Trabalho de Projeto e Análise de Algoritmos
(Selection-Sort e Merge-Sort)

- O que é um Algoritmo de ordenação?

Um algoritmo de ordenação é um conjunto de instruções ou regras lógicas utilizadas para organizar um conjunto de elementos em uma determinada ordem. Ele permite que os dados sejam colocados em uma sequência específica, como em ordem crescente ou decrescente, facilitando a busca, a comparação e a manipulação desses dados. Existem vários algoritmos de ordenação, cada um com suas próprias características e eficiência. Mas no relatório serão abordados dois que são:

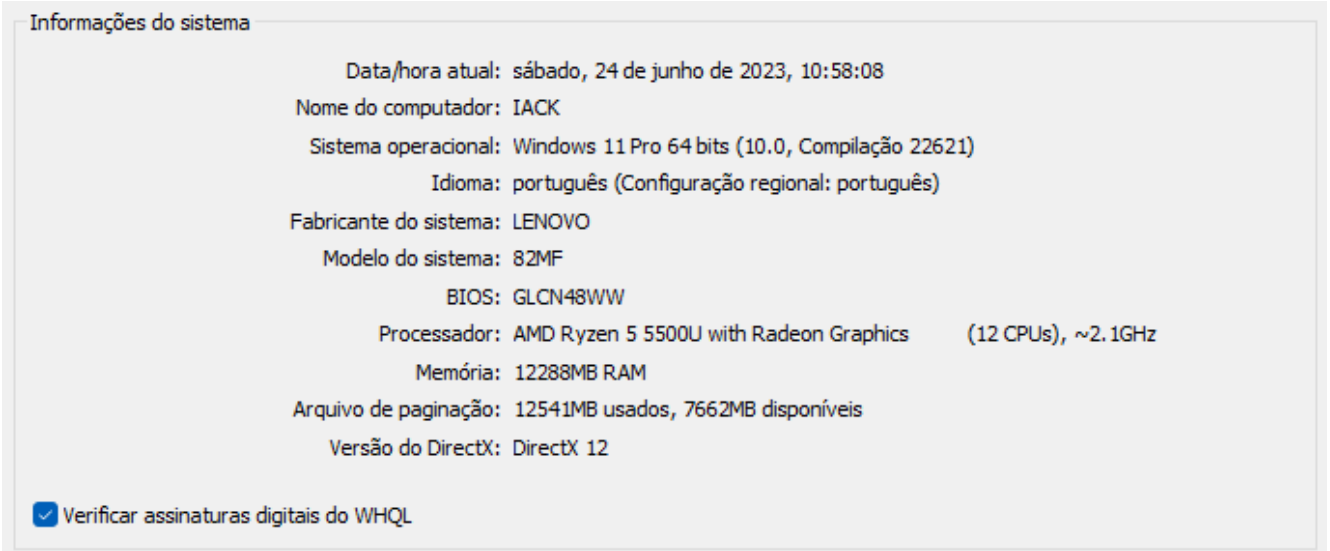
- **Selection-Sort (Ordenação por Seleção):** Nesse algoritmo, o menor elemento é selecionado e colocado na primeira posição, em seguida o segundo menor elemento é selecionado e colocado na segunda posição, e assim por diante. O processo continua até que a lista esteja completamente ordenada.

- **Merge-Sort (Ordenação por Junção):** Este algoritmo divide a lista em duas metades, ordena cada metade separadamente e, em seguida, combina as duas metades ordenadas para obter a lista final ordenada. Esse processo de divisão e combinação é repetido até que a lista esteja completamente ordenada.

- Linguagem Utilizada:

Nos programas, foram utilizados a linguagem computacional python, pois ele tem uma grande variedade de bibliotecas e a sintaxe é clara (simples e legível).

- Configuração do Computador:



