Ordenação

Victor Silva Camargo, Vinícius Silva Camargo

[1]Pontifícia Universidade Católica do Paraná (PUCPR)
Curitiba – PR – Brazil

1. Bubble Sort

O método de ordenação ao Bubble sort usa a lógica de percorrer o vetor comparando o elemento em que ele está com o próximo, caso esteja desordenado, ele troca as posições do elemento atual com o próximo, e repete esse processo em uma quantidade fixa de vezes, a quantidade é sempre o tamanho do vetor - 1, esse método demonstra ser um dos mais lentos métodos de ordenação, devido a ele ter que percorrer todo o vetor, o tamanho todo dele - 1, independentemente se todas as trocas já foram feitas, pode ser ajustado com algum tipo de validador, como um booleano, por exemplo.

Média de tempo de ordenação para 50 elementos:	33160 ns
Média de trocas para 50 elementos:	581
Média de iterações para 50 elementos:	49
Média de tempo de ordenação para 500 elementos	884080 ns
Média de trocas para 500 elementos:	63131
Média de iterações para 500 elementos:	499
Média de tempo de ordenação para 1000 elemento	1454980 ns
Média de trocas para 1000 elementos:	250553
Média de iterações para 1000 elementos:	999
Média de tempo de ordenação para 5000 elemento	15488080 ns
Média de trocas para 5000 elementos:	6242272
Média de iterações para 5000 elementos:	4999
Média de tempo de ordenação para 10000 element	95428380 ns
Média de trocas para 10000 elementos:	25010625
Média de iterações para 10000 elementos:	9999

Figure 1. Tabela de Execução com Tempo de execução, Número de trocas e Número de iterações Bubble Sort.

2. Insert Sort

O método de ordenação Insert Sort pega um vetor desordenado e faz a ordenação movendo os elementos não ordenados para a posição correta, pegando um elemento e fazendo comparação com os outros até chegar na sua devida posição.

Média de tempo de ordenação para 50 elementos:	21580 ns
Média de trocas para 50 elementos:	660
Média de iterações para 50 elementos:	49
Média de tempo de ordenação para 500 elementos:	484900 ns
Média de trocas para 500 elementos:	62379
Média de iterações para 500 elementos:	499
Média de tempo de ordenação para 1000 elemento	551660 ns
Média de trocas para 1000 elementos:	248702
Média de iterações para 1000 elementos:	999
Média de tempo de ordenação para 5000 elemento	1674180 ns
Média de trocas para 5000 elementos:	6218849
Média de iterações para 5000 elementos:	4999
Média de tempo de ordenação para 10000 element	7600020 ns
Média de trocas para 10000 elementos:	24959144
Média de iterações para 10000 elementos:	9999

Figure 2. Tabela de Execução com Tempo de execução, Número de trocas e Número de iterações Insert Sort.

3. Merge Sort

O método de ordenação Merge Sort usa a lógica de separar o vetor pela metade até que sobrem dois elementos. Alguns casos um elemento fica sozinho devido ao tamanho do vetor ser ímpar. Depois que divide, ele junta novamente por pares, só que dessa vez ordenando, até chegar em um vetor final ordenado. Esse método demonstrou ser mais eficiente em casos que o vetor está no pior caso de ordenação, além de que no geral, ele realiza menos trocas que os outros métodos de ordenação.

Média de tempo de ordenação para 50 elementos:	36620 ns
Média de trocas para 50 elementos:	103
Média de iterações para 50 elementos:	49
Média de tempo de ordenação para 500 elementos:	100240 ns
Média de trocas para 500 elementos:	1908
Média de iterações para 500 elementos:	499
Média de tempo de ordenação para 1000 elemento:	158040 ns
Média de trocas para 1000 elementos:	4314
Média de iterações para 1000 elementos:	999
Média de tempo de ordenação para 5000 elemento:	1484840 ns
Média de trocas para 5000 elementos:	26624
Média de iterações para 5000 elementos:	4999
Média de tempo de ordenação para 10000 element:	2498320 ns
Média de trocas para 10000 elementos:	58262
Média de iterações para 10000 elementos:	9999

Figure 3. Tabela de Execução com Tempo de execução, Número de trocas e Número de iterações Merge Sort.

4. Shell Sort

O Shell Sort funciona de maneira parecida com o Insert Sort. O Shell Sort percorre o vetor e faz a divisão dele em sub-listas. Podemos determinar quantas sub-listas, mas no caso deste trabalho ele é definido com um grande incremento (chamado ”span”) e, em cada iteração, divide as sublistas internamente para classificar essas sublistas com base no ”span”. O ”span” é determinado pelos valores em vetor[incr], que são recuperados a cada iteração.

Média de tempo de ordenação para 50 elementos:	1740 ns
Média de trocas para 50 elementos:	3
Média de iterações para 50 elementos:	49
Média de tempo de ordenação para 500 elementos	200840 ns
Média de trocas para 500 elementos:	5250
Média de iterações para 500 elementos:	499
Média de tempo de ordenação para 1000 elemento	480240ns
Média de trocas para 1000 elementos:	7392
Média de iterações para 1000 elementos:	999
Média de tempo de ordenação para 5000 elemento	7524720 ns
Média de trocas para 5000 elementos:	60173
Média de iterações para 5000 elementos:	4999
Média de tempo de ordenação para 10000 element	55392720ns
Média de trocas para 10000 elementos:	130762
Média de iterações para 10000 elementos:	9999

Figure 4. Tabela de Execução com Tempo de execução, Número de trocas e Número de iterações Shell Sort.

5. Comparando Bubble e Merge

O bubble sort demosntrou ser mais lento no geral que o merge sort, pois o tempo de execução do bubble sort é o quadrado do tamanho do vetor, ele tem que percorrer o tamanho do array -1, isso significa que quanto maior o array, bem maior será o tempo, mesmo que ja tenha ordenado tudo, ele continua rodando se não houver uma condição de parada, já o mergesort , a medida que o tamanho do array aumenta, o tempo de execução aumendo de forma bem menos acelerada, visto que ele consiste em dividir o array todo em sub arrays, até chegar em arrays de tamanho 1 e comparar a esquerda com a direita e fazer o merge.

6. Comparando Insert e Shell

O Shell provou ser superior em comparação ao Insert em todos os aspectos, devido à sua capacidade de comparar elementos distantes uns dos outros, criando sub-listas para a comparação. No entanto, o desempenho do Insert pode ser melhor se os elementos semelhantes estiverem próximos uns dos outros.

Para saber mais, confira este (vídeo de comparação entre os dois métodos: https://www.youtube.com/watch?v=g06hNBhoS1k).