

Ordenação Externa

1. Objetivo

O objetivo deste trabalho é projetar e implementar um sistema de programas para ordenação de arquivos que não cabem na memória primária, o que nos obriga a utilizar um algoritmo de ordenação externa [Ziviani, 2011; Capítulo 4, Exercício 26].

2. Problema

Existem muitos métodos para ordenação externa. Entretanto, a grande maioria utiliza a seguinte estratégia geral : blocos de entrada tão grandes quanto possível são ordenados internamente e copiados em arquivos intermediários de trabalho. A seguir, os arquivos intermediários são intercalados e copiados em outros arquivos de trabalho, até que todos os registros são intercalados em um único bloco final representando o arquivo ordenado.

Um procedimento simples e eficiente para realizar essa tarefa é o de colocar cada bloco ordenado em um arquivo separado até que a entrada é toda lida. A seguir, os f primeiros arquivos são intercalados em um novo arquivo e esses f arquivos removidos. f é uma constante que pode ter valores entre 2 e 10, chamada Ordem de Intercalação. Esse processo é repetido até que fique apenas um arquivo, o arquivo final ordenado. A cada passo, o procedimento de intercalação nunca tem de lidar com mais do que f arquivos de intercalação mais um único arquivo de saída.

Para tornar mais claro o que cada aluno tem de realizar, apresentamos no Programa 1 um primeiro refinamento do procedimento que permite implementar a estratégia descrita anteriormente. Pode-se observar que grande parte do procedimento lida com criação, abertura, fechamento e remoção de arquivos em momentos adequados.

A fase de intercalação utiliza dois índices, Low e High, para indicar o intervalo de arquivos ativos. O índice High é incrementado de 1, OrdemIntercalacao arquivos são intercalados a partir de Low e armazenados no arquivo High e, finalmente, Low é incrementado de OrdemIntercalacao. Quando Low fica igual ou maior do que High, a intercalação termina com o último bloco resultante High totalmente ordenado.

É importante observar que as interfaces dos vários procedimentos presentes no código do Programa 1 não estão completamente especificadas.

Para mostrar o funcionamento dos módulos do programa OrdeneExterno, você deve proceder da seguinte forma.

- a) Use um arquivo contendo 22 registros, em que a chave de cada registro é uma letra maiúscula, conforme mostrado a seguir :

INTERCALACAO BALANCEADA

Mais especificamente, a chave do primeiro registro é I, a chave do segundo registro é N, a chave do terceiro registro é T, e assim por diante.

Para fins de teste, você deve colocar em cada registro um campo associado ocupando 31 bytes, para que o registro fique com um total de 32 bytes. Permita apenas $m = 3$ registros na memória principal. Use uma ordem de intercalação $f = 2$.

- i. Leia as chaves dos registros de um arquivo denominado arquivoentrada.txt, em que cada linha contém a chave de um registro. Figura 1 mostra o arquivoentrada.txt.
- ii. Faça a impressão dos blocos ordenados obtidos na primeira fase do programa.

- iii. Na segunda fase do programa, para cada iteração do anel, mostre o conteúdo de: Low, Lim, High, nome dos arquivos de entrada abertos, conteúdo dos arquivos de entrada, nome do arquivo de saída aberto, e conteúdo do arquivo de saída.

I
N
T
E
R
C
A
L
A
C
A
O
B
A
L
A
N
C
E
A
D
A

Figura 1: arquivoentrada.txt.

- b) Gere arquivos contendo $n = 2^{25}$, 2^{26} e 2^{27} registros de 32 bytes de tamanho, cada registro contendo um campo chave constituído por um número inteiro obtido com o auxílio de um gerador de números pseudo-aleatórios. Faça a medida do tempo necessário para ordenar estes arquivos com $m = n/4$, $n/16$ e $n/256$ registros na memória principal e $f = 2$, $f = 3$ e $f = 4$ arquivos de intercalação. A Tabela 1 mostra o formato de impressão dos resultados.
- c) Gere um relatório contendo as impressões do item a especificadas nos itens a.ii e a.iii, e os resultados do item b no formato da Tabela 1 e de gráficos. Os gráficos devem mostrar o tempo de ordenação em função do tamanho do arquivo, n , para diferentes quantidades de registros em memória principal, m , e diferentes quantidades de arquivos de intercalação, f . Gere um gráfico para cada quantidade de arquivo de intercalação, f . Nos gráficos, o eixo-x deve denotar o tamanho do arquivo, n , o eixo-y o tempo de ordenação e as linhas a quantidade de registros em memória principal, m .

