Relatório

Fernando Ribeiro Martins

**Introdução**

Este relatório tem como objetivo descrever os resultados do segundo trabalho da disciplina de Algoritmo e estrutura de dados II, que se trata de uma comparação entre diferentes métodos de ordenação, sendo eles: Selectsort, Mergsort, Quicksort e Insertionsort. Para compará-los foram realizados testes com vetores de 1.000, 10.000, 100.000 e 2.000.000, foram feitos testes com vetores aleatórios, ordenados e inversamente ordenados.

**Implementação**

O algoritmo se inicia com a criação de um vetor e criado e é alocado números para cada posição do vetor, podendo ser com valores aleatórios, ordenados e inversamente ordenados, é dado a possibilidade de imprimi-los na tela ao usuário.

Posteriormente é realizada a ordenação do vetor, utilizando os meios outrora apresentados, é apresentada a opção de imprimi-los depois de ordena-los, além disso o código armazena o número de comparações realizadas, além do tempo gasto para cada ordenação, depois esses dados são armazenados em um arquivo junto ao nome do método utilizado, para que seja possível uma melhor visualização dos dados obtidos.

**ANALISE DE COMPLEXIDADE**

Para esta parte, foi feito um resumo completo da questão no período, onde foram obtidas algumas informações que serão tratadas individualmente para cada uma para que a análise de complexidade aqui demonstrada ser o método utilizado.

**SELECTIONSORT**

Este método usa dois loops onde o loop interno executa a comparação n-1 e o loop interno executa a comparação interna n-1 mais n-1 vezes, então o custo é 0 de desses.

Com relação às atribuições, a complexidade é 0, o que torna o algoritmo linear e esse tipo de algoritmo ideal para pequenas estruturas devido ao custo.

**INSERTIONSORT**

este algoritmo nos deparamos com 3 possibilidades, o melhor caso onde temos um algoritmo de complexidade O, o pior caso onde temos O complexidade e o caso médio que consideramos também como pior caso que também seria de O complexidade

**MERGESORT**

Este algoritmo é dividido em duas partes, uma é que tem O como complexidade linear, que depende da quantidade de intercalação realizada para determinar a complexidade e a do algoritmo, portanto, o resultado final complexidade disso é O (n log n).

**QUIKSORT**

Por funcionar no local como MergeSort, dividindo os problemas e vários problemas menores , também é considerado de complexidade O. (n log n).

|  |  |  |  |
| --- | --- | --- | --- |
| selectionsort | Tamanho | Tempo em segundos | Número de movimentos |
| Aleatório | 1000 | 5 | 999 |
| Crescente | 1000 | 14 | 999 |
| Decrescente | 1000 | 0 | 999 |
| Aleatório | 10000 | 557 | 9999 |
| Crescente | 10000 | 580 | 9999 |
| Decrescente | 10000 | 519 | 9999 |
| Aleatório | 100000 | 55000 | 99999 |
| Crescente | 100000 | 62000 | 99999 |
| Decrescente | 100000 | 53200 | 99999 |

|  |  |  |  |
| --- | --- | --- | --- |
| mergesort | tamanho | Tempo em segundos | Número de movimentos |
| Aleatório | 1000 | 0 | 9976 |
| Crescente | 1000 | 0 | 9976 |
| Decrescente | 1000 | 0 | 9976 |
| Aleatório | 10000 | 11,5 | 133616 |
| Crescente | 10000 | 0 | 133616 |
| Decrescente | 10000 | 13,9 | 133616 |
| Aleatório | 100000 | 120 | 1668928 |
| Crescente | 100000 | 335 | 1668928 |
| Decrescente | 100000 | 65 | 1668628 |

|  |  |  |  |
| --- | --- | --- | --- |
| quicksort | tamanho | Tempo em segundos | Número de movimentos |
| Aleatório | 1000 | 2 | 2630 |
| Crescente | 1000 | 0 | 1000 |
| Decrescente | 1000 | 0 | 1011 |
| Aleatório | 10000 | 16 | 34289 |
| Crescente | 10000 | 0 | 10000 |
| Decrescente | 10000 | 1 | 10906 |
| Aleatório | 100000 | 48 | 434013 |
| Crescente | 100000 | 29 | 100000 |
| Decrescente | 100000 | 14 | 115535 |

|  |  |  |  |
| --- | --- | --- | --- |
| insertionsort | tamanho | Tempo em segundos | Número de movimentos |
| Aleatório | 1000 | 3 | 261071 |
| Crescente | 1000 | 0 | 999 |
| Decrescente | 1000 | 17 | 500499 |
| Aleatório | 10000 | 330 | 2494858 |
| Crescente | 10000 | 1 | 9999 |
| Decrescente | 10000 | 650 | 50004999 |
| Aleatório | 100000 | 33300 | 73877008 |
| Crescente | 100000 | 3 | 99999 |
| Decrescente | 100000 | 68400 | 705082703 |

**Conclusão**

Este trabalho que tem como objetivo comparar os métodos de ordenação é complexo, longo e demorada, já que para ser realizado depende de uma grande quantia de números e de comparações, o que demanda grande quantia de tempo.

Os diferentes tipos de ordenação prévia dos vetores possuem uma significativa diferença de tempo, já que leva alguns tipos de ordenação a situações incomuns e problemáticas para eles, podendo levar alguns ao seu pior caso.

**BIBLIOGRAFIA**

http://linguagemc.com.br/exibindo-data-e-hora-com-time-h/