PESQUISA, ORDENAÇÃO E TÉCNICAS DE ARMAZENAMENTO

UNIDADE 3 - ORDENAÇÃO EXTERNA E PROCESSOS DE INTERCALAÇÃO DOS ARQUIVOS

Keila Barbosa Costa

Introdução

A classificação é uma das operações fundamentais para o processamento dos elementos em qualquer banco de dados. Com o crescente aumento da quantidade de informações, o problema de classificar ou ordená-las dentro de um prazo estipulado se tornou um dos principais problemas para quase todos os aplicativos.

Vamos começar esta unidade compreendendo o que é ordenação externa e intercalação de arquivos. Para isso, é interessante refletirmos sobre alguns pontos: Será que é possível colocar dois ou mais baralhos de tarô em ordem com todas aquelas quantidades de cartas usando apenas as duas mãos? Respeitando a ordem é provável que não se consiga; colocar os naipes, depois os números e assim por diante, normalmente acabamos fazendo motins separando por naipes ou números, certo? Porquê? Talvez seja porque essa ordenação ultrapassa nossa capacidade manual ou até cognitiva de fazer essa ordenação de uma só vez. Qual método podemos usar para fazer esse tipo de classificação? O que abordaremos neste capítulo é praticamente isso!

Mas, quando a intercalação é útil ou não, nesse processo de classificação? A ordenação externa é usada quando o processo de ordenação em memória principal não puder ser realizado e estoura a capacidade do nosso computador, ou seja, quando tivermos um volume de dados tão grande que não é possível transportá-lo para memória principal. Para introduzir a área de algoritmos de classificação, os mais apropriados são os métodos "elementares". Eles fornecem uma maneira fácil de aprender terminologia e mecanismos básicos para ordenar algoritmos, fornecendo um *background* mais sofisticado e adequado para classificações. Vamos conhecer os principais tópicos da ordenação externa? Acompanhe esta unidade com atenção!

3.1 Ordenação externa

A classificação diz respeito à operação ou técnica de organizar e reorganizar um grupo de dados em alguma ordem específica, ou seja, é a execução da operação que organiza os registros de um conjunto de dados em alguma ordem, de acordo com algum critério de ordem específico. As técnicas de classificação podem ser divididas em duas categorias: classificação interna ou classificação externa. Se todos os dados a serem classificados/ordenados puderem ser ajustados por vez na memória principal, o método de classificação interna estará sendo executado. (VIANA, 2015).

Nesta unidade, em específico, veremos detalhes do processo de ordenação externa ou também conhecida como classificação externa. A Ordenação externa é um termo para uma classe de algoritmos de classificação que podem lidar com grandes quantidades de dados; ela é necessária quando os dados que estão sendo ordenados não se encaixam na memória principal de um dispositivo de computação (geralmente RAM) e, em vez disso, devem residir na memória externa mais lenta (geralmente em um disco rígido), ou seja, a ordenação externa é o processo de ordenação com algum tipo de dispositivo de memória externa como (HD, Fitas, Pen Drives etc.) que tenha uma capacidade de memória grande, com muitos gigabytes, maior que a memória principal. (AGUILAR, 2008).

Usamos a ordenação externa quando não conseguimos transportar todos os nossos registos para a memória principal do computador e roda um algoritmo básico como o *Shellsort* ou *Mergesort* por exemplo. Este processo tem como objetivo ordenar os registros, ou seja, colocar todo mundo em uma ordem pré-estabelecida assim como a palavra já sugere.

VOCÊ QUER VER?

O método *ShellSort* criado por Donald Shell em 1959, é um algoritmo de classificação complexo e rápido que divide várias vezes de forma repetida toda a coleção em subcoleções, levando cada elemento *h-ésimo* para um intervalo *h* fixo e executando uma inserção de classificação a cada subcoleção, ele é uma extensão do algoritmo de ordenação por inserção. O *ShellSort* permite a troca de registros distantes, diferenciando do algoritmo de ordenação por inserção que possui a troca de itens adjacentes para determinar o ponto de inserção. Para compreender melhor o funcionamento do algoritmo assista a este vídeo: <https://www.youtube.com/watch?v=CmPA7zE8mx0>.

O problema de como ordenar os dados com eficiência tem sido amplamente discutido. Na maioria das vezes, a classificação é realizada por ordenação externa, na qual o arquivo de dados é muito grande para caber na memória principal e deve residir na memória secundária. As E/S (Entrada/Saída) de disco são as medidas mais apropriadas no desempenho da ordenação externa e outros problemas externos, porque a velocidade de E/S é muito mais lenta que a velocidade da CPU. Os algoritmos de ordenação externa geralmente se enquadram em dois tipos: ordenação de distribuição, que se assemelha à ordenação rápida, e ordenação externa de mesclagem. A ordenação por mesclagem, usada na computação, também conhecida como mesclagem comum, trata de um algoritmo bem usual de ordenação eficiente, que se baseia em comparação. (AGUILAR, 2008). A maioria das implementações produz uma ordenação estável, o que significa que a ordem dos elementos iguais é a mesma na entrada e na saída.

VOCÊ O CONHECE?

John von Neumann, foi um matemático que fez grandes contribuições na definição teorias como a ergódica e a dos jogos, mecânica, geometria contínua, economia, ciência da computação, análise numérica, hidrodinâmica e estatística. Além disso, em 1945, criou a classificação por mesclagem: um algoritmo de divisão e conquista. (KADAM, 2000).

A ordenação externa de distribuição é bem semelhante à ordenação de mesclagem, que veremos nos próximos tópicos; esta normalmente usa uma estratégia híbrida de ordenação e mesclagem. Na fase de ordenação, são lidos pedaços de dados pequenos o suficiente para caber na memória principal, ordenados e gravados em um arquivo temporário. Na fase de mesclagem, os subarquivos ordenados são combinados em um único arquivo maior.

A ordenação externa mais comumente usada ainda é a de mesclagem, conforme descrito por Knuth (1973) e Singh (1985). A seguir veremos algumas formas eficientes de processar uma intercalação também conhecida como mesclagem.

3.2 Definição de mecanismos

Neste tópico estudaremos as formas eficientes e os mecanismos de se processar uma intercalação. As técnicas abordadas objetivam reduzir a gravação dos registros de cada parte guardada em disco magnético, também conhecida como área de trabalho e o número de leitura desses registros. De acordo com Viana (2015), os fatores que têm relevância na eficiência de um algoritmo de classificação em geral são os que você poderá ver, clicando nos ícones a seguir.

O tamanho e o número de registros de cada um deles.
Usar ou não a memória auxiliar (necessita ou não da ordenação externa).
Meios e tipos de armazenamento usado.
Periodicidade de atualização e nível de ordenação já existente do arquivo.

Ainda conforme Viana (2015), os fatores que influem na eficiência de um algoritmo de ordenação externa são os que você poderá ver, clicando nos ícones a seguir.

Número de séries produzidas.
O tamanho das séries e se são do mesmo tamanho.
As características dos blocos e das memórias intermediárias utilizadas na fase de classificação e a forma da distribuição das séries pelos arquivos de trabalho.

