Trabalho Prático 02

Algoritmos de ordenação

Rafael Francisco Silva Granja

2021019629

Introdução:

O objetivo deste trabalho era a análise de diferentes métodos, algoritmos de ordenação em diferentes cenários, levando em conta o tamanho da entrada, o tempo de execução, quantidade de trocas e a quantidade de comparações realizadas.

A solução empregada para realizar esta análise foi a construção de uma estrutura que faz a leitura de um arquivo de entrada contendo em cada linha um tamanho de vetor, e para cada tamanho de vetor, criar um vetor daquele tamanho com elementos aleatórios, em seguida utilizar cada um dos métodos de ordenação propostos no trabalho, para ordenar o vetor, e ao final gerar um arquivo de saída contendo as estatísticas de cada método de ordenação para ser posteriormente analisado.

Método:

O uso de conceitos de POO garante a fluidez, o reuso, velocidade e fácil manutenção do código, apesar de não serem levados em conta neste TP.

Incialmente Existe uma Classe Vetor e uma Classe Analisa, a classe Vetor possui um objeto Analisa, que é onde acontecem as operações para gerar as estatísticas de cada método, além disso, a classe Vetor possui todos as funções comuns entre os métodos de ordenação, como a troca de elementos, preenchimento do vetor com elementos aleatórios, escrita das estatísticas no arquivo de saída, etc;

A implementação de fato dessa classes no código principal acontece da seguinte forma:

1. Leitura da linha de comando e tratamento da suas informações;
2. Identifica qual tipo de ordenação será usado;
3. Leitura de arquivo de entrada;
4. Loop para gerar estatísticas do método de ordenação escolhido para cada tamanho de vetor passado no arquivo de entrada;

Análise de Complexidade:

Em relação a complexidade do programa, a maior preocupação foi utilizar estruturas que não sobrepusessem a complexidade do próprio algoritmo de ordenação utilizado, então o programa só utiliza estruturas com O(n), sendo ‘n’ o tamanho do vetor de entrada, ou a quantidade de elementos no arquivo de entrada.

Dessa forma a Ordem de Complexidade, se resume a ordem de complexidade do algoritmo que está sendo utilizado Formalmente, todos são $\Omega(n*log(n))$.

Estratégia de Robustez:

O uso de conceitos de Programação Orientada a Objetos, reutilizando funções e adaptando classes para todos os métodos de ordenação garantem a robustez do código para qualquer tipo de manutenção, correção, adição ou remoção no código, da mesma forma que garantem segurança no acesso das informações de cada classe somente pelas classes permitidas.

Análise Experimental:

Para fazer a análise cada método de ordenação, foi utilizado a comparação em tabela, seguida de seleção para afunilamento, a seguir estão os resultados dos testes realizados:

Comparação entre Quicksort Recursivo, Quicksort Mediana, Quicksort Seleção, Quicksort não Recursivo, Quicksort Empilha Inteligente:

TEMPO DE EXECUÇÃO

	QuicksortRecursivo			QuicksortMediana			QuicksortSelecao			QuicksortNaoRecursivo			QuicksortEmpilha		
	TEMPO DE EXECUÇÃ	NUMERO DE TROCAS	NUMERO DE COMPARACOES	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES
1000	0.000617	6467	13766	0.000500	5991	12215	0.000566	7201	13709	0.002373	4473	10398	0.000840	4473	38371
5000	0.002067	36380	78608	0.001935	35968	70865	0.002458	41964	79582	0.002530	25809	56781	0.004254	25809	227445
10000	0.004421	86151	170892	0.004116	76807	153853	0.004539	95434	175770	0.004854	54281	116215	0.007817	54281	475139
50000	0.024759	505796	1012750	0.023255	428008	909457	0.025162	520141	1016086	0.026496	303484	626355	0.043235	303484	2773046
100000	0.051804	1015162	2280239	0.048254	980384	1964199	0.061972	1082427	2088347	0.053745	631333	1278699	0.089103	631333	5787479
500000	0.302689	6192156	12510824	0.280953	5686927	11052415	0.300099	6322575	11882288	0.280243	3294444	6268281	0.497822	3294444	31781274
1E+06	0.621315	12414993	26253287	0.597807	11926235	25059878	0.608367	12656835	24776732	0.575373	6853808	12800893	1.065.096	6853808	65671510

