Trabalho de Ordenação

Luiz Roberto da Silva Gonzaga | TADS IFRN - 2º Sem. | Jorgiano Vidal - Algoritmos

Ordenação

Neste trabalho de ordenação temos 3 funções principais de Ordenação (Seleção, Inserção e Merge_sort), tendo 5 pastas de casos-testes, cada um deles contendo cerca de 30 arquivos .txt, sendo cada arquivo desse um teste, variando de 10000 a 100000 números, sendo eles ordenados ou não.

Em um breve resumo, podemos perceber que o método Merge_sort foi mais eficiente em sua maioria, já o método de Seleção foi o menos eficiente.

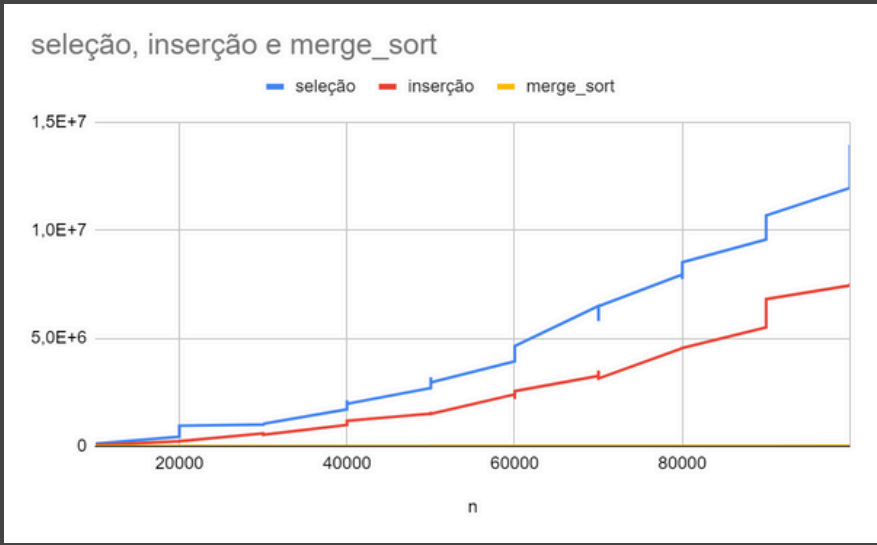
Big-Oh

A seguir temos as 3 funções e seus respectivos Big-Oh's:

-> Seleção = $O(n^2)$
-> Inserção = $O(n^2)$
-> Merge_sort = $O(n \log n)$

Caso-01

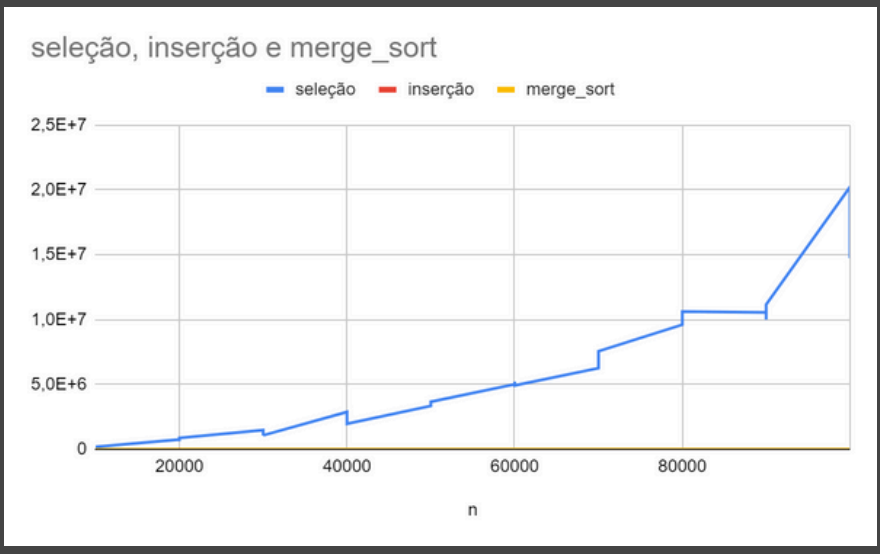
No caso 1, temos merge_sort como mais eficiente, função essa que possúi n log n como operação Big-oh, tendo também o método de seleção como menos eficiente (n^2)



	Caso 1		
n	seleção	inserção	merge_sort
10000	108449	59066	1446
10000	108803	58281	1602
10000	118957	60660	1407
20000	441764	230535	3950
20000	454827	231538	3103
20000	950745	233792	3709
30000	1006139	598495	4956
30000	1056102	568966	5648
30000	1039540	530778	5972
40000	1712257	985461	7354
40000	2126845	990474	8305
40000	1966808	1180202	7511
50000	2698399	1510026	9210
50000	3197058	1600436	8672
50000	2956391	1504150	9320
60000	3927310	2402035	10885
60000	4241255	2190316	10424
60000	4636295	2551601	10052
70000	6493117	3254635	17565
70000	5791151	3506606	25426
70000	6481818	3138078	13794
80000	7949554	4536058	15983
80000	7737327	4548752	14223
80000	8514753	4549053	16235
90000	9565705	5500738	15991
90000	9710251	6108491	19172
90000	10679988	6809201	16223
100000	11954531	7441289	24149
100000	12026161	7484581	18715
100000	13954556	7542467	18813

