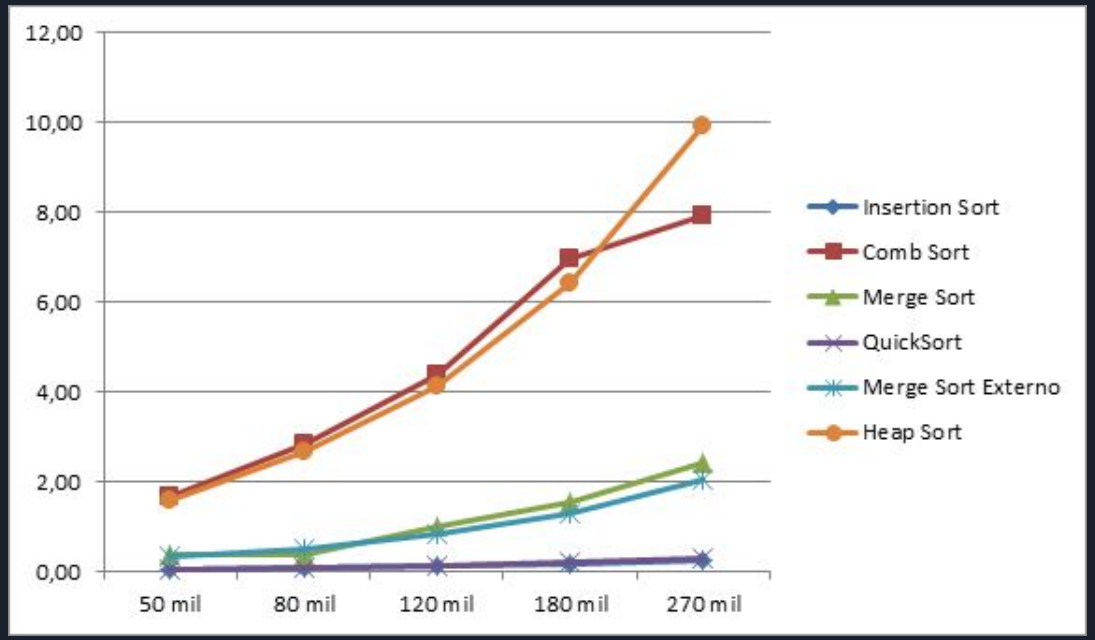
Vetor Aleatório



Resultados gráficos dos experimentos

Relação entre Comparações x Tamanho do Vetor

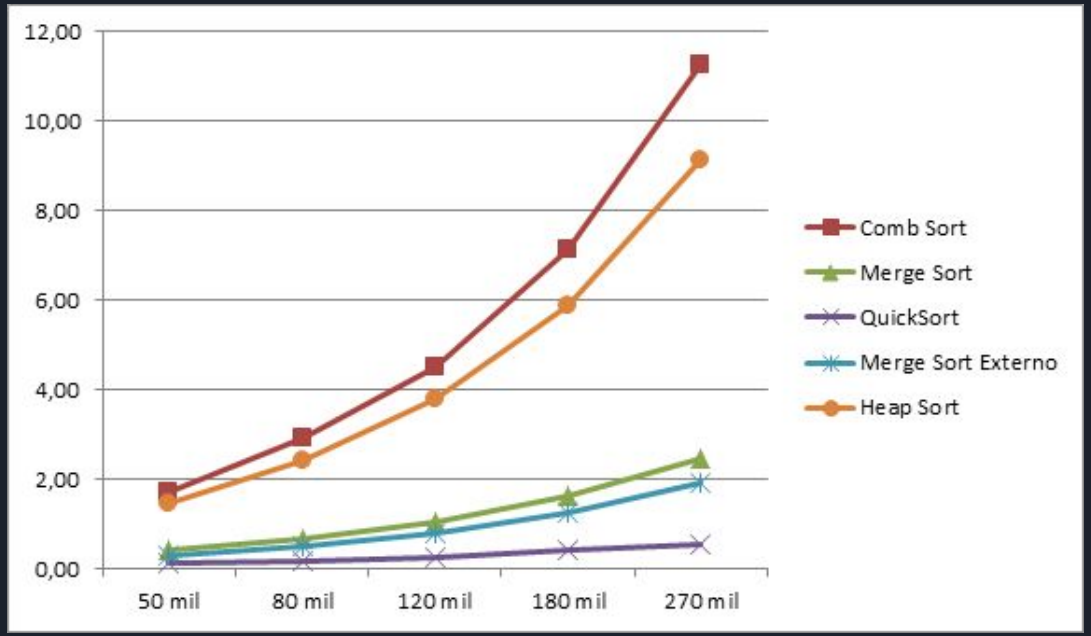
Vetor Ordenado



Resultados gráficos dos experimentos

Relação entre Comparações x Tamanho do Vetor