NÚMERO DE TROCAS

	QuicksortRecursivo			QuicksortMediana			QuicksortSelecao			QuicksortNaoRecursivo			QuicksortEmpilha		
	TEMPO DE EXECUÇÃ	NUMERO DE TROCAS	NUMERO DE COMPARACOES	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES
1000	0.000617	6467	13766	0.000500	5991	12215	0.000566	7201	13709	0.002373	4473	10398	0.000840	4473	38371
5000	0.002067	36380	78608	0.001935	35968	70865	0.002458	41964	79582	0.002530	25809	56781	0.004254	25809	227445
10000	0.004421	86151	170892	0.004116	76807	153853	0.004539	95434	175770	0.004854	54281	116215	0.007817	54281	475139
50000	0.024759	505796	1012750	0.023255	428008	909457	0.025162	520141	1016086	0.026496	303484	626355	0.043235	303484	2773046
100000	0.051804	1015162	2280239	0.048254	980384	1964199	0.061972	1082427	2088347	0.053745	631333	1278699	0.089103	631333	5787479
500000	0.302689	6192156	12510824	0.280953	5686927	11052415	0.300099	6322575	11882288	0.280243	3294444	6268281	0.497822	3294444	31781274
1E+06	0.621315	12414993	26253287	0.597807	11926235	25059878	0.608367	12656835	24776732	0.575373	6853808	12800893	1.065.096	6853808	65671510

NÚMERO DE COMPARAÇÕES

	QuicksortRecursivo			QuicksortMediana			QuicksortSelecao			QuicksortNaoRecursivo			QuicksortEmpilha		
	TEMPO DE EXECUÇÃ	NUMERO DE TROCAS	NUMERO DE COMPARACOES	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES	TEMPO DE EXECUÇÃ	NUMERO DE TROCAS	NUMERO DE COMPARACOES	TEMPO DE EXECUÇÃ	NUMERO DE TROCAS	NUMERO DE COMPARACOES
1000	0.000617	6467	13766	0.000500	5991	12215	0.000566	7201	13709	0.002373	4473	10398	0.000840	4473	38371
5000	0.002067	36380	78608	0.001935	35968	70865	0.002458	41964	79582	0.002530	25809	56781	0.004254	25809	227445
10000	0.004421	86151	170892	0.004116	76807	153853	0.004539	95434	175770	0.004854	54281	116215	0.007817	54281	475139
50000	0.024759	505796	1012750	0.023255	428008	909457	0.025162	520141	1016086	0.026496	303484	626355	0.043235	303484	2773046
100000	0.051804	1015162	2280239	0.048254	980384	1964199	0.061972	1082427	2088347	0.053745	631333	1278699	0.089103	631333	5787479
500000	0.302689	6192156	12510824	0.280953	5686927	11052415	0.300099	6322575	11882288	0.280243	3294444	6268281	0.497822	3294444	31781274
1E+06	0.621315	12414993	26253287	0.597807	11926235	25059878	0.608367	12656835	24776732	0.575373	6853808	12800893	1.065.096	6853808	65671510

Obs : Os dados podem ser consultados na planilha anexada junto ao arquivo de entrega na pasta src;

Foram analisados separadamente tempo de execução, número de trocas e número de comparações, apesar de em alguns testes o QuickSort Mediana apresentar menor tempo de execução, em relação aos outros quesitos, o quicksort não recursivo se mostrou dominante em relação aos testes e comparações realizadas.

