**Lívia Stéffanny de Sousa**

**Resumo Sobre Ordenação Externa**

**Quando é utilizada a ordenação externa?**

Quando todos os dados a serem ordenados não cabem em memoria principal.

**Quais os fatores que determinam as diferenças das técnicas de ordenação externa em relação à ordenação interna?**

O custo de acesso que a memória secundária tem em relação a memória primária por conta principalmente das transferências de dados entre a memória interna e externa.

**Porque para tratar com memoria secundaria é necessário se preocupar com as restrições vinculadas aos mecanismos de dispositivos de memória secundaria que a gente vá utilizar?**

Porque quando trabalhamos com dispositivos de memoria secundaria que pode ser disco ou fitas temos que a fita só pode ser acessada sequencialmente e os discos podem ser acessados de forma direta, porém o acesso direto é muito caro.

**O que quer dizer que os métodos de ordenação externa são dependentes do estado atual da tecnologia?**

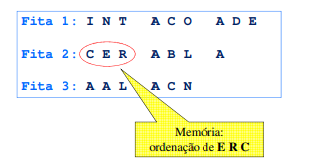
Que quanto mais rápido nos tivermos esse processo de transferência entre a memoria principal e a secundaria, melhor (mais rápido) vai ser o método.

**Ordenação por intercalação** é o método mais importante de ordenação externa. Intercalar significa combinar dois ou mais blocos ordenados em um único bloco ordenado, é utilizada como uma operação auxiliar na ordenação. O foco dos algoritmos para ordenação externa é reduzir o número de passadas sobre o arquivo.

**Estratégia geral dos métodos de ordenação externa:**

Se tenho um arquivo muito grande, eu levo blocos desse arquivo para memoria principal, pego um bloco **ordeno**, volto com ele até ter vários blocos ordenados (a cada passada são criados blocos ordenados cada vez maiores, até que todo o arquivo esteja ordenado) e depois intercalo/combino esses blocos ordenados. Mas isso é muito custoso por conta das transferências.

Na **Intercalação balanceada de vários caminhos ou intercalação balanceada de dois f caminhos**, onde esses dois f caminhos representam f caminho de entrada + f caminho de saída é preciso considerar a capacidade da memória interna para a quebra do arquivo em blocos e quanto mais unidades disponíveis de fita magnética melhor. Depois se seguir o que foi dito **na estratégia geral dos métodos de ordenação externa, o** processo de intercalação termina quando de todas as fitas, apenas uma tiver itens.



Passo a passo detalhado descrito nos slides:

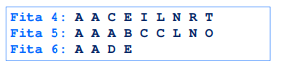
1. Leitura do primeiro registro de cada fita.

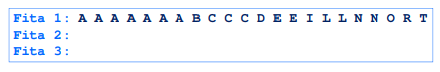
2. Retirada do registro contendo a menor chave, armazenando-o em uma fita de saída.

3. Leitura de um novo registro da fita de onde o registro retirado 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. Repetição do processo para os blocos restantes





**Quantas passadas são necessárias para ordenar um arquivo de tamanho arbitrário?**

O número de passadas vai ser log de n - sendo n a quantidade registros - sobre m - sendo m a capacidade da memória interna para a quebra do arquivo em blocos - na base f - sendo f a quantidade de fitas utilizadas em cada passada-.

**Logo, P(n) = log f (n/m)**

Esse método de intercalação de vários caminhos pode ser bem **demorado** dependendo da **quantidade** de arquivos a serem ordenados.

Na **intercalação balanceada de f +1 caminhos,** onde tem-se f fitas de entradas e 1 fitas de saída, ocorre da mesma forma que na intercalação balanceada de vários caminhos, mas usando somente uma fita de saída. O grande problema é que depois de gerar os blocos na fita de saída, tem que pegar os blocos e distribuir nas fitas de entrada, ou seja, é preciso ler todos os elementos e escrever dentro das fitas de entrada, que envolve mais uma passada dentro do arquivo.

Uma forma de ganhar eficiência seria trazer os blocos para memoria principal ao invés de ir lendo de dentro do arquivo, que diminuiria essas leituras feitas diretamente das fitas de entrada e saída.

A **Implementação por meio de substituição por seleção** ajuda a tornar o processo de intercalação balanceada muito mais fácil e rápida**.** Essa técnica usa a estrutura do heap.