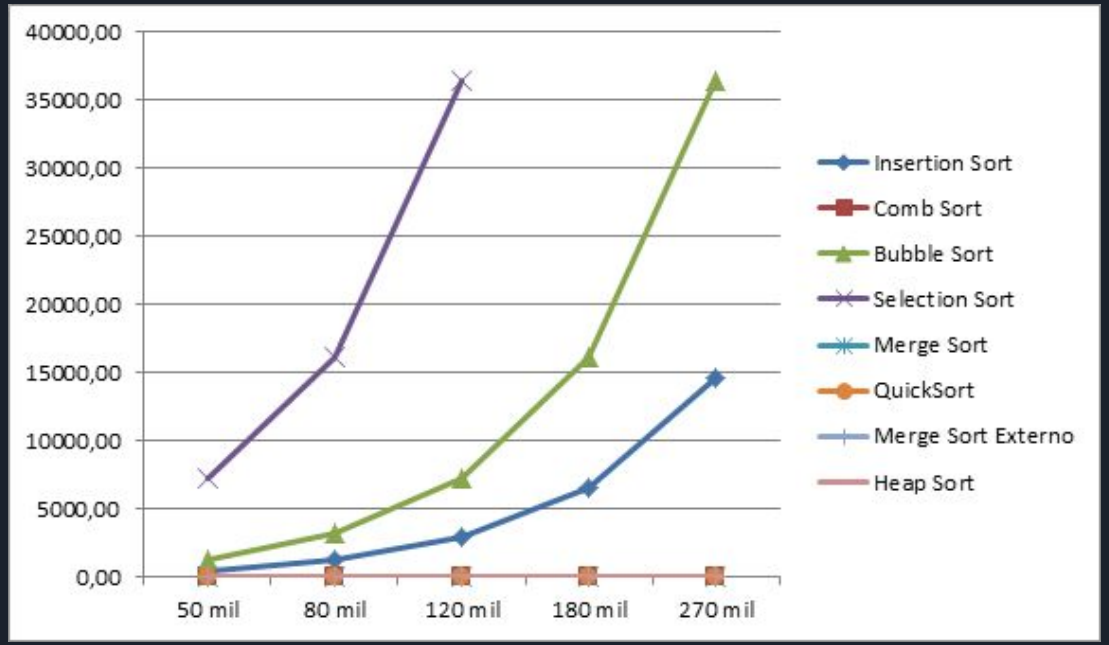
Vetor Ordenado Reverso



Resultados gráficos dos experimentos

Relação entre Comparações x Tamanho do Vetor

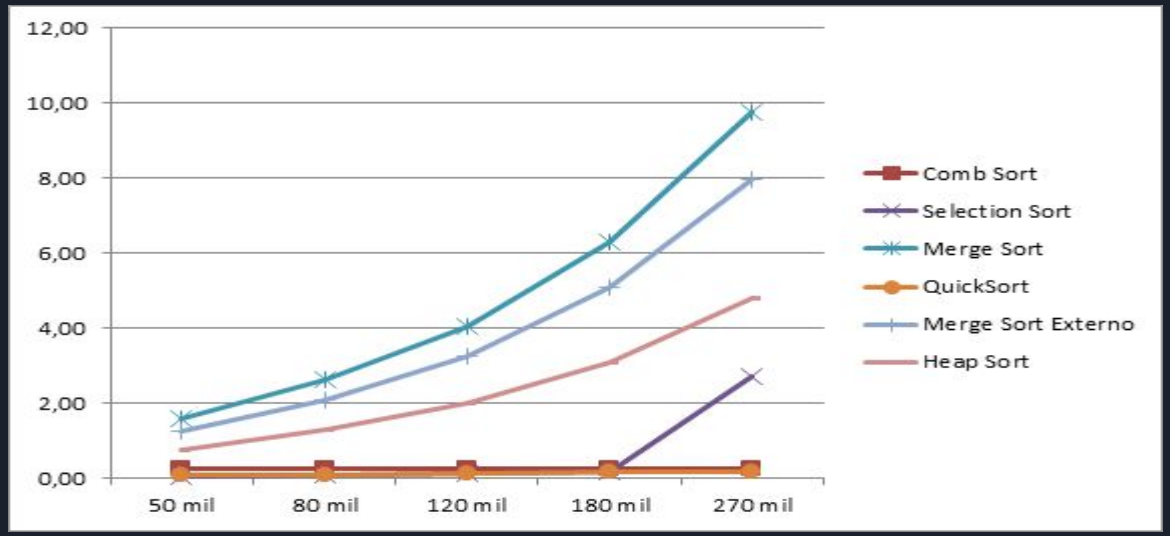
Vetor com valores distribuídos aleatoriamente, e cada elemento com qtd >= 5% do tamanho



