CENTRO PAULA SOUZA

FACULDADE DE TECNOLOGIA DE FRANCA

“Dr. THOMAZ NOVELINO”

TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

LAURA ANDRADE DE OLIVEIRA

VINÍCIUS CHIARELO GOMES

TRABALHO DE ESTRuTURA DE DADOS

Relatório com o Desempenho de Alguns Algoritmos de Ordenação

**FRANCA/SP**

**2024**

**TRABALHO 1º BIMESTRE**

1. BUBBLE SORT

#### Tabela com o tempo de execução de cada método:

**Crescente**

|  |  |  |
| --- | --- | --- |
| **Tamanho do Vetor** | **Ordem de Entrada dos Elementos** | **Tempo de Execução** |
| 100 | Crescente | 0.918**ms** |
| 1.000 | Crescente | 4.659**ms** |
| 10.000 | Crescente | 167.17**ms** |
| 100.000 | Crescente | 19.073**s** |

**Decrescente**

|  |  |  |
| --- | --- | --- |
| **Tamanho do Vetor** | **Ordem de Entrada dos Elementos** | **Tempo de Execução** |
| 100 | Decrescente | 1.768**ms** |
| 1.000 | Decrescente | 12.963**ms** |
| 10.000 | Decrescente | 224.11**ms** |
| 100.000 | Decrescente | 20.120**s** |

**Aleatório**

|  |  |  |
| --- | --- | --- |
| **Tamanho do Vetor** | **Ordem de Entrada dos Elementos** | **Tempo de Execução** |
| 100 | Aleatório | 0.097**ms** |
| 1.000 | Aleatório | 2.648**ms** |
| 10.000 | Aleatório | 281.767**ms** |
| 100.000 | Aleatório | 30.862**s** |

#### Comparação dos métodos:

### Vantagens

* **Simplicidade:** É um dos algoritmos de ordenação mais simples de entender e implementar.
* **Eficiência em dados quase ordenados:** Em conjuntos de dados já parcialmente ordenados, o Bubble Sort pode ser bastante eficiente, realizando poucas trocas.
* **Estabilidade:** O algoritmo preserva a ordem original de elementos iguais na lista. Se dois elementos iguais X e Y estão em suas posições X e Y antes da ordenação, eles permanecerão nas mesmas posições X e Y após a ordenação.

### Desvantagens

* **Ineficiência em grandes conjuntos:** O tempo de execução cresce quadraticamente com o tamanho do vetor, tornando-o inviável para ordenar grandes volumes de dados.
* **Número excessivo de comparações:** Mesmo em casos favoráveis, o Bubble Sort realiza um número significativo de comparações desnecessárias.
* **Lentidão:** Em comparação com algoritmos mais eficientes, como QuickSort ou MergeSort, o Bubble Sort é consideravelmente mais lento.

### Complexidade

* **Melhor Caso**: O(n^2)
* **Pior Caso:** O(n^2)
* **Média:** O(n^2)

1. SELECTION SORT (Seleção Direta)

#### Tabela com o tempo de execução de cada método:

**Crescente**

|  |  |  |
| --- | --- | --- |
| **Tamanho do Vetor** | **Ordem de Entrada dos Elementos** | **Tempo de Execução** |
| 100 | Crescente | 0.506**ms** |
| 1.000 | Crescente | 2.679**ms** |
| 10.000 | Crescente | 56.773**ms** |
| 100.000 | Crescente | 4.483**s** |

**Decrescente**

|  |  |  |
| --- | --- | --- |
| **Tamanho do Vetor** | **Ordem de Entrada dos Elementos** | **Tempo de Execução** |
| 100 | Decrescente | 0.34**ms** |
| 1.000 | Decrescente | 3.197**ms** |
| 10.000 | Decrescente | 45.611**ms** |
| 100.000 | Decrescente | 3.723**s** |