Resultados do Selection-Sort

- **Gráficos do Programa Selection-Sort:**

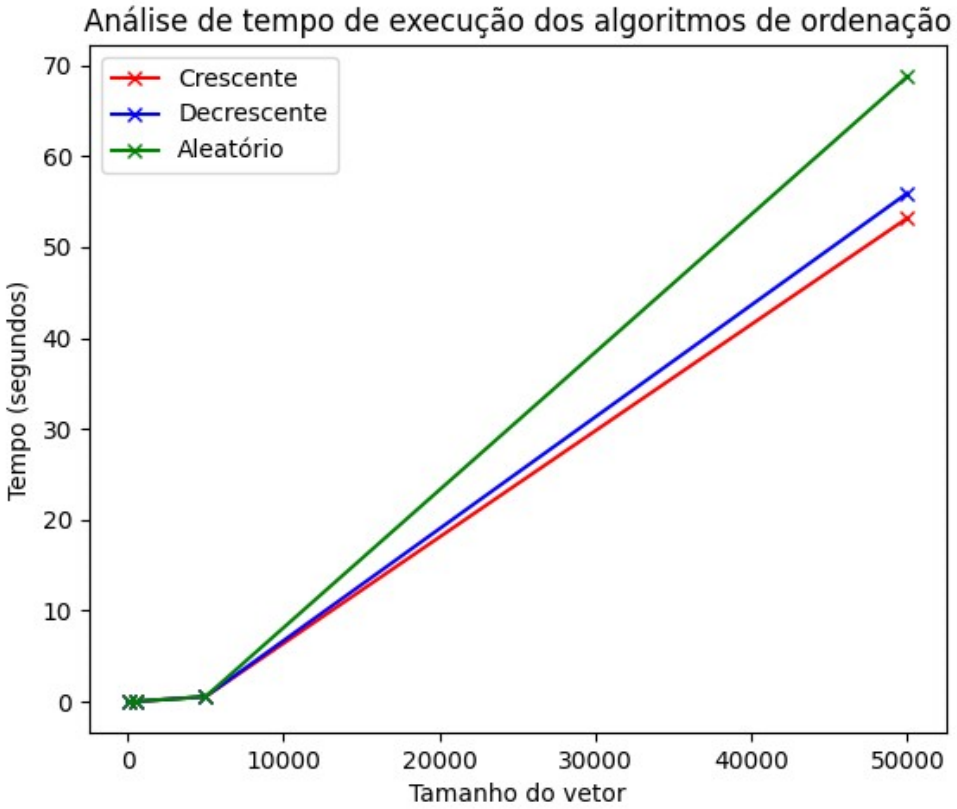


Figura 1 (Gráfico Selection-Sort)

Selection-Sort

Ordem	Tamanho dos vetores	Tempo de execução em seg.
Crescente	50	0,00006
Decrescente	50	0,000056
Aleatorio	50	0,000055
Crescente	500	0,005207
Decrescente	500	0,005498
Aleatorio	500	0,005318
Crescente	5000	0,531141
Decrescente	5000	0,534374
Aleatorio	5000	0,526208
Crescente	50000	53,209264
Decrescente	50000	55,951206
Aleatorio	50000	68,786323

Figura 2 (Tamanhos x Tempos Selection-Sort)

Observa-se que o tempo de execução na Figura 2 para vetores de tamanho 50, nas seguintes ordens: crescente, decrescente e aleatória, apresentam os seguintes resultados: 0.000060 segundos, 0.000056 segundos e 0.000055 segundos, respectivamente. Pode-se inferir que as variações de tempo de execução entre essas ordens são mínimas, entretanto, é notável que a ordem crescente demanda um tempo ligeiramente maior em comparação com as demais.

Ao analisarmos vetores de tamanho 500 na Figura 2, observa-se um tempo de execução de 0.005207 segundos para a ordem crescente, 0.005498 segundos para a ordem decrescente e 0.005318 segundos para a ordem aleatória. Conclui-se que as diferenças de tempo de execução ainda são insignificantes para esse tamanho de vetor, sendo a ordem decrescente a que demanda mais tempo em comparação com as outras ordens.

Considerando vetores de tamanho 5000 na Figura 2, o tempo de execução é de 0.531141 segundos para a ordem crescente, 0.534374 segundos para a ordem decrescente e 0.526208 segundos para a ordem aleatória. Observa-se que as diferenças de tempo de execução entre as ordens ainda são relativamente pequenas, sendo a ordem decrescente a que apresenta maior tempo de execução.

Na última análise, com vetores de tamanho 50000, o tempo de execução é de 53.209264 segundos para a ordem crescente, 55.951206 segundos para a ordem decrescente e 68.786323 segundos para a ordem aleatória. Nesta análise, nota-se que na Figura 1 as diferenças de tempo de execução aumentaram um pouco, sendo a ordem aleatória a que demanda mais tempo em comparação com as outras ordens.

Após considerar todas as análises de ordem, tamanho e tempo, conclui-se que as diferenças de tempo de execução entre vetores em ordem crescente, decrescente e aleatória tendem a ser relativamente pequenas. No entanto, é importante ressaltar que tais diferenças podem se tornar mais perceptíveis em vetores de tamanhos muito grandes.

