Relatório de Análise de Algoritmos de Ordenação

Este relatório foi elaborado como parte de um projeto para a disciplina de Estrutura de Dados, ministrada pelo Professor Thiago, na Universidade Federal de Alagoas (UFAL), Campus Arapiraca. A disciplina tem como objetivo introduzir os alunos aos conceitos fundamentais das estruturas de dados e algoritmos. Neste contexto, o presente trabalho busca aplicar e analisar a eficiência de algoritmos de ordenação em um conjunto de dados específico, utilizando os conhecimentos adquiridos durante o curso.

Equipe

- **Caio Teixeira da Silva - Número de Matrícula 22112243**
- **Noemy Torres Pereira - Número de Matrícula 22112110**

Introdução

Neste relatório, analisamos a performance de três algoritmos de ordenação (Merge Sort, Radix Sort e Bucket Sort) aplicados a um conjunto de dados de animes, que consiste em mais de 10.000 registros obtidos do Kaggle. Os dados foram pré-processados, removendo colunas desnecessárias e convertidos para o formato JSON. Para facilitar a comparação entre as execuções dos algoritmos, optamos por ordenar os animes com base na popularidade.

Utilizamos as implementações dos algoritmos de ordenação fornecidas no código Python e medimos o tempo de execução nos computadores dos membros da equipe. Os testes foram realizados no ambiente local usando o terminal e o Visual Studio Code (VSCode).

Configurações dos computadores utilizados

Caio

- Processador Intel Core i5-9300H CPU @ 2.40GHz 4.10 GHz
- RAM instalada 16,00 GB 3200 MHz
- Windows 10

Noemy

- Processador Intel Core i3-4005U CPU @ 1.70GHz 1.70 GHz
- RAM instalada 4,00 GB 1700 MHz
- Windows 10

Resultados

Algoritmo	Caio - VSCode (s)	Caio - Terminal (s)	Noemy - VSCode (s)	Noemy - Terminal (s)
Merge Sort	0.3600042	0.0758421	0.1840045	0.2319829
Radix Sort	0.1665537	0.0438836	0.0960023	0.0960050
Bucket Sort	0.0349073	0.0109599	0.0200100	0.0280182

Análise

No geral, os tempos de execução obtidos no terminal mostraram-se menores quando comparados àqueles obtidos no ambiente do Visual Studio Code. Essa diferença pode ser explicada por diversas variáveis, como a forma com que o terminal e o VSCode administram a alocação de recursos e a execução de processos, assim como a possível influência de extensões e outros fatores presentes no ambiente do VSCode.

Outro aspecto a ser levado em conta é a variação nas especificações de hardware dos computadores utilizados durante os testes. Máquinas equipadas com processadores mais velozes e maior quantidade de memória RAM tendem a apresentar tempos de execução reduzidos em comparação a dispositivos com processadores menos potentes e menor quantidade de memória RAM. Essa disparidade no desempenho é esperada, uma vez que algoritmos de ordenação podem ser afetados pelas capacidades de processamento e memória disponíveis.

É crucial destacar que os tempos de execução podem ser impactados por fatores externos, como a carga do sistema operacional e outros processos em execução durante os testes. Dessa forma, os resultados apresentados devem ser analisados levando em consideração possíveis variações provenientes desses fatores.

Complexidade dos Algoritmos

A tabela abaixo apresenta a complexidade dos algoritmos de ordenação em notação Big O, considerando o pior caso, caso médio e melhor caso para cada algoritmo.

Algoritmo	Pior caso	Caso médio	Melhor caso
Merge Sort	O(n log n)	O(n log n)	O(n log n)
Radix Sort	O(nk)	O(nk)	O(nk)
Bucket Sort	$O(n^2)$	O(n log n)	O(n+k)

Merge Sort

O Merge Sort é um algoritmo de ordenação baseado na técnica de dividir e conquistar. Sua complexidade é O(n log n) para todos os casos, uma vez que sempre divide o conjunto de dados em duas partes iguais e as combina. Esse algoritmo é estável, o que significa que mantém a ordem relativa dos registros com chaves iguais.

Radix Sort

O Radix Sort é um algoritmo de ordenação não comparativo que ordena os números inteiros processando seus dígitos de forma individual. Sua complexidade é O(nk), onde n é o número de elementos a serem ordenados e k é o número de dígitos do maior número. O Radix Sort funciona melhor quando os dígitos dos números a serem ordenados têm uma distribuição uniforme.

Bucket Sort

O Bucket Sort é um algoritmo de ordenação baseado na técnica de distribuição. Divide o intervalo dos elementos a serem ordenados em diversos "baldes" e, em seguida, ordena cada balde individualmente. A complexidade do Bucket Sort é $O(n^2)$ no pior caso, que ocorre quando todos os elementos estão no mesmo balde. No caso médio, a complexidade é

O(n log n) e no melhor caso é O(n+k), que ocorre quando os elementos estão distribuídos uniformemente entre os baldes e o número de baldes é apropriado para o conjunto de dados.

Referências

- Repositório GitHub com os algoritmos de ordenação: AnimeRanker
- Base de dados utilizada: MyAnimeList Dataset - Animes, Profiles, Reviews