Resultados gráficos dos experimentos

Relação entre Trocas x Tamanho do Vetor

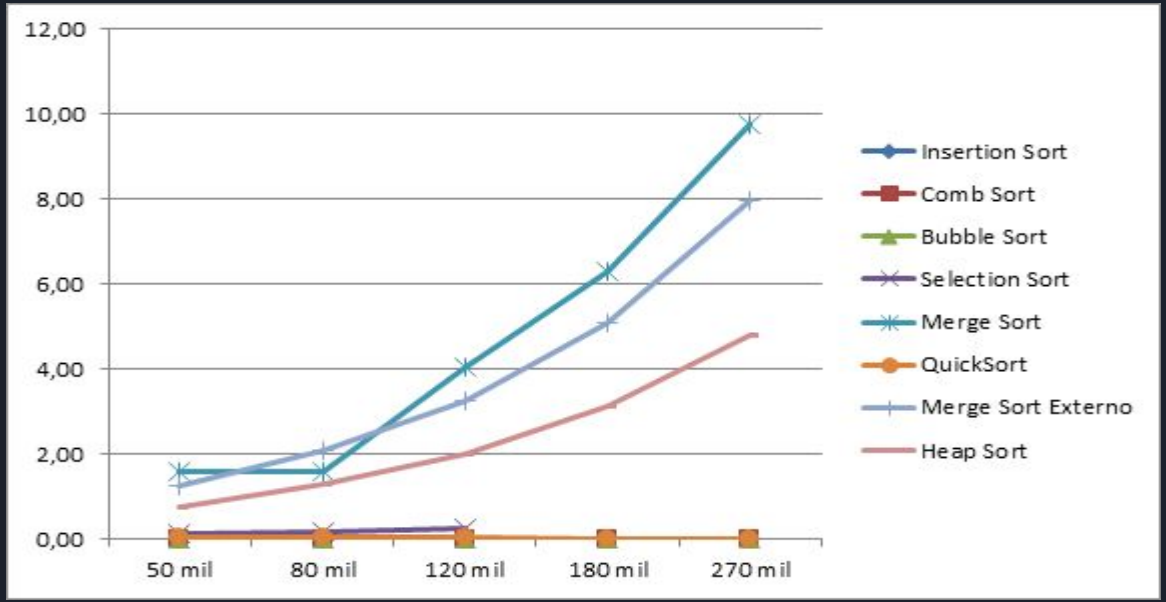
Vetor Aleatório



Resultados gráficos dos experimentos

Relação entre Trocas x Tamanho do Vetor

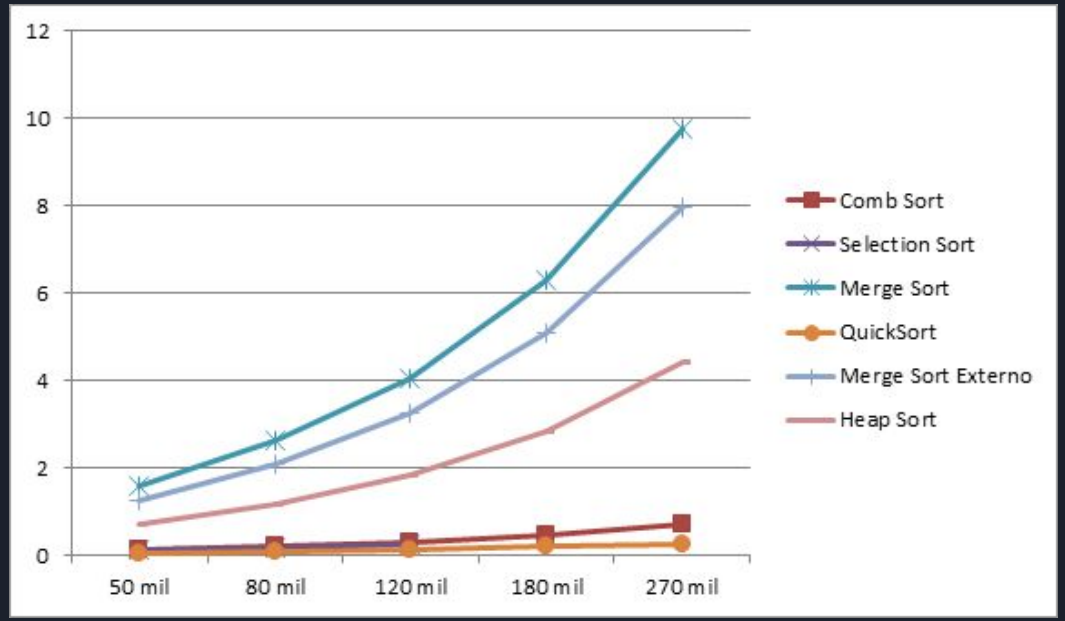
Vetor Ordenado



Resultados gráficos dos experimentos

