Relatório do Projeto – SIN 213

Eduardo Nunes de Oliveira - 6021

Universidade Federal de Viçosa – CRP

1 Introdução e Objetivos

Neste relatório, são reportadas algumas análises de execução dos algoritmos selection sort, insertion sort, shell sort e merge sort, a fim de, compará-los computacionalmente, realizando alguns casos de testes com entradas de dados diferentes e mostrando o tempo de execução que cada algoritmo gastou para cada uma delas.

2 Materiais e Métodos

Os algoritmos utilizados neste trabalho foram implementados sobre a linguagem Python (utilizando a IDE Spyder) e avaliados em um hardware composto por um processador Intel(R) Core(TM) i5-7300HQ, 1TB de HDD + 240GB de SSD e 8GB de RAM.

Para avaliar as diferenças dos algoritmos, foram realizados 3 casos de testes, utilizando 4 arquivos .txt como entrada, onde cada arquivo possui valores e quantidades de elementos diferentes.

Arquivos com:

• 10 elementos.

• 1.000 elementos.

• 10.000 elementos.

• 1.000.000 elementos.

Testes:

• (1º) Execução dos algoritmos utilizando um vetor preenchido em ordem aleatória com os elementos dos arquivos.

• (2 º) Execução dos algoritmos utilizando um vetor preenchido em ordem crescente com os elementos dos arquivos.

• (3 º) Execução dos algoritmos utilizando um vetor preenchido em ordem decrescente com os elementos dos arquivos.

3 Resultados

Nesta seção, serão apresentadas as tabelas com o tempo de execução gasto para cada algoritmo com seus respectivos dados de entrada.

• 1º Tabela (Teste – Ordem aleatória):

selection sort	Nº de elementos do arquivo	10	1.000	10.000	100.000	1.000.000
	Tempo de execução (s)	0.0000 (s)	0.0339 (s)	3.4496 (s)	6.0939 (m)	_
insertion sort	Nº de elementos do arquivo	10	1.000	10.000	100.000	1.000.000
	Tempo de execução (s)	0.0000 (s)	0.0519 (s)	4.4135 (s)	7.6623 (m)	_
shell sort	Nº de elementos do arquivo	10	1.000	10.000	100.000	1.000.000
	Tempo de execução (s)	0.0000 (s)	0.0040 (s)	0.0808 (s)	1.1636 (s)	18.3268 (s)
merge sort	Nº de elementos do arquivo	10	1.000	10.000	100.000	1.000.000
	Tempo de execução (s)	0.0000 (s)	0.0040 (s)	0.0568 (s)	0.6097 (s)	9.1728 (s)

- 2º Tabela (Teste – Ordem crescente):

selection sort	Nº de elementos do arquivo	10	1.000	10.000	100.000	1.000.000
	Tempo de execução (s)	0.0000 (s)	0.0369 (s)	3.7539 (s)	9.0109 (m)	_

insertion sort	Nº de elementos do arquivo	10	1.000	10.000	100.000	1.000.000
	Tempo de execução (s)	0.0000 (s)	0.0000 (s)	0.0020 (s)	0.0249 (s)	_

shell sort	Nº de elementos do arquivo	10	1.000	10.000	100.000	1.000.000
	Tempo de execução (s)	0.0000 (s)	0.0000 (s)	0.0010 (s)	0.0229 (s)	_

merge sort	Nº de elementos do arquivo	10	1.000	10.000	100.000	1.000.000
	Tempo de execução (s)	0.0000 (s)	0.0010 (s)	0.0109 (s)	0.1389 (s)	2.2682 (s)

- 3º Tabela (Teste – Ordem decrescente):

selection sort	Nº de elementos do arquivo	10	1.000	10.000	100.000	1.000.000
	Tempo de execução (s)	0.0000 (s)	0.0419 (s)	4.2678 (s)	9.7271 (s)	_