Dessa forma, conforme Ziviani (2004), podemos concluir que existem três fatores importantes que diferenciam os algoritmos de ordenação externa dos algoritmos de ordenação interna, são os que você poderá ver, clicando no recurso a seguir.

- O valor de acesso do item

  É a ordem de grandeza maior do que os valores de processamento na memória interna. O valor principal na ordenação externa está relacionado com o valor de mudar dados entre a memória externa e a memória interna.

- Existem regras graves de acesso aos dados

  Por exemplo, os itens guardados em fita magnética só podem ser localizados em sequências; os itens guardados em disco podem ser acessados de forma direta, mas a um valor maior do que o valor para acessar de forma sequencial, o que contraindica o uso de acesso direto.

- A depender do estado atual da tecnologia podemos medir o desempenho do desenvolvimento de métodos de ordenação externa

  Muitos tipos e variedades de unidades de memória externa podem tornar os métodos de ordenação externa dependentes de vários parâmetros que afetam sua performance.

CASO

Muitas vezes, em nosso dia-a-dia precisamos consultar dados ordenados. Como por exemplo, várias listas de telefones ou até mesmo muitas listas de remédios, em se tratando de uma farmácia. O simples fato de ter que procurar um remédio ou um telefone de contato em uma lista telefônica daria muito trabalho no caso de não estarem em ordem alfabética ou numérica. O principal papel da ordenação é facilitar a busca e recuperação da informação. Dispor as coisas em ordem está presente em boa parte das aplicações, principalmente quando temos informações que precisamos recuperar no futuro, como no caso da lista telefônica ou da lista de remédios. Já imaginou ter que procurar nome por nome, de cada remédio, em várias tabelas armazenadas nos inúmeros dispositivos como o *Pen drive*, enquanto um cliente espera na farmácia por sua reposta sobre a disponibilidade do mesmo? As listas ordenadas ajudam muito em nosso cotidiano quando temos objetos que foram armazenados e precisam ser pesquisados e recuperados.

Medimos a eficiência de uma classificação externa a partir da quantidade total do tempo necessário para que os dados sejam lidos, classificados e gravados na área de trabalho, dando início a primeira fase do processo de ordenação ou seja, basicamente os algoritmos vão separar os dados em diversas fontes e ordenar essas fontes.
Na segunda fase, as fontes classificadas são lidas e intercaladas em uma sequência de duas a duas, três a três etc. O número de fontes intercaladas em cada etapa corresponde ao número de caminhos. Quanto maior for o número de caminhos, mais complexo será o algoritmo de intercalação também chamado de *Merge*. De um modo geral, usamos dois ou três caminhos já que este processo é usado diversas vezes para alcançar o agrupamento de todas as fontes. Na sessão a seguir apresentaremos os algoritmos de intercalação de dois caminhos, de três caminhos e de *k* caminhos.

3.3 Intercalação de arquivos: intercalação de dois caminhos; intercalação de três caminhos; intercalação de k caminhos

A intercalação é o processo de misturar diferentes tópicos e habilidades; em boa parte das vezes, é a mistura de tópicos ou de fontes realmente diferentes, como *HD's*, *Pen Drives* ou, ainda a mistura de outras fontes com novas fontes. No caso, por exemplo, de uma escola trabalhar misturando tópicos realmente diferentes, como matemática e química, ou a mistura de material antigo e novo. No segundo caso, em vez de estudar/revisar as coisas em uma ordem cronológica, é misturado os assuntos melhorando a retenção dos tópicos.
Intercalar é a maneira de combinar dois, três ou mais blocos agrupados em um único grupo ordenado. A intercalação é utilizada para auxiliar na ordenação como se fosse uma operação. Quando usamos o termo intercalar, estamos realmente nos referindo ao ato de anexar um item em ordem classificada à uma estrutura temporária da matriz, que foi construída ao longo do tempo, à medida que continuamos classificando e acrescentando mais itens. Essa etapa envolve comparar as listas e, em seguida, anexar a maior ao final da menor.
Mas, quando a intercalação é útil (e quando não é)? Voltaremos à ideia do *baralho de Tarô* descrito na introdução; quando pegamos um baralho completo para ordenar qual a primeira coisa a fazer? É provável que o baralho seja separado por *naipes*, ordenando; o próximo passo é que se junte todo baralho, fazendo a

intercalação. Nesse exemplo talvez não seja preciso fazer muitas intercalações, porém, a ideia é dividir em grupos menores e depois reintegrar em algo maior e ordenado. É o que iremos ver nesta sessão, porém, utilizando uma quantidade de fontes que chamamos de caminhos; às vezes esses algoritmos serão identificados como intercalação balanceada de dois caminhos, três caminhos e *k* caminhos.

VOCÊ QUER LER?

Atualmente, com as facilidades presentes para manipulação de bases de dados e arquivos, as técnicas de fusão de arquivos para a intercalação ou ordenação acabam esquecidas. Contudo, essas técnicas são facilmente implementadas em ambientes de computação com poucos recursos. Um estudo bem interessante de Arbex (2009), utiliza técnicas de fusão de arquivos para a intercalação ou ordenação de arquivos sequenciais. Veja o artigo completo clicando aqui <https://seer.cesjf.br/index.php/cesRevista/article/viewFile/706/559>.

O que esses caminhos querem dizer? Dizem respeito a quantidade de fontes que será intercalada para se criar algo mais ordenado. Sendo assim, voltando ao exemplo do *baralho de tarô*, quando pegamos, separamos o baralho em quatro naipes e juntamos tudo em algo único, diríamos que são quatro fontes. Outro exemplo concreto é utilizando um conjunto de *HD`s* externos para fazer a ordenação; dependendo das quantidades de *HD`s* externos, temos a quantidades de caminhos. Assim, poderíamos definir a intercalação de arquivos como sendo a combinação de dois ou mais blocos ordenados em um único bloco agrupado. A intercalação é utilizada na ordenação como uma operação auxiliar. (ZIVIANI, 2012).
Conforme Ziviani (2012), com as fontes já formadas, podemos então fazer a intercalação entre elas. Para isso, precisamos de alguns métodos que sejam responsáveis por fazer esse trabalho. Estes métodos, chamaremos de *Merge2, Merge3* e o *Mergek*, responsáveis por fazer a intercalação de duas fontes, três fontes, e *k* fontes respectivamente, veremos detalhes deste processo a seguir na próxima subseção.

3.3.1 Intercalação de dois caminhos

A intercalação de dois caminhos também conhecida como *Merge2*, é o método responsável por mesclar duas fontes. Para exemplificar, tomemos um arquivo com dez registros, ou seja, de A_1 a A_{10} de acordo com a figura a seguir, levando em consideração que esses dez registros não são suportados pela memória *RAM*. O processo de intercalação externa, utilizando o processo intercalação de dois caminhos, se dá da seguinte forma: dividindo esse arquivo ao meio (ou próximo do meio), de forma que ele consiga utilizar nessa primeira parte do arquivo um dos métodos de ordenação e na segunda parte do arquivo o mesmo método de ordenação; em seguida é realizado o processo de intercalação entre os dois arquivos.
Perceba que aquele único arquivo apresentado anteriormente foi dividido em dois arquivos, o *Arquivo1* que foi iniciado em A_1 até A_4 e o *Arquivo2* que foi iniciado em A_5 até A_{10}, conforme ilustração a seguir. Temos ainda, um terceiro arquivo com valores de B_1 até B_{10}, onde foi armazenado todos os registros do arquivo original já em ordem, ou seja, foi realizado uma intercalação entre o *Arquivo1* e o *Arquivo2*, dando origem à uma única série ordenada, como mostra a figura abaixo.

