Ordenação Externa

Consiste em ordenar de arquivos maior que a memória interna possível. Seus métodos são diferentes dos que ordenação interna, seus algoritmos devem diminuir o número de acessos às unidades externa.

Fatores de que determinam a diferença entre técnicas de ordenação externa e interna:

- Custo de acesso a memória secundária é muito maior que o acesso a memória principal. O custo principal na ordenação externa está relacionado à transferência de dados entre a memória interna e externa;

- Restrições de acesso a dados. Fitas são acessadas somente sequencialmente. Em disco. Acesso direto é muito caro;

- Os métodos de ordenação externa são dependentes do estado atual da tecnologia.

Métodos

- Ordenação por intercalação:

É o método mais importante de ordenação externa é o de ordenação por intercalação. Intercalar significa combinar dois ou mais blocos ordenados em um bloco único. Utilizada para auxiliar na ordenação;

- Foco dos algoritmos:

Reduzir os números de vezes que se passa por um arquivo, sendo assim, uma boa medida de complexidade dos algoritmos é o número de vezes que um item é lido ou escrito na memória interna. Um bom número é >= 10;

- Estratégia geral dos métodos:

1 - Quebrar o arquivo em blocos no espaço de memória interna disponível;

2 – Ordenar cada bloco na memória interna;

3 – Intercalar os blocos ordenados, fazendo várias passadas sobre o arquivo. *Cada passada cria blocos ordenados cada vez maiores, até que o arquivo esteja totalmente ordenado.

Intercalação balanceada de Vários Caminhos

- Fase de criação dos blocos ordenados envolvem:

1- Quebra do arquivo em blocos do tamanho da memória interna disponível;

2- Ordenação de cada bloco na memória interna.

- Fase de intercalação envolve:

1- Leitura do primeiro registro de cada fita;

2- Retirada do registro contendo a menor chave, armazenando-o em uma fita de saída;

3- Leitura de um novo registro da fita de onde o registro é proveniente.

- Ao ler o terceiro registro de um dos blocos, a fita correspondente fica inativa;

- A fita é reativada quando os terceiros registros das outras fitas forem lidos;

- Neste momento, um bloco de nove registros ordenados foi formado na fita de saída.

4- Repetir o processo para os blocos restantes.

Exemplificação:

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

Fita 1: I N T | A C O | A D E

Fita 2: C E R | A B L | A

Fita 3: A A L | A C N

➔ Primeira passada nos blocos

Fita 4: A A C E I L N R T (Primeira parte de cada fita)

Fita 5: A A A B C C L N O (Segunda parte de cada fita)

Fita 6: A A D E (Terceira parte de cada fita)

➔ Segunda passada nos blocos

Fita 1: A A A A A A A B C C C D E E I L L N N O R T

Fita 2:

Fita 3:

➔ Arquivo ordenado

Nesse exemplo foram utilizadas 2f fitas, mas é possível usar apenas f+1 fitas, sendo que nesse método são feitas mais passagens e existe uma redistribuição entre os blocos**,** isso causa uma passada a mais oara cada intercalação.

Implementação por meio de substituição por seleção

A implementação do método anterior (intercalação balanceada) pode ser feita utilizando filas de prioridade. As fases de quebra e intercalação podem ser implementadas de forma eficiente e elegante, substituindo o menor item existente na memória interna pelo próximo item da fita de entrada.

Estrutura ideal para a implementação: heap

Operação:

- Retirar o menor item da fila;

- Colocar um novo item em seu lugar;

- Reconstituir a propriedade do heap.

Processo de funcionamento para gerar os blocos ordenados:

- M itens são inseridos na fila de prioridades inicialmente vazia;

- O menor item da fila de prioridade é substituído pelo próximo item de entrada. *Se o próximo item é menor do que o que está saindo, então ele deve ser marcado como membro do próximo bloco, sendo tratado como o maior tem do bloco atual.

- Quando o item marcado vai para o início da fila, o bloco atual é encerrado e um novo bloco de ordenação é criado.

Após gerados os blocos ordenados, faz-se intercalação deles utilizando fila de prioridades:

- Monte uma fila de prioridade de tamanho F a partir dos primeiros itens de cada um do F blocos ordenados;

- Repita o processo abaixo até que não haja mais itens nos blocos ordenados:

1- Substitua o item do topo da fila de prioridades, escrevendo-o em uma fita de saída, pelo próximo item do mesmo bloco do item que está sendo substituído;

2- Reconstitua a propriedade da fila de prioridades.