Relação entre Trocas x Tamanho do Vetor

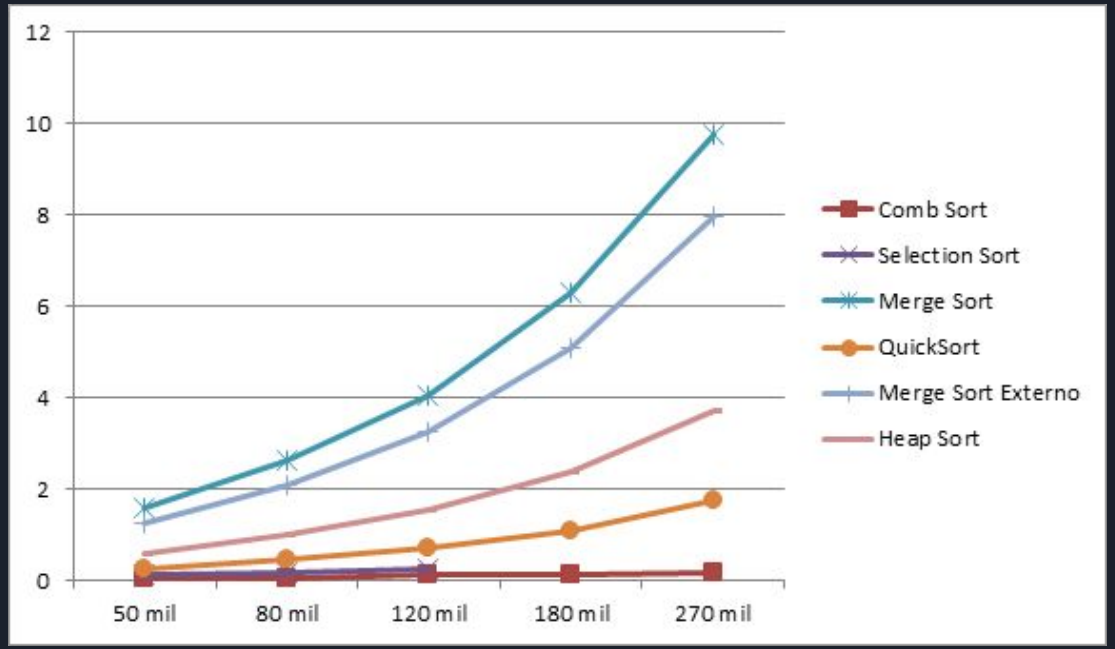
Vetor Ordenado de Forma Reversa



Resultados gráficos dos experimentos

Relação entre Trocas x Tamanho do Vetor

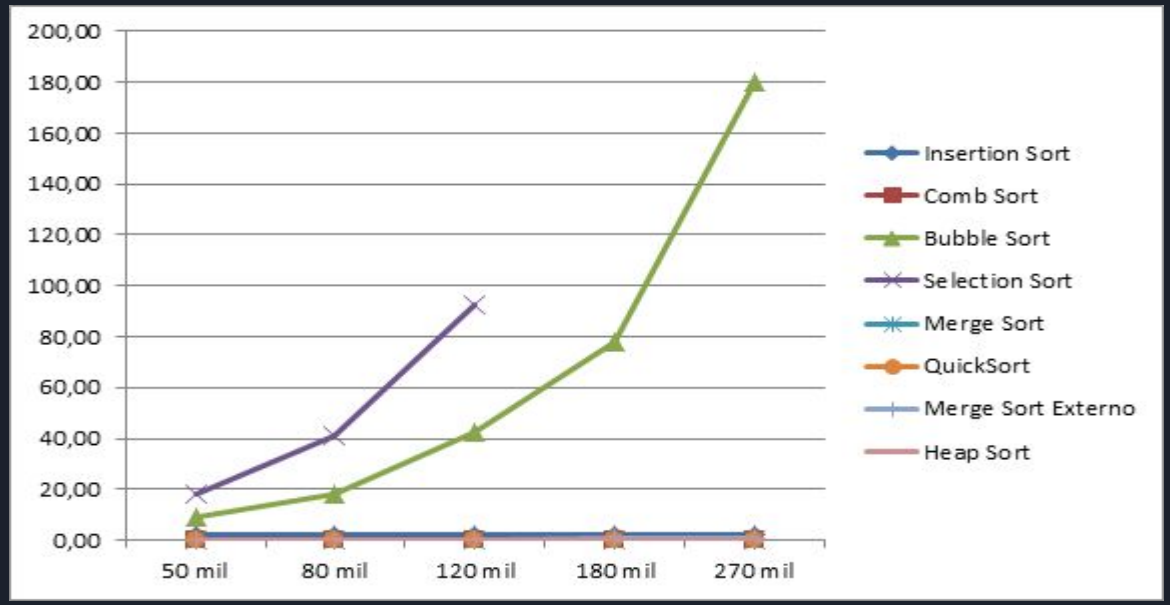
Vetor com valores distribuídos aleatoriamente, e cada elemento com qtd >= 5% do tamanho



