Universidade Federal de Viçosa – Campus Florestal

Ciência da Computação

Carlos Alberto de Carvalho Antônio - 2283

Samuel Jhonata Soares Tavares - 2282

Métodos de Ordenação e Funções de Complexidade

Florestal

2015

Sumário

1.0 INTRODUÇÃO

Neste relatório, estarão contidas informações acerca da implementação dos métodos de ordenação vistos em aula: Seleção, Inserção, Quicksort, Heapsort e Shellsort. Que serão executados sobre um vetor de 10000 posições. A partir disso, é possível fazer contabilidade dos gastos dessas ordenações, sendo estas: comparações, movimentações e o tempo gasto. Sendo que tais dados, serão armazenados através de arquivos de texto no formato .txt que serão gerados pelo próprio programa na main.c. E posteriormente, estes dados serão utilizados para uma análise de complexidade que, será útil para descobrir qual é o algoritmo mais interessante para cada grupo de números solicitado.

Para execução das ordenações dos 13 conjuntos de números dados, 30 vezes cada, foi criado um vetor de 13 posições, que juntamente à um comando de repetição FOR, faz a passagem desses 13 grupos de números, executando cada ordenação às 30 vezes como solicitado.

Lembrando, que todas as operações são feitas sobre o mesmo vetor de 10000 posições, sendo que para cada nova execução, de acordo com o grupo de números a ser ordenado, são gerados novos números neste mesmo vetor, através de uma função chamada “Criar”.

2.0 DESENVOLVIMENTO

Para criação do programa, inicialmente, foi preciso criar as funções de ordenação, sendo o vetor.h o cabeçalho que contêm os escopos das funções. O vetor.c, que é o arquivo que contém as funções e suas operações. E finalmente, o arquivo principal main.c que é utilizada para execução das ordenações, e também, para registro dos dados em arquivos externos.

Na main.c, foram criadas variáveis do tipo inteiro, para execução dos comandos de repetição necessários, vetor de inteiros, que será o conjunto de números a ser ordenado, outros inteiros, que serão para armazenar as quantidades de comparações, movimentações e o tempo gasto para ordenação, e também uma variável do tipo arquivo, que é onde serão armazenadas as informações acerca da ordenação.

Essas quantidades de comparações e operações serão advindas de contadores a serem utilizados nas funções contidas no arquivo vetor.c. Sendo que de acordo com as comparações e movimentações feitas na execução do algoritmo, serão retornados os valores e armazenados nos inteiros destinados a salvar esses dados.

Para cada algoritmo de ordenação, é usado um comando que cria um arquivo, para armazenamento de informações, e então. Usamos dois comandos de repetição FOR aninhados, sendo que um deles irá percorrer o vetor que contêm os grupos de números, e o outro irá efetuar as 30 repetições solicitadas. Desta forma, dentro destes comandos, inicialmente são gerados números para o vetor de 10000 posições até o limite do grupo de inteiros, e repetidos 30 vezes. Sendo que para cada uma das 30 vezes, um novo vetor com números diferentes será, criado, e assim posteriormente, ordenado. E durante a execução da ordenação, é usada uma função de cálculo de tempo, que nos retornará quanto tempo foi desprendido para ordenar aquela quantidade de números. E finalmente, estes valores, juntamente com a quantidade de comparações e movimentações, serão impressos no arquivo criado no início do programa para cada caso feito.

Neste trabalho, não contamos com nenhuma parte direcionada à um usuário final, todas as informações e necessidades do programa são supridas com os recursos até então citados.

2.1 SELEÇÃO

a) Inicialmente, através da análise das comparações e movimentações, pudemos obter valores mínimos, máximos e médios para a quantidade de cada um, variando em acordo com a quantidade de elementos contidos na entrada (N). Sendo que tais valores podem ser representados pela tabela:

Seleção	Comparações			Movimentações		
Entradas	Mínimo	Máximo	Média	Mínimo	Máximo	Média
10	45	45	45	27	27	27
100	4950	4950	4950	297	297	297
500	124750	124750	124750	1497	1497	1497
1000	499500	499500	499500	2997	2997	2997
2000	1999000	1999000	1999000	5997	5997	5997
3000	4498500	4498500	4498500	8997	8997	8997
4000	7998000	7998000	7998000	11997	11997	11997
5000	12497500	12497500	12497500	14997	14997	14997
6000	17997000	17997000	17997000	17997	17997	17997
7000	24496500	24496500	24496500	20997	20997	20997
8000	31996000	31996000	31996000	23997	23997	23997
9000	40495500	40495500	40495500	26997	26997	26997
10000	49995000	49995000	49995000	29997	29997	29997

b) Posteriormente, conseguimos também obter valores médios para o tempo gasto, em cada teste, novamente, variando de acordo com a alteração da quantidade de elementos (N). Como pode ser observado nesta tabela:

Seleção	Tempo
Entradas	Média
10	7,42E-07
100	3,2E-05
500	0,00068
1000	0,002349
2000	0,009076
3000	0,019296
4000	0,030348
5000	0,050564
6000	0,077218
7000	0,104829
8000	0,136616
9000	0,16156
10000	0,195226

2.2 INSERÇÃO

a) Da mesma maneira, através da análise das comparações e movimentações, pudemos obter valores mínimos, máximos e médios para a quantidade de cada um, variando em acordo com a quantidade de elementos contidos na entrada (N). Sendo que tais valores podem ser representados pela tabela:

Inserção	Comparações			Movimentações		
Entradas	Mínimo	Máximo	Média	Mínimo	Máximo	Média
10	17	36	25,3	33	53	41,2
100	2121	2814	2476,6	2314	3012	2670,1
500	58558	65454	62063,57	59552	66444	63055,57
1000	239347	261030	250609,5	241337	263024	252600,8
2000	965794	1036790	1000072	969789	1040779	1004063
3000	2183917	2287400	2245849	2189905	2293392	2251840
4000	3923816	4093472	4012533	3931798	4101466	4020523
5000	6116773	6367554	6238913	6126765	6377543	6248903
6000	8830977	9138066	8997136	8842967	9150055	9009126
7000	1,2E+07	12551993	12233282	12106300	12565981	12247271
8000	1,6E+07	16258717	15998197	15759611	16274705	16014186
9000	2E+07	20542616	20247688	20037179	20560607	20265677
10000	2,4E+07	25233390	24964669	24514518	25253381	24984658

b) E ainda, conseguimos também obter valores médios para o tempo gasto, em cada teste, novamente, variando de acordo com a alteração da quantidade de elementos (N). Como pode ser observado nesta tabela:

Inserção	Tempo
Entradas	Média
10	3,2E-07
100	1,25E-05
500	0,000283
1000	0,001525
2000	0,005039
3000	0,012072
4000	0,020727
5000	0,035339
6000	0,045832
7000	0,068272
8000	0,088509
9000	0,122893
10000	0,145744

2.3 QUICKSORT

a) Analogamente, através da análise das comparações e movimentações, pudemos obter valores mínimos, máximos e médios para a quantidade de cada um, variando em acordo com a quantidade de elementos contidos na entrada (N). Sendo que tais valores podem ser representados pela tabela:

Quicksort	Comparações			Movimentações		
Entradas	Mínimo	Máximo	Média	Mínimo	Máximo	Média
10	16	27	21,53333333	24	36	30,7
100	466	677	562,5666667	519	594	547,9
500	3567	5312	4097,5	3411	3630	3516
1000	7923	11319	8942,633333	7593	7956	7790,5
2000	18050	22460	19839,46667	16599	17376	17015,9
3000	27860	37401	31351,93333	26283	27219	26873,2
4000	39364	49232	43379,46667	36573	37548	37117,9
5000	52147	64329	55578,26667	47055	48252	47703,5
6000	61991	83939	68731,53333	56829	59493	58418,1
7000	74073	93015	82288,46667	68079	70860	69474,6
8000	86226	108152	94326,5	78849	81663	80692,7
9000	99291	118056	106808,0667	90978	93507	92277,6
10000	112320	134093	119772,6333	102501	105042	103711,7

b) E assim, conseguimos também obter valores médios para o tempo gasto, em cada teste, novamente, variando de acordo com a alteração da quantidade de elementos (N). Como pode ser observado nesta tabela:

Quicksort	Tempo
Entradas	Média
10	9,05E-07
100	1,53E-05
500	9,24E-05
1000	0,000194
2000	0,000405
3000	0,000658
4000	0,000932
5000	0,001164
6000	0,001176
7000	0,001509
8000	0,001602
9000	0,001969
10000	0,002195