Figura 1 - Processo de intercalação de dois caminhos.

VAMOS PRATICAR?

Imagine um banco de dados contendo dez milhões de registros, cada um com 100 bytes de comprimento. Forneça uma estimativa de quanto tempo levaria para classificar o banco de dados em uma estação de trabalho típica.

Observe agora o pseudocódigo, adaptado de Viana (2015), do método *Merge2* descrito anteriormente:

PROCEDIMENTO DE 2 CAMINHOS (A, B, c, d)
ENTRADA: *INTEIROS x, y Registro H com duas fontes a intercalar*
SAÍDA: *Registro Z com uma única série ordenada*
// A (1:c) (Primeira fonte ordenada)
// A (c+1:d) (Segunda fonte ordenada)
// B (1:d) (Fonte ordenada/intercalada)
//Variáveis auxiliares
// K (1:d) Vetor de inteiros
// i, j, k, ini, fim: ponteiros/contadores
i = 1;
k = 1;
j = c+1;
ENQUANTO *(i<=c AND j<=d){*
SE *(Bi <= Bj)*
Bk = Ai
i++;
SENÃO
Bk = Aj
j++;

SE *(i >c)*
ini = j
fim = d
SENÃO
ini = i
fim = c
//Copiar o restante da fonte para a fonte resultado
PARA *i=ini ATÉ fim*
Bk = Ai
k++
FIM_PARA
FIM_ENQUANTO
Retorna

Aprendemos nesta subseção como se procede o funcionamento das fases de intercalação de dois caminhos, também chamado de *Merge2*, na seguinte subseção abordaremos detalhes da intercalação de três caminhos, também conhecido como *Merge3*.

3.3.2 Intercalação de três caminhos

A intercalação de três caminhos, também conhecida como *Merge3* é o método responsável por intercalar três fontes. Bem semelhante ao método *Merge2*, o *Merge3* segue o mesmo princípio e o mesmo raciocínio do processo de intercalação de dois caminhos, o que os diferencia é que um utiliza dois caminhos ou seja duas fontes, e o outro, três arquivos para fazer a intercalação (*Arquivo1, Arquivo2* e *Arquivo3*) e um arquivo para fazer a ordenação desses três arquivos (*Arquivok*).

Uma outra característica do processo de intercalação de três caminhos é que ele começa utilizando o procedimento de três caminhos, ou seja, digamos que temos três arquivos para serem comparados e ordenados, pegamos o menor valor e armazenamos no valor resultado; em determinado momento, um desses três arquivos finaliza seu processo de intercalação, restando dois arquivos, então o procedimento de ordenação *Merge2* se inicia até que um dos arquivos finalize e termine de ordenar o arquivo solução, deixando o arquivo solução todo ordenado. Um exemplo de estrutura de algoritmo (pseudocódigo) de intercalação de três caminhos pode ser observado a seguir, adaptado do pseudocódigo de Viana (2015):

PROCEDIMENTO DE 3 CAMINHOS (A, B, c, d, E)
ENTRADA: *INTEIROS c, d, E*
Registro A com três séries a intercalar
A1:c (primeira fonte ordenada)
Ap+1:d (segunda fonte ordenada)
Ad+1:E (terceira fonte ordenada)
SAÍDA: *Registro B com uma única fonte ordenada*
// B1:E (fonte ordenada/intercalada)
// Variáveis auxiliares
K: vetor [1:n] de inteiros
i, j, k, l, m, n: ponteiros/contador
FUNÇÃO MENOR *(X, Y, Z): inteiro*
ENTRADA: *X, Y, Z inteiros*
SAÍDA: *MENOR*
// retorna 1 se o menor é X
// retorna 2 se o menor é Y
// retorna 3 se o menor é Z
// Variáveis auxiliares

```
v: vetor [1:3] de inteiros
w: inteiro
v[1] = X; v[2] = Y; v[3] = Z;
MENOR = 1; w = v[1];
PARA i = 2 ATÉ 3
SE v[i] < w
w = v[i]
MENOR = i
// fim para
DEVOLVA MENOR
FIM {MENOR}
i = 1; k = 1; j = c + 1; l
= d + 1
ENQUANTO ( i ≤ c E j ≤ d E l ≤ n )
CASO MENOR( Ki , Kj , Kl )
1: Bk = Ai
i = i + 1
2: Bk = Aj
j = j + 1
3: Bk = Al
l = l + 1
// fim caso
k = k + 1
// O trecho seguinte define os parâmetros dos dois
// últimos arquivos para o MERGE2.
SE ( i > c )
m = d – j + 1
n = E - l + 1
SENÃO
SE ( i > c )
m = c – i + 1
n = E - l + 1
SENÃO
m = c – i + 1
n = d – j + 1
MERGE2 ( A, B, m, n )
// fim enquanto
RETORNA
FIM {MERGE3}
```

O algoritmo que faz a intercalação de *Merge3*, identificado também como intercalação de três caminhos, utiliza respectivamente representantes das chaves dos seus registros *Ai, Bj, Al* e *Bk*. A seguir daremos início a intercalação de *k* caminhos.

3.3.3 Intercalação de k caminhos

Em computação, os algoritmos de intercalação de *k* caminhos ou mesclagem de múltiplas vias, são um tipo específico de algoritmos de mesclagem de sequência que se especializam em receber várias listas classificadas e

fundi-las em uma única lista ordenada. Esses algoritmos de mesclagem geralmente se referem a aqueles que recebem um número de listas classificadas maiores que duas ou três. Mesclagens bidirecionais também são chamadas de mesclagens binárias.

Em ordenação externa, uma mesclagem de várias vias, ou seja, *k* vias, permite que os arquivos fora da memória sejam intercalados em menos passagens do que em uma mesclagem binária. Se houver seis execuções que precisam ser mescladas, uma mesclagem binária precisaria receber três passes de mesclagem, diferente da mesclagem de seis vias. Essa redução de passes de mesclagem é especialmente importante, considerando a grande quantidade de informações que geralmente estão sendo ordenadas em primeiro lugar, permitindo maiores acelerações e reduzindo a quantidade de acessos à memória mais lenta.

No geral, algoritmos que fazem intercalação *Mergek* também conhecido como intercalação de *k* caminhos, são implementados de forma bem similar ao *Merge3*, ou seja, os registros são intercalados pegando o primeiro registro de cada série e transferindo para a próxima série, assim, o menor é excluído da série, cedendo lugar ao próximo registro até que se tenha a fita ordenada.

Exemplo: para o Merge3 (*k=3*), a função *MENOR* selecionaria o menor valor entre os três números e no momento em que é atingido o fim de um dos arquivos, deveria ser "chamado" o procedimento *Merge2* para finalizar o processo de intercalação.