Resultados gráficos dos experimentos

Relação entre Tempo x Tamanho do Vetor

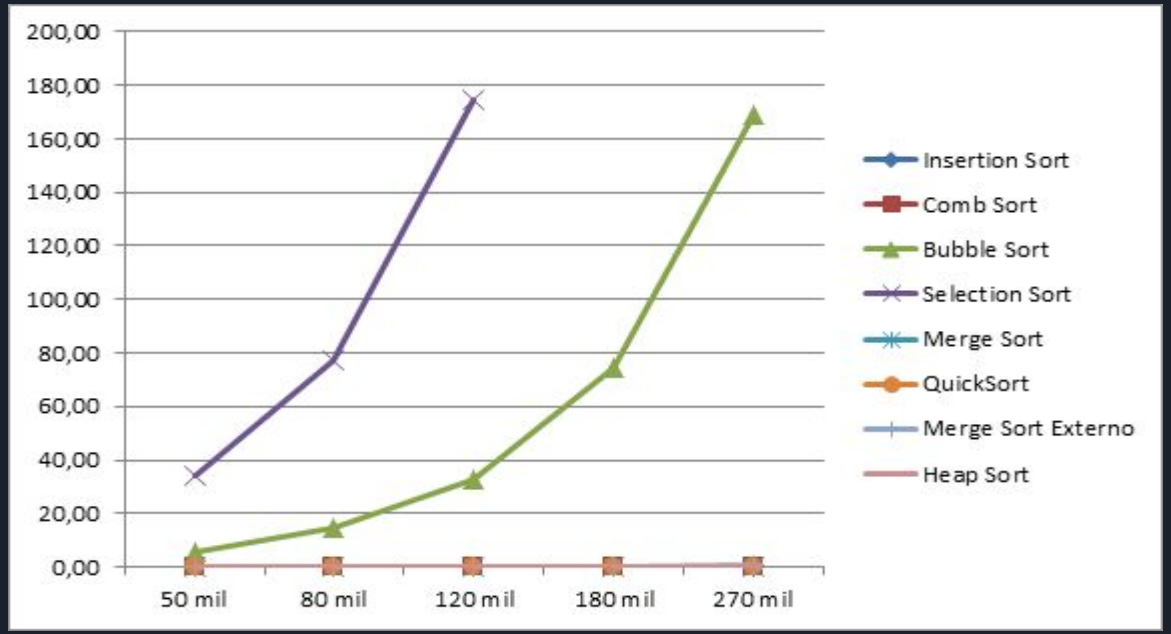
Vetor Aleatório



Resultados gráficos dos experimentos

Relação entre Tempo x Tamanho do Vetor

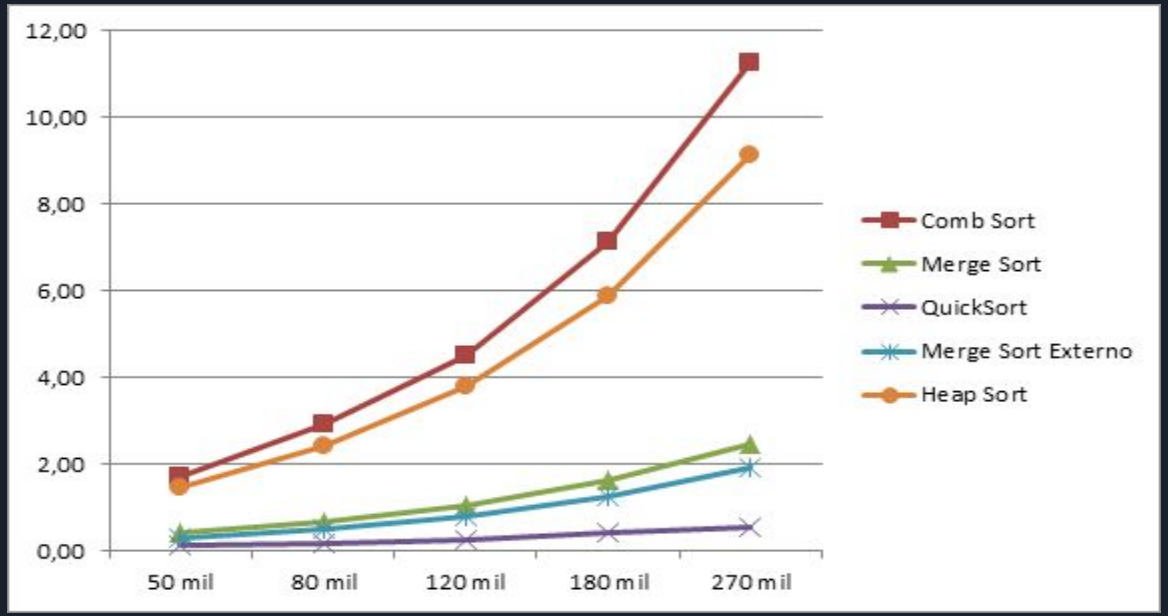
Vetor Ordenado



Resultados gráficos dos experimentos

Relação entre Tempo x Tamanho do Vetor

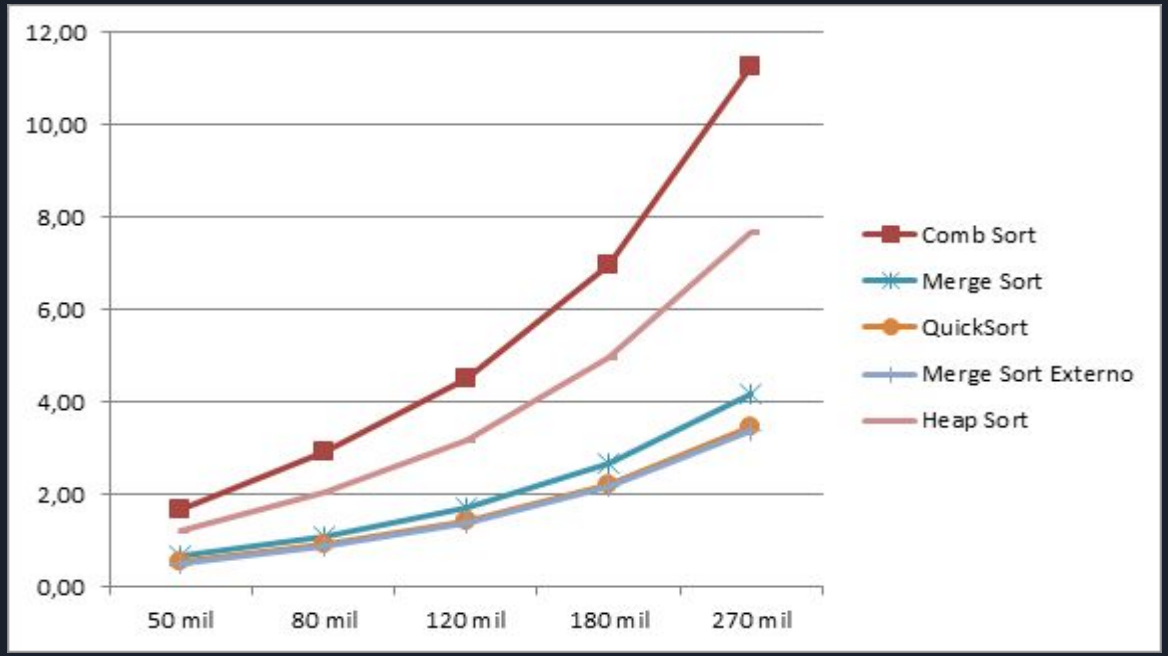
Vetor Ordenado de Forma Reversa



Resultados gráficos dos experimentos

Relação entre Tempo x Tamanho do Vetor

Vetor com valores distribuídos aleatoriamente, e cada elemento com qtd >= 5% do tamanho



Discussão sobre os resultados obtidos

Quantidade de comparações

1° Caso -> Vetores com elementos distintos e aleatórios.

QuickSort faz **menos** comparações
Bubble e Selection Sort fazem **mais** comparações

2°Caso -> Vetores com elementos ordenados:

Insertion Sort faz **menos** comparações
Comb, Bubble, Selection Sort fazem **mais** comparações.

3° Caso -> Vetores com elementos ordenados de forma reversa

QuickSort faz **menos** comparações
Bubble, Selection e Insertion Sort fazem **mais** comparações

4°Caso -> Vetores com elementos aleatório (cada elemento com qtd >= 5% do tamanho)