2.4 SHELLSORT

a) Novamente, através da análise das comparações e movimentações, pudemos obter valores mínimos, máximos e médios para a quantidade de cada um, variando em acordo com a quantidade de elementos contidos na entrada (N). Sendo que tais valores podem ser representados pela tabela:

Shellsort	Comparações			Movimentações		
Entradas	Mínimo	Máximo	Média	Mínimo	Máximo	Média
10	16	26	20,13333333	39	52	44
100	498	662	571,8333333	1027	1197	1113,3
500	4214	4928	4604,5	8099	8881	8537,4
1000	10280	12442	10901,83333	19079	21333	19765,9
2000	24472	30177	26081	44548	50477	46174,2
3000	39392	48089	42932,76667	71028	79990	74702,16667
4000	55742	66739	60972,33333	99669	111323	105249,3667
5000	74485	86588	80069,56667	131522	144363	137437,4333
6000	94857	106186	99625,4	165202	177131	170258,6333
7000	113853	129731	121913,6333	197469	214038	205907,4333
8000	133286	154190	142518,6	230280	251615	239786,4667
9000	157512	183580	165415,4	267999	294649	276254
10000	174207	195718	186667	298273	320440	311222,9667

b) E finalmente, conseguimos também obter valores médios para o tempo gasto, em cada teste, novamente, variando de acordo com a alteração da quantidade de elementos (N). Como pode ser observado nesta tabela:

Shellsort	Tempo
Entradas	Média
10	7,02E-07
100	9,54E-06
500	7,14E-05
1000	0,000197
2000	0,000393
3000	0,000711
4000	0,001042
5000	0,001397
6000	0,001771
7000	0,001865
8000	0,002114
9000	0,002505
10000	0,002874

2.5 HEAPSORT

a) De novo, através da análise das comparações e movimentações, pudemos obter valores mínimos, máximos e médios para a quantidade de cada um, variando em acordo com a quantidade de elementos contidos na entrada (N). Sendo que tais valores podem ser representados pela tabela:

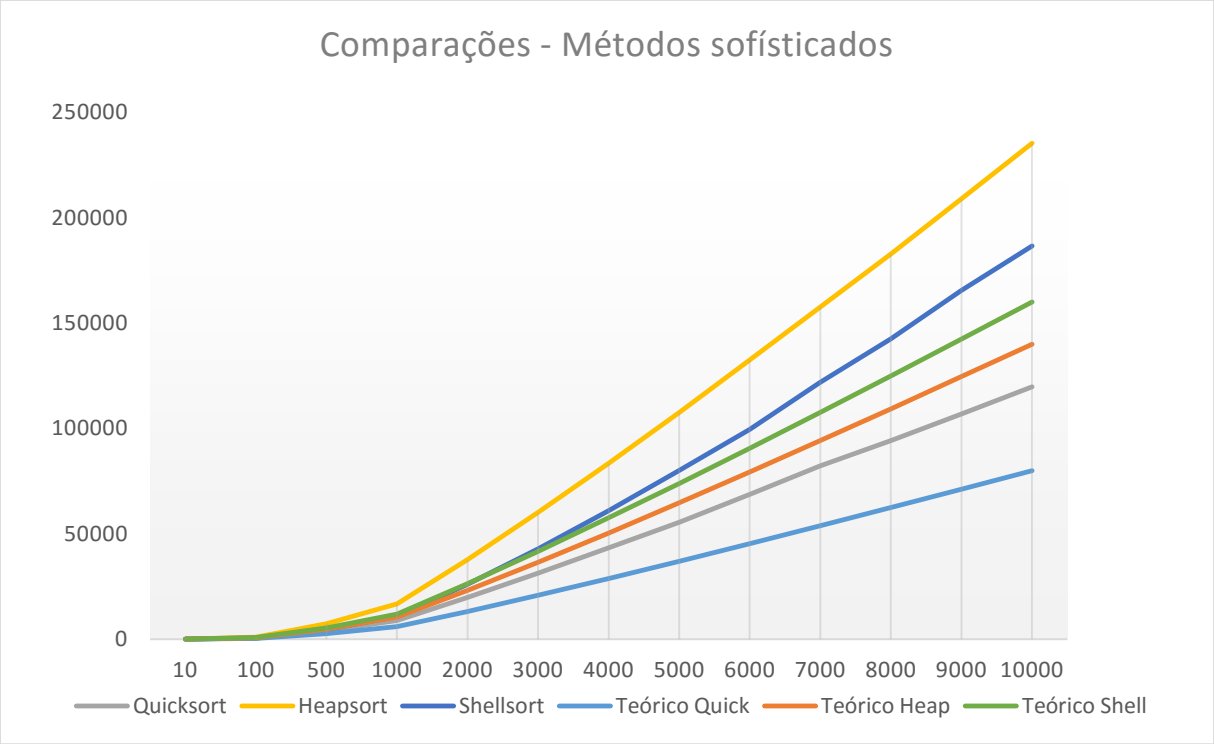
Heapsort	Comparações			Movimentações		
Entradas	Mínimo	Máximo	Média	Mínimo	Máximo	Média
10	31	41	36,2	66	75	70,8
100	1006	1047	1026,2	1059	1088	1076,733333
500	7381	7463	7424,533333	6498	6567	6537
1000	16782	16915	16846,83333	14032	14123	14074,06667
2000	37637	37765	37696,56667	30101	30235	30149,43333
3000	60091	60347	60231,23333	46980	47157	47062,86667
4000	83312	83584	83427,33333	64224	64419	64316,63333
5000	107558	107764	107670,1333	82015	82181	82088,73333
6000	132331	132581	132465,4333	99999	100235	100149,7
7000	157447	157748	157599	118268	118503	118370,8667
8000	182627	182958	182815,7333	136463	136769	136618,8
9000	208687	209113	208895,9	155167	155487	155313,9667
10000	235077	235479	235346,3667	174033	174316	174184,8667

