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:

	QuicksortRecursivo			QuicksortMediana			QuicksortSelecao			QuicksortNaoRecursivo			QuicksortEmpilha		
	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	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES
1000	0.002476	6467	13766	0.004717	5991	12215	0.002497	7201	13709	0.003272	4473	10398	0.002773	4473	38371
5000	0.010539	36380	78608	0.012953	35968	70865	0.012123	41964	79582	0.012983	25809	56781	0.013880	25809	227445
10000	0.023326	86151	170892	0.030499	76807	153853	0.035005	95434	175770	0.025668	54281	116215	0.038425	54281	475139
50000	0.133500	505796	1012750	0.228168	428008	909457	0.154172	520141	1016086	0.134142	303484	626355	0.188428	303484	2773046
100000	0.268195	1015162	2280239	1.020.914	980384	1964199	0.237019	1082427	2088347	0.255617	631333	1278699	0.348561	631333	5787479
500000	1.467.345	6192156	12510824	6.330.667	5686927	11052415	1.838.044	6322575	11882288	1.950.176	3294444	6268281	1.517.729	3294444	31781274
1000000	6.652.489	12414993	26253287	10.338.401	11926235	25059878	9.912.579	12656835	24776732	2.833.070	6853808	12800893	3.243.460	6853808	65671510

NÚMERO DE TROCAS

	QuicksortRecursivo			QuicksortMediana			QuicksortSelecao			QuicksortNaoRecursivo			QuicksortEmpilha		
	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	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES
1000	0.002476	6467	13766	0.004717	5991	12215	0.002497	7201	13709	0.003272	4473	10398	0.002773	4473	38371
5000	0.010539	36380	78608	0.012953	35968	70865	0.012123	41964	79582	0.012983	25809	56781	0.013880	25809	227445
10000	0.023326	86151	170892	0.030499	76807	153853	0.035005	95434	175770	0.025668	54281	116215	0.038425	54281	475139
50000	0.133500	505796	1012750	0.228168	428008	909457	0.154172	520141	1016086	0.134142	303484	626355	0.188428	303484	2773046
100000	0.268195	1015162	2280239	1.020.914	980384	1964199	0.237019	1082427	2088347	0.255617	631333	1278699	0.348561	631333	5787479
500000	1.467.345	6192156	12510824	6.330.667	5686927	11052415	1.838.044	6322575	11882288	1.950.176	3294444	6268281	1.517.729	3294444	31781274
1000000	6.652.489	12414993	26253287	10.338.401	11926235	25059878	9.912.579	12656835	24776732	2.833.070	6853808	12800893	3.243.460	6853808	65671510