**Aleatório**

|  |  |  |
| --- | --- | --- |
| **Tamanho do Vetor** | **Ordem de Entrada dos Elementos** | **Tempo de Execução** |
| 100 | Aleatório | 0.017**ms** |
| 1.000 | Aleatório | 1.081**ms** |
| 10.000 | Aleatório | 40.235**ms** |
| 100.000 | Aleatório | 3.078**s** |

#### Comparação dos métodos:

### Vantagens

* **Simplicidade:** O Selection Sort é um dos algoritmos de ordenação mais simples de entender e implementar.
* **Eficiência de memória:** O algoritmo não requer memória extra além do vetor original a ser ordenado. Isso o torna vantajoso em ambientes com recursos limitados.
* **Estabilidade:** O Selection Sort preserva a ordem relativa de elementos iguais no vetor original. Isso pode ser importante em algumas situações, como ao ordenar uma lista de objetos com chaves duplicadas.
* **Desempenho decente em vetores pequenos:** O Selection Sort geralmente é mais rápido que outros algoritmos de ordenação, como o Bubble Sort, em vetores com um número pequeno de elementos.

### Desvantagens

* **Ineficiência em vetores grandes:** O tempo de execução do Selection Sort aumenta quadraticamente com o tamanho do vetor (O(n²)). Isso significa que o algoritmo se torna cada vez mais lento à medida que o número de elementos aumenta.
* Número de trocas: Apesar de realizar menos comparações que o Bubble Sort, o Selection Sort geralmente faz mais trocas de elementos, o que pode afetar o desempenho em alguns casos.

### Complexidade

## Tempo de execução:

## Melhor caso: O(n²)

## Pior caso: O(n²)

## Média: O(n²)

## Número de comparações: n\*(n-1) / 2, independente da ordem inicial do vetor.

## Número de trocas:

## Melhor caso: (vetor já ordenado): 0

## Pior caso: (vetor ordenado inversamente): n-1

## Média: n/2

1. INSERTION SORT (Inserção Direta)

#### Tabela com o tempo de execução de cada método:

**Crescente**

|  |  |  |
| --- | --- | --- |
| **Tamanho do Vetor** | **Ordem de Entrada dos Elementos** | **Tempo de Execução** |
| 100 | Crescente | 0.498**ms** |
| 1.000 | Crescente | 1.910**ms** |
| 10.000 | Crescente | 70.371**ms** |
| 100.000 | Crescente | 4.469**s** |

**Decrescente**

|  |  |  |
| --- | --- | --- |
| **Tamanho do Vetor** | **Ordem de Entrada dos Elementos** | **Tempo de Execução** |
| 100 | Decrescente | 0.527**ms** |
| 1.000 | Decrescente | 3.524**ms** |
| 10.000 | Decrescente | 105.698**ms** |
| 100.000 | Decrescente | 9.268**s** |

**Aleatório**

|  |  |  |
| --- | --- | --- |
| **Tamanho do Vetor** | **Ordem de Entrada dos Elementos** | **Tempo de Execução** |
| 100 | Aleatório | 0.02**ms** |
| 1.000 | Aleatório | 0.785**ms** |
| 10.000 | Aleatório | 86.421**ms** |
| 100.000 | Aleatório | 7.513**s** |

#### Comparação dos métodos:

### Vantagens

## Simplicidade: O Insertion Sort é um dos algoritmos de ordenação mais simples de entender e implementar.

## Eficiência em listas pequenas: O Insertion Sort é muito eficiente para ordenar listas pequenas, especialmente quando os elementos já estão quase ordenados.

## Estabilidade: O Insertion Sort é um algoritmo estável, o que significa que elementos com chaves iguais manterão suas posições relativas na lista ordenada.

## Eficiência em inserções: O Insertion Sort é muito eficiente para inserir novos elementos em uma lista já ordenada.

### Desvantagens

## Ineficiência em listas grandes: O Insertion Sort pode ser muito ineficiente para ordenar listas grandes, especialmente quando os elementos estão em ordem inversa.