b) E finalmente, conseguimos também obter valores médios para o tempo gasto, em cada teste, novamente, variando de acordo com a alteração da quantidade de elementos (N). Como pode ser observado nesta tabela:

Heapsort	Tempo
Entradas	Média
10	6,01E-07
100	1,18E-05
500	7,77E-05
1000	0,000171
2000	0,000362
3000	0,000584
4000	0,000791
5000	0,001007
6000	0,001211
7000	0,001453
8000	0,001679
9000	0,001964
10000	0,002139

4.0 GRÁFICOS

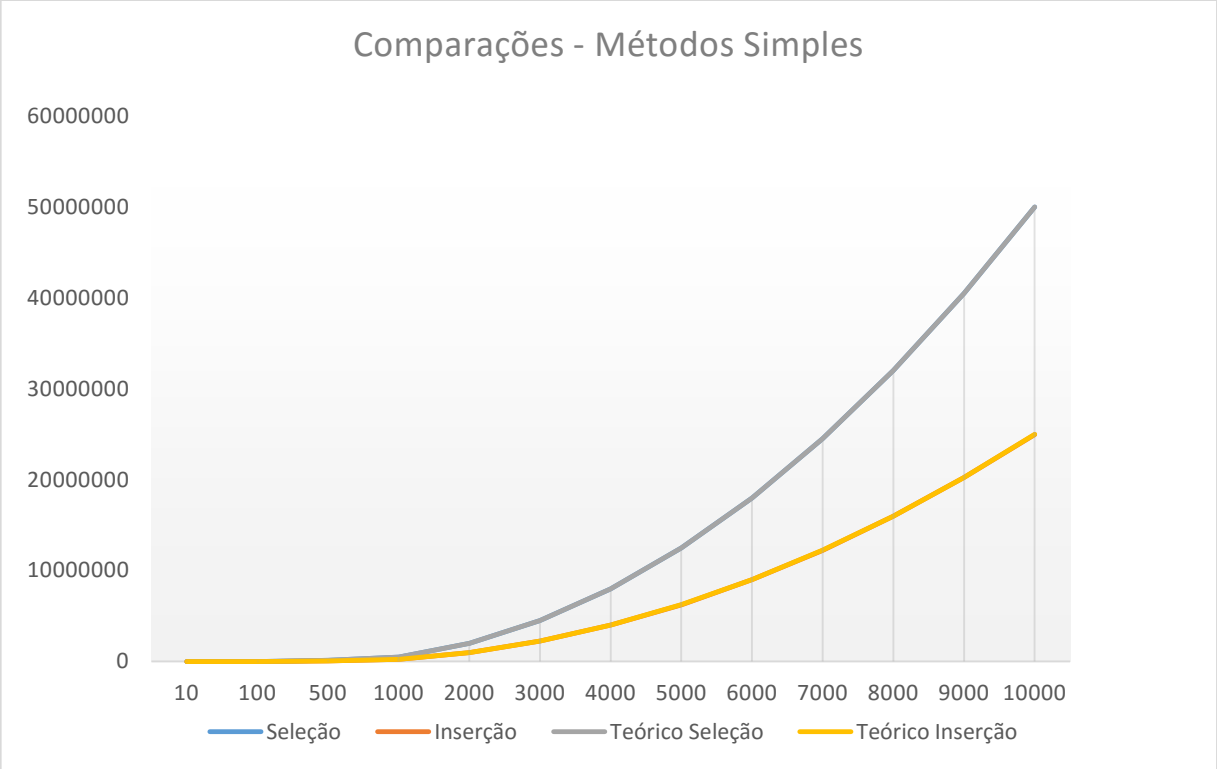
A partir da execução das ordenações solicitadas, conseguimos obter os seguintes gráficos:



Inicialmente, analisamos o gráfico de comparação que engloba as funções Quicksort, Heapsort e Shellsort, e também os valores teóricos de quantidade de comparações para cada uma dessas ordenações. Sendo que, para as três foram utilizados O(n)=nlogn sendo que assim, são desconsideradas constantes. E desta maneira, foram feitas multiplicações por constantes para que seus valores se aproximassem do ideal. Foram valores teoricamente bem próximos, se analisado de perto.

Dentre tais resultados, pode-se observar que nesta escala, os desempenhos dos três métodos de ordenação para a comparação são bem semelhantes. A se destacar que o Quicksort tem o menor número de comparações, seguido do Shellsort, e finalmente o Heapsort. Portanto, obtemos que em termo de comparações:

- Quicksort < Shellsort < Heapsort.



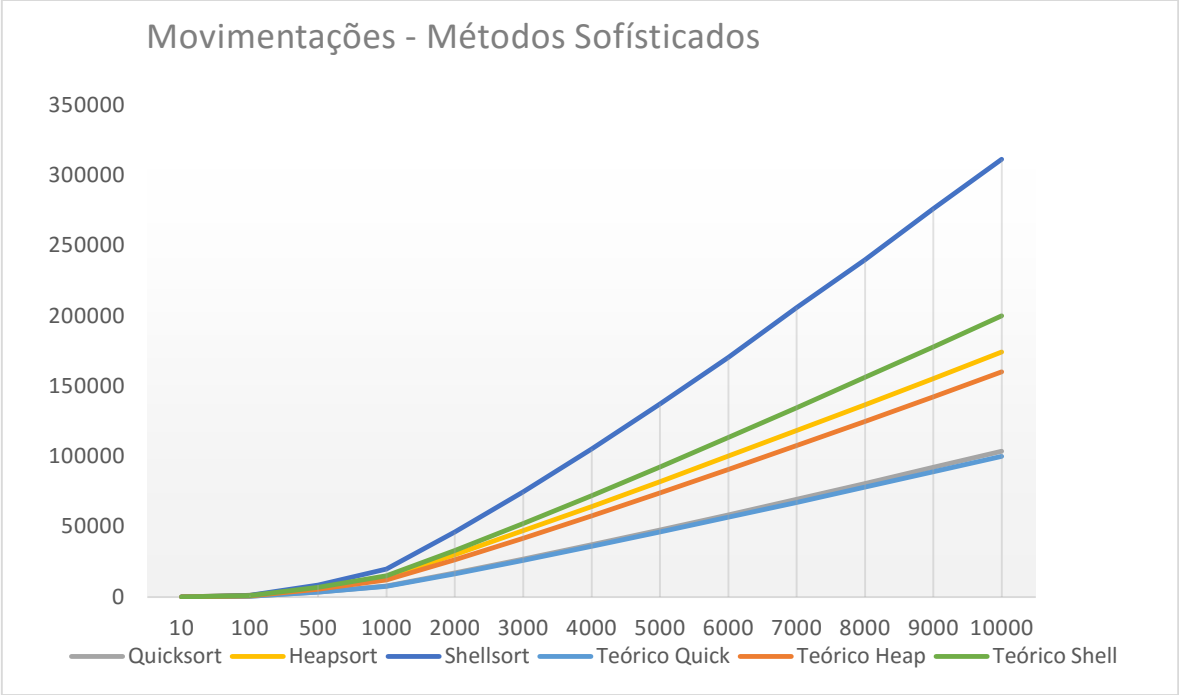
Agora, a análise feita foi a dos métodos de ordenação simples Seleção e Inserção, com seus respectivos, também, valores teóricos para número de comparações. Sendo que, como pode-se ver, só se enxerga duas funções, e isso se deve ao fato de que na prática, essas duas ordenações tiveram valores muito semelhantes, senão, iguais. As funções para Seleção e Inserção foram O(n)=((n^2)-n)/2 e O(n)=((n^2)/4)+(n/4)-(1/2) respectivamente.