MergeSort Externo, faz **menos** comparações
Bubble, Selection e Insertion Sort fazem **mais** comparações

Discussão sobre os resultados obtidos

Quantidade de trocas

1º Caso -> Vetores com elementos distintos e aleatórios.

QuickSort faz **menos** trocas
Bubble e Insertion Sort fazem **mais** trocas

2ºCaso -> Vetores com elementos ordenados:

MergeSort Externo faz **menos** trocas
Comb, Bubble, Insertion Sort fazem **mais** trocas

3º Caso -> Vetores com elementos ordenados de forma reversa

QuickSort faz **menos** trocas
Bubble e Insertion Sort fazem **mais** trocas

4ºCaso -> Vetores com elementos aleatório (cada elemento com qtd >= 5% do tamanho)

Combo Sort faz **menos** trocas
Bubble e Insertion Sort fazem **mais** trocas

Discussão sobre os resultados obtidos

Tempo de execução

1° Caso -> Vetores com elementos distintos e aleatórios.

QuickSort leva **menos** tempo
Bubble Sort leva **mais** tempo

2°Caso -> Vetores com elementos ordenados:

QuickSort leva **menos** tempo
Selection Sort leva **mais** tempo

3° Caso -> Vetores com elementos ordenados de forma reversa

QuickSort leva **menos** tempo
Insertion Sort leva **mais** tempo

