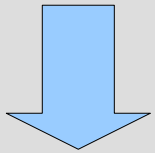
Replacement Selection

Prof. Muriel de Souza Godoi

msgodoi@din.uem.br
Bloco 13 – Sala 1

Efeitos do aumento dos tamanhos das corridas

- E se pudéssemos criar corridas de 2MB cada?
  - Teríamos 400 corridas (800MB/2MB)
  - 400 *buffers* de 2,5KB
  - 800 *seeks* por corrida (2MB/2,5KB)
  - Total de *seek* para a leitura do *merge*:
    - 800 *seeks* X 400 corridas = 320.000 *seeks*



  - Número de *seeks* diminuído pela metade!

Problema

- Como aumentar o tamanho das corridas sem aumentar a quantidade de memória disponível no sistema?
- **Mais uma vez**
  - Sacrificar as operações em memória em detrimento de um número menor de seeks
- **Solução:**
  - Replacement Selection

Replacement Selection

- **Idéia básica:**
  - Selecionar na memória a menor chave
  - Escrever essa chave no arquivo de saída
  - Usar seu lugar (replace it) para uma nova chave (da lista de entrada).

Replacement Selection - Passos

- **Passo 1:** Leia um conjunto de registros e ordene-os utilizando heapsort, criando uma heap primária.
- **Passo 2:** Ao invés de escrever, neste momento, a primary heap inteira ordenadamente e gerar uma corrida, escreva apenas o registro com menor chave.
- **Passo 3:** Busque um novo registro no arquivo de entrada
  - Se chave lida for maior que a já chave escrita, insira o registro normalmente na heap
  - Se a chave lida for menor que qualquer chave já escrita, insira o registro numa secondary heap(usando a mesma memória!)
- **Passo 4:** Repita o passo 3 enquanto existirem registros a serem lidos. Ficando a primary heap vazia, transforme a secondary heap em primary heap, e repita passos 2 e 3.

Replacement Selection – Exemplo Simples

Input:
21, 67, 12, 5, 47, 16

Input restante	**Memoria**(p=3)	**Saída da corrida**
21, 67, 12	5 47 16	-
21, 67	12 47 16	5
21	67 47 16	12, 5
-	67 47 21	16, 12, 5
-	67 47 -	21, 16, 12, 5
-	67 - -	47, 21, 16, 12, 5
-	- - -	67, 47, 21, 16, 12, 5

Replacement Selection – Exemplo Completo (1/2)

Entrada

33, 18, 24, 58, 14, 17, 7, 21, 67, 12, 5, 47, 16

↑ **Início da string**

Restante da entrada	**Memoria**(P=3)	**Saída da corrida(A)**
33, 18, 24, 58, 14, 17, 7, 21, 67, 12	5 47 16	-
33, 18, 24, 58, 14, 17, 7, 21, 67	12 47 16	5
33, 18, 24, 58, 14, 17, 7, 21	67 47 16	12, 5
33, 18, 24, 58, 14, 17, 7	67 47 21	16, 12, 5
33, 18, 24, 58, 14, 17	67 47 (7)	21, 16, 12, 5
33, 18, 24, 58, 14	67 (17) (7)	47, 21, 16, 12, 5
33, 18, 24, 58	(14) (17) (7)	67, 47, 21, 16, 12, 5

Replacement Selection – Exemplo Completo (2/2)

- Fim da primeira corrida → Início da Segunda

Restante da entrada	Memoria(p=3)	Saída da corrida(B)
33, 18, 24, 58	14 17 7	-
33, 18, 24	14 17 58	7
33, 18	24 17 58	14, 7
-	24 18 58	17, 14, 7
-	24 33 58	18, 17, 14, 7
-	- 33 58	24, 18, 17, 14, 7
	- - 58	33, 24, 18, 17, 14, 7
	-	58, 33, 24, 18, 17, 14, 7

Replacement Selection

- Dadas P posições de memória, qual o tamanho, em média, da corrida que o algoritmo de *replacement selection* vai produzir?
  - **Resp:** A corrida terá, em média, tamanho 2P.

- Quais são os custos de se usar *replacement selection*?
  - **Resp:** Precisaremos de *buffers* p/ I/O e, portanto, não poderemos utilizar toda a MEMÓRIA disponível para ordenação

Replacement Selection - Custo

- Se o buffer suporta 2.500 registros
  - podemos executar leituras seqüencias de 2.500 registros de cada vez, e portanto precisamos de 8.000.000/2.500 = 3.200 seeks para acessar todos os registros do arquivo.

