Relatório
Ordenação

Luca Takemura Piccoli

1

Resumo. *Este meta-artigo relata a performance de diversos algoritmos de ordenação, incluindo o algoritmo de ordenação Quick Sort, o algoritmo de ordenação Bubble Sort e o algoritmo de Merge Sort. A pesquisa envolveu a análise comparativa da eficiência desses algoritmos em termos de tempo de execução, número de trocas e numero de iterações, usando conjuntos de dados de diferentes tamanhos.*

1. Introdução

A eficiência na ordenação de conjuntos de dados é um aspecto crítico em computação e processamento de informações. A seleção do algoritmo de ordenação adequado desempenha um papel essencial na otimização de sistemas e aplicativos que manipulam dados. Nesse artigo, veremos o desempenho de vários algoritmos de ordenação. Esse relatório pode ajudar em tomadas de decisões ao escolher o algoritmo de ordenação mais apropriado para suas necessidades.

2. Métodologia

É essencial descrever explicitamente o metodo usado e a máquina que foi usada para obter os resultados, em particular para facilitar os pesquisadores que pretendem reverificar-los.

O Algoritimo de Bubble Sort usou um metodo iterativo, enquanto os algoritimos de Merge Sort e Quick Sort usaram metodos recursivos. Por tanto para contar o numero de iteracoes, no algoritimo de Bubble Sort foi contado o numero de iterações feitas, enquanto que nos algoritimos de Merge Sort e Quick Sort foi contado o numero de chamadas recursivas feitas.

Para medir o tempo de execução foi criado em java uma classe contendo os algoritimos, o codigo foi executado em um ambiente online Replit. Foram testados 5 tamanhos de vetores, 50, 500, 1000, 5000, 10000. Foi usada a função random do java para inserir em cada vetor numeros interiros aleatorios. Para facilitar a replicabilidade dos experimentos foi usada na função random a seed ”1”. Cada algoritimo foi executado 5000 vezes para cada tamanho de vetor, para obter a media de tempo de execução.

Para medir o número de trocas e de iterações foi criado outras classes, para que a contagem não interferice com a performance.

3. Resultados

A análise comparativa dos algoritmos de ordenação revelou insights valiosos sobre seu desempenho:

Table 1. Comparações da média do tempo de execução de cada algoritimo para cada tamanho (tempo em nanosegundos)

Tamanho	Bubble	Merge	Quick
50	12078	29660	14207
500	640180	160058	289814
1000	2483297	433797	1203400
5000	38461470	1239597	12967981
10000	163100157	1718216	57739569

Table 2. Comparações do numero de iterações de cada algoritimo para cada tamanho

Tamanho	Bubble	Merge	Quick
50	1225	99	57
500	124750	999	607
1000	499500	1999	1207
5000	12497500	9999	5983
10000	49995000	19999	11951

Table 3. Comparações do numero trocas de cada algoritimo para cada tamanho

Tamanho	Bubble	Merge	Quick
50	22	286	28
500	197	4488	303
1000	396	9976	603
5000	2008	61808	2991
10000	4024	133616	5975

4. Discução

Fizemos um estudo comparativo de três algoritmos de ordenação (Bubble, Merge e Quick) em vetores de tamanho 50, 500, 1000, 5000, 10000. Para vetores de tamnho 50 o Bubble Sort teve a melhor performance, enquanto que o Merge Sort teve a pior. No entanto com o tamanho dos vetores aumentando o Bubble acaba sendo o pior e o Merge o Melhor. Curioasamente mesmo tendo em todos os casos o menor numero de iterações o Quick sort em segundo lugar em termos de performance. Além disso mesmo tendo o maior número de trocas o Merge tem uma otima performance com conjuntos de dados maiores.

5. Conclusão

Concluindo, as comparações dos algoritmos de ordenação mostra que o Bubble Sort, pode não ser ruim com conjuntos de dados pequenos, mas é ineficiente para conjuntos de dados maiores. O Quick Sort oferece um desempenho sólido e consistente. O Merge Sort se destaca como o algoritmo mais eficiente para conjuntos de dados grandes. Portanto seleção do algoritmo de ordenação deve ser feita com base nas necessidades específicas de cada aplicação, levando em consideração o tamanho do conjunto de dados a ser processado.

6. Referencias

GitHub: Link