A ideia é usar o heap durante a intercalação. Quando formos intercalar os blocos, pegamos os elementos e adicionamos na estrutura para encontrar o menor elemento, e reconstituímos o heap. A grande complexidade em fazer o heap é construir ele pela primeira vez, mas depois de constituído, a reconstituição é logarítmica então acharemos o menor elemento para assim colocá-lo nas fitas de uma a maneira mais rápida.

Nessa segunda fase do método que é a intercalação propriamente dita, usar o heap diminui a quantidade de complexidade de tempo para achar o menor elemento. O heap é utilizado para ajudar na busca do menor item no caso da intercalação, mas além disso pode ser utilizado na primeira fase do método de intercalação balanceada, que como já dito é formada por duas fases. A primeira é a ordenação do bloco inicial, e a segunda é a intercalação. Um porém do uso do heap é que com ele perdemos a referência da fita no momento de reconstituir o heap. Para solucionar esse problema faz-se necessário guardar também a fita de onde o item veio.

Além disso, o heap também pode ajudar na primeira fase de geração dos blocos inicialmente ordenados, justamente essa ajuda que deu o nome do método de como gerar blocos ordenados de uma maneira mais eficiente.

Nessa primeira fase onde ocorre a geração dos blocos inicialmente ordenados é necessário pegar o conjunto de dados do arquivo, gerar os blocos iniciais que caibam em memoria principal e nesses blocos são aplicados algum método de ordenação interna. Para gerar os blocos ordenados é necessário uma única passada dentro do arquivo. Já na segunda fase que é a intercalação, pegamos os blocos inicialmente ordenados e faz-se varias passadas dentro do arquivo ate que tenha o arquivo ordenado.

Usando a **substituição por seleção**, e não a ordenação interna, a primeira fase usando heap se inicia semelhante a outra, mas se o próximo elemento for menor que o elemento que foi pra fita ele é marcado como maior que os que estão no vetor. Quando todos os elementos do vetor são marcados, eles são desmarcados e um novo bloco é colocado na próxima fita de entrada, utilizando essa técnica que se chama substituição por seleção, são gerados blocos com tamanhos diferentes que é uma coisa muito boa para a intercalação. Se meu arquivo tiver ordenado na substituição por seleção o próximo elemento a ser colocado no heap sempre vai ser maior, então nenhum elemento vai ser marcado e vai ser gerado um único bloco já ordenado, a substituição de seleção gera blocos grandes já ordenados e quanto menor numero de blocos e quanto mais itens eles tiverem a ordenação vai ocorrer de forma mais rápida.

Se estiver ordenado decrescentemente, vai ser o pior caso, porque todos os próximos elementos jogados na estrutura vão ser marcados, então todos os blocos ordenados vão ter o tamanho do vetor de heap e também o mesmo tamanho que o método de ordenação externa retorna.

Como dessa forma acessa muitas vezes o arquivo, então daria pra criar um vetor temporário e ler uma quantidade maior de elementos e armazenar nesse vetor.

Para valores pequenos de f, não é vantajoso utilizar seleção por substituição para intercalar blocos, 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 se obter o menor item.

Executar os métodos de ordenação com muitos registros é demorado porque tem que ficar fazendo acesso aos arquivos tanto para leitura quanto para gravação colocando elementos ordenados ali dentro. Os algoritmos de ordenação externa envolvendo intercalação exigem muitas operações de entrada e saída envolvendo arquivo, fazendo com que demore mais.

Escolha da ordem de intercalação f:

Fitas magnéticas:

f deve ser igual ao número de unidades de fita disponíveis menos um, já que a fase de intercalação pode usar f fitas de entrada e uma fita de saída.

O número de fitas de entrada deve ser no mínimo dois.

Discos magnéticos:

Mesmo raciocínio apresentado, porém o acesso direto é menos eficiente que o sequencial.

Porém, a melhor escolha para f depende de parâmetros relacionados com o sistema de computação disponível.

**Intercalação polifásica**

Um grande problema é o custo de uma copia. Se eu tenho todo o meu arquivo nas f fitas de entrada, quando eu realizo a minha primeira fase de intercalação balanceada eu vou gerar blocos maiores nas fitas de saída, então uma vez terminada as fases de intercalação, vamos ter a cópia adicionada nas fitas de saída. Que pode ser considerado um problema em termos de armazenamento. Ai surgiu a intercalação polifásica.