- Portanto, o passo de ordenação requer 3.200 seeks para leitura, 3.200 para escrita, **6.400 total.**

Replacement Selection - Custo

- Se os registros ocorrem em seqüência aleatória de chaves:
  - Tamanho médio da corrida será 2 X 7.500 = 15.000 registros, e teremos ~ 8.000.000/15.000 = 534 dessas corridas sendo produzidas.
- Para o passo de intercalação:
  - Dividimos a memória total (1 MB) em 534 buffers que podem conter cerca de 1 MB/534 = 18.73 registros.
  - Portanto, realizaremos cerca de 15.000/18.73 = 801 seeks por corrida

- **Total:** 831 seeks X 534 corridas = **427.734 seeks**

Replacement Selection

- **Resumindo**:
  - **K-way merge:** 800 ordenações em memória: 800 corridas, 681.600 seeks, **4h 58min** para seek + rotational delay
  - **Replacemente selection**: 534 corridas: 434.134 seeks, **3h 48 min**
  - **Replacement selection:** 200 corridas: 206.400 seeks, **1h 30 min**

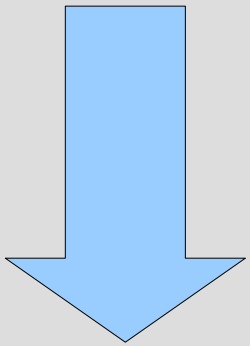
  - **Obs:** O último caso ilustra ganhos quando os registros estão parcialmente ordenados.

Replacement Selection + Multiple Step Merging

- Se tivéssemos utilizado Multiple Step Merging teríamos:
  - **2-Step Merging:** 800 corridas, 25x32 way + 25 way merge, 127.000 seeks, **56 min**
  - **Replacement selection**: 534 corridas, 19x28 way + 19 way merge, 124.438 seeks, **55 min**
  - **Replacement selection:** 200 corridas, 20x10 way + 20way merge, 110.400 seeks, **48 min**

Replacement Selection + Multiple Step Merging

- Portanto o uso de intercalação em 2 passos melhora todos os casos.



- O método utilizado para formar as corridas não é tão importante quanto utilizar intercalações múltiplas!

Replacement Selection + Uso de 2 discos rígidos

- Se dois discos rígidos estiverem disponíveis:
  - Um poderia ser utilizado para escrita e outro para leitura.
  - Deste modo podemos sobrepor input e output.
- A memória deve ser configurada para tirar vantagem dos discos:
  - Dois buffers são utilizados para double buffering, e o resto da memória é utilizado para heap.
- Como o processo de ordenação pode tirar vantagem dessa situação ?

Replacement Selection + Uso de 2 discos rígidos

- **Fase de ordenação**
  - Sobrepondo processamento e input
    - Enquanto um buffer tem seus registros passados para a heap, o outro pode estar recebendo registros do disco.
  - Sobrepondo seleção e output
    - Enquanto um buffer está sendo enchido com registros da árvore, o conteúdo do outro pode estar sendo enviado para o disco
- **Fase de intercalação**
    - O disco de saída torna-se o de entrada e vice-versa. Como as corridas estão no mesmo disco, teremos seeking na entrada. A saída é sequencial.

Pontos Básicos para Ordenação Externa

- Uma lista de ferramentas conceituais para melhorar ordenação externa inclui:
  - Para ordenação interna, em-MEMÓRIA, use heapsort para formar corridas. Com heapsort e double buffering, é possível sobrepor processamento com input e output.
  - Use o máximo de MEMÓRIA possível. Isso permite corridas maiores e buffers maiores na fase de intercalação.
  - Se o número de corridas é muito grande, de modo que o seek e rotation times são maiores que o tempo de transmissão, use intercalação em múltiplos passos: aumenta o tempo de transmissão, mas diminui em muito o número de seeks.

Pontos Básicos para Ordenação Externa

- Considere utilizar Replacement Selection para formação da corrida inicial, especialmente se existe a possibilidade das corridas estarem parcialmente ordenadas.
- Utilize mais de um drive de disco para sobrepor I/O e processamento. Principalmente se não existem outros usuários no sistema.
- Lembre-se dos elementos fundamentais da ordenação externa, e os seus custos relativos.
- E procure maneiras de tirar vatagens das novas arquiteturas.