Resolução de Problemas Estruturados em Computação

RA04 - Ordenação - Professor Andrey Cabral Meira

[1]PUCPR- Kauany Almeida Silva – Ciência da Computação 4a

Códigos no GitHub: https://github.com/kauany-almeida/recuperacao-ordenacao

1. Bubble Sort

Analisando os resultados do algoritmo de ordenação BubbleSort em quesitos de tempo de inserção, quantidade de trocas e números de iterações para cinco vetores de tamanhos diferentes, executando cinco vezes para cada um dos tamanhos de vetor. Em seguida, após testar cinco vezes para cada um dos tamanhos há também uma tabela com a média de resultadas obtidas para cada.

1.1. Bubble Sort - 50 elementos — Executando 05 vezes

Qtde. elementos	ID execução	Tempo de Inserção	Número de Trocas	Número de Iterações
50	01	7ms	583	1225
50	02	10ms	701	1225
50	03	6ms	661	1225
50	04	12ms	443	1225
50	05	6ms	633	1225

Table 1. BUBBLE SORT COM VETOR DE 50 ELEMENTOS

1.2. Bubble Sort - 500 elementos — Executando 05 vezes

Qtde. elementos	ID execução	Tempo de Inserção	Número de Trocas	Número de Iterações
500	01	19ms	60816	124750
500	02	15ms	59901	124750
500	03	13ms	57888	124750
500	04	15ms	61090	124750
500	05	18ms	59250	124750

Table 2. BUBBLE SORT COM VETOR DE 500 ELEMENTOS

1.3. Bubble Sort - 1000 elementos — Executando 05 vezes

Qtde. elementos	ID execução	Tempo de Inserção	Número de Trocas	Número de Iterações
1.000	01	10ms	236765	499500
1.000	02	16ms	241211	499500
1.000	03	15ms	237664	499500
1.000	04	13ms	242883	499500
1.000	05	22ms	240265	499500

Table 3. BUBBLE SORT COM VETOR DE 1000 ELEMENTOS

1.4. Bubble Sort - 5000 elementos — Executando 05 vezes

Qtde. elementos	ID execução	Tempo de Inserção	Número de Trocas	Número de Iterações
5.000	01	56ms	5982115	12497500
5.000	02	64ms	6061392	12497500
5.000	03	59ms	6023608	12497500
5.000	04	75ms	5960032	12497500
5.000	05	69ms	5925652	12497500

Table 4. BUBBLE SORT COM VETOR DE 5000 ELEMENTOS

1.5. Bubble Sort - 10000 elementos — Executando 05 vezes

Qtde. elementos	ID execução	Tempo de Inserção	Número de Trocas	Número de Iterações
10.000	01	243ms	24081976	49995000
10.000	02	185ms	24252602	49995000
10.000	03	184ms	24317596	49995000
10.000	04	205ms	24273039	49995000
10.000	05	242ms	24045538	49995000

Table 5. BUBBLE SORT COM VETOR DE 10000 ELEMENTOS

1.6. BUBBLE SORT - Média de Desempenho

Qtde. elementos	Média - Tempo de Inserção	Média de Trocas	Média de Iterações
50	8.2ms	604	1225
500	16.0ms	59789	124750
1.000	15.2ms	239757	499500
5.000	65.2ms	5990559	12497500
10.000	211.8ms	24194150	49995000

Table 6. BUBBLE SORT - MÉDIA DE DESEMPENHO

A tabela acima mostra a média de resultados que cada um dos vetores obteve. Comparando esse algoritmo com os dois outros algoritmos e ordenação (MergeSort, Shell Sort) este aqui é o segundo com melhor desempenho em relação às 3 médias, tendo resultados inferiores apenas ao algoritmo Shell Sort.

2. MERGE SORT

Agora iremos analisar o desempenho do algoritmo Merge Sort. Serão feitas 05 execuções para análise de desempenho para 05 vetores de tamanhos diferentes, cada vetor terá suas cinco execuções individuais. Logo em seguida, a média de cada um dos resultados obtidos será armazenada em uma tabela após os 05 testes.

2.1. Merge Sort - 50 elementos — Executando 05 vezes

Qtde. elementos	ID execução	Tempo de Inserção	Número de Trocas	Número de Iterações
50	01	19ms	531	772
50	02	4ms	540	772
50	03	6ms	529	772
50	04	14ms	543	772
50	05	8ms	536	772

Table 7. MERGE SORT COM VETOR DE 50 ELEMENTOS

2.2. Merge Sort - 500 elementos — Executando 05 vezes

Qtde. elementos	ID execução	Tempo de Inserção	Número de Trocas	Número de Iterações
500	01	1ms	8682	10976
500	02	19ms	8714	10976
500	03	12ms	8685	10976
500	04	0ms	8734	10976
500	05	15ms	8676	10976

Table 8. MERGE SORT COM VETOR DE 500 ELEMENTOS

2.3. Merge Sort - 1000 elementos — Executando 05 vezes

Qtde. elementos	ID execução	Tempo de Inserção	Número de Trocas	Número de Iterações
1.000	01	17ms	19374	23952
1.000	02	11ms	19367	23952
1.000	03	17ms	19382	23952
1.000	04	5ms	19376	23952
1.000	05	12ms	19333	23952

