EP3 - MAC0121 - Algoritmos e Estrutura de Dados I

Aluno: Fernando Gouveia Lima Número Usp: 13672710

Introdução:

O principal objetivo deste trabalho foi o estudo e a implementação de alguns dos principais algoritmos de ordenação de vetores, sendo estes o “Heapsort”, “Insertion sort”, “Mergesort” e o “Quicksort”. Além de produzir tais códigos, um ponto crucial deste trabalho é aprofundar-se no mundo destes 4 algoritmos e compreender quando a utilização de cada um destes é mais benéfica. Está análise será feita observando o número total de comparações e de trocas que cada um destes códigos realiza, dado vetores com diferentes tamanhos e instâncias.

Descrição Breve dos Métodos Utilizados:

Os métodos utilizados dos quatros algoritmos foram os clássicos com singelas alterações. No “Heapsort”, a parte da “filadeprioridades” não houve a necessidade de implementar códigos padrões deste algoritmo como o “corrigeSubindo”, “InserenaHeap” ou “Alteraprioridade”, pois o “Heapsort” somente necessita do “constroiHeap” e “corrigeDescendo”. Ademais, ao invés de implementar o “Heapsort” com uma struct foi optado pela utilização de um “vetor de chars”, como feito nos outros 3 algoritmos.

No “Mergesort”, o código do professor utilizava dois vetores auxiliares na função “intercala”, e neste código foi optado pela utilização de somente um e com singelas alterações nesta mesma função.

EP3 - MAC0121 - Algoritmos e Estrutura de Dados I

Descrição do Tipo de Instância:

Neste trabalho foram utilizadas 4 instâncias, sendo todas arquivos .txt, onde cada linha encontrava-se uma palavra de até 10 caracteres. Duas destas foram dicionários (um inglês e um português), outra instância foi utilizado a Bíblia e por último um arquivo criado por mim contendo algumas dezenas de palavras. A Bíblia foi utilizada de forma contendo palavras em modo aleatório, ordenadas, e ordenadas ao contrário. Já os dicionários foram utilizados para testar de modo ordenado, parcialmente ordenado e ordenado ao contrário, enquanto meu arquivo pessoal foi testado das 4 formas. Para cada um destes modos (ordenado, parcialmente, reverso e aleatório) iniciou-se os testes como recomendado pelo professor, utilizando 250 palavras e dobrando o número de palavras até quando for possível (os dicionários possuem aproximadamente 256.000 palavras, portanto este número foi utilizado como teto máximo para os testes). Quando alguns dos testes apresentaram um excessivo tempo de execução foi considerado também como o máximo possível.

Resultados Gráficos e Tabelas:

InsertionSort:

Insertion_Sort											
Quanridade de palavras:	250	500	1000	2000	4000	8000	16000	32000	64000	128000	256000
Comparações											
Ordenado:	249	499	999	1999	3999	7999	15999	31999	63999	127999	255999
Aleatorio:	12728	56815	229336	958110	3965050	16109194	63320401	250909382	979558584	3939002002	16715488901
Ordenado ao contrario	28624	98305	240805	679196	6226753	28790775	124137877	496894601	2013044377	8062781574	----------
Parcialmente Ordenado	357	606	1190	2403	4689	9112	17023	39134	71892	134545	277911
Trocas											
Ordenado:	0	0	0	0	0	0	0	0	4095	4096	4323
Aleatorio:	12977	57314	230335	960109	3969049	16117193	63336400	250941381	979622583	3939130001	16715744900
Ordenado ao contrario	28375	97806	239806	677197	6222754	28782776	124121878	496862602	2012980378	8062653575	----------
Parcialmente Ordenado	45	123	301	432	674	1208	3789	6984	9020	15382	21984

EP3 - MAC0121 - Algoritmos e Estrutura de Dados I

MergeSort:

MergeSort											
Quanridade de palavras:	250	500	1000	2000	4000	8000	16000	32000	64000	128000	256000
Comparações											
Ordenado:	983	2216	4932	10864	23728	51456	110912	237824	507648	1079296	2286592
Aleatorio:	1689	3868	8739	19423	42871	93595	203196	438294	940703	2009806	4275019
Ordenado ao contrario	1158	2562	5494	12534	27594	58555	125761	267552	569283	1194158	2514197
Parcialmente Ordenado	1068	2301	5017	10979	23872	51935	111651	246263	546475	1192520	2541767
Trocas											
Ordenado:	1994	4488	9976	21952	47904	103808	223616	479232	1022464	2172928	4601856
Aleatorio:	1994	4488	9976	21952	47904	103808	223616	479232	1022464	2172928	4601856
Ordenado ao contrario	1994	4488	9976	21952	47904	103808	223616	479232	1022464	2172928	4601856
Parcialmente Ordenado	1994	4488	9976	21952	47904	103808	223616	479232	1022464	2172928	4601856

QuickSort:

Quick_Sort											
Quanridade de palavras:	250	500	1000	2000	4000	8000	16000	32000	64000	128000	256000
Comparações											
Ordenado:	31125	124750	269853	1050110	5600701	26646105	116322765	487389033	----------	----------	----------
Aleatorio:	2192	7162	15294	40920	105327	294117	544676	1346601	4541450	10099721	26925876
Ordenado ao contrario	6737	36364	279114	1391486	1975278	3653583	5360383	20253650	55721486	224079587	1300453959
Parcialmente Ordenado	9378	54029	263244	340321	445891	897812	----------	----------	----------	----------	----------
Trocas											
Ordenado:	0	0	4061	4073	4081	4098	4154	4186	----------	----------	----------
Aleatorio:	740	2028	4106	11501	26836	58565	125228	292918	822968	1934243	3678861
Ordenado ao contrario	1991	5970	14470	31780	93093	220450	518848	1625700	4920214	17244388	80581792
Parcialmente Ordenado	481	1561	2274	6720	11293	26781	----------	----------	----------	----------	----------

HeapSort:

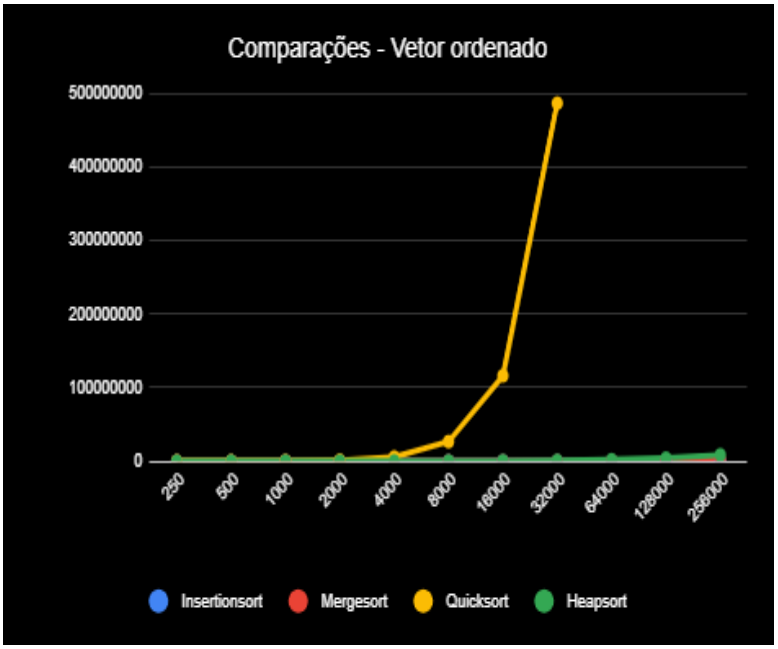
Heap_Sort											
Quanridade de palavras:	250	500	1000	2000	4000	8000	16000	32000	64000	128000	256000
Comparações											
Ordenado:	745	1495	8758	30554	81066	187499	414892	877924	1911620	4062871	8548641
Aleatorio:	3331	7640	17286	38514	84764	185189	402638	870096	1869723	3998289	8495438
Ordenado ao contrario	2843	4799	6535	14219	71481	169991	381893	811995	1748947	3609145	7730273
Parcialmente Ordenado	4507	10492	24785	55546	122529	270892	589334	1273432	2739879	5866002	12010600
Trocas											
Ordenado:	0	0	3701	15303	43448	101419	224868	469212	1031042	2170696	4546450
Aleatorio:	1656	3797	8646	19194	42136	92089	200402	433716	932888	1995900	4234528
Ordenado ao contrario	1208	1735	1770	4396	31926	78823	179919	383520	825820	1699938	3653822
Parcialmente Ordenado	2756	6496	15816	35619	78676	175184	382104	826978	1783003	3823633	7804238

Obs: Campos preenchidos com “—”, são aqueles testes que ultrapassaram o “máximo possível” (devido ao tempo de execução demasiadamente grande).

