Universidade Federal do Maranhão
Disciplina: Estrutura de Dados II
Discentes: André Filipe S. Barreto, João Víctor C. Gonçalves

Trabalho Prático I

São Luís-MA
2022

1) Ambiente de trabalho

Antes de iniciar a apresentação dos resultados, é importante contextualizar o ambiente de trabalho em que foram realizados os testes. Utilizou-se o Sistema Operacional Windows 11, 64 bits, e a IDE utilizada para rodar os programas em Java foi o Intellij IDEA, por conta do bom suporte e das ótimas ferramentas de debug, otimizando, assim, o processo de escrita e de resolução de erros.

Na coleta de dados para os testes, as entradas de mesmo tamanho foram repetidas 3 vezes, excluindo a de maior valor e a de menor valor, evitando possíveis resultados parciais. Foram coletados dados com tamanhos de 1000, 100.000 e 1.000.000, a fim de demonstrar a diferença do tempo de execução em cada um desses conjuntos. É importante ressaltar que mesmo com esse método, os valores encontrados são aproximados, uma vez que processos internos do sistema podem causar alterações.

2) Testes

a. Selection-Sort

Chave: String Valor: Double						
Tamanho	1000	100000	1000000	1000	100000	1000000
SelectSort	Crescente			Decrescente		
Comparações	499500	704982704	INDEF	499500	704982704	INDEF
Atribuições	9526	1407848	INDEF	9467	1416917	INDEF
Tempo	0,0208845s	116,0208069 s	INDEF	0,0203063s	127,1066546s	INDEF
Chave: Double Valor: String						
Comparações	499500	499500	INDEF	499500	704982704	INDEF
Atribuições	9662	1410088	INDEF	9473	1409969	INDEF
Tempo	0,0127311s	84,9231274s	INDEF	0,0167198s	86,3135213	INDEF
Chave: Double Valor: Integer						
Comparações	499500	704982704	INDEF	499500	704982704	INDEF
Atribuições	9758	1410229	INDEF	9233	1410404	INDEF
Tempo	0,0166513s	21,4329618	INDEF	0,0149917s	21,2962722	INDEF

b. Selection-Sort → MinMax

Chave: String Valor: Double						
Tamanho	1000	100000	1000000	1000	100000	1000000
SelectionSort-MinMax	Crescente			Decrescente		
Comparações	499500	704982704	INDEF	499500	704982704	INDEF
Atribuições	9278	1411706	INDEF	9634	1404262	INDEF
Tempo	0,0220554s	119,7118556s	INDEF	0,0158492s	118,989913s	INDEF
Chave: Double Valor: String						
Comparações	501500	705182704	INDEF	501500	705182704	INDEF
Atribuições	16098	2529944	INDEF	15624	2503380	INDEF
Tempo	0,0157762s	69,2259359s	INDEF	0,0171379s	68,3246191s	INDEF
Chave: Double Valor: Integer						
Comparações	501500	705182704	INDEF	501500	705182704	INDEF
Atribuições	16636	2507016	INDEF	16070	2508398	INDEF
Tempo	0,0163979s	17,3316445s	INDEF	0,0333198s	17,0952388s	INDEF

c. HeapSort

Chave: String Valor: Double						
Tamanho	1000	100000	1000000	1000	100000	1000000
HeapSort	Crescente			Decrescente		
Comparações	28665	4874028	58644408	28602	4873821	58644138
Atribuições	22666	3937303	47668628	22622	3937073	47666603
Tempo	0,0036616s	0,1593533s	1,8571019s	0,0034693s	0,1640363s	1,8722346s
Chave: Double Valor: String						
Comparações	28743	4873047	58647828	28845	4875018	58643652
Atribuições	22786	3936563	47671185	22839	3938253	47667745
Tempo	0,002659s	0,2077259s	1,6207067s	0,0029647s	0,1774179s	1,7238199s
Chave: Double Valor: Integer						
Comparações	28641	4875540	58649847	28713	4873911	58645434
Atribuições	22708	3938467	47672893	22756	3937451	47668744
Tempo	0,002523s	0,0873944s	1,5920099s	0,0019666s	0,0951938s	1,6200879s

d. MergeSort

Chave: String Valor: Double						
Tamanho	1000	100000	1000000	1000	100000	1000000
MergeSort	Crescente			Decrescente		
Comparações	9977	1668929	19951425	9977	1668929	19951425
Atribuições	32925	5306781	62854269	32925	5306781	62854269
Tempo	0,0033864s	0,1792204s	1,3385476s	0,0039206s	0,1514609s	1,4359801s
Chave: Double Valor: String						
Comparações	9977	1668929	19951425	9977	1668929	19951425
Atribuições	32925	5306781	62854269	32925	0,098012	62854269
Tempo	0,0025115s	0,1165386s	0,9233549s	0,0024557s	0,098012s	1,0846974
Chave: Double Valor: Integer						
Comparações	9977	1668929	19951425	9977	1668929	9951425
Atribuições	32925	5306781	62854269	32925	0,070774	62854269
Tempo	0,0019528s	0,0609774s	0,8940742s	0,0028493s	0,070774s	0,9089059s