4°Caso -> Vetores com elementos aleatório (cada elemento com qtd >= 5% do tamanho)

QuickSort leva **menos** tempo
Bubble Sort leva **mais** tempo

Conclusão

- O algoritmo mais rápido é o Quick Sort e é usado para ordenar qualquer tipo de vetor de maneira eficiente.
- Os algoritmos Bubble Sort, Insertion Sort e Selection Sort são opções inviáveis para ordenação em larga escala, são algoritmos de fácil implementação e são usados somente em aplicações mais simples. O Comb Sort se demonstrou mais eficiente que o Bubble Sort, sendo rápido para diversos tamanhos.
- Merge Sort é o algoritmo mais estável e é usado em aplicações que necessitam de um tempo de execução fixo.
- Os Algoritmos Heap Sort e Merge Sort Externo são algoritmos rápidos porém possuem uma complexidade maior na hora da implementação, e como o Quick Sort se mostra, na maioria dos casos, mais rápido que estes dois é mais viável utilizá-lo, com exceção a arquivos muitos grandes para caber na memória, podendo usar o Merge Sort Externo.

Equipe:

Lucas Eduardo Barbosa de Siqueira
Matheus de Souza
Rodrigo de Lima Dias
Rodrigo Duarte Silva Luz
Yasmin Karolyne Aniceto Carvalho

Muito obrigado!