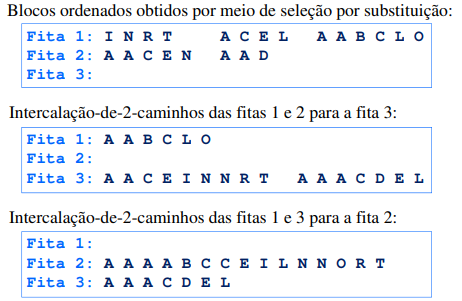
Processo de funcionamento da intercalação polifásica:

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

Uma fita é deixada livre.

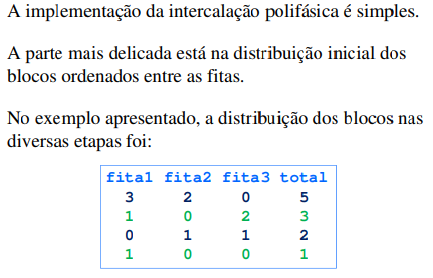
Em seguida, a intercalação de blocos ordenados é executada até que uma das fitas de entrada se esvazie.

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



A primeira fase já foi feita então esta incialmente ordenada e já se encontram nas fitas. O processo termina quando tem somente um bloco e uma fita.

Se os blocos tiverem tamanhos iguais, vai ter um bloco grande que impede de intercalar os blocos, fazendo com que tenhamos que verificar esse caso onde uma fita contem mais de um bloco.



Em cada fase não envolve uma passada completa, então não tem como fazer a analise.

Desculpe por essa parte meio ruim, mas meu notebook resolveu reiniciar e eu não tinha salvo, e acabei fazendo essa parte somente com algumas anotações :/

**Quicksort externo**

Utiliza o paradigma de divisão e conquista, temos um pivô inicial em cima de um conjuntos de elementos presentes em um arquivo e é feita a escolha de um pivô e a partir dessa escolha eu faço uma organização de todos os outros elementos do arquivo, pego todos menores que o pivô de um lado e os maiores do outro lado, a partir dai eu gero dois subarquivos , onde eu realizo da mesma forma. Ordena in situ, no próprio arquivo. O pivô não é formado por um elemento só, mas por um conjunto.

Primeiro é necessário particionar, elementos menores de um lado e maiores do outro.

Depois, é necessário fazer o mesmo com os subaquivos, ate o subqarquivo acabar na memória interna, ai eu aplico a ordenação com o quicksort interno.

Para a partição do arquivo, é utilizada uma área de memória interna para o armazenamento do pivô.

* Tamanho da área = j − i − 1, sendo necessariamente ≥ 3.

Nas chamadas recursivas, deve-se considerar que:

* deve ser ordenado, inicialmente, o subarquivo de menor tamanho;
* subarquivos vazios ou com um único registro são ignorados;
* caso o arquivo de entrada A possua no máximo (j − i − 1) registros, ele é ordenado em um único passo.

Considerações sobre o funcionamento de uma partição:

* Os valores das chaves de Ri e Rj são denominados limite inferior (Linf) e limite superior (Lsup) da partição.
* A leitura de A é controlada por apontadores de leitura inferior (Li) e superior (Ls). 29 inferior (Li) e superior (Ls);
* A escrita em A é controlada por apontadores de escrita inferior (Ei) e superior (Es).

Se eu tenho li e ls, vou lendo elementos do meu arquivo e jogando esses elementos pra dentro da memória interna , levando pra dentro do pivô. Quando faltar apenas 1 elemento dentro pivô que não foi preenchido, depois eu leio mais um elemento do arquivo chave c. Quando o pivô ficar completo é preciso remover um registro pra continuar o processo. Ou pego o menor elemento do pivô ou o maior. Mas é preciso verificar os elementos de a1 e a2 e colocar o pivô no menor arquivo. O valor da leitura inferior não pode ser menor que a escrita inferior

O quicksort externo segue a mesma analise que o interno, n logarítmico.

Seja N a qtd de registros e b a qtd que eu vou estar lendo, o melhor caso é dado por O (n / b) e o caso médio por O (n / b × log (n / TamArea) ).

Quanto maior o pivô melhor o valor da complexidade, tamanho do pivô influencia diretamente, o pior não acontece porque as partições são uniformes.