Aqui conhecemos como se dá o método de ordenação externa para *k* caminhos, porém não é comum a utilização de uma intercalação com mais de três caminhos, devido à complexidade deste algoritmo, mesmo sabendo que o número de leituras e gravações poderia ser reduzido. O raciocínio da intercalação *Mergek* é o mesmo da intercalação *Merge2*, quebrando o arquivo único em dois, três ou quatro arquivos, quantos valores forem o *k* e o processo de intercalação entre esses arquivos é iniciado, ao final temos um único arquivo ordenado. A seguir conheceremos a comparação entre distribuições equilibradas.

3.4 Comparação entre distribuições equilibradas

Para exemplificar a comparação entre distribuições equilibradas vamos adaptar o caso de Viana (2015). Suponhamos que desejamos classificar *9000* registros, porém, a memória que temos disponível para a ordenação interna é pequena e não suporta mais de *1000* registros a cada vez; ou seja, será realizada a cada *1000* registros lidos uma classificação; estes serão ordenados de formas correspondentes e armazenados em uma fonte auxiliar de modo ordenado e em sequência (arquivo todo sequencial), de modo que os números de registros sejam múltiplos de *1000*, caso contrário a última série ficará incompleta, o que não é difícil para o processo de intercalação do tipo *Mergek*, já visto nas seções anteriores.

O número de arquivos de trabalho, assim como a distribuição das séries nestes arquivos, depende do número *k* de fontes a serem intercalados. (VIANA, 2015). Podemos ver esse processo na figura a seguir, na qual para *k=2* serão necessários cinco arquivos de trabalho, ou seja, precisarão cinco arquivos para conseguir fazer esse processo de classificação e ordenação. Poderemos observar, na mesma figura, um arquivo principal com *9000* registros, classificados de *1 à 1000*, e de *1001 à 2000* e assim sucessivamente até alcançar os *9000* registros. Em um primeiro passo é realizado um processo de intercalação, armazenado nos arquivos auxiliares (*Arquivo1* e *Arquivo2)*. Pega-se o primeiro valor na primeira classe de *1* até *1000* e armazena-se no *Arquivo1*, pega o segundo intervalo ou segunda classe que é de *1001* até *2000* e armazena no *Arquivo2* e o processo é feito até chegar ao final do arquivo principal; por último teremos "A" registros que vai de *8001* a *9000*, que foi armazenado no *Arquivo1*. Nota-se também que é feita uma intercalação no *Arquivo1* e no *Arquivo2*.

Para concluirmos esse processo de ordenação é preciso mais três arquivos que seguem como mostrado nas etapas do processo. Dando continuidade às etapas descritas na figura, começa-se o processo de classificação dando início a partir da coluna, ou seja, pega o resultado da primeira coluna e armazena no primeiro intervalo do *Arquivo3*, pega a segunda coluna e armazena no *Arquivo4*, a terceira coluna no *Arquivo3*, a quarta coluna no

Arquivo4 e assim por diante, realizando o processo de intercalação e resultando no último arquivo contendo todo o intervalo (*A1* até *A9000)* já ordenado, então esse é o processo de comparação entre distribuições equilibradas.

Primeiro realiza-se uma divisão de mil em mil, em seguida unifica-se as colunas armazenando os registros no próximo arquivo, dando continuidade ao processo até que se finalize toda a ordenação, gerando o último arquivo com todas as séries (*1* até *9000*) classificadas.

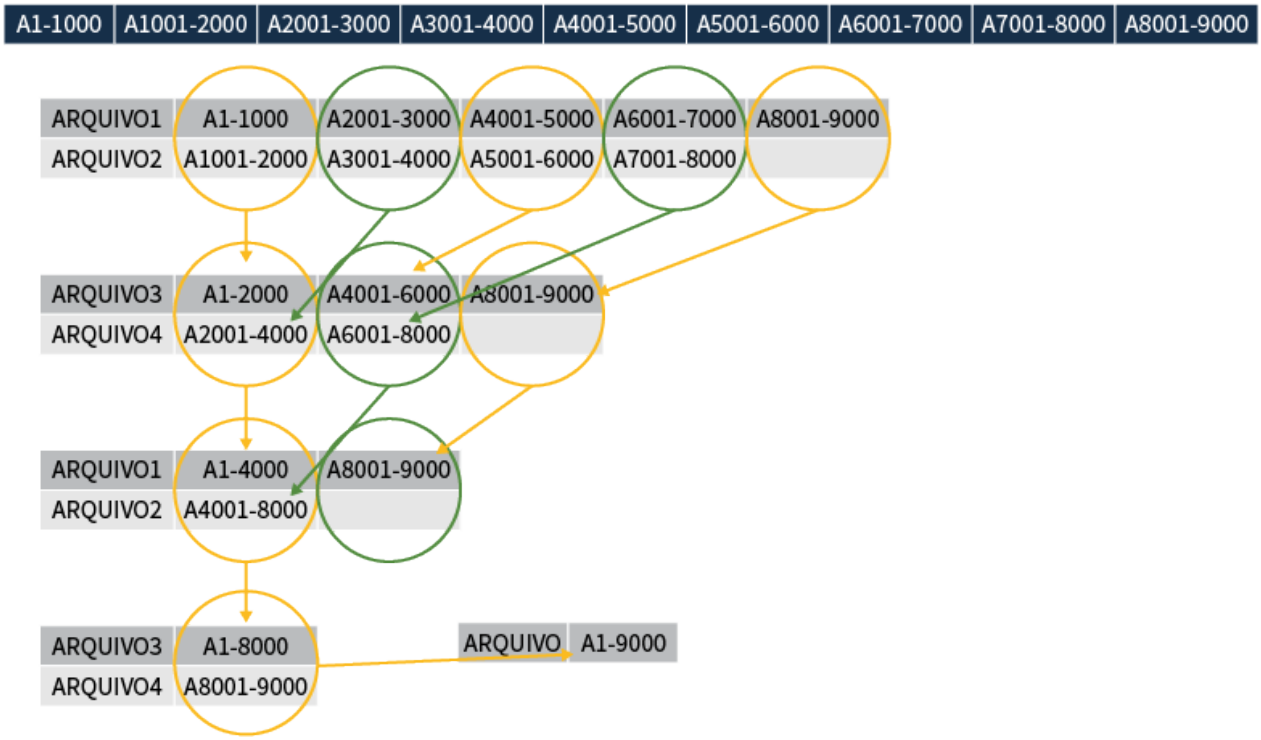

Figura 2 - Processo de classificação e ordenação.

VAMOS PRATICAR?

Classifique *15000* registros *(B1* até *B15000)* usando o processo de ordenação que você acabou de ver, adote k=2.

Nesta seção, foi possível aprender sobre comparação entre distribuição equilibrada e como o método é aplicado com os dados simulando o *k=2*. Nas seções seguintes faremos uma simulação para *k= 2, 3* e *4* e veremos os tópicos da classificação dos métodos de ordenação externa.

3.5 Classificação dos métodos de ordenação externa

Quando falamos de ordenação externa, estamos falando de métodos para organizar arquivos que não cabem na memória tradicional do computador; atualmente, o método mais importante para classificação externa é o método por intercalação. (VIANA, 2015).
Alguns métodos de ordenação externa são os que você poderá ver, clicando nos ícones a seguir.