Considerações:

- Para pequenos valores de F, não é vantajoso utilizar esse método, já que o menor item pode ser obtido por F-1 comparações;

- Quando F>=8, o método é considerado adequado, realizando log2 F comparações para obter o menor item;

- Deve-se procurar implementar leitura, escrita e processamento interno dos dados simultaneamente;

- Computadores de maior porte possuem uma ou mais unidades independentes para processamento de entrada e saída, podendo realizar processamento e operações de E/S simultaneamente;

- Sedegwick (1988) sugere considerar F grande o suficiente para completar a ordenação em poucos passos, porém, a melhor escolha para F depende de parâmetros relacionados com o sistema de computação disponível.

Exemplificação:

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

(*) Marcados por serem menos que o elemento que saiu

ENTRADA	FILA 1	FILA 2	FILA 3
E	I	N	T

Bloco 1: I

R	N	E*	T

Bloco 1: I N

C	R	E*	T

Bloco 1: I N R

A	T	E*	C*

Bloco 1: I N R T

L	A*	E*	C*

*Criação do novo bloco por todos os elementos estarem marcados

Bloco 1: I N R T

Bloco 2: A

A	C	E	L

Bloco 1: I N R T

Bloco 2: A C

C	E	A*	L

Bloco 1: I N R T

Bloco 2: A C E

A	L	A*	C*

Bloco 1: I N R T

Bloco 2: A C E L

O	A*	A*	C*

*Criação do novo bloco por todos os elementos estarem marcados

Bloco 1: I N R T

Bloco 2: A C E L

Bloco 3: A

B	A	O	C

Bloco 1: I N R T

Bloco 2: A C E L

Bloco 3: A A

A	B	O	C

Bloco 1: I N R T

Bloco 2: A C E L

Bloco 3: A A B

L	C	O	A*

Bloco 1: I N R T

Bloco 2: A C E L

Bloco 3: A A B C

A	L	O	A*

Bloco 1: I N R T

Bloco 2: A C E L

Bloco 3: A A B C L

N	O	A*	A*

Bloco 1: I N R T

Bloco 2: A C E L

Bloco 3: A A B C L O

C	A*	N*	A*

*Criação do novo bloco por todos os elementos estarem marcados

Bloco 1: I N R T

Bloco 2: A C E L

Bloco 3: A A B C L O

Bloco 4: A

E	A	N	C

Bloco 1: I N R T

Bloco 2: A C E L

Bloco 3: A A B C L O

Bloco 4: A A

A	C	N	E

Bloco 1: I N R T

Bloco 2: A C E L

Bloco 3: A A B C L O

Bloco 4: A A C

D	E	N	A*

Bloco 1: I N R T

Bloco 2: A C E L

Bloco 3: A A B C L O

Bloco 4: A A C E

A	N	D*	A*

Bloco 1: I N R T

Bloco 2: A C E L

Bloco 3: A A B C L O

Bloco 4: A A C E N

-	A*	D*	A*

*Criação do novo bloco por todos os elementos estarem marcados

Bloco 1: I N R T

Bloco 2: A C E L

Bloco 3: A A B C L O

Bloco 4: A A C E N

Bloco 5: A

-	A	D	-

Bloco 1: I N R T

Bloco 2: A C E L

Bloco 3: A A B C L O

Bloco 4: A A C E N

Bloco 5: A A

-	D	-	-

Bloco 1: I N R T

Bloco 2: A C E L

Bloco 3: A A B C L O

Bloco 4: A A C E N

Bloco 5: A A D

Intercalação polifásica:

- Desenvolvida como solução para os problemas da intercalação balanceada de vários caminhos.

- Processo de funcionamento:

- Os blocos são distribuídos de forma desigual entre as fitas disponíveis.

- Uma fita é SEMPRE deixada livre

- Em seguida, a intercalação de blocos ordenados é executada até que uma das fitas se esvazie.

- A fita vazia torna-se a próxima fita de saída.

- Observações:

- A intercalação é feita em farias fases;

- As fases não envolvem todos os blocos;

- Nenhuma copia direta é feita entre fitas.

Implementação:

- A implementação desse método é simples;

- A distribuição inicial dos blocos nas fitas é a parte mais delicada do processo.