Table 9. MERGE SORT COM VETOR DE 1000 ELEMENTOS

2.4. Merge Sort - 5000 elementos — Executando 05 vezes

Qtde. elementos	ID execução	Tempo de Inserção	Número de Trocas	Número de Iterações
5.000	01	48ms	119906	143616
5.000	02	57ms	120028	143616
5.000	03	60ms	119947	143616
5.000	04	52ms	119868	143616
5.000	05	52ms	120056	143616

Table 10. MERGE SORT COM VETOR DE 5000 ELEMENTOS

2.5. Merge Sort - 10.000 elementos — Executando 05 vezes

Qtde. elementos	ID execução	Tempo de Inserção	Número de Trocas	Número de Iterações
10.000	01	99ms	259684	307232
10.000	02	96ms	259838	307232
10.000	03	87ms	259753	307232
10.000	04	92ms	259697	307232
10.000	05	87ms	259611	307232

Table 11. MERGE SORT COM VETOR DE 10000 ELEMENTOS

2.6. MERGE SORT - Média de Desempenho

Qtde. elementos	Média Inserção	Média Trocas	Média Iterações
50	10.2ms	535	772
500	9.4ms	8698	10976
1.000	12.4ms	19366	23952
5.000	53.8ms	119961	143616
10.000	391.4ms	207777	307232

Table 12. MERGE SORT - MÉDIA DE DESEMPENHO

Estes foram os resultados obtidos ao testar o algoritmo Merge Sort. Comparando os resultados deste algoritmo com os resultados obtidos nos algoritmos BubbleSort e Shell Sort, este aqui foi o que obteve o desempenho mais lento em comparação aos outros.

3. SHELL SORT

O último algoritmo em que será feita uma análise de desempenho será o Shell Sort. Será executado cinco vezes diferentes para cada um dos cinco tamanhos diferentes de vetor, e em seguida terá sua média de resultados anexada em uma tabela e comparada às outras duas médias dos outros algoritmos.

3.1. Shell Sort - 50 elementos — Executando 05 vezes

Qtde. elementos	ID execução	Tempo de Inserção	Número de Trocas	Número de Iterações
50	01	10ms	167	203
50	02	12ms	147	203
50	03	0ms	138	203
50	04	6ms	193	203
50	05	8ms	98	203

Table 13. SHELL SORT COM VETOR DE 50 ELEMENTOS

3.2. Shell Sort - 500 elementos — Executando 05 vezes

Qtde. elementos	ID execução	Tempo de Inserção	Número de Trocas	Número de Iterações
500	01	11ms	1994	3506
500	02	8ms	2054	3506
500	03	10ms	2129	3506
500	04	8ms	2098	3506
500	05	8ms	1917	3506

Table 14. SHELL SORT COM VETOR DE 500 ELEMENTOS

3.3. Shell Sort - 1000 elementos — Executando 05 vezes

Qtde. elementos	ID execução	Tempo de Inserção	Número de Trocas	Número de Iterações
1.000	01	0ms	4174	8006
1.000	02	9ms	4289	8006
1.000	03	9ms	4442	8006
1.000	04	0ms	4633	8006
1.000	05	11ms	5224	8006

Table 15. SHELL SORT COM VETOR DE 1000 ELEMENTOS

3.4. Shell Sort - 5000 elementos — Executando 05 vezes

Qtde. elementos	ID execução	Tempo de Inserção	Número de Trocas	Número de Iterações
5.000	01	22ms	29506	55005
5.000	02	17ms	28380	55005
5.000	03	25ms	32860	55005
5.000	04	15ms	33040	55005
5.000	05	22ms	29305	55005

Table 16. SHELL SORT COM VETOR DE 5000 ELEMENTOS

3.5. Shell Sort - 10000 elementos — Executando 05 vezes

Qtde. elementos	ID execução	Tempo de Inserção	Número de Trocas	Número de Iterações
10.000	01	42ms	62698	120005
10.000	02	50ms	61399	120005
10.000	03	78ms	60054	120005
10.000	04	42ms	62411	120005
10.000	05	44ms	61834	120005

Table 17. SHELL SORT COM VETOR DE 10000 ELEMENTOS

3.6. SHELL SORT - Média de Desempenho

Qtde. elementos	Média Inserção	Média Trocas	Média Iterações
50	7.2ms	154	203
500	9.0ms	2038	3506
1.000	5.8ms	3625	8006
5.000	20.2ms	30618	55005
10.000	51.2ms	61679	120005

Table 18. SHELL SORT - MÉDIA DE DESEMPENHO

Comparando dessa vez os resultados obtidos com o algoritmo de Shell Sort, este teve o melhor desempenho nos 3 itens que foram medidos. Ele foi o que teve os melhores resultados em relação ao tempo de inserção em cada ”vetor resultado”, poucas trocas e também uma menor média de iterações.