UNIVERSIDADE FEDERAL DE SANTA MARIA

CURSO DE CIÊNCIA DA COMPUTAÇÃO

RELATÓRIO DE DESEMPENHO DE ALGORITMOS DE ORDENAÇÃO HÍBRIDOS

Disciplina de Pesquisa e Ordenação de Dados

Docente: Antônio Marcos de Oliveira Candia

Lucas Xavier Pairé

Miguel Brondani

Santa Maria, RS

Sobre

O presente relatório faz parte do método de avaliação da disciplina de Pesquisa e Ordenação de Dados da Universidade Federal de Santa Maria no curso de Ciência da Computação ministrado pelo docente Antônio Marcos de Oliveira Candia. Seu conteúdo tem por finalidade descrever a implementação, análise e comparação de desempenho de algoritmos híbridos de ordenação de dados em contextos de ordenação interna e externa. Foram implementados algoritmos que utilizam conceitos do Bucket, Quick, Bubble e Merge sort.

Algoritmo Híbrido de BucketSort e QuickSort:

O algoritmo implementado utiliza o BucketSort como principal método de ordenação. O BucketSort divide o conjunto de dados em baldes e ordena cada balde separadamente, antes de concatená-los. No entanto, para lidar com o caso em que um balde contém muitos elementos, foi introduzido um segundo método de ordenação: o QuickSort. Quando um balde excede um determinado tamanho limite, definido pelo usuário, o QuickSort é utilizado para ordenar os elementos desse balde. Isso proporciona uma maneira eficiente de lidar com situações em que o BucketSort pode ter desempenho inferior devido à distribuição desigual dos elementos nos baldes. BucketSort: Escolhido como algoritmo principal devido à sua eficiência para distribuir os elementos em baldes com base em seus valores. Isso facilita a ordenação subsequente de cada balde. QuickSort: Selecionado como algoritmo secundário devido à sua rapidez e eficiência em ordenar pequenas partições de dados. É especialmente útil quando um balde do BucketSort contém muitos elementos, garantindo um desempenho satisfatório em todas as situações.

Algoritmo Híbrido de MergeSort e BubbleSort:

O segundo algoritmo implementado é a hibridização entre o MergeSort e BubbleSort. A ideia de mesclagem destas formas de ordenação se deu de forma semelhante ao outro algoritmo híbrido: a possibilidade de utilizar os pontos fortes de ordenação com quantidades diferentes. A simplicidade do Bubble na comparação e troca a posição de elementos em listas o torna eficiente em listas pequenas. Já o Merge se destaca com listas grandes, realizando consecutivas divisões e comparando partes menores, para posteriormente reorganizar os dados. A implementação do algoritmo consiste na divisão da lista de inteiros em porções de tamanho igual ou inferior à 30, unidade arbitrariamente escolhida para a execução do Bubble Sort. Cada sublista é reorganizada em ordem crescente. Após os elementos passarem pelo Bubble e estarem em uma lista geral parcialmente ordenada, as sublistas são finalmente mescladas na ordem correta. O processo é repetido recursivamente até todos os elementos estarem ordenados.

Comparação dos algoritmos:

Os dados apresentados foram obtidos através da mediana do tempo de execução para uma determinada quantidade de inteiros em uma das formas de acesso aos dados. Por padrão, todas as iterações do BucketQuick utilizaram um tamanho de Bucket igual a 30.

BucketQuick

Quantidade Inteiros	Tempo Externo	Tempo Interno
5000	0.01	0.01
10000	0.02	0.02
100000	0.01	0.01
10000000	7.78	4.63
100000000	92.52	59.08

BubbleMerge

Quantidade Inteiros	Tempo Externo	Tempo Interno
5000	0.001	0.0
10000	0.002	0.001
100000	0.016	0.014
10000000	1.777	Não foi capaz
100000000	20.012	Não foi capaz

Conclusões

O algoritmo externo e interno do BucketQuick tiveram desempenho constante e semelhante até aproximadamente 100000 inteiros. Com dados maiores a ordenação Interna apresentou-se mais eficiente, conseguindo desempenhar cerca de 30 segundos mais rápida a ordenação de 100000000 números.

Comparativamente, o BubbleMerge mostrou-se melhor em ambos os contextos até 100000 inteiros. Aproximadamente 10% mais rápido que o BucketQuick. Entretanto, grandes datasets exigem a alocação de mais memória para as sublistas, o que impossibilitou a ordenação interna. Apesar disto, o tempo da ordenação externa do BubbleMerge foi menor que a obtida com o BucketQuick.