Resumo II - Ordenação Externa

Bernardo Cavanellas Biondini - 20.1.4112

Nicolle Canudo Nunes - 20.1.4022

Ordenação Externa

Os métodos de ordenação externa são úteis quando precisa-se ordenar uma grande quantidade de dados, que não cabem em memória principal. Os métodos de ordenação usados em memória principal, como buble sort, insert sort, quick sort, merge sort, entre outros, possuem como base de parâmetro para a complexidade do algoritmo o número de comparações e de movimentações. Entretanto, quando os dados se encontram em memória secundária, esses parâmetros não são suficientes para determinar o custo de um algoritmo, uma vez que o número de transferências entre memória principal e secundária tem uma considerável importância sobre ele e precisa ser considerado.

As principais diferenças entre esses métodos estão relacionadas a diferença de complexidade, e a restrição de acesso aos dados (em qual dispositivo os dados se encontram) e, por isso, se diferencia também pela dependência do estado da tecnologia. Dessa forma, o acesso pode ser mais rápido ou mais devagar, dependo do dispositivo que os dados estão localizados.

Métodos de Intercalação

Intercalar é combinar blocos ordenados em um único bloco ordenado, servindo como uma operação auxiliar na ordenação, assim ordena-se vários blocos que caibam em memória principal, fazendo as passadas sobre o arquivo. Dessa forma, os métodos passam a intercalar, ao invés de só movimentarem e compararem os dados.

- O foco dos métodos está na capacidade de realizar menores transferências entre memórias, diminuindo significativamente a complexidade.

Estratégia Geral:

- Quebrar o arquivo em blocos (passada sobre o arquivo).
- Ordenar cada bloco em memória interna (utilizar os métodos mais eficientes para ordenar os dados quando estes já foram divididos em blocos).
- Intercalar os blocos ordenados (são feitas várias passadas sobre o arquivo, criando-se blocos ordenados cada vez maiores).

Intercalação Externa de Vários Caminhos

- Utiliza dispositivos de memória secundária de forma temporária (fitas e discos por exemplo).
- Utiliza de fitas auxiliares para o processo de intercalação, metade delas são chamadas de fitas de entrada e a outra metade fitas de saída.

Fase 1, de criação dos blocos ordenados:

- Quebra o arquivo em blocos de tamanho da memória interna.
- Ordenação de cada bloco de memória interna.

Primeiramente, é lido certa quantidade de itens que são armazenados em um vetor. Nele é aplicado algum dos métodos de ordenação (ordenação interna) e depois colocado nas fitas auxiliares de entrada. Os blocos ordenados vão sendo colocados nessas fitas e o processo se repete para os próximos blocos. Nesse primeiro processo só é realizada uma passada no arquivo.

Fase 2, de intercalação:

- Leitura do primeiro registro de cada fita.
- Encontra-se o menor elemento (complexidade linear) entre os primeiros registros de cada fita e o retira, armazenando em uma das fitas de saída .
- Após a retirada, lê⁻se um novo registro da fita a qual o antigo registro estava.

Ou seja, o menor registro é substituído pelo seu sucessor na fita auxiliar de origem.

Ao ler o último registro do bloco, a fita fica inativa e só é reativada quando os últimos registros das outras fitas forem lidos.

- O processo se repete para os blocos restantes.
- Quando o processo é feito para todos os blocos, as fitas de saída resultantes (ordenadas) sofrem uma outra intercalação.

Nesse momento, as fitas de saída funcionariam como fitas de entrada e vice-versa.

- Caso a ordenação ainda precise ser feita, o processo de intercalação é feito novamente, gerando blocos cada vez maiores até que os dados estejam realmente ordenados.

