Algoritmos e Estruturas de Dados II

Processamento Co-sequencial & Ordenação Externa I

Prof. Ricardo J. G. B. Campello

Operações Co-sequenciais

- Processamento coordenado (simultâneo) de duas ou mais "listas" de entradas seqüenciais, produzindo uma única lista como saída
- Exemplos Típicos:
  - **merging** (união) ou **matching** (intersecção) de dois ou mais conjuntos de registros mantidos em arquivos separados e ordenados por chave
  - Exemplo (só chaves dos registros):

Lista 1	Lista 2
Adams	Anderson
Davis	Foster
Foster	Rosewald
Garwich	Schmidt
Rosewald	
Turner	

matching	merging
Foster	Adams
Rosewald	Anderson
	Davis
	Foster
	Garwich
	Rosewald
	Schmidt
	Turner

Operações Co-sequenciais

- **Merging:**
  - lê uma entrada de cada lista/arquivo e as compara
  - se ambas são iguais, copia a entrada na lista/arquivo de saída e avança para a próxima entrada em cada lista
  - se uma das entradas é menor:
    - copia essa entrada na saída e lê a próxima entrada da respectiva lista
  - retorna ao segundo passo até que ambas as listas terminem

Merging

- **Exemplo:**

```
#include <stdio.h>
#define VALOR_ALTO 100000000
bool MAIS_CHAVES_EXISTEM = true;

void main(void) {
     int chave1 = 0, chave2 = 0;
     FILE *pt_arq_in1, *pt_arq_in2, *pt_arq_out;
     pt_arq_in1 = fopen("arq_in1.txt","r");
     pt_arq_in2 = fopen("arq_in2.txt","r");
     pt_arq_out = fopen("arq_out.txt","w");
     chave1 = input(pt_arq_in1,chave2);
     chave2 = input(pt_arq_in2,chave1);
     while (MAIS_CHAVES_EXISTEM){
           if (chave1 < chave2){
               fprintf(pt_arq_out,"%d\n",chave1);
               chave1 = input(pt_arq_in1,chave2); }
           else if (chave2 < chave1){
                     fprintf(pt_arq_out,"%d\n",chave2);
                     chave2 = input(pt_arq_in2,chave1); }
                else { fprintf(pt_arq_out,"%d\n",chave1);
                     chave1 = input(pt_arq_in1,chave2);
                     chave2 = input(pt_arq_in2,chave1); }
     }
}                                                  // continua...
```

Nota: Implementação acima assume que listas são simplesmente de números inteiros (arquivos texto com um número por linha)

Merging

- **Exemplo:**
  (continuação)

```
int input(FILE *pt_arq, int CHAVE_OUTRA_LISTA){
      int flag;
      int chave;
      flag = fscanf(pt_arq,"%d",&chave);
      if (flag == EOF){
          chave = VALOR_ALTO; // lista atual finalizada
          if (CHAVE_OUTRA_LISTA == VALOR_ALTO)
              MAIS_CHAVES_EXISTEM = false; // duas listas finalizadas
      }
      return chave;
}
```


Operações Co-sequenciais

- **Matching:**
  - lê uma entrada de cada lista/arquivo e as compara
  - se ambas são iguais, copia a entrada na lista/arquivo de saída e avança para a próxima entrada em cada lista
  - se uma das entradas é menor:
    - lê a próxima entrada da respectiva lista
  - retorna ao segundo passo até que uma das listas termine

Matching

- **Exemplo:**

```
#include <stdio.h>
bool MAIS_CHAVES_EXISTEM = true;

void main(void) {
     int chave1 = 0, chave2 = 0;
     FILE *pt_arq_in1, *pt_arq_in2, *pt_arq_out;
     pt_arq_in1 = fopen("arq_in1.txt","r");
     pt_arq_in2 = fopen("arq_in2.txt","r");
     pt_arq_out = fopen("arq_out.txt","w");
     chave1 = input(pt_arq_in1);
     chave2 = input(pt_arq_in2);
     while (MAIS_CHAVES_EXISTEM){
          if (chave1 < chave2){
               chave1 = input(pt_arq_in1); }
          else if (chave2 < chave1){
                    chave2 = input(pt_arq_in2); }
               else { fprintf(pt_arq_out,"%d\n",chave1);
                      chave1 = input(pt_arq_in1);
                      chave2 = input(pt_arq_in2); }
     }
}                                              // continua...
```

