ALGORITMO DE ORDENAÇÃO EXTERNA

NATAN LUIZ PAETZHOLD BERWALDT

DENES VARGAS TEIXEIRA

nlberwaldt@inf.ufsm.br e dvteixeira@inf.ufsm.br

1 INTRODUÇÃO

O presente trabalho tem por objetivo implementar um algoritmo de ordenação externa de textos grandes na linguagem de programação C++, utilizando o método QuickSort para a ordenação interna dos itens com tamanho de memória definidos, e a Intercalação Balanceada de vários conjuntos ordenados de dados em arquivos para trabalhar com a relação de memórias interna e externa, gerando um arquivo de saída com as palavras ordenadas após todo o processo de ordenação.

2 ALGORITMO

2.1 - Funcionamento

A princípio temos um arquivo de texto muito grande que não cabe em memória interna para a execução de algoritmos de ordenação, sabendo disso o principio do algoritmo baseia-se em ler sequencialmente o arquivo por pequenos conjuntos e ordenando cada uma dessas partes isoladamente, definidas por um tamanho fixo no programa.

Após as ordenações, o conteúdo estará dividido entre os arquivos temporários e inicia-se o processo de intercalação entre os arquivos ‘temp’ e ‘out’, na primeira execução teremos pequenos conjuntos ordenados divididos nos arquivos ‘temp’ e se intercala os elementos de cada conjunto correspondente de cada arquivo ‘temp’ para um único arquivo ‘out’, assim esses conjuntos de cada arquivo se juntam e formam um conjunto maior de dados ordenados. Entretanto, cada arquivo ‘temp’ possui vários conjuntos ordenados e cada um desses novos conjuntos maiores gerados das intercalações vão ser distribuídos dos arquivos ‘temp’ para um mesmo número de arquivos ‘out’.

Após todos os conjuntos serem intercalados e divididos entre os novos arquivos, os arquivos ‘temp’ serão sobrescritos com a nova intercalação dos arquivos ‘out’ anteriores, e assim gerando conjuntos internos ordenados ainda maiores. Esse processo de intercalação entre esses dois grupos de arquivos temporários é recursivo, até que o conjunto gerado seja do mesmo tamanho do arquivo de entrada e fique em um único arquivo de saída.

Após o arquivo de saída ser gerado, todos os arquivos temporários usado no processo de intercalação são excluídos.

O número de ciclos de intercalação necessárias entre esses arquivos temporários é calculado previamente pelo programa com base no número de conjuntos ordenados em memória interna, para assim ter melhor controle das execuções.

2.2 – Implementação

Logo no início do programa temos dois Defines que controlam o número de arquivos usados entre a intercalação, onde cada ciclo de intercalação usará o dobro de arquivos ali definidos, pois um é o arquivo onde estão os dados temporariamente, e o outro servirá de destino da ordenação parcial.

O segundo Define é o número de Bytes máximo usados pelos conjuntos iniciais de dados no processo de ordenação interna.

O arquivo de entrada é lido no inicio do programa e ordenados os pequenos conjuntos de palavras (respeitando o número de Bytes máximo) por um QuickSort interno.

Cada conjunto ao ser ordenado é enviado para a função ‘grava’, que coloca os conjuntos de dados ordenados nos arquivos, intercalando os arquivos. Após isso, o numero de intercalações é definido pelo calculo baseado no numero de arquivos e pelo número de conjuntos ordenados.

Em seguida, ele inicia a ordenação externa com a função recursiva ‘interpolacao’ que usa dois vetores de arquivos, sendo um onde estão os dados e o outro o destino dos dados da intercalação.

Dependendo do ciclo de execução ele verifica onde estão os dados e para quais conjuntos de arquivos eles irão no ciclo atual da recursão e coloca as primeiras palavras de cada conjunto ordenado em um vetor onde o índice é correspondente ao arquivo temporário no vetor de arquivos.

Procura-se o menor valor entre os arquivos e o escreve em um outro arquivo temporário, quando acaba-se o conjunto atual de cada arquivo, ele vai para o próximo conjunto de cada arquivo novamente, mas alterando o arquivo de destino para distribuir os novos conjuntos.

Ao terminar todos os conjuntos de um grupo de arquivos ele recalcula o número de execuções restantes para sobrar apenas um arquivo e quando o número de execuções previsto chega ao máximo, o ciclo de intercalação dos arquivos acaba chamando a função ‘encerra’.

A função ‘encerra’ é responsável por reescrever todo o código ordenado para um arquivo final de saída. Ao sair dela o programa deleta todos os arquivos temporários e termina a execução.

3 APLICAÇÕES

As aplicações do algoritmo de ordenação são muito variadas. Em especial a de palavras, cujo relacionada ao presente trabalho, pode servir na criação de dicionários, ou listas ordenadas em um banco de dados, os quais permitem uma busca mais rápida por um nome específico.

Essas listas por exemplo podem ser de uma tabela com nomes de clientes com cadastro em uma empresa ordenados por ordem alfabética, listas telefônicas, ou qualquer

outra lista que possa ter utilidade em estar ordenada, tanto para utilização de buscas automáticas, utilizando métodos de pesquisa computacionais, como também na busca humana, facilitando aos indivíduos encontrar algo apenas passando os olhos pela lista.

Referências

Nunes, Graça. Ordenação Externa. Inst. de Ciências Matemática e de Computação-USP, http://wiki.icmc.usp.br/images/1/1e/SCC0203-1o-2012-15.OrdenacaoExterna.pdf.

Knuth, D. E.(1998) The Art of Computer Programming, Volume 3: Sorting and Searching, Second Edition. Addison-Wesley, Section 5.4: External Sorting, pp.254-[...].

Kutova, Marcos. (2013) Intercalação Balanceada –Ciencia da Computação PUC Minas. Vídeo: http://www.showme.com/sh/?h=0a0OGGW.

De Assis, Guilherme. (2012) Universidade Federal de Ouro Preto. Ordenação Externa. Material de Estruturas de Dados II. www.decom.ufop.br/guilherme/BCC203/geral.

Dittrich, Jens. (2014) External Merge Sort. (13m:34s), https:/www.youtube.com/watch?v=ATK74YSzwxg .

Quicksort, Wikipedia, a enciclopédia livre. https://pt.wikipedia.org/wiki/Quicksort.