Exemplo

- Considerando os itens de chave: I N T E R C A L A C A O B A L A N C E A D A
- Considerando 3 fitas de entrada e 3 fitas de saída
- Considerando memória interna para 3 itens

A primeira fase colocará os itens nas fitas em blocos ordenados, utilizando algum método de ordenação. Os 3 primeiros itens já estam em ordem INT, já os próximos ERC precisam ser ordenados, ficando CER e sendo colocados dessa forma na fita. Esse processo se repete para os demais blocos de 3 registros.

Quando esse processo é feito para o resto dos itens, começa a fase de intercalação.

Os primeiros elementos de cada fita sofrem o processo de intercalação. No exemplo, a primeira vez do processo pegamos os itens ICA (primeiros registros das fitas) e encontra-se o menor deles (A). Após isso, o elemento é

substituído pelo próximo de sua fita, ou seja, A. Depois disso, verifica-se qual o menor item e ele é colocado na fita de saída. O processo se repete até todos os blocos sofrerem intercalação.

Após esse processo as fitas de saída fazem o papel das fitas de entrada e o processo de intercalação acontece novamente.

Complexidade

- Seja n o número de registros
- Seja m o número de posições disponíveis na memória interna

A primeira fase produz n/m blocos ordenados.

- Seja P(n) o número de passadas na fase de intercalação
- Seja f o número de fitas utilizadas em cada passada.

> Distribuir os blocos em fitas, dividindo o problema. Quanto mais fitas, maior a distribuição e menos blocos por fitas

Logo, P(n) = log f (n/m) [log de (n/m) na base f]

O método também é chamado de Intercalação Balanceada de 2f Caminhos, pois são f caminhos de entrada e f caminhos de saída.

- É possível utilizar f caminhos de entrada e apensas 1 de saída, ou seja, f+1 fitas

  Nesse caso a primeira fase é a mesma. Já na fase de intercalação, os blocos são colocado de forma ordenada na fita de saída e na etapa seguinte os blocos da fita de saída serão distribuídos de forma homogênea entre as fitas de entrada.

  O problema de se utilizar somente 1 fita de saída seria de ter uma a passada mais no arquivo, para que os registros passem para as fitas de entrada novamente, passando pela divisão entre elas. Entretanto, essa implementação envolveria menos intercalações, o que poderia ser mais vantajoso, dependendo do número de registros.

Uma forma de diminuir o número de transferências entre memória principal e memória externa seria ler blocos inteiros, armazenando-os em uma estrutura, dessa forma, é possível ter acesso a eles temporariamente sem

precisar realizar novas leituras no arquivo. Assim, os blocos que estivessem sendo usados ficariam em memória principal.

Substituição por seleção

A implementação do método de intercalação balanceada pode ser feita utilizando filas de prioridade. Idealmente, a estrutura para implementação é o heap.

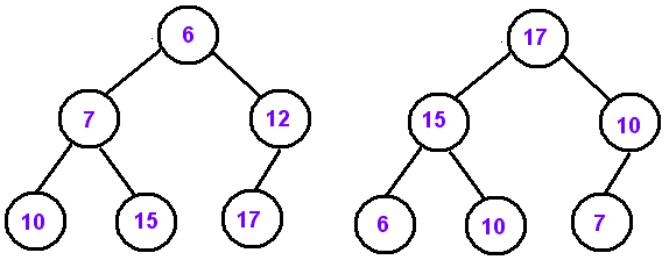
- As fases do método podem ser mais eficientes.
- Substitui o menor item pelo próximo da fita de entrada.

A substituição consiste na retirada do menor item na fila, a inserção de um novo item no lugar e reconstruir a propriedade da estrutura.

Heap

Um heap é uma estrutura capaz de manter certa ordenação de elementos de um vetor. Um heap pode ser construido de forma crescente (heap minimo), quando a raiz é o menor elemento, e de forma decrescente (heap máximo), tendo o maior item como raiz.