e. QuickSort

Chave: String **Valor**: Double						
Tamanho	1000	100000	1000000	1000	100000	1000000
QuickSort	Crescente			Decrescente		
Comparações	10908	1946696	24274129	11794	1974247	24542775
Atribuições	25016	4075346	52069900	26896	4485982	52677334
Tempo	0,0034655s	0,1208128s	1,0804512s	0,0038242s	0,1072606s	1,1049773s
Chave: Double **Valor**: String						
Comparações	11675	2118657	23753095	10856	2000588	24105382
Atribuições	24008	4717942	51279406	25454	4207824	51666582
Tempo	0,0034545s	0,1134583s	0,8437183s	0,0027673s	0,0908774s	0,8200369s
Chave: Double **Valor**: Integer						
Comparações	11368	1973384	25539147	11115	2004466	25956051
Atribuições	24498	4282256	59306498	27498	4512474	55892086
Tempo	0,0028204s	0,0723036s	0,848666s	0,0035345s	0,0745709s	0,8274375s

f. MergeSort JDK

Chave: String **Valor**: Double						
Tamanho	1000	100000	1000000	1000	100000	1000000
MergeSort JDK	Crescente			Decrescente		
Comparações	-	-	-	-	-	-
Atribuições	-	-	-	-	-	-
Tempo	0,0003s	0,0003s	0,001s	0,0003s	0,0003s	0,001s
Chave: Double **Valor**: String						
Comparações	-	-	-	-	-	-
Atribuições	-	-	-	-	-	-
Tempo	0,0003s	0,0003s	0,001s	0,0003s	0,0003s	0,0010s
Chave: Double **Valor**: Integer						
Comparações	-	-	-	-	-	-
Atribuições	-	-	-	-	-	-
Tempo	0,0003s	0,0043s	0,0145s	0,0003s	0,0045s	0,012s

3) Análise dos resultados e Conclusão

Para começar a análise dos resultados, é preciso observar primeiramente o algoritmo do Selection-Sort. Como esse é um algoritmo de custo O(n^2), é esperado que para grandes valores de n, o tempo de execução cresça quase que exponencialmente. Quando a entrada tem n == 1000 elementos, o tempo de execução é de aproximadamente 0,0166513s. Quando esse valor passa para 100.000 elementos, o tempo de execução é de 21,4329618s, ou seja, percebe-se um evidente aumento. Quando n recebe o valor de 1.000.000 de elementos, o tempo de execução excede ainda mais esse crescimento, sendo, portanto, indefinido para nossos testes. A expectativa era de que esse tempo fosse mais de 6 horas.

Analisando o algoritmo SelectionSort - MinMax, a partir da teoria vista em aula, era esperado que seu tempo de execução fosse melhor que o algoritmo SelectionSort. Tal expectativa se confirma na prática, pois os resultados observados se mostraram melhores. Para n == 1000, foi observado um tempo de execução de aproximadamente 0,0163979 segundos. Para n == 100.000, o tempo de execução encontrado foi de aproximadamente 17,3316445 segundos. Para n == 1000, os tempos de execuções ficaram bem parecidos, mas para n == 100.000, percebe-se uma diferença de mais de 4 segundos do segundo algoritmo para o primeiro. A tendência é que à medida que n cresça, essa diferença cresça também.

Os algoritmos que possuem tempo de execução na ordem O($n \log n$) se demonstraram muito mais eficientes que os demais algoritmos. Essa diferença pode ser bem evidente quando observamos os tempos de execução para n == 1.000.000. Em quase todos esses algoritmos, o tempo de execução permaneceu na faixa de um segundo, sendo mais rápido o algoritmo do QuickSort, que apresentou um T.E de 0,848666 segundos (Chama-se a atenção para a diferença em relação aos algoritmos quadráticos, cujo custo foi indefinido)

Outro ponto observável é a diferença entre os tempos de execução dos algoritmos na ordem crescente e decrescente. No geral, o T.E na ordem crescente teve melhor desempenho que na ordem decrescente, o que já era esperado, uma vez que são necessárias mais comparações para ordenar. Contudo, em alguns casos específicos, pôde-se notar que os algoritmos que ordenaram em ordem decrescente tiveram um desempenho melhor, Isso pode ser explicado pelo fato de terem sido utilizados vetores com valores totalmente aleatórios, causando essas pequenas exceções..

Por fim, durante a análise dos algoritmos, foi observado que as chaves que eram do tipo String demoraram um tempo maior para serem ordenadas que as demais chaves. Tal ocorrência se dá pelo fato de que ordenar Strings requer um esforço maior que ordenar números reais e números inteiros. Analisando o algoritmo de melhor desempenho, QuickSort (para n = 1.000.000), a diferença entre a chave String para a chave Double chega a ser de aproximadamente 0,24 segundos. Essa diferença também pode ser notada nos demais algoritmos.

(Obs: Os valores retirados como referência foram os de ordenamento crescente, com chave: Double e valor: Integer)

Referências

Maneiras de medir o tempo em Java sem bibliotecas externas
https://thiagovespa.com.br/blog/2015/09/29/maneiras-de-medir-o-tempo-em-java-sem-bibliotecas-externas/

Como fazer MergeSort em Java
https://www.youtube.com/watch?v=yj8igr9DjeY

Gerar String Aleatória em Java
https://www.delftstack.com/pt/howto/java/random-alphanumeric-string-in-java/

Algoritmos - Teoria e Prática Cormen
https://computerscience360.files.wordpress.com/2018/02/algoritmos-teoria-e-prc3a1tica-3ed-thomas-cormen.pdf

Generic Types (The Java™ Tutorials > Learning the Java Language>Generics (Updated))
https://docs.oracle.com/javase/tutorial/java/generics/types.html