Intercalação Balanceada de vários caminhos.
Implementação por meio de Seleção por substituição.
Intercalação Polifásica.
Quicksort externo, também conhecido como Heapsort.

Definimos a intercalação balanceada de vários caminhos como a intercalação de blocos, na qual é realizada uma varredura na memória externa, dividindo-a em blocos compatíveis com a memória interna. Os blocos são intercalados, realizando leituras sobre este arquivo, criando blocos classificados cada vez maiores. Já a implementação por meio de Seleção por substituição é uma implementação muito parecida com a intercalação balanceada, porém, sofre um acréscimo da abordagem de filas por prioridade, implementadas por um *heap*. (ZIVIANI, 2012).

VOCÊ SABIA?

Na ciência da computação, uma fila de prioridade é um tipo de dados abstrato baseado em uma fila regular ou estrutura de dados de pilha, porém, cada elemento tem uma "prioridade" associada a ele. Já um *heap* é uma estrutura de dados baseada em árvore, na qual todos os nós da árvore estão em uma ordem específica. Em uma pilha, o elemento de prioridade mais alta (ou mais baixa) é sempre armazenado na raiz, daí o nome "pilha". Um *heap* não é uma estrutura classificada e pode ser considerado como parcialmente ordenado.

A intercalação polifásica excluí a necessidade de cópias adicionais pois distribui as fitas ordenadas por meio de uma seleção desigual. Convido você a ver com mais detalhes a classificação de cada um desses métodos de ordenação externa nas seções seguintes.

3.6 Classificação Equilibrada de dois caminhos

Um exemplo de classificação equilibrada de dois caminhos pode ser visto na Figura 3, onde é adotado um limite de classificação interna de três registros, então a intercalação será realizada de três em três, lembrando que na classificação equilibrada de dois caminhos, ou seja para *k=2* será preciso quatro arquivos para realizar a ordenação, chamados de *Arquivo1*, *Arquivo2*, *Arquivo3* e *Arquivo4*.

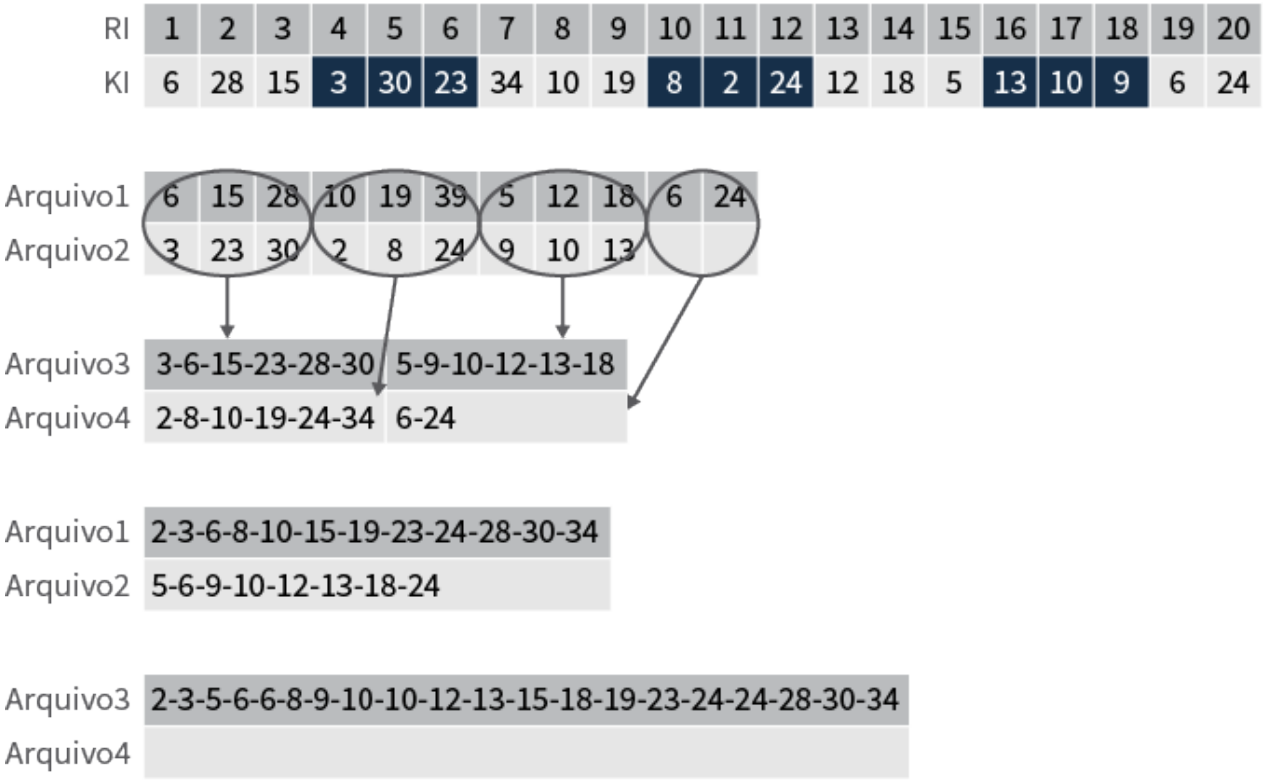

Figura 3 - Classificação equilibrada de 2 caminhos.

O primeiro passo é pegar os três primeiros valores e trazer para o primeiro arquivo (*Arquivo1*), ordenando (*6, 28* e *15*); os próximos três (*3, 30* e *23*) serão levados para o próximo arquivo (*Arquivo2*), dando continuidade ao processo de intercalação de três em três. Ao final sobraram (*6* e *24*). Como mencionado anteriormente os últimos valores não precisam ter o valor completo da taxa interna de registro, após realizado o primeiro passo com os dois arquivos (*Arquivo1* e *Arquivo2*), deve-se continuar realizando o processo de intercalação, nesse caso, o próximo passo será pegar a primeira coluna que se refere aos dois arquivos e realizar a classificação, ou seja armazenar no *Arquivo3*. Então, teremos (*3, 6, 15, 23, 28* e *30*), ou seja, os valores que estão ordenados no *Arquivo3*; deve ser realizado o mesmo procedimento para os demais valores, até que de fato cada fonte tenha um único bloco todo ordenado.
Nesta sessão aprendemos sobre classificação equilibrada de dois caminhos com um exemplo clássico de *Merge2* de ordenação externa a seguir veremos o processo de classificação equilibrada usando três caminhos.

3.7 Classificação Equilibrada de três caminhos

Para a classificação equilibrada com *k=3* podemos usar também quatro arquivos de trabalho, de modo que no início da fase de classificação seja possível criar fitas de comprimento *1000* e as distribuir também em dois arquivos; a única coisa que diferencia das demais é que em determinado momento da segunda fase, usando a classificação de dois caminhos, um dos arquivos fica vazio, sendo utilizada a classificação de três caminhos.
Observe na figura abaixo como ficaria essa classificação.