Exemplo de heap mínimo e heap máximo com os elementos 6, 7, 10, 12, 15 e 17:



Heap Mínimo ***Heap Máximo***

fonte: https://www.andrew 121/lectures/Binary Heaps/heaps.html

Fase de intercalação, o heap, diminui consideravelmente o número de comparações. Enquanto que os métodos utilizados para encontrar o menor item tem custo linear, o heap consegue otimizar esse custo com uma complexidade logaritma no pior caso.

O funcionamento consiste em construir um heap com os itens do vetor, realizar a validação, para termos acesso ao menor item pela raiz e realizar o mesmo processo com o próximo item, reconstruindo a estrutura e tendo acesso ao menor registro.

Um problema gerado com a implementação do heap nessa fase seria a perda da referência da fita ao qual o item pertence, uma vez que essa referência se dava pela posição do vetor. Dessa forma, é preciso que o vetor não armazene mais apenas o item, mas uma referência para a fita também.

Fase de criação dos blocos, os registros serão lidos sequencialmente e colocados em um vetor. Logo depois, construi-se um heap com eles, obtendo o menor registro do bloco pela raiz que é colocado em um bloco. Depois disso, pega-se o próximo registro e reconstitui-se o heap com os antigos registros do vetor. Caso o registro trocado seja menor que o último elemento inserido no bloco, ele recebe uma marca e a partir desse momento, consideramos que ele seja maior, nesse momento o heap precisa ser refeito para obtermos o menor elemento

(considerando que o menor seja maior). Caso todos os elementos do vetor estejam marcados, passamos a coloca-los em outro bloco.

Quando todos os registros são lidos e colocados em blocos, o processo para e os elemento são colocados nas fitas de entrada. Dessa forma, não é preciso utilizar os métodos de ordenação interna na quebra do registro em blocos.

Na substituição por seleção, caso os itens já estejam ordenados, não é preciso passar pelo processo de intercalação, uma vez que só será gerado 1 bloco. Isso ocorre pois os itens vão ser sempre colocados em ordem, o menor será o primeiro colocado no bloco e os próximos serão seus "sucessores". Considerando que os itens estejam em ordem decrescente, os blocos serão gerados de forma idêntica ao método de utilização de ordenação interna, pois todos elementos inseridos vão ser marcados, maiores, gerando blocos de tamanho igual quando se comparado aos gerados pelos métodos de ordenação interna.

O problema desse método está na quantidade de leituras no arquivo, que serão feitas sequencialmente. Dessa forma, é possível criar uma estrutura para ler uma maior quantidade de registros, para que eles sejam armazenada de forma temporária, realizando menos leituras no arquivo.

Estratégias fase 1, de criação dos blocos ordenados:

- método de ordenação interna
- substituição por seleção

Estratégias fase 2, de intercalação:

- balanceada de vários caminhos (2f fitas)
- balanceada de vários caminhos (f+1 fitas)
- polifásica

A estratégia utilizada na primeira fase pode ser usada com qualquer uma das estratégias para segunda fase e vice-versa. A utilização de uma estratégia não influencia na escolha de outra. O ideal é analisar qual delas seria mais eficiente para a quantidade de registros que se deseja ordenar.

Em geral, para valores de f pequenos, não é vantajoso utilizar seleção por substituição na fase de intercalação, já que o menor item pode ser obtido por f-1 comparações. Caso f seja maior que 8, o método é adequado, realizando log 2 (f) [log de f na base 2] comparações para se obter o menor item.

Considerações:

- Os métodos de intercalação envolvem muita entrada e saída de dados e por esse motivo acabam sendo mais demorados, sendo mais vantajosos quando envolvem mais de um processador.
- F deve ser grande o suficiente para completar a ordenação em poucos passos (Sedegwick- 1998).