Comparação entre Quicksort não Recursivo, MergeSort e HeapSort:

TEMPO DE EXECUÇÃO

Seed 1

	QuicksortNaoRecursivo			HeapSort			MergeSort		
	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES
1000	0.00066	4473	10398	0.00084	8118	29014	0.00078	8727	23715
5000	0.00247	25809	56781	0.00276	52080	179780	0.00347	55123	141946
10000	0.00503	54281	116215	0.00602	114148	389392	0.00613	120308	303940
50000	0.02715	303484	626355	0.03495	687583	2297353	0.03499	718206	1752688
100000	0.05533	631333	1278699	0.07168	1475330	4895464	0.07350	1536525	3705472
500000	0.30187	3294444	6268281	0.42809	8523711	27920793	0.39904	8837345	20813078
1000000	0.61093	6853808	12800833	0.91819	18048047	58842635	0.84623	18674438	43625884

Seed 2

	QuicksortNaoRecursivo			HeapSort			MergeSort		
	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES
1000	0.00069	4498	10377	0.00065	8097	28995	0.00072	8739	23727
5000	0.00252	25863	56858	0.00277	52106	179800	0.00302	55204	142027
10000	0.00523	54400	116474	0.00574	114076	389312	0.00603	120382	304014
50000	0.02701	304703	627965	0.03435	687387	2297021	0.03467	718180	1752662
100000	0.05589	635266	1282136	0.07177	1475089	4894861	0.07085	1536550	3705497
500000	0.28373	3295632	6267151	0.41923	8524341	27922069	0.40492	8837212	20812945
1000000	0.59952	6840033	12785129	0.92494	18047069	58840629	0.83722	18674657	43626103

Seed 3

	QuicksortNaoRecursivo			HeapSort			MergeSort		
	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES
1000	0.00067	4518	10343	0.00074	8069	28921	0.000756	8678	23666
5000	0.00249	26080	57039	0.00292	52148	179860	0.002937	55136	141959
10000	0.00530	54427	116462	0.00741	114258	389696	0.006062	120350	303982
50000	0.02728	303615	627712	0.03336	687798	2297780	0.034006	717969	1752451
100000	0.05630	631341	1278107	0.07187	1474690	4894126	0.070912	1536256	3705203
500000	0.29277	3290501	6264415	0.42281	8523748	27919708	0.403860	8837180	20812913
1000000	0.59309	6866165	12811269	0.93769	18048997	58842865	0.832454	18674499	43625945

NÚMERO DE TROCAS

Seed 1

	QuicksortNaoRecursivo			HeapSort			MergeSort		
	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES
1000	0.00066	4473	10398	0.00084	8118	29014	0.00078	8727	23715
5000	0.00247	25809	56781	0.00276	52080	179780	0.00347	55123	141946
10000	0.00503	54281	116215	0.00602	114148	389392	0.00613	120308	303940
50000	0.02715	303484	626355	0.03495	687583	2297353	0.03499	718206	1752688
100000	0.05533	631333	1278699	0.07168	1475330	4895464	0.07350	1536525	3705472
500000	0.30187	3294444	6268281	0.42809	8523711	27920793	0.39904	8837345	20813078
1000000	0.61093	6853808	12800833	0.91819	18048047	58842635	0.84623	18674438	43625884

Seed 2

	QuicksortNaoRecursivo			HeapSort			MergeSort		
	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES
1000	0.00069	4498	10377	0.00065	8097	28995	0.00072	8739	23727
5000	0.00252	25863	56858	0.00277	52106	179800	0.00302	55204	142027
10000	0.00523	54400	116474	0.00574	114076	389312	0.00603	120382	304014
50000	0.02701	304703	627965	0.03435	687387	2297021	0.03467	718180	1752662
100000	0.05589	635266	1282136	0.07177	1475089	4894861	0.07085	1536550	3705497
500000	0.28373	3295632	6267151	0.41923	8524341	27922069	0.40492	8837212	20812945
1000000	0.59952	6840033	12785129	0.92494	18047069	58840629	0.83722	18674657	43626103

Seed 3

	QuicksortNaoRecursivo			HeapSort			MergeSort		
	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES
1000	0.00067	4518	10343	0.00074	8069	28921	0.000756	8678	23666
5000	0.00249	26080	57039	0.00292	52148	179860	0.002937	55136	141959
10000	0.00530	54427	116462	0.00741	114258	389696	0.006062	120350	303982
50000	0.02728	303615	627712	0.03336	687798	2297780	0.034006	717969	1752451
100000	0.05630	631341	1278107	0.07187	1474690	4894126	0.070912	1536256	3705203
500000	0.29277	3290501	6264415	0.42281	8523748	27919708	0.403860	8837180	20812913
1000000	0.59309	6866165	12811269	0.93769	18048997	58842865	0.832454	18674499	43625945