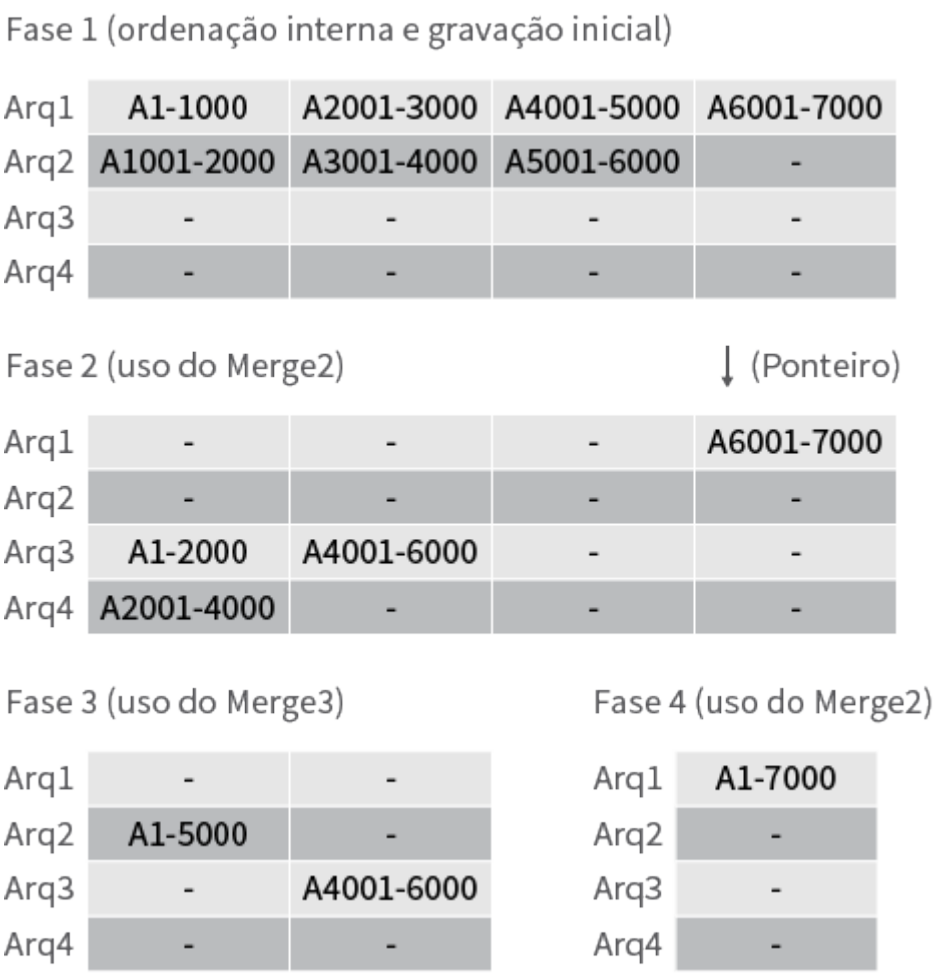

Figura 4 - Processo de classificação Equilibrada de três caminhos.
Fonte: Elaborada pela autora, baseada em VIANA, 2015.

É possível notar que a fita *A1-5000* é a saída no *Merge3* com entrada obtida a partir de (*A6001-7000, A1-2000 e A2001-4000)* praticamente nesta ordem. A fase que finaliza diz respeito ao *Merge2.* Observe que para todas as fases deste tipo de classificação pode ser usado o *Merge3*, dado que de acordo com sua descrição, quando um dos arquivos está com o ponteiro no final o *Merge2* é acionado.

VAMOS PRATICAR?

Classifique *15000* registros *(B1* até *B15000)* usando o processo de ordenação que você acabou de ver, adote k=3.

A seguir, veremos detalhes da ordenação equilibrada de múltiplos caminhos.

3.8 Classificação Equilibrada de múltiplos caminhos

De acordo com Viana (2015), quando tratamos de classificação equilibrada de vários caminhos também conhecida como classificação equilibrada *Mergek,* levamos em consideração que para *k>4* é necessário *(k+1)* arquivos de trabalho. A fase que dá início a essa classificação cria fitas de tamanhos fixos e as insere alternadamente em *k* arquivos, ou seja (*Arq1* e *Arq2*) deixando a ordem (*k+1*) livre. A próxima fase utiliza o procedimento de intercalação de vários caminhos o *Mergek*.
Veremos na figura abaixo um exemplo de classificação equilibrada de *Mergek*.

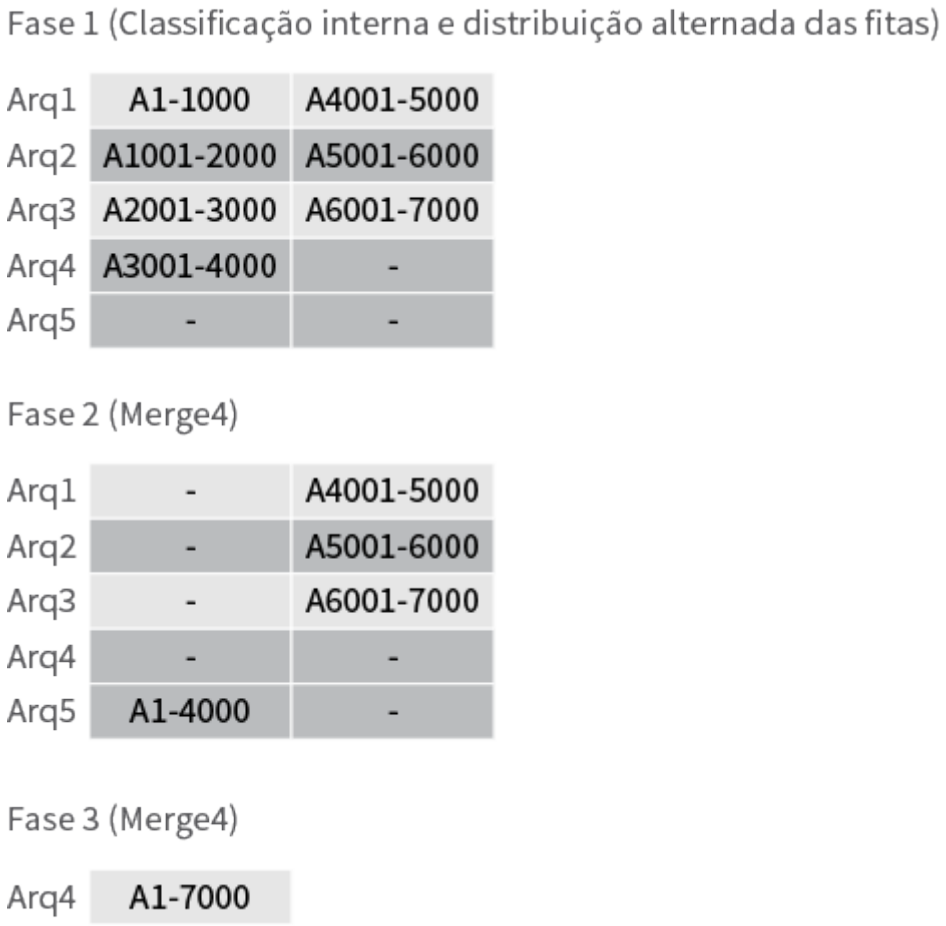

Figura 5 - Processo de classificação Equilibrada de múltiplos caminhos.
Fonte: Elaborada pela autora, baseada em VIANA, 2015.

