Comparação de Métodos de Ordenação

Thiago Moraes

[1]Instituto de Computação – Universidade Federal do Amazonas (UFAM)
Manaus – AM – Brasil

tmr@icomp.ufam.edu.br

Parte 1 - Média dos Algoritmos de Ordenação

						Aleatórios
Métrica	N	1k	5k	10k	20k	50k
Comparações	Bolha	499500	12497500	49995000	199990000	1249975000
	Seleção	499500	12497500	4999500	199990000	1249975000
	Insercao	999	4999	9999	19999	49999
	Heapsort	16857	107695	235382	510751	1409814
	MergSort	18706	117061	254095	548162	1502681
	Quicksort	6547	43560	96499	211447	589331
Número de Trocas	Bolha	248658	6244174	24883340	9980330	623351270
	Seleção	990	4989	9988	19987	49989
	Insercao	2496657	6249173	624777	100000329	623401269
	Heapsort	10566	64588	139188	298384	812456
	MergSort	28756	180013	389984	839926	2298271
	Quicksort	2826	16820	35957	76517	206586
Tempo	Bolha	0.004295	0.166094	0.541298	2.234208	13.927709
	Seleção	0.001688	0.040487	0.097624	0.676024	3.948554
	Insercao	0.001154	0.029645	0.156975	0.540641	2.989432
	Heapsort	0.000210	0.001018	0.002209	0.004689	0.013044
	MergSort	0.000283	0.001358	0.002863	0.006154	0.016372
	Quicksort	0.000154	0.000711	0.001512	0.003219	0.008764

Essa foi a média das 10 execuções de cada algoritmo para : 1k, 5k, 10k, 20k e 50k

Parte 2 - Perguntas

Nessa questão eu testei alguns vetores ordenados inversamentes e crescente e aleatórios. Pois se eu fosse me basear diretaemente somente por tamanho do vetor sempre teríamos que com 1k é melhor e 50k seria pior.

Para cada método de ordenação. Qual foi seu melhor e pior caso observado?

BubleSort

Melhor caso é quando o vetor ja estava ordenado. E pior caso é quando o vetor estava ordenado na ordem inversa.

SelectionSort

Melhor caso é quando o vetor ja estava ordenado. E pior caso é quando o vetor encontra-se numa aleatoriedade muito grande, números a serem comparados muitos distantes dos

valores da proxima posição

InsertionSort
Melhor caso é quando o vetor ja estava ordenado. E pior caso é quando o vetor encontra-se ordenado inversamente

HeapSort
Melhor caso é quando o vetor encontra-se ordenaddo inversamente, ao contrário do insertion que seu pior caso é esse de está ordenado inversamente
Mais uma vez sendo o inverso do insertion, seu pior caso é quando o vetor está ordenado de forma crescente

MergeSort
Melhor caso foi quando o vetor já se encontrou parcialmente ordenado e não precisou fazer muitas trocas
Pior caso foi quando vetor encontrou muita aleatoriedade. Onde há muitas trocas

QuickSort
Melehor caso foi quando ordenou vetores grandes
Pior caso foi quando os vetores eram pequenos, o insertion estava melhor que o quick para vetores pequenos.

Qual função melhor descreve o desempenho de cada método?

Abaixo a complexidade assintótica de cada método.
BubleSorte $0(N^2)$
Selection $0(N^2)$
Insertion $0(N^2)$
MergeSort $0(nlogn)$
HeapSort $0(nlogn)$
QuickSort $0(nlogn)$

Em quais métodos(e quantidade de dados nos vetores), o número de comprações é um bom substituto para o tempo de execução (ou seja, as duas métricas dão resultados relativos parecidos)?
Como podemos observar na tabela, é evidente que seja o QuickSort.
De forma geral, quais os melhores algoritmos dentre os testados?
QuickSort possui o melhor desempenho, HeapSort segundo, MergeSort terceiro e BubleSort como o pior entre todos.