Intercalação Polifásica

No contexto de ordenação externa, a intercalação polifásica veio para solucionar problemas da intercalação balanceada, tais como:

- Necessitar de um grande número de fitas, o que resulta em uma grande quantidade de operações de leitura e escrita
- Existir um custo de uma cópia adicional do arquivo

Nesse caso, a intercalação polifásica é a solução. Seu funcionamento é dado da seguinte forma:

1. Os blocos ordenados são distribuídos de forma desigual entre as fitas disponíveis
2. Uma fita fica livre
3. A intercalação dos blocos ordenados é feita 20 vezes, até que uma das fitas de entrada fique vazia, então, essa fita se torna a próxima fita de saída

- Blocos ordenados obtidos por meio de 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-de-2-caminhos das fitas 1 e 2 para a 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-de-2-caminhos das fitas 1 e 3 para a 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-de-2-caminhos das fitas 2 e 3 para a 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
Fita 2:
Fita 3:
```

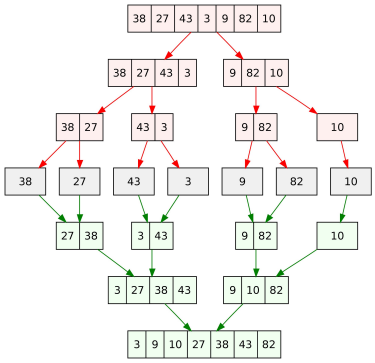
É importante destacar que o estado final de ordenação se dá somente quando existe apenas um bloco em apenas uma fita (outras fitas estarão todas vazias). Além disso, sabe-se que as fases não necessariamente envolvem todos os blocos e que não é realizada uma cópia direta entre as fitas.

Em caso de os blocos ordenados serem distribuídos de forma igual entre as fitas disponíveis, isso irá gerar uma única fita com esses blocos intercalados. Nesse caso será necessário copiar um dos blocos para outra fita, para que o processo continue normalmente.

Fazer a análise da intercalação polifásica não é uma tarefa simples. O que se sabe é que ela se faz ligeiramente melhor que a intercalação balanceada para pequenos valores de F.

No caso de F>8, a intercalação balanceada de vários caminhos pode vir a ser mais rápida.

QuickSort



O quicksort adota a estratégia de divisão e conquista. A estratégia consiste em ordenar as duas sublistas de chaves menores e maiores recursivamente até que a lista completa se encontre ordenada. Os passos do quicksort padrão são:

1. Escolher um elemento da lista, denominado pivô
2. Rearranjar a lista de forma que todos os elementos anteriores ao pivô sejam menores que ele, e todos os elementos após o pivô sejam maiores que ele. Após isso, o pivô estará em sua posição final e haverá duas sub listas não ordenadas.
3. Ordenar recursivamente a sub lista dos elementos menores e a dos elementos maiores; O caso base da recursão são as listas de tamanho zero ou um.

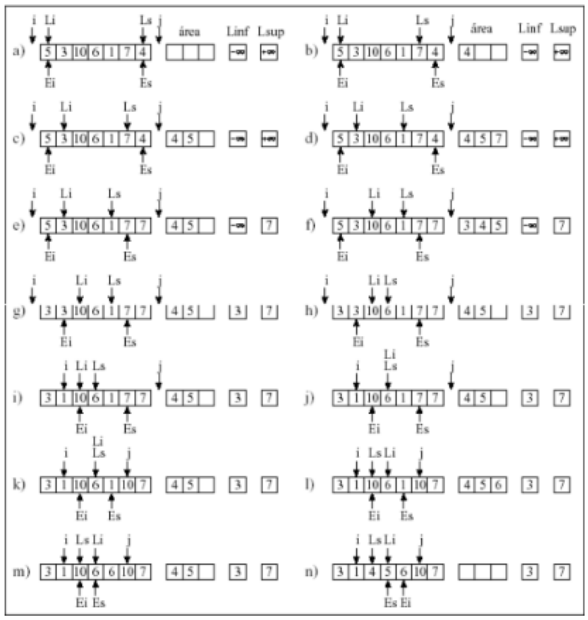
Quicksort Externo

Os métodos mostrados trabalham com intercalação, enquanto o quicksort trabalha com a estratégia de divisão e conquista.

O algoritmo faz a ordenação dentro do próprio arquivo. A diferença entre o quicksort externo e interno é que, o pivô não mais será apenas um elemento, e sim, um conjunto de elementos.

Considere um arquivo A = {R1, ..., Rn}. Ele será ordenado da seguinte forma:

1. Particionar: {R1, ... , Ri} ≤ Ri+1 ≤ Ri+2 ≤ ... ≤ Rj−2 ≤ Rj−1 ≤ {Rj, ... , Rn};
    a. Os registros ordenados {Ri+1, ..., Rj-1} correspondem ao pivô do algoritmo e serão transferidos para a memória interna.
    b. Para a partição do arquivo, é utilizada uma área de memória interna para o armazenamento do pivô.
    c. Tamanho da área = j − i − 1, sendo necessariamente ≥ 3
2. Chama-se recursivamente o algoritmo em cada um dos subarquivos gerados
    a. A1 = {R1 , . . . ,Ri} e A 2= {R1 , . . . ,Rn}
    b. A1 contém registros menores que Ri+1
    c. A2 contém registros maiores que Rj-1
    d. deve ser ordenado, inicialmente, o subarquivo de menor tamanho;
    e. subarquivos vazios ou com um único registro são ignorados;
    f. caso o arquivo de entrada A possua no máximo (j − i − 1) registros, ele é ordenado em um único passo.



Esq: limite da posição inferior

Dir: limite da posição superior

Li: ponteiro de leitura inferior

Ls: ponteiro de leitura superior

Ei: ponteiro de escrita inferior

Es: ponteiro de escrita superior

Linf: limite inferior do pivô

Lsup: limite superior do pivô

i e j: valor retornado pela partição

Como funciona a partição

Os primeiros tamArea - 1 são lidos, alternando entre superior e inferior dos extremos de A. Serão armazenados na área de memória interna: os pivôs. Quando houver apenas uma posição livre para eles na memória interna, o processo finaliza.

Ao ler o tamArea-ésimo registro, ao depender do valor da chave C, temos 3 situações:

1. se C>Lsup, j=Es e o registro é escrito em A2
2. se não, se C<Linf, i = Ei e o registro é escrito em A1
3. se não, se Linf ≤ C ≤ LSup, o registro é inserido na memória interna de forma ordenada

Quando essa área lota, deve-se remover um registro dela, considerando os tamanhos de A1 e A2. Considerando o balanceamento dos dois subarquivos, podem ocorrer duas situações:

1. Retira-se o menor, adiciona em A1 e atualiza o Linf para esse valor

2. Armazena-se o maior, adiciona em A2 e atualiza-se o Lsup para esse valor

O objetivo é pegar o registro que foi removido da memória interna e escrevê-lo no subarquivo de menor tamanho:

1. Tendo ESQ e DIR a primeira e última posição de A, respectivamente, os tamanhos serão: A1 = Ei-ESQ; a2=DIR-Es
2. Se TamA1 < TamA2, o registro de menor chave é removido da memória e escrito em Ei(A1) e o Linf é atualizado com essa chave
3. Se não, se T2≤T1, o registro de maior chave é removido da memória, sendo escrito em Es(A2) e Lsup é atualizado com essa chave

Esse processo continua até que Li e Ls se cruzem.

Análise

Seja n o número de registros a serem ordenados e seja b o tamanho do bloco de leitura ou gravação do SO.
Melhor caso: O (n / b)

Ocorre quando o arquivo de entrada já está ordenado.

Pior caso: O (n2 / TamArea)

Ocorre quando as partições geradas possuem tamanhos inadequados: maior tamanho possível e vazio.

A medida que n cresce, a probabilidade de ocorrência do pior caso tende a zero.

Caso Médio: O (n / b × log (n / TamArea))

Maior probabilidade de ocorrer.