Caso-02

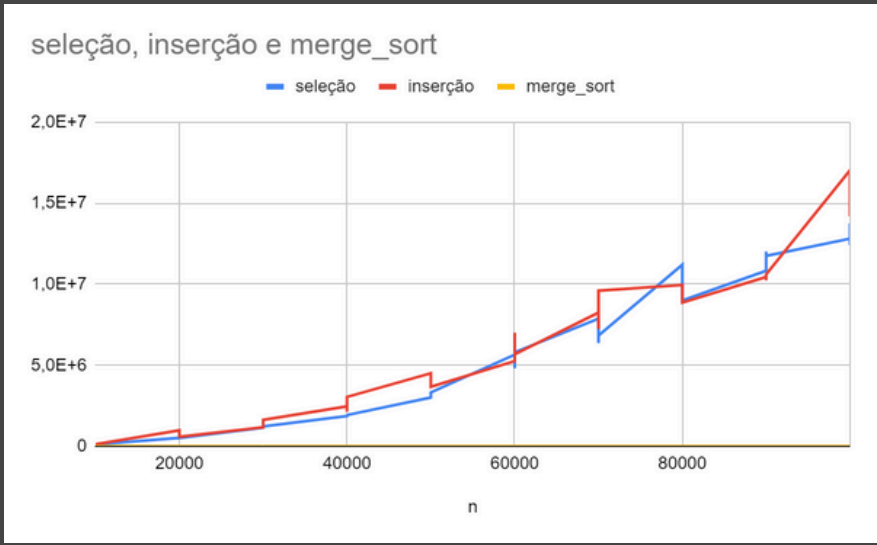
No caso 2, temos uma diferença com relação aos outros casos-teste, neste caso, o método de Inserção foi mais eficiente que o merge_sort (método mais eficiente até então).
(Método de Seleção continuando a ser o menos eficiente)



	Caso 2		
n	seleção	inserção	merge_sort
10000	209615	29	788
10000	207938	31	897
10000	191024	31	1136
20000	749285	77	10168
20000	631446	63	13857
20000	876825	61	10708
30000	1477753	131	2131
30000	1058019	129	2540
30000	1076026	93	2088
40000	2880910	120	3051
40000	2818901	128	2768
40000	1969317	165	3134
50000	3349325	171	3977
50000	3535945	157	3785
50000	3664986	280	4318
60000	5008943	258	5248
60000	5212282	177	5188
60000	4926666	186	5523
70000	6257842	220	6103
70000	6871834	218	5779
70000	7563370	208	5966
80000	9608890	390	6571
80000	10083780	238	6952
80000	10615804	249	7082
90000	10555926	282	8310
90000	10006553	441	9978
90000	11158857	284	8776
100000	20231859	366	10504
100000	16406070	296	10007
100000	14746258	297	9549

Caso-03

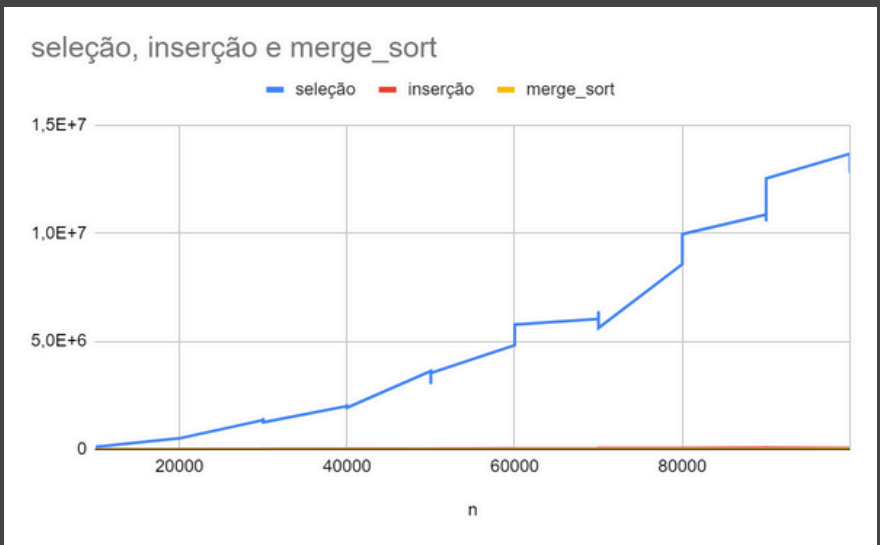
Já no caso 3, temos o método merge_sort como o mais eficiente de todos O(n log n), e uma disputa entre Seleção e Inserção como os menos eficientes.