Parte 3 - Gráficos dos Algoritmos de Ordenação

Figura 1. Comparações entre os Métodos de Ordenação

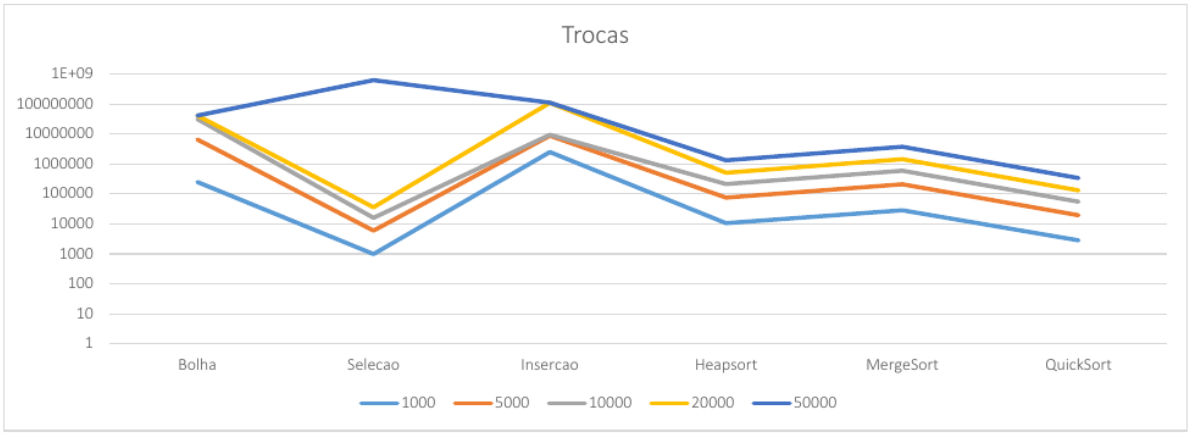

Figura 2. Gráfico de Trocas

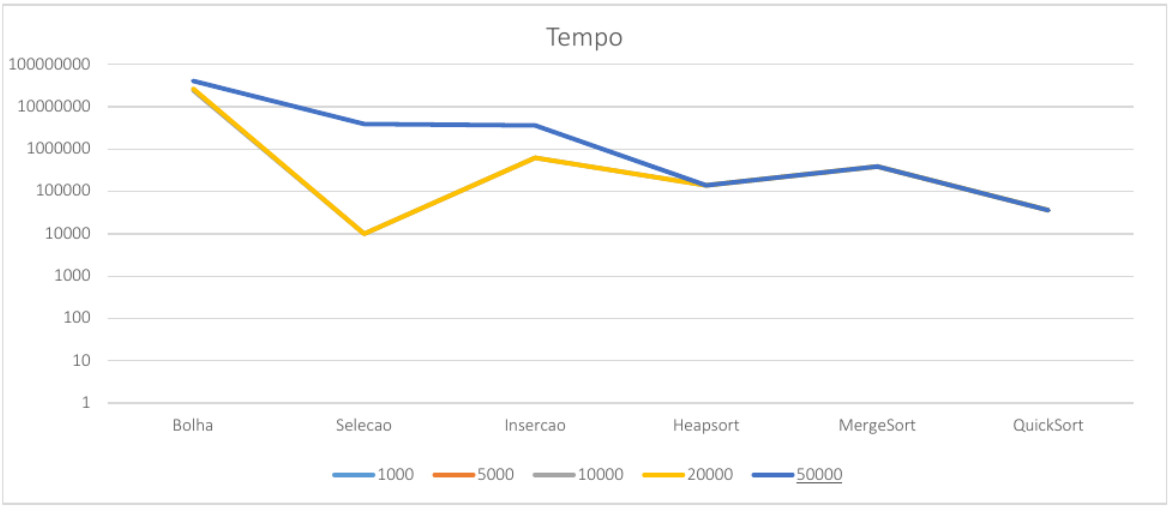

Figura 3. Gráfico de Tempo