Tabela 1: Tempo total real em segundos para ordenar arquivos com $n = 2^{25}$, 2^{26} e 2^{27} registros de 32 bytes de tamanho, $m = n/4$, $n/16$ e $n/256$ registros na memória principal e $f = 2$, $f = 3$ e $f = 4$ arquivos de intercalação.

$f = 2$				
n	$m = n/4$	$m = n/16$	$m = n/256$	
2^{25}				
2^{26}				
2^{27}				
$f = 3$				
n	$m = n/4$	$m = n/16$	$m = n/256$	
2^{25}				
2^{26}				

$$\begin{array}{ccccccc}
 & & 2^{27} & & & & \\
 & & f = 4 & & & & \\
 & n & & m = n/4 & & m = n/16 & & m = n/256 \\
 & 2^{25} & & & & & & \\
 & 2^{26} & & & & & & \\
 & 2^{27} & & & & & &
 \end{array}$$

Programa 1: Primeiro refinamento do procedimento OrdeneExterno, o qual está disponível em <http://www2.dcc.ufmg.br/livros/algoritmos/cap4/codigo/c/4.25-refinaordexterno.c>.

```

void OrdeneExterno()
{
    int OrdemIntercalacao=2;
    int NBlocos = 0;
    ArqEntradaTipo ArqEntrada, ArqSaida;
    ArqEntradaTipo[OrdemIntercalacao] ArrArqEnt;
    short Fim;
    int Low, High, Lim;
    NBlocos = 0;
    ArqEntrada = abrir arquivo a ser ordenado;
    do /*Formacao inicial dos NBlocos ordenados */
    {
        NBlocos++;
        Fim = EnchePaginas(NBlocos, ArqEntrada);
        OrdeneInterno;
        ArqSaida = AbreArqSaida(NBlocos);
        DescarregaPaginas(ArqSaida);
        fclose(ArqSaida);
    } while (!Fim);
    fclose(ArqEntrada);
    Low = 0;
    High = NBlocos-1;
    while (Low < High) /* Intercalacao dos NBlocos ordenados */
    {
        Lim = Minimo(Low + OrdemIntercalacao -1, High);
        AbreArqEntrada(ArrArqEnt, Low, Lim);
        High++;
        ArqSaida = AbreArqSaida(High);
        Intercale(ArrArqEnt, Low, Lim, ArqSaida);
        fclose(ArqSaida);
        for(i= Low; i < Lim; i++)
        {
            fclose(ArrArqEnt[i]);
            Apague_Arquivo(ArrArqEnt[i]);
        }
        Low += OrdemIntercalacao;
    }
    Mudar o nome do arquivo High para o nome fornecido pelo usuario;
}

```

3. Instruções de Desenvolvimento

- Este trabalho deverá ser desenvolvido em grupos de dois alunos.
- O programa deste trabalho deverá ser implementado usando a linguagem C.
- O programa deverá também ser modularizado. Crie arquivos de extensão .c e .h para cada módulo do programa. Crie arquivos exclusivos para definir tipos abstratos de dados.
- O programa deverá também ser compilado e executado com o aplicativo make e as regras all, runa, e runb. O programa deverá ser compilado com o comando make all; o item a executado com o comando make runa; e o item b executado com o comando make runb.

4. Instruções de Entrega

A definir.

5. Maiores Detalhes

Maiores detalhes serão discutidos em sala de aula. Considerações feitas em sala terão valor superior ao daquelas contidas nesta especificação.

6. Referência

N. Ziviani. *Projeto de Algoritmos: com Implementações em PASCAL e C*. 3a. edição revista e ampliada. São Paulo: CENGAGE Learning, 2011.