Entre os dois métodos simples estudados, pode-se observar que o Inserção tem o menor número de comparações feitas, portanto, neste quesito, torna-se o melhor. Entre eles, obtemos:

- Inserção < Seleção.

Se for efetuada uma comparação entre todos os métodos de ordenação estudados no quesito comparações, obtêm-se a ordem.

- Quicksort < Shellsort < Heapsort < Inserção < Seleção



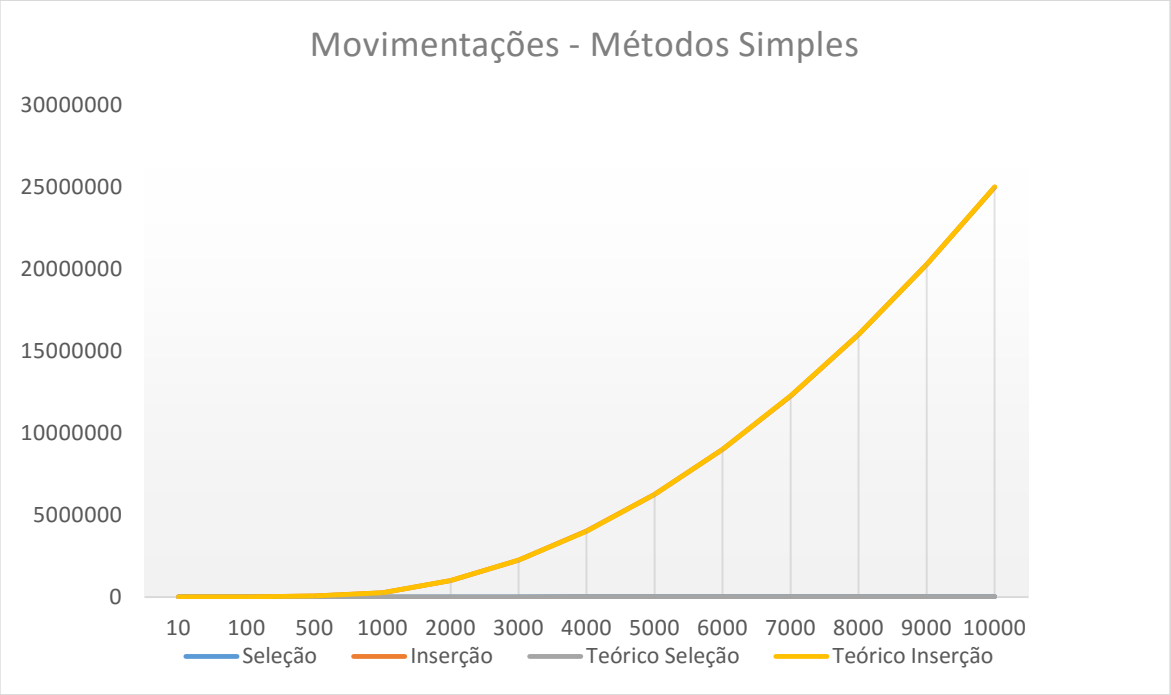
A partir deste gráfico de Movimentações, pudemos notar os valores de movimentações desprendidas no métodos de ordenação sofisticados Quicksort, Heapsort e Shellsort, e também seus respectivos valores teóricos para quantidade de movimentações.

As funções usadas para tal análise foram O(n)=nlogn para os três, havendo pequenas alterações nas constantes multiplicadas, uma vez que a complexidade assintótica ignora as constantes das funções de complexidade.