Resultados do Merge-Sort

- **Gráficos do Programa Merge-Sort:**

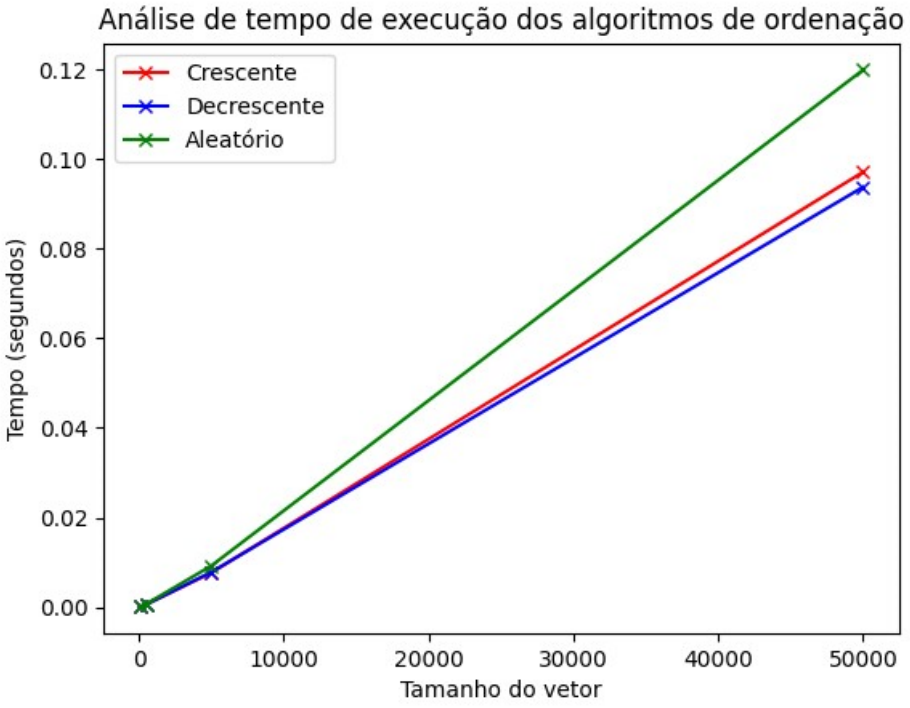


Figura 3 (Gráfico Merge-Sort)

Merge-Sort

Ordem	Tamanho dos vetores	Tempo de execução em seg.
Crescente	50	0.000103
Decrescente	50	0.000070
Aleatorio	50	0.000085
Crescente	500	0.000595
Decrescente	500	0.000574
Aleatorio	500	0.000665
Crescente	5000	0.007668
Decrescente	5000	0.007790
Aleatorio	5000	0.009210
Crescente	50000	0.097112
Decrescente	50000	0.093866
Aleatorio	50000	0.119910

Figura 4 (Tamanhos x Tempos Merge-Sort)

Observa-se que o tempo de execução na Figura 4 para vetores de tamanho 50, nas seguintes ordens: crescente, decrescente e aleatória, apresentam os seguintes resultados: 0.000103 segundos na ordem crescente, 0.000070 segundos na ordem decrescente e 0.000085 segundos na ordem aleatória, respectivamente. Verifica-se que a diferença de tempo de execução entre as ordens não é insignificante, sendo a ordem crescente a que demanda mais tempo.

Ao analisar vetores de tamanho 500, observa-se na Figura 4 um tempo de execução de 0.000595 segundos para a ordem crescente, 0.000574 segundos para a ordem decrescente e 0.000665 segundos para a ordem aleatória. Conclui-se que as diferenças de tempo de execução diminuíram um pouco em relação aos vetores de tamanho 50, sendo a ordem aleatória a que apresenta maior tempo de execução.

Considerando vetores de tamanho 5000 conforme apresentado na Figura 4, o tempo de execução é de 0.007668 segundos para a ordem crescente, 0.007790 segundos para a ordem decrescente e 0.009210 segundos para a ordem aleatória. Observa-se que as diferenças de tempo de execução entre as ordens crescente e decrescente são relativamente pequenas, porém a ordem aleatória apresenta uma diferença mais significativa em relação ao tempo de execução, sendo a que demanda mais tempo.

Na última análise, com vetores de tamanho 50000, na Figura 3 o tempo de execução é de 0.097112 segundos para a ordem crescente, 0.093866 segundos para a ordem decrescente e 0.119910 segundos para a ordem aleatória. Nesta análise, nota-se que as diferenças de tempo de execução entre as ordens crescente e decrescente são pequenas, enquanto a ordem aleatória continua apresentando uma diferença relativamente maior, sendo a que demanda mais tempo em comparação com as outras ordens.

Após considerar todas as análises de ordem, tamanho e tempo, podemos concluir que as diferenças de tempo de execução entre vetores em ordem crescente, decrescente e aleatória podem variar dependendo do tamanho do vetor. Em tamanhos pequenos, as diferenças podem ser mínimas, mas à medida que o tamanho do vetor aumenta, o desempenho do Merge Sort começa a se destacar em relação a algoritmos de ordenação menos eficientes.

