Laboratório de Estrutura de Dados

Primeira versão do projeto da disciplina

Comparação entre os algoritmos de ordenação elementar

Dupla: Matheus Victor Rocha Rodrigues e Luiz Filipe Marinho dos Santos

1. Introdução

Este relatório corresponde ao relato dos resultados obtidos no projeto da disciplina de LEDA que, onde foi realizada uma análise comparativa de algoritmos de ordenação em diferentes conjuntos de testes. Algoritmos de ordenação que servem para organizar elementos em uma sequência específica.

Os principais objetivos deste trabalho são os seguintes:

- Implementar e comparar o desempenho dos algoritmos de ordenação mencionados em 3 cenários de teste.
- Realizar análises estatísticas do desempenho de cada algoritmo, levando em consideração o cenário da amostra de melhor caso, médio caso e pior caso.
- Observar e relatar como os diferentes cenários de teste impactam no desempenho dos algoritmos de ordenação.
- Fornecer informações relevantes acerca da eficiência e aplicabilidade de cada algoritmo em diversas situações.
- Os testes realizados foram respectivamente: Ordenação do arquivo games.csv de acordo com sua data de lançamento em ordem crescente; Ordenação de acordo com o preço em ordem crescente e Ordenação de acordo com o número de conquista em ordem decrescente.

2. Descrição geral sobre o método utilizado

A relevância da análise comparativa está intrinsecamente ligada à considerável disparidade de desempenho que diferentes algoritmos de ordenação podem apresentar, dependendo das características dos dados de entrada. Portanto, a seleção do algoritmo de ordenação mais apropriado para uma tarefa específica deve ser respaldada pelas peculiaridades do conjunto de dados a ser ordenado. Através desta análise comparativa, será possível identificar as virtudes e limitações de cada algoritmo, bem como avaliar sua eficiência em diferentes cenários de teste. Tais resultados podem fornecer valiosos insights para o desenvolvimento de aplicações que requerem operações de ordenação, bem como auxiliar na seleção do algoritmo mais adequado para atender às demandas de cada situação específica.

Descrição geral do ambiente de testes

Os insumos utilizados para a execução dos algoritmos de ordenação foram obtidos a partir das especificações do dispositivo empregado, um Notebook que abriga um processador Intel(R) Core(TM) i5-11400H, 16 GB de RAM, GTX 1650 8GB Nvidia e Sistema Operacional Windows 11 de 64 bits. Essas informações são fundamentais para contextualizar o ambiente em que os testes foram conduzidos e podem exercer influência sobre os resultados obtidos.

3. Resultados e Análise

Estes são os resultados do tempo de ordenação dos algoritmos com base no teste de Ordenação por Preço em ordem Crescente:

Algoritmo	Melhor Caso (s)	Médio Caso (s)	Pior Caso (s)
QuickSort	20,0	3,84	30,4
InsertionSort	0,03	84	216
CountingSort	0,04	0,07	0,40
HeapSort	0,50	0,55	0,45
MergeSort	0,18	0,24	0,18
SelectionSort	330	331	319
QuickSort Mediana de Três	18,5	1,53	4,19

- **QuickSort:** Este algoritmo demonstra um ótimo desempenho no médio caso, mas o tempo de execução aumenta significativamente alto no melhor caso e no pior caso. Isso é típico do QuickSort, que tem desempenho pior quando a escolha do pivô é desfavorável.
- **InsertionSort:** Consistente com sua natureza, o InsertionSort tem um belo desempenho no melhor caso, mas seu tempo de execução cresce rapidamente nos outros casos devido à sua complexidade quadrática.
- **CountingSort:** Tem um tempo de execução muito bom no melhor e médio caso, mas mostra um aumento considerável no pior caso. Isso pode ser devido à implementação específica ou às características dos dados, pois o CountingSort geralmente tem um desempenho consistente devido à sua complexidade linear.
- **HeapSort:** Apresenta um desempenho estável em todos os casos, o que reflete a sua complexidade de tempo O(n log n). Não há mudanças dramáticas entre os diferentes casos, o que indica boa performance no geral.
- **MergeSort:** Assim como o HeapSort, o MergeSort mostra um bom equilíbrio entre todos os casos, com um ligeiro aumento no tempo de execução do médio caso. Sua complexidade O(n log n) faz dele um algoritmo estável.
- **SelectionSort:** Apresenta um comportamento bem pior com relação aos demais algoritmos, com tempo de execução menor no melhor caso e um leve aumento nos outros casos.
- **QuickSort Mediana de Três:** Este parece ser o mais eficiente entre todos os QuickSorts, com o melhor tempo de execução em todos os casos. A técnica da mediana de três ajuda a evitar o pior caso do QuickSort tradicional, tornando-o mais confiável.

Observações gerais: A tabela mostra que algoritmos com complexidade O(n log n), como MergeSort e HeapSort, têm desempenho previsível e robusto em diferentes casos. O QuickSort mediana de três demonstra melhorias significativas em relação ao QuickSort padrão, especialmente no pior caso. Algoritmos com complexidade quadrática, como InsertionSort e SelectionSort, podem ser adequados para conjuntos de dados pequenos ou quase ordenados, mas não são eficientes para entradas maiores. O comportamento atípico do CountingSort em todos os casos sugere que ele pode não ter sido utilizado no tipo de dados ideal ou pode haver questões de implementação.

4. Conclusão

Em resumo, neste trabalho, realizamos uma análise comparativa de algoritmos de ordenação em diferentes cenários de teste (melhor, médio e pior caso). Os algoritmos de ordenação desempenham um papel fundamental na organização dos dados, e entender como eles se comportam em diferentes situações é crucial para a escolha do algoritmo mais adequado para uma tarefa específica.

O principal objetivo foi implementar e comparar o desempenho de diversos algoritmos de ordenação em diferentes cenários de teste, realizar análises estatísticas do desempenho de cada algoritmo, investigar como os diferentes cenários de teste impactam o desempenho e fornecer informações relevantes sobre a eficiência e aplicabilidade de cada algoritmo.

Durante o projeto, executamos sete algoritmos de ordenação: HeapSort, InsertionSort, MergeSort, SelectionSort, QuickSort, CountingSort e QuickSort(com mediana 3). Cada um deles possui características distintas que se adequam a diferentes situações.

- **Melhor caso:** No melhor caso o InsertionSort se mostrou melhor que os demais algoritmos de ordenação.

- **Médio caso:** No médio caso o CountingSort foi o mais rápido dentre os outros algoritmos de ordenação.

- **Pior caso:** No pior caso o CountingSort continuou sendo melhor do que os outros algoritmos de ordenação.

A escolha do algoritmo de ordenação geralmente depende das características do conjunto de dados e dos requisitos de desempenho. Existem vantagens e desvantagens associados a cada algoritmo, e nossa análise ajuda na tomada de decisões sobre quais algoritmos usar em diferentes situações. Concluímos que não existe um algoritmo que funciona melhor em todos os cenários, em vez disso, existem várias opções que devem ser levadas em consideração de acordo com as necessidades específicas de cada aplicação.