O processo de classificação equilibrada de múltiplos, consiste em fundir algoritmos e ocorre geralmente na segunda fase, usando algoritmos de ordenação externa, da mesma forma como é realizado com o *Mergesort*. Uma junção com vários pontos permite que os arquivos externos sejam incorporados em menos tempo do que em uma mesclagem binária. Esta redução de passagem é especialmente importante, considerando a enorme quantidade de informação que é usualmente classificada em primeiro lugar, permitindo maiores acelerações e ao mesmo tempo, reduzindo a quantidade de acessos à memória mais lenta.
Agora que você reconhece a classificação equilibrada de múltiplos caminhos, já está preparado para compreender o conceito de intercalação polifásica definida como a distribuição dos blocos ordenados por meio de uma seleção desigual, assunto do tópico a seguir.

3.9 Intercalação polifásica

Na intercalação polifásica, os blocos ordenados são distribuídos de forma desequilibrada entre as fitas disponíveis; uma fita é deixada de forma livre e logo após, a intercalação de blocos ordenados é executada até que uma das fitas esvazie. Neste ponto, uma das fitas de saída troca de papel com a fita de entrada. (ZIVIANE, 2012).

Ainda conforme Ziviane (2012), a intercalação é realizada em diversas fases e essas fases não envolvem todos os blocos e nenhuma cópia direta entre fitas é realizada. De acordo com Viana (2015), o objetivo deste tipo de intercalação é distribuir as séries iniciais de forma desequilibradas, de modo que menos passos sejam necessários para a classificação total.

Podemos observar na figura abaixo, alguns passos do processo de uma ordenação externa usando intercalação polifásica. Navegue na interação a seguir para entender cada passo da figura abaixo.

No passo um, os blocos são ordenados obtidos por meio de seleção por substituição, a partir da frase "INTERCALACAOBALANCEADA".

No passo dois, intercala-se para fita três deixando a fita dois livre.

No passo três, intercala-se a fita um e a fita três na fita dois, deixando a fita um livre.

No passo quatro, intercala-se para fita três, deixando a fita dois livre.

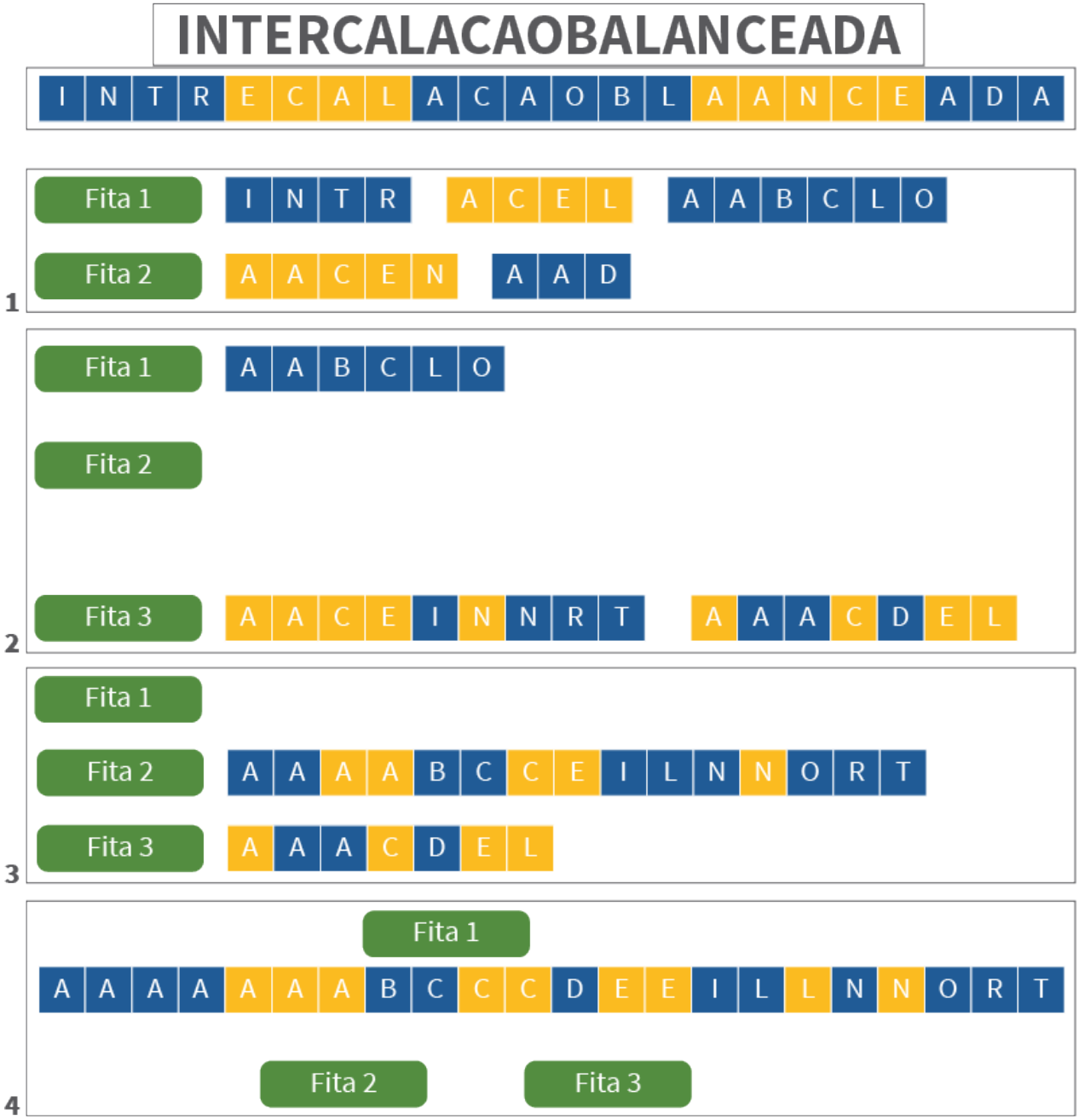

Figura 6 - Processo de ordenação externa usando Intercalação Polifásica.

Os blocos são distribuídos entre unidades de forma desigual, na intercalação polifásica, deixando apenas uma unidade reservada para guardar os blocos restantes das intercalações. Em cada fase, uma unidade se esvazia e assim sucessivamente, até que a intercalação seja finalizada. A intercalação é realizada em muitas fases; estas não envolvem todos os blocos, sendo ligeiramente melhor do que a intercalação balanceada.

O algoritmo *Quicksort*, de ordenação rápida, introduzido em 1960 por Charles Antony Richard Hoare, pertence à classe dos chamados algoritmos de divisão e conquista, semelhantes à classificação de mesclagem. É popular por não ser difícil de implementar, além de ser uma boa classificação de uso geral, já que funciona em várias situações e consome menos recursos do que qualquer outro algoritmo de classificação.

VOCÊ QUER VER?

O *Quicksort* é um algoritmo que foi desenvolvido em 1960, por Charles A. R. Hoare, conhecido também como Tony Hoare; é o método de classificação interna com maior rapidez para uma ampla variedade de situações. Ele ganhou o Prêmio Turing (*Association for Computing Machinery*), de 1980. (CORMEM, 2002). Para conhecer mais sobre a execução desse algoritmo, clique aqui: <https://youtu.be/ywWBy6J5gz8>.