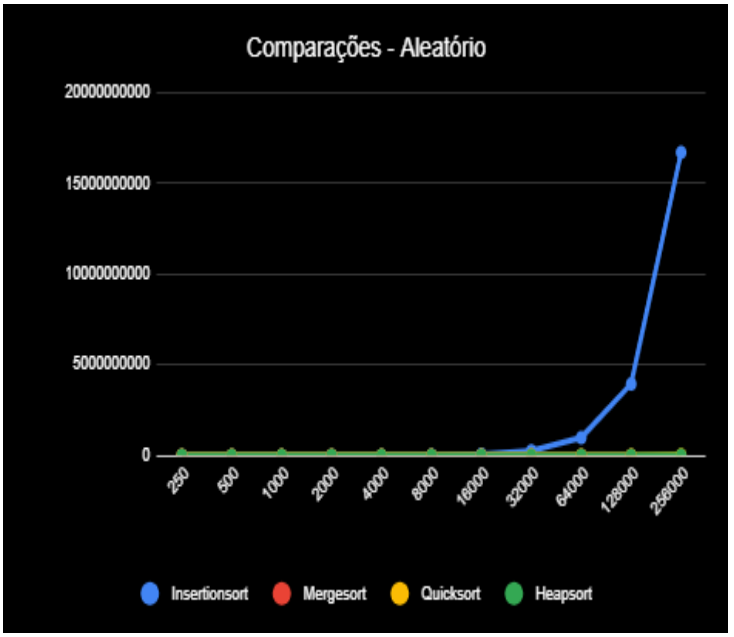
EP3 - MAC0121 - Algoritmos e Estrutura de Dados I

Gráficos Comparações:

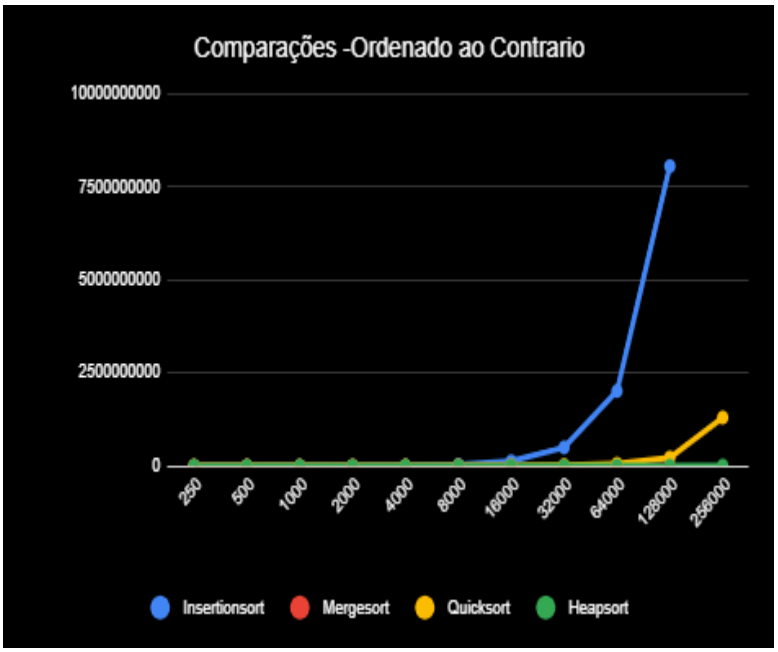
Ordenado:



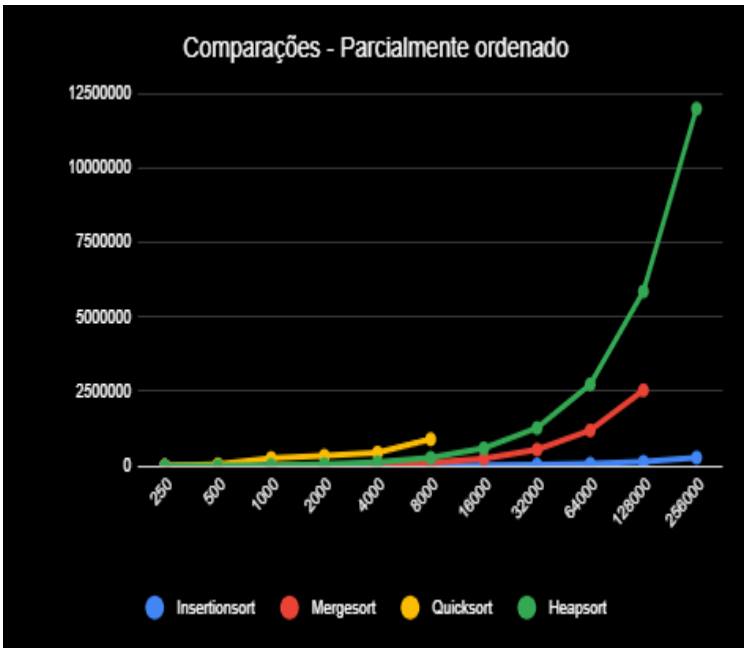
Aleatório:



Ordenado ao contrário:



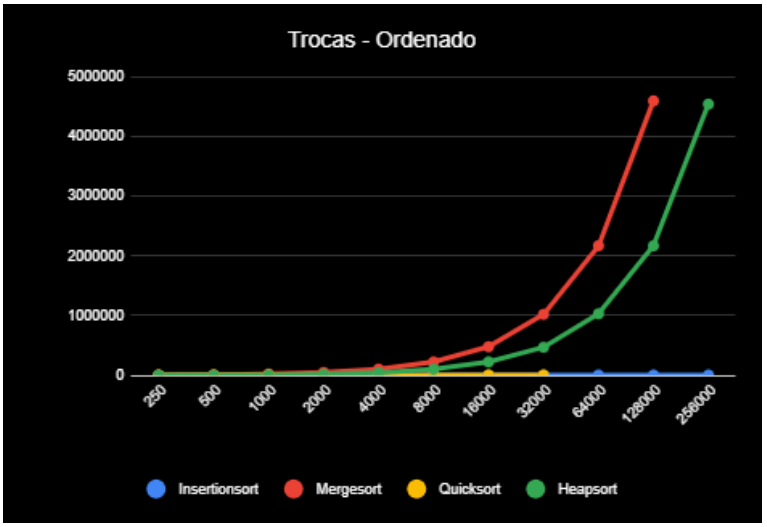
Parcialmente Ordenado:



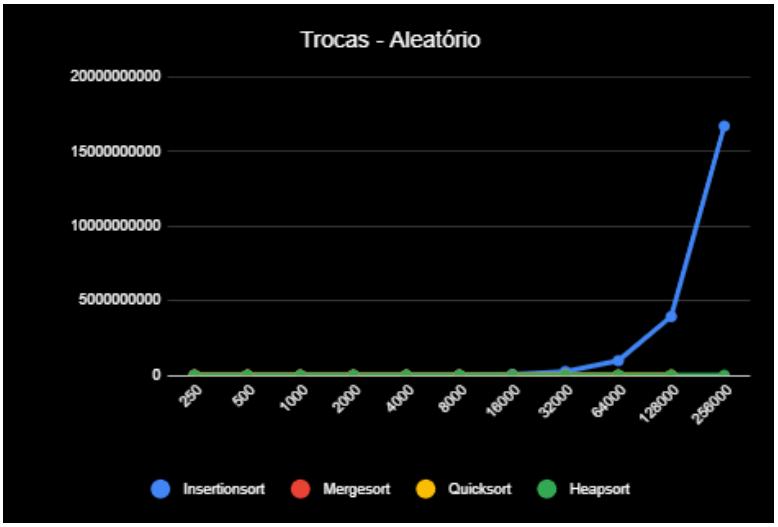
EP3 - MAC0121 - Algoritmos e Estrutura de Dados I

Gráficos Trocas:

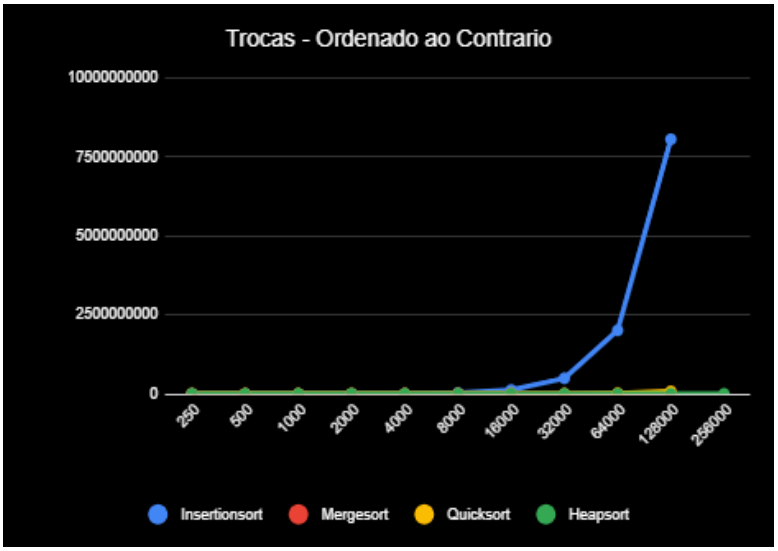
Ordenado:



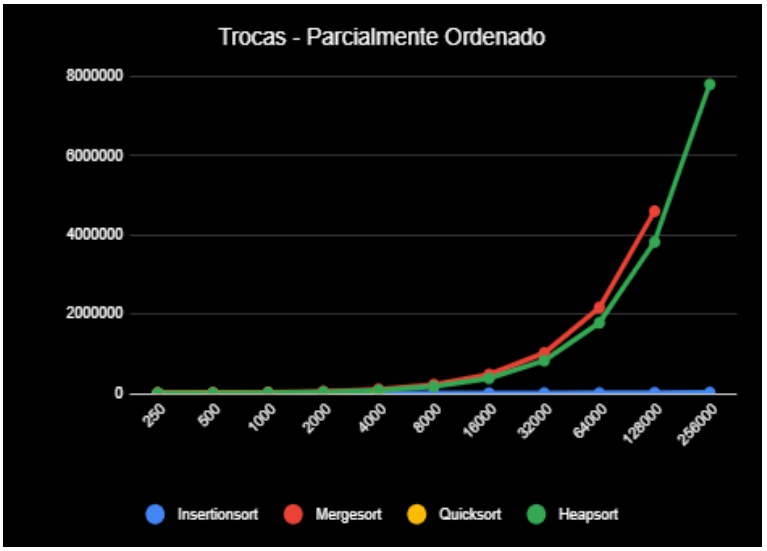
Aleatório



Ordenado ao contrário:



Parcialmente Ordenado:



EP3 - MAC0121 - Algoritmos e Estrutura de Dados I

Conclusões:

Ao observar os gráficos e os dados da tabela, acaba sendo notório os pontos fortes e fracos de cada algoritmo de ordenação. O “InsertionSort” é extremamente eficiente para casos em que o vetor está ordenado e parcialmente ordenado independentemente do tamanho dele. No entanto, este mesmo é o mais ineficiente disparado entre os 4 para os casos de o vetor encontrar-se de modo aleatório ou ordenado ao contrário, isso decorre do modo que ele é implementado que exige um número excessivo de trocas nestes casos, atingindo $O(n^2)$ no caso ao contrário. Com relação ao “HeapSort”, este demonstra-se bem ineficiente quando o vetor está ordenado ou parcialmente ordenado, pois este algoritmo transforma o vetor, primeiramente, em um heap (uma espécie de árvore binária), destruindo a ordem que estava instaurada, para somente depois retorna-lá. Contudo, ele é um dos mais eficientes para o caso do vetor está de modo aleatório ou ordenado ao contrário. Já o “QuickSort” é o mais ineficiente para o caso do vetor está ordenado, isso decorre pela sua excessiva comparação de “mini-vetores”. E por último o “MergeSort” é um dos mais “equilibrados” algoritmos de ordenação, suas trocas são constantes devido seu método de intercalação e pode ser bem útil para ordenar vetores aleatórios e decrescentes de valores pequenos/medianos.