Considerações:

- A Análise da intercalação polifásica é complicada;

- O que se sabe é que ela é ligeiramente melhor que a intercalação balanceada para valores pequenos de F;

- Para valores de F>8, a intercalação balanceada pode ser mais eficiente.

Exemplificação:

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

*Blocos já ordenadas por meio da seleção por substituição.

Fita 1: I N R T | A C E L | A A B C L O

Fita 2: A A C E N | A A D

Fita 3:

➔ Intercalação entre os dois primeiros blocos da fita 1 e 2 na fita 3

Fita 1: A A B C L O

Fita 2:

Fita 3: A A C E I N N R T | A A A C D E L

➔ Intercalação entre o primeiro bloco da fita 1 e 3 na fita 2

Fita 1:

Fita 2: A A A A B C C E I L N N O R T

Fita 3: A A A C D E L

➔ Intercalação entre o primeiro bloco das fitas 2 e 3 na fita 1

Fita 1: A A A A A A A B C C C D E E I L L N N O R T

Quicksort externo:

- Proposto em 1980 por Monard, o algoritmo utiliza o paradigma de divisão e conquista. Ele ordena *in situ* um arquivo A = {R1, ..., Rn} de n registros, esses registros se encontram em memória secundária de acesso randômico.

- O algoritmo utiliza somente O(log n) unidades de memória interna, não necessitando de qualquer memória externa adicional.

Ordenação:

- Para ordenar o arquivo A, o algoritimo:

-Divide A em:

{R1, ..., Ri}<=Ri+1<=Ri+2<=...<=Rj-2<=Rj-1<={Rj, ..., Rn}

-E chama recursivamente os arquivos gerados:

A1 = {R1, ..., Ri} e A2 = {Rj, ..., Rn}

- Os registros {Ri+1, ..., Rj-1} ordenados são o pivô do algoritmo, encontrando-se na memória interna durante a execução do mesmo, os subarquivos gerados possuem os registros maiores que o último registro e menores que o primeiro.

- Para a partição do arquivo, é utilizada uma área de memória interna para armazenar o pivô, e essa área é >=3.

- Considerar que, deve ser ordenado o subarquivo de menor tamanho inicialmente, subarquivos vazios ou com registro único são ignorados e caso os arquivos de entrada possuam no máximo (j – i – 1) registros, ele é ordenado em etapa única.

Funcionamento:

- Os primeiros "tamanho da área" (TamArea) - 1 registros são lidos, alternativamente, dos extremos de A e armazenados na área de memória interna.

- Ao ler o TamArea-ésimo registro, cuja chave é C:

- C é comparada com Lsup e, sendo maior, j recebe Es e o registro é escrito em A2; 3

- Caso contrário, C é comparada com Linf e, sendo menor, i recebe Ei e o registro é escrito em A1;

- Caso contrário (Linf ≤ C ≤ Lsup), o registro é inserido na área de memória interna.

- Para garantir que os apontadores de escrita estejam atrás dos apontadores de leitura, a ordem alternada de leitura é interrompida se (Li = Ei) ou (Ls = Es). *Nenhum registro pode ser destruído durante a ordenação in situ.

- Quando a área de memória enche, deve-se remover um registro dela, considerando os tamanhos atuais de A1 e A2.

- Sendo Esq e Dir a 1ª e a última posição de A, os tamanhos de A1 e A2 são, respectivamente, (T1 = Ei – Esq) e (T2 = Dir – Es).

- Se (T1 < T2), o registro de menor chave é removido da memória, sendo escrito em Ei (A1), e Linf é atualizado com tal chave. Se (T2 ≤ T1), o registro de maior chave é removido da memória,

- Se (T2 ≤ T1), o registro de maior chave é removido da memória, sendo escrito em Es (A2), e Lsup é atualizado com tal chave.

- O objetivo é escrever o registro removido da memória no subarquivo de menor tamanho, no intuito de dividir A de forma uniforme e, assim, balancear a árvore gerada pelas recursões, isso minimiza a quantidade de operações de leitura e escrita efetuadas pelo algoritmo.

-O processo de partição continua até que Li e Ls se cruzem, ou seja, (Ls < Li).