Nota: Implementação acima assume que listas são simplesmente de números inteiros (arquivos texto com um número por linha)

Matching

- **Exemplo:**
  (continuação)

```
int input(FILE *pt_arq){
      int flag;
      int chave;
      flag = fscanf(pt_arq,"%d",&chave);
      if (flag == EOF) MAIS_CHAVES_EXISTEM = false; // lista finalizada
      return chave;
}
```

Matching

- **Nota:**
  - Os códigos anteriores fazem a leitura e escrita de registros um a um, o que pode tornar as operações ineficientes
  - Na prática, para reduzir o no. de acessos ao dispositivo externo, pode-se lidar com leitura/escrita de **blocos** de registros
    - Registros são lidos em blocos para **buffers** de memória em RAM
      - 1 buffer para cada lista/arquivo de entrada
    - Operações são executadas nos buffers em RAM
      - Cada buffer é recarregado sempre que finalizado
    - Resultado é escrito em um buffer de saída também em RAM
      - Buffer é descarregado no arquivo de saída sempre que cheio

Operações Co-sequenciais

- **Muti-Way:**
  - Operações co-sequenciais, tais como merging e matching, não precisam se restringir a operar em apenas duas listas (2-way)
  - Versões **k-way** são obtidas como generalizações de **2-way**
  - Muda apenas a comparação:
    - Avança-se para a próxima entrada em toda lista cuja entrada corrente for mínima dentre todas as entradas correntes
  - Exemplo 3-way:

Lista 1	Lista 2	Lista 3	Matching
Adams	Anderson	Adams	Foster
Davis	Foster	Foster	Rosewald
Foster	Rosewald	Rosewald	
Garwich	Schmidt	Schmidt	
Rosewald		Turner	
Turner			

Ordenação Externa

- **Ordenação Externa via Muti-Way Merging:**
  - Pode-se utilizar uma modificação da operação co-sequencial de merging multi-way para ordenar um arquivo grande em disco
    - Merging como soma (chaves repetidas) e não como união: vide exercícios
  - **Idéia**:
    - Carrega-se toda a RAM disponível com parte do arquivo
    - Ordena-se os registros em RAM com um algoritmo *in-place*
    - Escreve-se os registros ordenados em um arquivo separado
    - Repete-se os passos acima até encerrar o arquivo original
      - Se a RAM disponível comporta (1/k * no. de regs. do arq. original), então ter-se-ão **k** arquivos ordenados
    - Aplica-se então multi-way merging nos arquivos ordenados

Ordenação Externa

- **Multi-Way Merging nos Arquivos Ordenados**:
  - Conforme descrito anteriormente, para maximizar a eficiência da operação de merging, opera-se com L/E de blocos em RAM
  - RAM disponível é sub-dividida em **k** buffers de entrada
    - 1 buffer para cada um dos **k** arquivos ordenados
    - "RAM disponível" já desconta uma porção reservada para buffer de saída
  - Buffers são preenchidos com 1/**k** regs. dos respectivos arquivos
  - Merging é realizado em RAM
    - Cada buffer de entrada é recarregado sempre que vazio
      - Como cada arquivo ordenado é exatamente do tamanho da RAM disponível, cada buffer é carregado **k** vezes
    - Buffer de saída é descarregado no arquivo de saída sempre que cheio

Ordenação Externa

- **Exemplo (40-way):**
  - Arquivo com 40Gb (40.000.000 de registros de 1Kb)
  - 1GB de RAM disponível
    - RAM comporta 1.000.000 de registros
  - Com 40 ordenações em RAM produz-se 40 arquivos ordenados
    - Cada arquivo com 1.000.000 de registros do arquivo original
  - Com merging 40-way produz-se um único arquivo ordenado
    - RAM é dividida em 40 blocos de 1/40 Gb