NÚMERO DE COMPARAÇÕES

	QuicksortRecursivo			QuicksortMediana			QuicksortSelecao			QuicksortNaoRecursivo			QuicksortEmpilha		
	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	TEMPO DE EXECUÇÃO	NUMERO DE TROCAS	NUMERO DE COMPARACOES
1000	0.002476	6467	13766	0.004717	5991	12215	0.002497	7201	13709	0.003272	4473	10398	0.002773	4473	38371
5000	0.010539	36380	78608	0.012953	35968	70865	0.012123	41964	79582	0.012983	25809	56781	0.013880	25809	227445
10000	0.023326	86151	170892	0.030499	76807	153853	0.035005	95434	175770	0.025668	54281	116215	0.038425	54281	475139
50000	0.133500	505796	1012750	0.228168	428008	909457	0.154172	520141	1016086	0.134142	303484	626355	0.188428	303484	2773046
100000	0.268195	1015162	2280239	1.020.914	980384	1964199	0.237019	1082427	2088347	0.255617	631333	1278699	0.348561	631333	5787479
500000	1.467.345	6192156	12510824	6.330.667	5686927	11052415	1.838.044	6322575	11882288	1.950.176	3294444	6268281	1.517.729	3294444	31781274
1000000	6.652.489	12414993	26253287	10.338.401	11926235	25059878	9.912.579	12656835	24776732	2.833.070	6853808	12800893	3.243.460	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	0.002179	4473	10398	0.002524	8118	29014	0.002570	8727	23715
5000	0.010749	25809	56781	0.013522	52080	179780	0.012444	55123	141946
10000	0.049286	54281	116215	0.028638	114148	389392	0.030328	120308	303940
50000	0.173887	303484	626355	0.179750	687583	2297353	0.150989	784481	1752688
100000	0.288865	631333	1278699	0.507915	1475330	4895464	0.245615	1668946	3705472
500000	1.463.744	3294444	6268281	3.524.897	8523711	27920793	1.372.291	9475732	20813078
1000000	18.370.902	6853808	12800893	7.818.524	18048047	58842635	5.457.547	19951445	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	0.002851	4498	10377	0.002868	8097	28995	0.003437	9987	23727
5000	0.012060	25863	56858	0.012373	52106	179800	0.017316	61822	142027
10000	0.023967	54400	116474	0.025774	114076	389312	0.029937	133631	304014
50000	0.144063	304703	627965	0.126962	687387	2297021	0.149111	784481	1752662
100000	0.231431	635266	1282136	0.262891	1475089	4894861	0.298477	1668946	3705497
500000	1.323.619	3295632	6267151	1.605.682	8524341	27922069	1.364.556	9475732	20812945
1000000	2.663.101	6840033	12785129	3.585.563	18047069	58840629	2.986.111	19951445	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	0.002778	4518	10343	0.002727	8069	28921	0.003112	9987	23666
5000	0.012719	26080	57039	0.012028	52148	179860	0.002937	61822	141959
10000	0.024896	54427	116462	0.025142	114258	389696	0.012696	133631	303982
50000	0.159086	303615	627712	0.134724	687798	2297780	0.126653	717969	1752451
100000	0.230627	631341	1278107	0.274697	1474690	4894126	0.244787	1668946	3705203
500000	1.239.889	3290501	6264415	1.694.084	8523748	27919708	1.279.707	9475732	20812913
1000000	2.787.940	6866165	12811269	3.611.629	18048997	58842865	2.826.193	19951445	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	0.002179	4473	10398	0.002524	8118	29014	0.002570	8727	23715
5000	0.010749	25809	56781	0.013522	52080	179780	0.012444	55123	141946
10000	0.049286	54281	116215	0.028638	114148	389392	0.030328	120308	303940
50000	0.173887	303484	626355	0.179750	687583	2297353	0.150989	784481	1752688
100000	0.288865	631333	1278699	0.507915	1475330	4895464	0.245615	1668946	3705472
500000	1.463.744	3294444	6268281	3.524.897	8523711	27920793	1.372.291	9475732	20813078
1000000	18.370.902	6853808	12800893	7.818.524	18048047	58842635	5.457.547	19951445	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	0.002851	4498	10377	0.002868	8097	28995	0.003437	9987	23727
5000	0.012060	25863	56858	0.012373	52106	179800	0.017316	61822	142027
10000	0.023967	54400	116474	0.025774	114076	389312	0.029937	133631	304014
50000	0.144063	304703	627965	0.126962	687387	2297021	0.149111	784481	1752662
100000	0.231431	635266	1282136	0.262891	1475089	4894861	0.298477	1668946	3705497
500000	1.323.619	3295632	6267151	1.605.682	8524341	27922069	1.364.556	9475732	20812945
1000000	2.663.101	6840033	12785129	3.585.563	18047069	58840629	2.986.111	19951445	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	0.002778	4518	10343	0.002727	8069	28921	0.003112	9987	23666
5000	0.012719	26080	57039	0.012028	52148	179860	0.002937	61822	141959
10000	0.024896	54427	116462	0.025142	114258	389696	0.012696	133631	303982
50000	0.159086	303615	627712	0.134724	687798	2297780	0.126653	717969	1752451
100000	0.230627	631341	1278107	0.274697	1474690	4894126	0.244787	1668946	3705203
500000	1.239.889	3290501	6264415	1.694.084	8523748	27919708	1.279.707	9475732	20812913
1000000	2.787.940	6866165	12811269	3.611.629	18048997	58842865	2.826.193	19951445	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	0.002179	4473	10398	0.002524	8118	29014	0.002570	8727	23715
5000	0.010749	25809	56781	0.013522	52080	179780	0.012444	55123	141946
10000	0.049286	54281	116215	0.028638	114148	389392	0.030328	120308	303940
50000	0.173887	303484	626355	0.179750	687583	2297353	0.150989	784481	1752688
100000	0.288865	631333	1278699	0.507915	1475330	4895464	0.245615	1668946	3705472
500000	1.463.744	3294444	6268281	3.524.897	8523711	27920793	1.372.291	9475732	20813078
1000000	18.370.902	6853808	12800893	7.818.524	18048047	58842635	5.457.547	19951445	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	0.002851	4498	10377	0.002868	8097	28995	0.003437	9987	23727
5000	0.012060	25863	56858	0.012373	52106	179800	0.017316	61822	142027
10000	0.023967	54400	116474	0.025774	114076	389312	0.029937	133631	304014
50000	0.144063	304703	627965	0.126962	687387	2297021	0.149111	784481	1752662
100000	0.231431	635266	1282136	0.262891	1475089	4894861	0.298477	1668946	3705497
500000	1.323.619	3295632	6267151	1.605.682	8524341	27922069	1.364.556	9475732	20812945
1000000	2.663.101	6840033	12785129	3.585.563	18047069	58840629	2.986.111	19951445	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	0.002778	4518	10343	0.002727	8069	28921	0.003112	9987	23666
5000	0.012719	26080	57039	0.012028	52148	179860	0.002937	61822	141959
10000	0.024896	54427	116462	0.025142	114258	389696	0.012696	133631	303982
50000	0.159086	303615	627712	0.134724	687798	2297780	0.126653	717969	1752451
100000	0.230627	631341	1278107	0.274697	1474690	4894126	0.244787	1668946	3705203
500000	1.239.889	3290501	6264415	1.694.084	8523748	27919708	1.279.707	9475732	20812913
1000000	2.787.940	6866165	12811269	3.611.629	18048997	58842865	2.826.193	19951445	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;