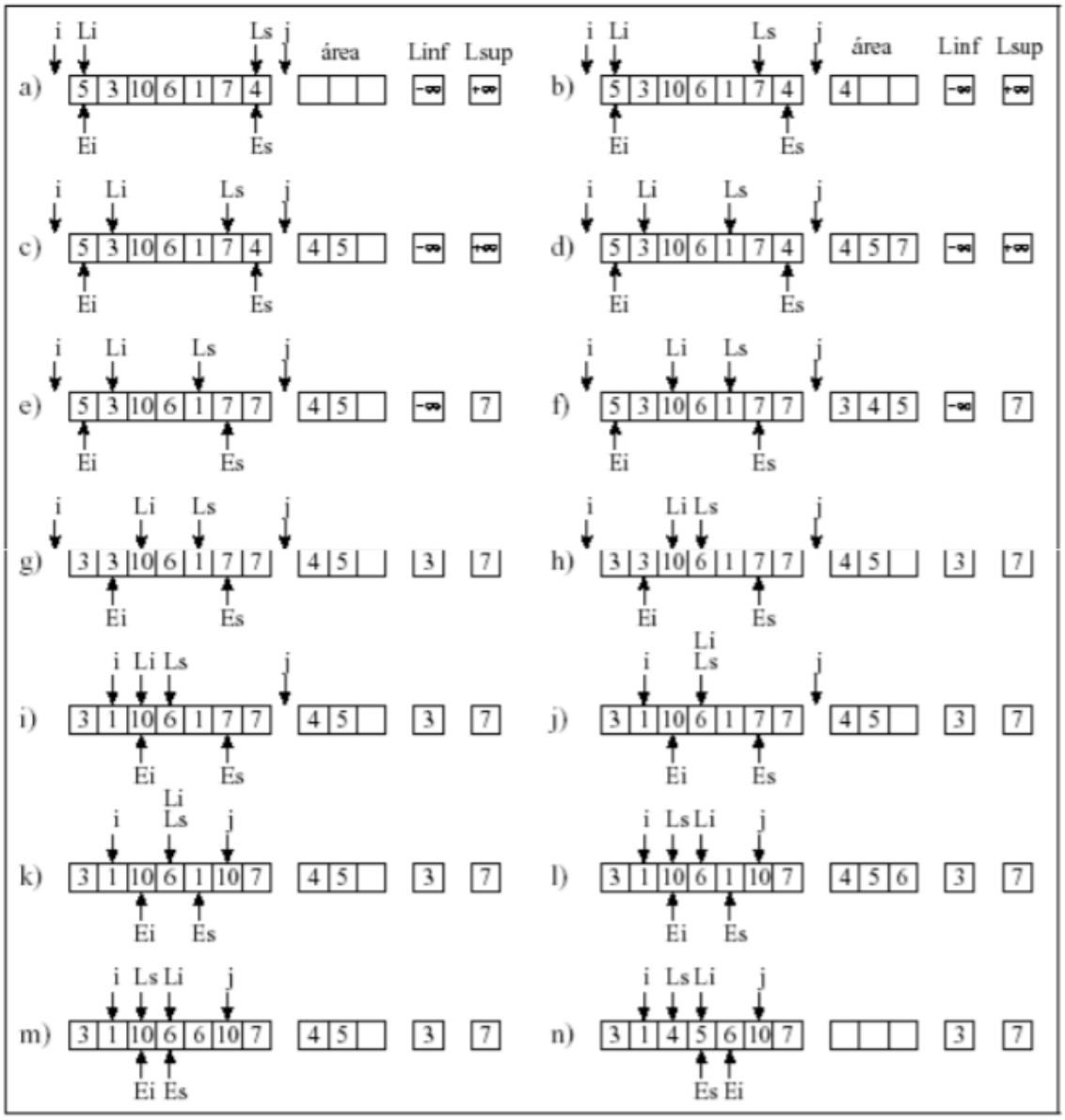
-Neste momento, os registros armazenados na área de memória interna devem ser copiados, já ordenados, em A.

-Enquanto existir registros na área de memória, o menor deles é removido e escrito na posição indicada por Ei em A.

Análise:

Melhor	Médio	Pior
O (n/b)	O (n/b x Log (n/Tam_Área))	O (n^2 / Tam_Área)
Arquivo já ordenado	Maior probabilidade de acontecer	Quando as partições possuem tamanhos inadequados: maior possível e vazio *Quanto Maior o n, menor a chance disso acontecer

Exemplificação: *Um arquivo {Ri+1, ..., Rj-1}



***imagens utilizadas dos slides da aula.**