NÚMERO DE COMPARAÇÕES

Seed 1

	QuicksortNaoRecursivo			HeapSort			MergeSort		
	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES
1000	0.00066	4473	10398	0.00084	8118	29014	0.00078	8727	23715
5000	0.00247	25809	56781	0.00276	52080	179780	0.00347	55123	141946
10000	0.00503	54281	116215	0.00602	114148	389392	0.00613	120308	303940
50000	0.02715	303484	626355	0.03495	687583	2297353	0.03499	718206	1752688
100000	0.05533	631333	1278699	0.07168	1475330	4895464	0.07350	1536525	3705472
500000	0.30187	3294444	6268281	0.42809	8523711	27920793	0.39904	8837345	20813078
1000000	0.61093	6853808	12800833	0.91819	18048047	58842635	0.84623	18674438	43625884

Seed 2

	QuicksortNaoRecursivo			HeapSort			MergeSort		
	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES
1000	0.00069	4498	10377	0.00065	8097	28995	0.00072	8739	23727
5000	0.00252	25863	56858	0.00277	52106	179800	0.00302	55204	142027
10000	0.00523	54400	116474	0.00574	114076	389312	0.00603	120382	304014
50000	0.02701	304703	627965	0.03435	687387	2297021	0.03467	718180	1752662
100000	0.05589	635266	1282136	0.07177	1475089	4894861	0.07085	1536550	3705497
500000	0.28373	3295632	6267151	0.41923	8524341	27922069	0.40492	8837212	20812945
1000000	0.59952	6840033	12785129	0.92494	18047069	58840629	0.83722	18674657	43626103

Seed 3

	QuicksortNaoRecursivo			HeapSort			MergeSort		
	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES
1000	0.00067	4518	10343	0.00074	8069	28921	0.000756	8678	23666
5000	0.00249	26080	57039	0.00292	52148	179860	0.002937	55136	141959
10000	0.00530	54427	116462	0.00741	114258	389696	0.006062	120350	303982
50000	0.02728	303615	627712	0.03336	687798	2297780	0.034006	717969	1752451
100000	0.05630	631341	1278107	0.07187	1474690	4894126	0.070912	1536256	3705203
500000	0.29277	3290501	6264415	0.42281	8523748	27919708	0.403860	8837180	20812913
1000000	0.59309	6866165	12811269	0.93769	18048997	58842865	0.832454	18674499	43625945

Obs : Os dados podem ser consultados na planilha anexada junto ao arquivo de entrega na pasta src;

Neste caso foram utilizadas 3 seeds diferentes para geração dos vetores e da mesma forma como no primeiro teste, foram analisados separadamente tempo de execução, número de trocas e número de comparações, em 98% das comparações o QuickSortNãoRecursivo se mostrou superior;

Conclusão:

De forma objetiva, pelos testes realizados, o QuickSort não recursivo se mostrou o melhor método de ordenação de vetores, e em relação ao trabalho de forma geral, podem existir erros quanto ao método de contagem de comparações ou trocas, porém fiz questão de contabilizar o tempo de execução apenas do algoritimo, tendo confiança para usar como base na tomada de decisão.

O trabalho se mostrou muito útil para mostrar o quão aprofundada pode ser uma análise de algoritmo, porém em relação as atividades propostas, foram um pouco confusas e não muti claras, que dificultaram o uso dos diferentes métodos.

Instruções para compilação:

Para utilizar o makefile, existem 3 comandos:

- make clean : limpas os arquivos na pasta bin e obj;
- make first: compila o código, limpa o arquivo de saida e o gera novamente, atualizado com o teste1, o objeto na pasta obj e o executável na pasta bin;
- make second: compila o código, limpa o arquivo de saida2 e o gera novamente, atualizado com o teste2, o objeto na pasta obj e o executável na pasta bin;