TDE 3
Ordenação

Henrique Tetilha Golias[1]

[1]Pontifícia Universidade Católica do Paraná - PUCPR

1. Explicação

Para este trabalho foram escolhidos 3 tipos de ordenação, que são o Bubble Sort, o Merge Sort e o Quick Sort. Cada método de ordenação foi testado pelo menos 5 vezes com cada quantidade requerida e no final feito a média entre estes testes, os valores da tabela de cada método são estas médias. As seeds que foram utilizadas são **123, 456, 789, 987, 321**. Todos os testes foram feitos em meu Desktop pessoal.

A configuração do computador usado: Processador **Intel(R) Core(TM) i5-9400F CPU @ 2.90GHz 2.90 GHz** RAM instalada **8,00 GB** DDR4 Tipo de sistema Sistema operacional de 64 bits, processador baseado em x64

2. Bubble Sort

O primeiro metodo de ordenação escolhido foi o Bubble Sort, que é um algoritmo de classificação simples que funciona comparando repetidamente pares de elementos adjacentes em uma lista e trocando-os se estiverem na ordem errada. O processo continua até que nenhuma troca seja necessária, o que indica que a lista está ordenada.

No main temos a parte principal, como inicialização do scanner e a geração da lista usando a quantidade escolhida pelo usuário, então ele imprime a lista e começa a contagem do tempo em nanosegundos, logo após o termino da ordenação ele para o contador, imprime novamente a lista e mostra o tempo de execução e a quantidade de trocas feitas pelo método.

Na classe BubbleSort, há o método bubbleSort que é responsavel pela ordenação da lista, ele percorre a lista verificando se os elementos são maiores que o seu subsequente, no final ele retornará o valor de trocas para impressão. Logo em seguida temos duas funções simples para geração da lista e para a impressão da lista.

Bubble Sort			
Quantidade de Elementos	Tempo (nano)	Número de Trocas	Iterações
50	0.0000385	615	1
500	0.0024105	6288	1
1000	0.0056537	247489	1
5000	0.0866586	6212698	1
10000	0.1327482	24748321	1

3. Merge Sort

O Merge Sort divide a lista original em duas metades, classifica cada metade separadamente e depois combina as duas metades classificadas em uma única lista classificada. O processo de divisão e combinação é repetido recursivamente até que a lista esteja completamente classificada.

No main temos basicamente a mesma coisa que o bubble sort, porém com a pequena diferença que é impresso também a quantia de iterações.

Na classe MergeSort temos o método mergeSort que é responsavel pela ordeção, ele vai dividindo a lista recursivamente, e depois o método merge que é responsável pela ordenação, compara os elementos das sub-linhas e retorna eles em ordem crescente, voltando até que todas as chamadas recursivas do método mergeSort tenham sido terminadas para ordenar a lista. Temos também métodos para pegar os valores das trocas e iterações e também o método para imprimir o array.

Merge Sort			
Quantidade de Elementos	Tempo (nano)	Número de Trocas	Iterações
50	0.0000331	221	98
500	0.0003580	3846	998
1000	0.0007204	8691	1998
5000	0.0013534	55157	9998
10000	0.0026866	120273	19998

4. Quick Sort

A principal ideia por trás do Quick Sort é escolher um elemento pivot da lista e rearranjar os elementos de forma que os elementos menores que o pivot estejam à esquerda, e os elementos maiores estejam à direita. Em seguida, o algoritmo é aplicado recursivamente às duas sublistas, à esquerda e à direita do pivô.

No main nós temos basicamente a mesma coisa que o merge sort.

Na classe QuickSort possuímos o método quickSort que é responsável pela divisão. Dentro desete mesmo método temos o método partition que é responsável por encontar o índice do pivô e trocar a ordem dos elementos com base no pivô usando o método swap, que simplesmente troca os elementos indicados pelo partition, que no final retorna o indice do pivô da lista anterior até a lista original estar ordenada.

Quick Sort			
Quantidade de Elementos	Tempo (nano)	Número de Trocas	Iterações
50	0.0000202	160	65
500	0.0002208	2313	810
1000	0.0004398	4909	1800
5000	0.0019684	25353	9800
10000	0.0042019	52278	19800

5. Análise Final

Estudando sobre o assunto, desenvolvendo e testando cheguei a conclusão que se você precisa de um algoritmo simples para listas muito pequenas, o Bubble Sort pode ser a melhor escolha. Para listas maiores, o Merge Sort e o Quick Sort são opções melhores. Entre o Merge Sort e o Quick Sort, a escolha depende das características específicas do problema. Em muitos casos, o Quick Sort é a melhor escolha devido à sua eficiência média.