	Caso 3		
n	seleção	inserção	merge_sort
10000	108865	123637	806
10000	148601	123360	862
10000	120581	127705	1547
20000	517915	985536	1875
20000	598525	668988	2224
20000	515928	586149	1817
30000	1151637	1174241	3017
30000	1104133	1273106	3139
30000	1227840	1629355	2975
40000	1861766	2464478	4053
40000	1952548	2131355	3976
40000	1928627	3045360	4155
50000	3006734	4504670	5593
50000	2941314	4041470	4956
50000	3321870	3670090	5405
60000	5650966	5256957	6526
60000	4817625	7007082	6940
60000	5798535	5676974	5994
70000	7874519	8247338	7364
70000	6366681	7239457	7352
70000	6828943	9597772	7463
80000	11201512	9949591	8537
80000	10876232	8908125	8498
80000	8997496	8863861	8684
90000	10828212	10444267	10925
90000	12015841	10235330	12533
90000	11743998	10606429	11630
100000	12806329	17028420	9725
100000	13751059	14185433	12423
100000	12420061	15717590	10250

Caso-04

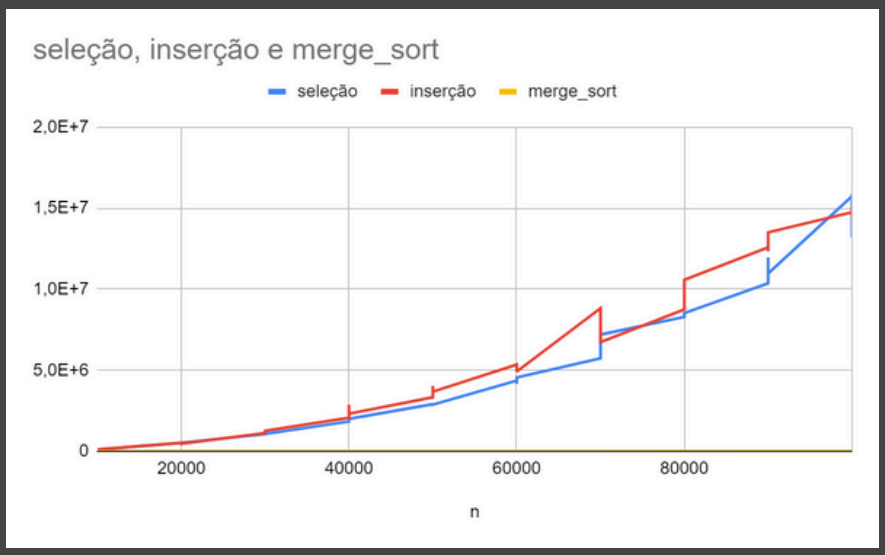
No caso 4, o método de merge_sort é o mais eficiente, porém, o método de inserção foi bem páreo no começo, e não se mostrou ser tão ruim eficientemente quanto o de seleção.



	Caso 4		
n	seleção	inserção	merge_sort
10000	112305	7164	1698
10000	111933	6724	1279
10000	117015	5723	1440
20000	508424	14939	3542
20000	523810	13305	3767
20000	505196	13782	3561
30000	1350190	19026	6946
30000	1447008	18150	4981
30000	1245660	18534	4143
40000	2001183	28094	9488
40000	2086215	29063	7053
40000	1923931	27768	5908
50000	3623900	30018	11091
50000	3013509	31203	8066
50000	3530499	32640	7230
60000	4814775	39928	13321
60000	5013533	46152	8987
60000	5770352	42081	11318
70000	6029311	49159	13584
70000	6390857	50248	13627
70000	5610595	56908	12606
80000	8564149	58834	13716
80000	9792560	58959	16091
80000	9954689	58822	13042
90000	10856267	68833	15729
90000	10539718	66827	22720
90000	12532262	68633	16262
100000	13674939	59015	20017
100000	12787676	65759	15197
100000	13292393	58878	25568

Caso-05

No caso 5 temos o merge_sort (O(n log n)) como o mais eficiente, e os métodos de Seleção e Inserção parelhos como os menos eficientes!



	Caso 5		
n	seleção	inserção	merge_sort
10000	108371	123496	1320
10000	115336	122093	1456
10000	108136	115120	1452
20000	513895	520343	2962
20000	566454	495975	3290
20000	528051	468948	2646
30000	1062716	1123730	4314
30000	1151685	1261669	5039
30000	1069389	1265889	4835
40000	1838074	2070359	6085
40000	1919346	2879825	7220
40000	2000995	2319624	6669
50000	2903946	3330149	6743
50000	2936361	4043812	6870
50000	2883193	3677108	7294
60000	4355766	5349581	9208
60000	4148764	4885538	9107
60000	4550547	4913296	9054
70000	5737801	8825521	10629
70000	5747693	6698655	9769
70000	7200522	6735559	10004
80000	8292636	8748387	13606
80000	8309111	9576038	12951
80000	8526364	10572776	11889
90000	10359987	12577561	13633
90000	11957501	12334276	14968
90000	10969725	13501592	12846
100000	15734532	14747633	15444
100000	15896876	14439251	17815
100000	13194233	15113465	14939