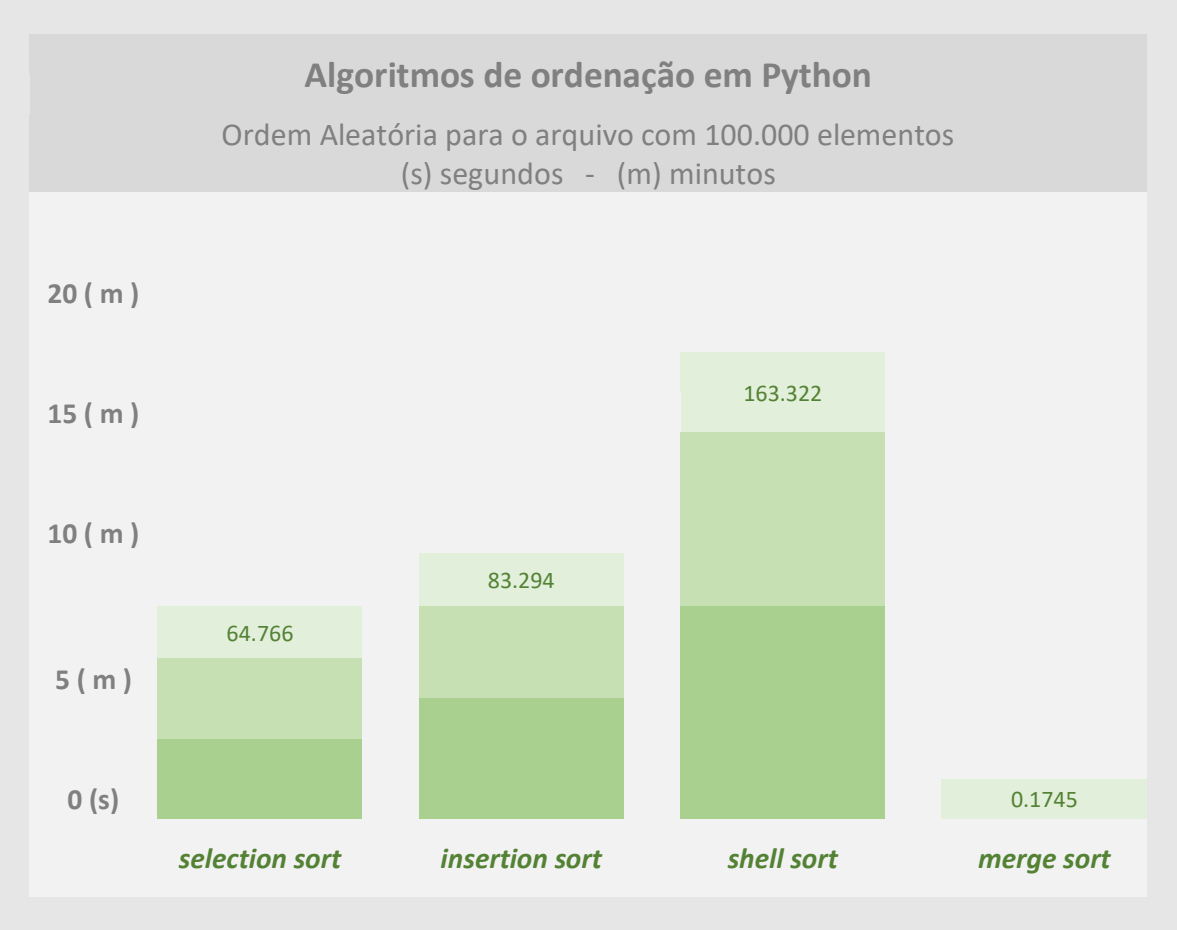
insertion sort	Nº de elementos do arquivo	10	1.000	10.000	100.000	1.000.000
	Tempo de execução (s)	0.0000 (s)	0.0868 (s)	9.6737 (s)	20.7729 (s)	_

shell sort	Nº de elementos do arquivo	10	1.000	10.000	100.000	1.000.000
	Tempo de execução (s)	0.0000 (s)	0.1263 (s)	14.0129 (s)	20.9109 (m)	_

merge sort	Nº de elementos do arquivo	10	1.000	10.000	100.000	1.000.000
	Tempo de execução (s)	0.0000 (s)	0.0010 (s)	0.0120 (s)	0.1576 (s)	2.5546 (s)

• Gráfico:

Este gráfico foi construído para mostrar a diferença entre os tempos de execução para cada algoritmo de forma mais clara, porém ele representa apenas o caso de teste com o vetor em ordem Aleatória e o arquivo de entrada contendo 100.000 números.



4 Conclusão

Após os resultados obtidos durante os testes, observou-se que os tempos de execução obtiveram diferenças consideráveis. Em praticamente todos os testes, com as possibilidades citadas anteriormente, o algoritmo Merge Sort foi o que conseguiu os melhores resultados na ordenação de um

vetor, se tratando de um vetor com uma grande quantidade de elementos, pelo fato desse algoritmo “quebrar o problema em vários problemas menores”, acabou sendo mais eficiente e conseguiu grande vantagem no tempo comparado com os outros algoritmos. Quanto a tabela de Ordem Crescente (2º), o único algoritmo que saiu do padrão foi o Selection Sort, pois gastou cerca de 9 minutos, enquanto os outros algoritmos gastaram menos de 1 segundo. O teste realizado com os vetores em ordem decrescente mostrou realmente como o Merge Sort se diferencia dos outros, pois foi o único algoritmo que conseguiu resolver a ordenação em segundos, seguido do Selection Sort, que gastou um pouco mais de 9 minutos, e dos outros dois, Insertion Sort e Shell Sort. Por fim, de maneira geral, os testes com o vetor de 100.000 elementos ordenado em ordem decrescente foram os que possuíram os piores tempos.