## Movimentação de elementos: O Insertion Sort pode mover muitos elementos durante a ordenação, o que pode ser um problema para alguns tipos de dados.

### Complexidade

## Melhor caso: O(n)

## Pior caso: O(n²).

## Média: O(n²).

1. QUICK SORT

#### Tabela com o tempo de execução de cada método:

**Crescente**

|  |  |  |
| --- | --- | --- |
| **Tamanho do Vetor** | **Ordem de Entrada dos Elementos** | **Tempo de Execução** |
| 100 | Crescente | 0.203**ms** |
| 1.000 | Crescente | 0.6**ms** |
| 10.000 | Crescente | 4.072**ms** |
| 100.000 | Crescente | 33.462**ms** |

**Decrescente**

|  |  |  |
| --- | --- | --- |
| **Tamanho do Vetor** | **Ordem de Entrada dos Elementos** | **Tempo de Execução** |
| 100 | Decrescente | 0.055**ms** |
| 1.000 | Decrescente | 0.228**ms** |
| 10.000 | Decrescente | 2.292**ms** |
| 100.000 | Decrescente | 32.593**ms** |

**Aleatório**

|  |  |  |
| --- | --- | --- |
| **Tamanho do Vetor** | **Ordem de Entrada dos Elementos** | **Tempo de Execução** |
| 100 | Aleatório | 0.048**ms** |
| 1.000 | Aleatório | 0.394**ms** |
| 10.000 | Aleatório | 3.520**ms** |
| 100.000 | Aleatório | 47.824**ms** |

#### Comparação dos métodos:

### Vantagens

## Eficiência: O Quick Sort é um dos algoritmos de ordenação mais eficientes, especialmente para ordenar listas grandes.

## Velocidade: O Quick Sort é geralmente muito mais rápido que outros algoritmos de ordenação, como o Insertion Sort e o Bubble Sort.

## Menos movimentação de elementos: O Quick Sort geralmente move menos elementos do que outros algoritmos de ordenação.

### Desvantagens

## Instabilidade: O Quick Sort não é um algoritmo estável, o que significa que elementos com chaves iguais podem ter suas posições relativas alteradas na lista ordenada.

## Pior caso: O Quick Sort tem um pior caso com complexidade O(n²), que ocorre quando a lista está em dados já ordenados ou quase ordenados.

## Memória adicional: O Quick Sort é um algoritmo recursivo, o que significa que ele chama a si mesmo para ordenar sub-partes do vetor original. Cada chamada recursiva requer espaço na memória para armazenar informações sobre o estado da ordenação. Embora a quantidade de memória extra seja relativamente pequena, pode ser uma desvantagem para ordenação de conjuntos de dados muito grandes, principalmente em sistemas com memória limitada.

### Complexidade

## Melhor caso: O (n log n).

## Pior caso: O(n²).

## Média: O (n log n).

1. CONCLUSÃO

Com os resultados obtidos, chegamos à conclusão de que a escolha do método de ordenação depende da quantidade dos dados a serem tratados. Em pequenas quantidades a diferença pode ser mínima, já em altas, a diferença de tempo começa a se tornar mais perceptível.

O método que apresentou pior caso foi o Bubble Sort, por ser mais lento que os outros. Em contrapartida, quem apresentou o melhor caso foi o Quik Sort, principalmente nos grandes vetores. Os que apresentaram caso médio foram o Selection Sort e o Insertion Sort, pois não foram tão lentos como o Bubble Sort nem chegaram no desempenho do Quik Sort.

Sobre complexidade, apesar do Quik Sort ser o mais complexo e apresentar recursividade, ele se destacou aos demais que não são tão complexos.

Portanto, concluímos que a primeira coisa que deve ser feita é analisar o problema como um todo, para que assim seja escolhido o método que melhor se encaixa na ocasião. Não existe “melhor” ou “pior”, existem casos e casos.