Ordenação Externa

- **Desempenho:**
- Apenas a fase de multi-way merging do exemplo anterior necessita, no mínimo, **1600 seeks** no disco:
  - Mesmo que cada bloco de registros possa ser lido com um único seek, serão necessários **k** = 40 seeks para cada arquivo
  - Como são 40 arquivos, tem-se ao menos $40 \times 40 = 1600$ seeks
    - Fora os seeks para escrita do arquivo de saída

Ordenação Externa

- **Estratégias para Melhorar o Desempenho:**
  - Aumento do tamanho e redução da quantidade de arquivos ordenados para serem fundidos:
    - Técnica de *Replacement Selection*
    - Seção 8.4.2 de (Folk & Zoellick, 1987)
  - Merging em múltiplos passos (a seguir)
  - Leitura/Escrita paralelizada:
    - Arquivos em múltiplos dispositivos de armazenamento externo
    - Próxima aula...

Ordenação Externa

- **Merging em Múltiplos Passos:**
  - No exemplo anterior, ao invés de fazer o merging dos 40 arquivos em um único de uma só vez, será que é vantajoso fazer o merging em múltiplas etapas ? Por exemplo:
    - 1ª etapa: 5 mergings de 8 arquivos, produzindo 5 arquivos
      - cada merging demanda $8 \times 8 = 64$ seeks
      - total de $64 \times 5 =$ **320 seeks**
    - 2ª etapa: 1 merging dos 5 arquivos, produzindo 1 arquivo
      - RAM disponível comporta 1/8 de cada arquivo
        - logo, cada um dos 5 blocos em RAM comporta 1/40 de cada arquivo
      - 40 seeks por arquivo $\Rightarrow$ total de **200 seeks**

Ordenação Externa

- **Merging em Múltiplos Passos:**
  - A estratégia de 2 passos no exemplo resultou em um total de 320+200 = **520 seeks**, contra **1600** da estratégia anterior
    - No entanto, esses valores são apenas **limitantes inferiores**:
      - Consideram que um bloco de registros sempre pode ser lido em um único seek, independente de seu tamanho
    - Mesmo com essa simplificação, em cada caso os seeks estarão associados à transmissão de quantidades diferentes de registros:
      - As análises anteriores ignoram o tempo de transmissão de registros, dentre vários outros aspectos que dependem de hardware e software
  - Escolha de uma determinada estratégia em um projeto deve considerar em detalhes todos esses aspectos...

Exercícios

- Modifique o código em C da operação **merging** de forma tal que o usuário não precise definir o valor limitante superior denominado VALOR_ALTO, ou seja, de forma que essa variável seja eliminada.
- Modifique o código em C da operação **merging** de forma a produzir uma operação que resulte na soma das listas originais, isto é, que resulte em uma lista de saída que inclua as entradas repetidas em ambas as listas de entrada.
- Modifique os códigos em C das operações **merging** e **matching** para o caso mais geral de processamento simultâneo de múltiplas listas (**multi-way**).

Exercícios

- O exemplo dado de ordenação de um arq. de 40Gb via merging de 2 passos utilizou uma configuração denominada de 8:8:8:8:8, que significa 5 mergings 8-way (1º passo) seguidos de um merging 5-way (2º passo)
  - Calcule o limitante inferior para o número de seeks em uma estratégia 5:5:5:5:5:5:5:5, ou seja, 8 mergings 5-way seguidos de um merging 8 way.
- Calcule a quantidade estimada de registros transferidos em cada seek das duas diferentes estratégias do exercício anterior
  - Note que sua resposta deve ser calculada de forma independente para cada um dos dois passos do procedimento de merging

Bibliografia

- **M. J. Folk and B. Zoellick, *File Structures: A Conceptual Toolkit*, Addison Wesley, 1987.**