Conclusão

O Merge-Sort(e baseado em dividir para conquistar, ele ordena cada metade recursivamente e em seguida combina as duas metades ordenadas para ter uma lista final ordenada) e o Selection-Sort (itera a lista não ordenada repetidamente, selecionando o elemento mínimo e colocando na posição certa) são algoritmos de ordenação com abordagens diferentes. Comparação em termos de:

- **Eficiência:**

Merge-Sort: E eficiente em termos de tempo, pois ele tem uma complexidade de tempo média e pior caso de O(n log n), onde n e o numero de elementos na lista. E esse algoritmo é o mais adequado para lidar com grandes conjuntos de dados. Mas requer um espaço adicional para armazenar as subdivisões durante o processo de combinação.

Selection-Sort: Têm uma complexidade de tempo média e pior caso de $O(n^2)$, onde n e o número de elementos na lista. Isso o torna menos eficiente do que o Merge-Sort para grandes conjuntos de dados. E o Selection-Sort têm um desempenho ruim na maioria dos casos, mesmo quando a lista está quase ordenada.

- **Estabilidade:**

Merge-Sort: É um algoritmo de classificação estável, o que significa que a ordem relativa dos elementos iguais é preservada após a classificação.

Selection-Sort: Não é um algoritmo de classificação estável, pois as trocas podem reorganizar a ordem relativa dos elementos iguais.

- **Complexidade de espaço:**

Merge-Sort: Requer espaço adicional para armazenar as subdivisões da lista durante o processo de mesclagem.

Selection-Sort: Realiza as trocas na própria lista, sem exigir espaço adicional.

.

Comparação entre os algoritmos de ordenação resumidamente:

Para vetores pequenos, como os de tamanho 50 e 500, é possível que o algoritmo de ordenação Selection-Sort tenha um tempo de execução menor do que o Merge-Sort. Isso ocorre porque o Selection Sort tem uma complexidade de tempo de $O(n^2)$, enquanto o Merge-Sort tem uma complexidade de tempo de O(n log n). Para conjuntos de dados pequenos, a diferença entre essas complexidades não é tão significativa, o que pode resultar em tempos de execução semelhantes ou até mesmo mais rápidos para o Selection-Sort.

No entanto, à medida que o tamanho do vetor aumenta, como para os vetores de tamanho 5000 e 50000, a diferença de desempenho entre os algoritmos se torna muito mais notável. O Merge-Sort, devido à sua complexidade de tempo mais eficiente, tende a ter um tempo de execução consideravelmente menor em comparação com o Selection-Sort para grandes conjuntos de dados. Portanto, o Merge-Sort é geralmente considerado mais eficiente nessas situações.

Além disso, como mencionado anteriormente, o Merge-Sort requer espaço adicional para armazenar as subdivisões da lista durante o processo de mesclagem, enquanto o Selection-Sort não exige espaço adicional. Portanto, a escolha entre os dois algoritmos também depende dos recursos disponíveis de espaço e tempo.

Resumindo, para vetores de tamanhos pequenos, o Selection-Sort pode ter um tempo de execução menor ou similar ao Merge-Sort. No entanto, para conjuntos de dados maiores, o Merge-Sort é mais eficiente e tem um tempo de execução consideravelmente menor, devido à sua complexidade de tempo mais otimizada. A decisão entre os algoritmos de ordenação dependerá do tamanho dos dados a serem classificados e dos recursos disponíveis de espaço e tempo.

Referências

- DEVMEDIA:https://www.devmedia.com.br/algoritmos-de-ordenacao-analise-e-comparacao/28261

- GEEKHUNTER:https://blog.geekhunter.com.br/aprendizado-de-maquina-e-seus-algoritmos/#:~:text=Os%20algoritmos%20de%20classificação%20tem,às%20variáveis%20para%20prever%20valores.

- WIKIPEDIA: https://pt.wikipedia.org/wiki/Selection_sort

- WIKIPEDIA:https://en.wikipedia.org/wiki/Merge_sort#:~:text=In%20computer%20science%2C%20merge%20sort,in%20the%20input%20and%20output.

- TREINAWEB:https://www.treinaweb.com.br/blog/conheca-os-principais-algoritmos-de-ordenacao

- FREECODECAMP:https://www.freecodecamp.org/portuguese/news/algoritmos-de-ordenacao-explicados-com-exemplos-em-python-java-e-c/#:~:text=Baseado%20na%20estabilidade%3A%20Algoritmos%20de,que%20estavam%20antes%20da%20ordenação.