UNIVERSIDADE DO VALE DO ITAJAI

CURSO DE Ciencia da computação

TÍTULO:

MÉTODOS DE ORDENAÇÃO

giulia santos lima e thiago andrade ferreira

itajai

2024

RESUMO

O relatório apresenta a análise comparativa de seis algoritmos de ordenação: Insertion Sort, Shell Sort, Bubble Sort, Quick Sort, Selection Sort e Merge Sort. A análise é realizada através da implementação dos algoritmos em C++ e da medição dos tempos de execução em termos de tempo máximo, mínimo e médio de execução para cada um deles;

Palavras-chave: Algoritmos de Ordenação; média dos tempos de execução dos Algoritmos.

SUMÁRIO

[1 introdução 3](#_Toc170760719)

[1.1 descrição dos algoritmos de ordenação 3](#_Toc170760720)

[1.1.1 RESULTADO DA COMPILAÇÃO DOS MÉTODOS DE ORDENAÇÃO 4](#_Toc170760721)

[2 Considerações finais 5](#_Toc170760722)

# introdução

A ordenação de dados possui um impacto direto no desempenho de sistemas, sendo utilizada na organização e acessibilidade dos dados, onde a ordenação facilita a busca e recuperação de informações. Ela é fundamental também na otimização das operações. A eficiência dos algoritmos de ordenação pode afetar significativamente a performance de aplicações e sistemas computacionais. Este relatório apresenta uma análise comparativa de seis algoritmos de ordenação: Insertion Sort, Shell Sort, Bubble Sort, Quick Sort, Selection Sort e Merge Sort, implementados em C++ e testados em diferentes cenários

## descrição dos algoritmos de ordenação

1. Insertion Sort

O Insertion Sort é um algoritmo simples e intuitivo, que constrói a lista ordenada um elemento de cada vez. Embora eficiente para listas pequenas ou quase ordenadas, sua complexidade de tempo O(n²) no pior caso o torna impraticável para grandes conjuntos de dados.

1. Shell Sort

O Shell Sort é uma extensão do Insertion Sort, que melhora sua eficiência ao inicializar com uma grande distância entre elementos comparados e reduzi-la gradualmente trocando elementos que estão mais distantes uns dos outros em suas posições corretas, com o objetivo de reduzir a quantidade de movimentação até que a lista fique ordenada.

1. Bubble Sort

O Bubble Sort é um algoritmo simples que repetidamente percorre a lista, comparando e trocando elementos adjacentes que estão fora de ordem. Sua simplicidade vem ao custo de uma complexidade de tempo O(n²), tornando-o inadequado para grandes listas.

1. Quick Sort

O Quick Sort é um algoritmo de ordenação eficiente, que usa a técnica de divisão e conquista. Ele seleciona um pivô e particiona a lista em sublistas menores e maiores que o pivô, ordenando-as recursivamente. Apresenta uma complexidade média de O(n log n), embora possa degradar para O(n²) em casos específicos, como listas já ordenadas.

1. Selection Sort

O Selection Sort é um algoritmo simples que seleciona o menor elemento de uma lista não ordenada e o coloca na posição correta em cada iteração. Com uma complexidade de tempo O(n²), é geralmente menos eficiente que outros algoritmos de ordenação.

1. Merge Sort

O Merge Sort é um algoritmo de divisão e conquista que divide repetidamente a lista em metades até que cada sublista tenha um único elemento, e então as mescla de forma ordenada. Apresenta uma complexidade de tempo O(n log n), sendo eficiente e estável, independentemente da configuração inicial dos dados.

### RESULTADO DA COMPILAÇÃO DOS MÉTODOS DE ORDENAÇÃO

Para uma visualização clara e objetiva dos tempos de execução dos algoritmos de ordenação, foi gerado um gráfico que compara os tempos médios de execução de cada algoritmo. Este gráfico é fundamental para identificar rapidamente quais algoritmos apresentam melhor desempenho em diferentes cenários.

A comparação dos algoritmos de ordenação foi feita com as seguintes condições:

* Valor de vezes para cada algoritmo: 4
* Número de vetores: 6
* Tamanho dos vetores: 1000

|  |  |  |  |
| --- | --- | --- | --- |
| **Algoritmo** | **Pior Caso (s)** | **Melhor Caso (s)** | **Caso Médio (s)** |
| Insertion Sort | 0.001023 | 0.000617 | 0.000766 |
| Shell Sort | 0.000165 | 0.000110 | 0.000135 |
| Bubble Sort | 0.004242 | 0.001710 | 0.002074 |
| Quick Sort | 0.000093 | 0.000030 | 0.000062 |
| Selection Sort | 0.001418 | 0.000716 | 0.000812 |
| Merge Sort | 0.000432 | 0.000191 | 0.000238 |

# Considerações finais

Os resultados mostram variações significativas nos tempos de execução entre os diferentes algoritmos. Os algoritmos como Quick Sort e Merge Sort apresentaram tempos de execução mais eficientes na maioria dos casos, enquanto Bubble Sort e Insertion Sort foram menos eficientes para grandes tamanhos de vetore

* Quick Sort e Merge Sort se destacam pela eficiência, especialmente em grandes conjuntos de dados, devido às suas complexidades médias de O(n log n).
* Insertion Sort e Bubble Sort apresentam desempenho inferior, com complexidades de O(n²) no pior caso, tornando-os inadequados para grandes conjuntos de dados.
* Shell Sort e Selection Sort têm desempenho intermediário, sendo mais eficientes que Insertion Sort e Bubble Sort, mas menos eficientes que Quick Sort e Merge Sort.

Este relatório comparativo evidenciou as diferenças de desempenho entre diversos algoritmos de ordenação. A análise dos tempos de execução forneceu insights sobre quais algoritmos são mais adequados para diferentes cenários. Para aplicações que exigem alta eficiência, algoritmos como Quick Sort e Merge Sort são recomendados

###### REFERÊNCIAS

**DE, T. UNIVALI - Universidade do Vale do Itajai - Intranet**. Disponível em: <https://siaiap27.univali.br/material/?control=Arquivo&action=download&idArquivo=308961&id=173441>. Acesso em: 2 jul. 2024.

###### APÊNDICE A – Título

[Inserir apêndice, se houver].

###### ANEXO A – Título

[Inserir anexo, se houver].