Ele funciona particionando um arquivo em duas partes e, em seguida, classificando as partes independentemente. Os elementos são divididos em duas submatrizes repetidamente, até que apenas um elemento seja deixado. A classificação de mesclagem usa armazenamento adicional para classificar a matriz auxiliar, ou seja, usando três matrizes, duas são usadas para armazenar cada metade e a terceira é usada para armazenar a lista final classificada, mesclando outras duas; cada matriz é classificada recursivamente. Por fim, todas as submatrizes são mescladas para tornar o tamanho do elemento da matriz.
Um exemplo pode ser observado na figura abaixo.

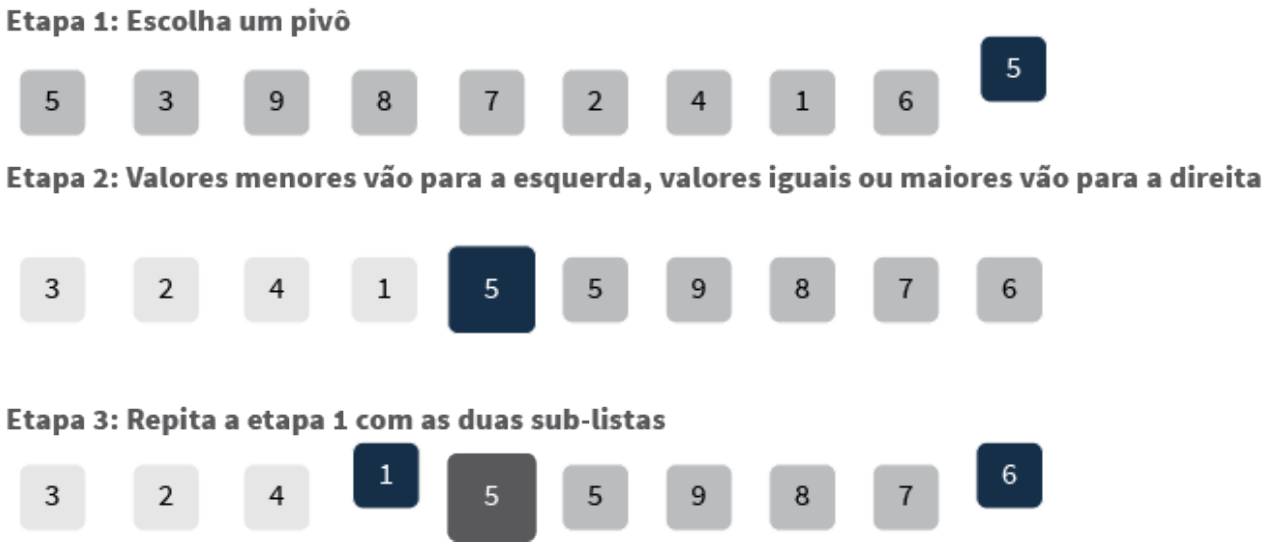

Figura 7 - Processo de ordenação usando o Algoritmo Quicksort.
Fonte: Elaborada pela autora, baseada em CORMEM (2002).

Temos como elemento o *P (Pivô)* e a reorganização é feita em três sub-blocos, nos quais os elementos iguais ou menores *(a <= P)*, são agrupados a esquerda e os elementos *P*, no bloco do meio. Aqueles maiores ou igual *(a > = P)*, são agrupados no bloco da direta. O processo é repetido recursivamente para os sub-blocos esquerdo e direito. O *Quicksort* acaba sendo o algoritmo de classificação mais rápido na prática e muito eficiente para classificar os arquivos de dados.

Síntese

Concluímos a unidade introdutória relativa à ordenação externa e processos de intercalação dos arquivos. Agora que você já conhece o processo de ordenação externa, e sabe que ela é empregada quando a quantidade de dados processada é grande e não cabe na memória interna do computador. Também vimos os principais métodos de

ordenação com foco no conceito e funcionamento de cada um deles. Além disso, percebemos que, caso o arquivo siga uma sequência, é necessário empregar outros métodos baseados em processos de divisão e medida conhecido como intercalação de arquivos.

Nesta unidade, você teve a oportunidade de:

- reconhecer os mecanismos de ordenação externa;
- conhecer os processos de intercalação dos arquivos;
- verificar a eficiência entre as diversas distribuições.

Bibliografia

AGUILAR, L. J. **Fundamentos de Programação: Algoritmos, estruturas de dados e objetos**. 3. ed. Porto Alegre: AMGH, 2008.

ARBEX, W. *et al*. Intercalação e Ordenação de arquivos por algoritmos de fusão. **CES revista**, Juiz de Fora, v. 23, p. 227- 237, 2009. Disponível em: <https://seer.cesjf.br/index.php/cesRevista/article/viewFile/706/559>. Acesso em: 10/10/2019.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. **Introduction to algorithms-string matching**. Massachusetts: MIT Press, 2002.

DASGUPTA, S.; PAPADIMITRIOU, C.; VAZIRANI, U. **Algoritmos**. Porto Alegre: AMGH, 2011.

DEITEL. P.; DEITEL H. **Java: como programar**. 8. ed. São Paulo: Pearson, 2015.

DOBRUSHKIN, V. A. **Métodos para Análise de Algoritmos**. Rio de Janeiro: LTC, 2012.

GUIMARÃES, A. M.; LAGES, N. A. C. **Algoritmos e estruturas de dados**. Rio de Janeiro: LTC, 2015.

KNUTH, D. E. **Sorting and Searching: The Art of Computer Programming**. 3. vol. Massachussets, Addison-Wesley, 1973.

MAHMOUD, H. M. **A Distribution Theory**. New Jersey, Wiley – Interscience, 2000.

QUICK-SORT WITH HUNGARIAN FOLK DANCE. Criado por: Sapientia University, Tirgu Mures (Marosvásárhely), România. Dirigido por: Kátai Zoltán and Tóth László. 6:54 min. Disponível em: <https://www.youtube.com/watch?v=ywWBy6J5gz8&feature=youtu.be>. Acesso em: 08/10/2019.

SHELL-SORT WITH ROMANIAN FOLK DANCE. Criado por: Sapientia University, Tirgu Mures (Marosvásárhely), România. Dirigido por: Kátai Zoltán and Tóth László. Disponível em: <https://www.youtube.com/watch?v=CmPA7zE8mx0>. Acesso em: 08/10/2019.

SINGH, B.; NAPS, T. L. **Introduction to Data Structure**. Minnesota: West Publishing Co, 1985.

TAMASSIA, R.; GOODRICH, M. T. **Estruturas de Dados e Algoritmos em Java**. Porto Alegre: Grupo A, 2011.

VIANA, G. V. R.; CINTRA, G. F.; NOBRE; R. H. **Pesquisa e ordenação de Dados**. 2. ed. Fortaleza: EdeuECE, 2015.

ZIVIANI, N. **Projeto de Algoritmos: com implementações em JAVA e C++**. São Paulo: Cengage Learning Editores, 2012.