Os valores obtidos, se comparados aos teóricos, foram muito próximos, uma vez que, no gráficos, os valores se aproximam bastante.

Ao se comparar os três métodos de ordenação, obtêm-se:

- Quicksort < Heapsort < Shellsort



Agora é apresentado, as curvas que representam a quantidade de movimentação para cada caso nos métodos de ordenação simples, Seleção e Inserção, e também, seus respectivos valores teóricos para essas quantidades. Sendo que as funções utilizadas para Seleção e Inserção foram respectivamente O(n)=3(n-1) e O(n)=((n^2)+n-2)/4

Pode-se observar que não se enxergam facilmente os valores de movimentações para o método seleção, isso deve ao fato de suas movimentações serem muitos baixas e lineares às entradas e também, à grandeza de movimentações do método de inserção. Neste caso, obtém-se:

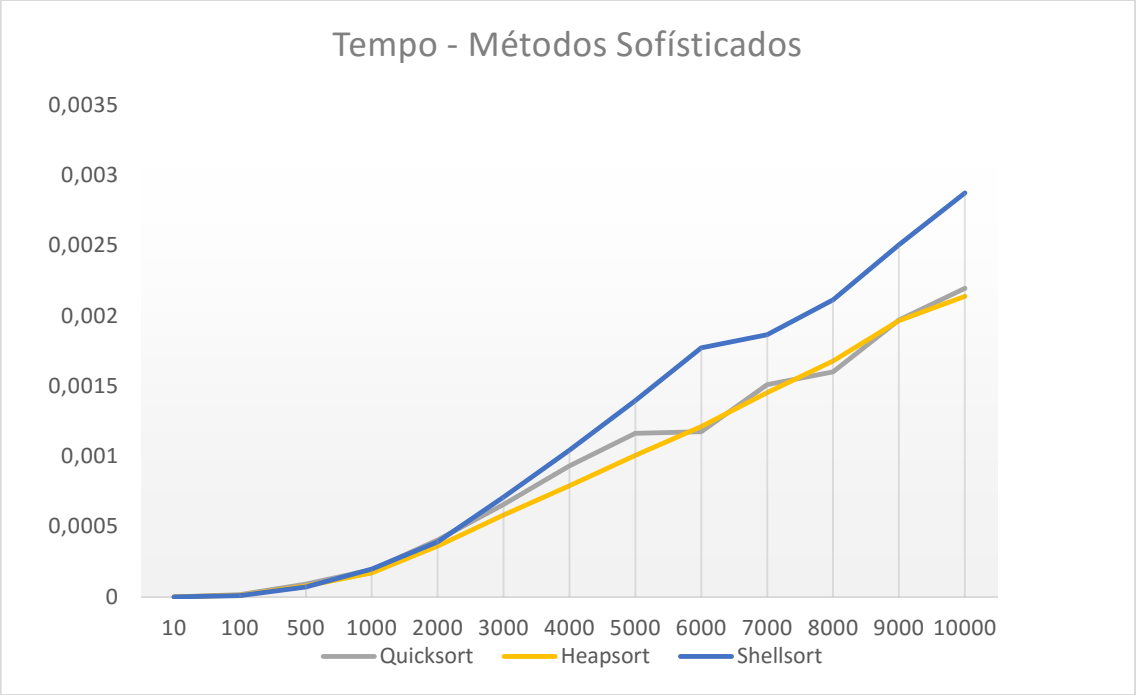
- Seleção < Inserção

E ainda, podemos observar que, os valores teóricos esperados são exatamente iguais para o Seleção, e extremamente próximos para o Inserção, fazendo com que no gráfico, nem notamos essa diferença, em decorrência do grande tamanho de movimentações.

Se analisarmos as movimentações do métodos simples junto aos métodos sofisticados, obteremos:

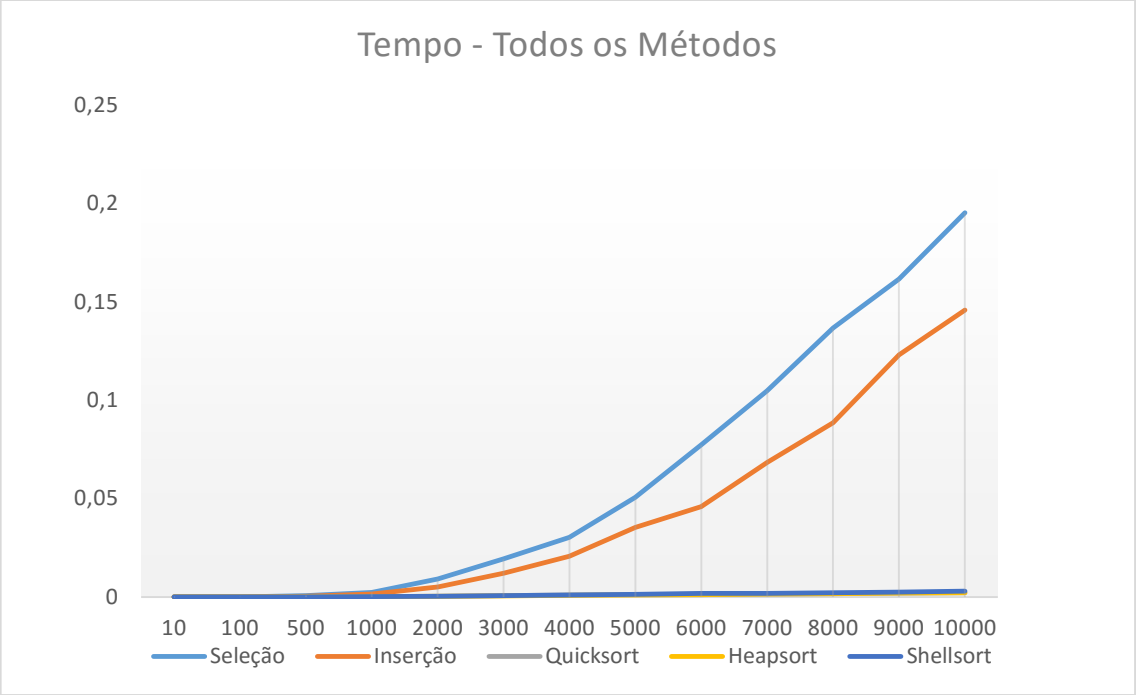
- Seleção < Quicksort < Heapsort < Shellsort < Inserção

Pode-se observar que neste caso, um algoritmo simples teve melhor desempenho que todos os outros, isso se deve ao fato de que, as movimentações no algoritmo Seleção são lineares a entrada, ou seja, nunca vai variar muito, tornando-se assim melhor que todos os outros, para quase todos os casos.



Para análise do tempo, fizemos um gráfico que demonstra a variação deste, em acordo com a variação das entradas. Separadamente, fizemos acima a análise dos métodos de ordenação sofisticados Quicksort, Heapsort e Shellsort, sendo que, para os três, pode-se observar que a média dos tempos, são bastante semelhantes. Tendo como pior método em disparada, o Shellsort. Assim obtém-se:

- Quicksort <= Heapsort < Shellsort



Neste última análise de tempo, pudemos observar todos os métodos de ordenação ao mesmo tempo, sendo que, os métodos sofisticados não puderam nem ser vistos, de tão pequenos em comparação com os métodos simples de ordenação.

Entre o Seleção e o Inserção, pode-se observar, que o Inserção tem menor tempo de execução, tornando-se assim o melhor entre os dois, e como já dito, ainda assim, muito pior que os métodos sofisticados.

Pode-se inferir a seguinte ordem geral em questão de tempo:

- Quicksort <= Heapsort < Shellsort < Inserção < Seleção

5.0 OBJETIVO

Este trabalho tem como objetivo desenvolver de forma mais ampla, a interação do aluno com as maneiras de implementação dos métodos de ordenação e também com a funções de complexidades, uma vez que, este é solicitado para implementar tais métodos e assim, fazer analises algorítmicas. Fazendo com que este além do conteúdo visto em sala de aula, tenha um conhecimento prático sobre o assunto.

6.0 CONCLUSÃO

Este trabalho foi proposto de uma maneira adequada e com o intuito de agregar conhecimento. À medida que sua execução foi sendo desenvolvida, pôde-se perceber pouco a pouco, o intuito de cada função, e seu modo de funcionar. Apesar de bem extenso, o trabalho cumpriu com o que era previsto. Disponibilizou a nós alunos a oportunidade de pensar maneiras e aspectos sobre cada parte do trabalho. E desta maneira, agregou grande valor acadêmico.