

# AEDS III

Ordenação Externa  
Prof. Olga Goussevskaia

# Ordenação Externa

- Sumário:
  - Exemplo de aplicação: MapReduce do Google
  - Intercalação balanceada
  - Seleção por substituição
  - Intercalação polifásica
  - Quicksort externo

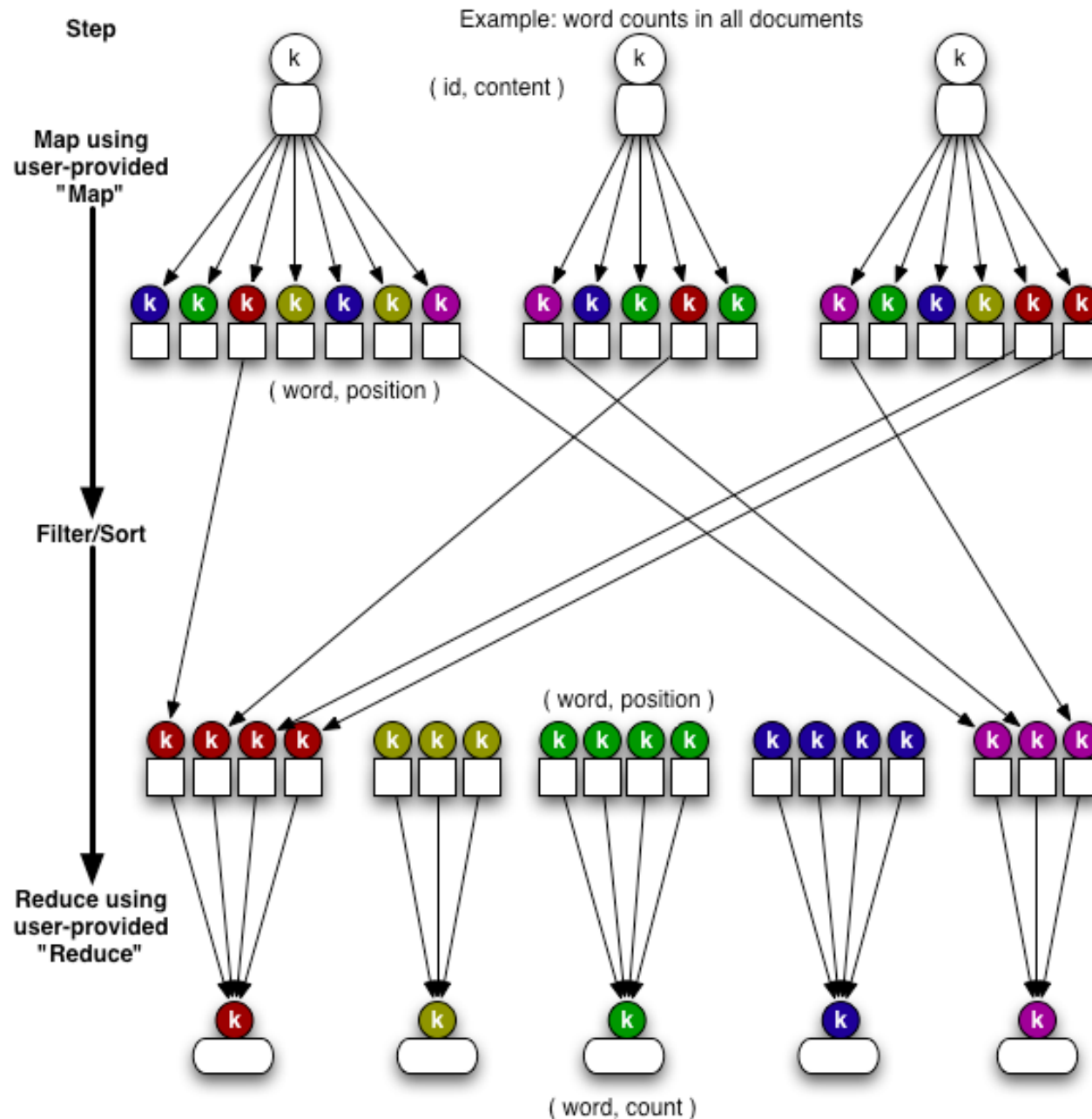
# Ordenação Externa

- Exemplo de aplicação: MapReduce do Google
  - Framework de computação distribuída e paralela.
  - Realiza operações simples sobre entradas muito grandes de dados, onde a computação sobre uma parte não afeta a computação sobre a outra parte (admite paralelização trivial)

# MapReduce do Google

- Exemplo de problema: contar quantas vezes cada URL foi acessada.
  - Entrada: vários arquivos de log de acessos
  - Estimativa do número de documentos indexados pelo Google em 2008: 1 trilhão ( $10^{12}$ ).
- **Map**: dividir a entrada de dados e distribuir entre vários processadores ( $p_i$ ). Retorno: lista de pares  $\langle \text{URL}, \text{count}_i \rangle$
- **Reduce**: agregar as saídas da fase Map através de uma *ordenação externa* e emitir o resultado total  $\langle \text{URL}, \text{count}_{\text{total}} \rangle$

# MapReduce do Google



Exemplo 2: contar ocorrências de palavras em todos os documentos.

---

## Ordenação Externa

---

- A ordenação externa consiste em ordenar arquivos de tamanho maior que a memória interna disponível.
- Os métodos de ordenação externa são muito diferentes dos de ordenação interna.
- Na ordenação externa os algoritmos devem diminuir o número de acesso as unidades de memória externa.
- Nas memórias externas, os dados ficam em um arquivo seqüencial.
- Apenas um registro pode ser acessado em um dado momento. Essa é uma restrição forte se comparada com as possibilidades de acesso em um vetor.
- Logo, os métodos de ordenação interna são inadequados para ordenação externa.
- Técnicas de ordenação diferentes devem ser utilizadas.

---

## Ordenação Externa

---

Fatores que determinam as diferenças das técnicas de ordenação externa:

1. Custo para acessar um item é algumas ordens de grandeza maior.
2. O custo principal na ordenação externa é relacionado a transferência de dados entre a memória interna e externa.
3. Existem restrições severas de acesso aos dados.
4. O desenvolvimento de métodos de ordenação externa é muito dependente do estado atual da tecnologia.
5. A variedade de tipos de unidades de memória externa torna os métodos dependentes de vários parâmetros.
6. Assim, apenas métodos gerais serão apresentados.

---

## Ordenação Externa

---

- O método mais importante é o de ordenação por intercalação.
- Intercalar significa combinar dois ou mais blocos ordenados em um único bloco ordenado.
- A intercalação é utilizada como uma operação auxiliar na ordenação.
- Estratégia geral dos métodos de ordenação externa:
  1. Quebre o arquivo em blocos do tamanho da memória interna disponível.
  2. Ordene cada bloco na memória interna.
  3. Intercale os blocos ordenados, fazendo várias passadas sobre o arquivo.
  4. A cada passada são criados blocos ordenados cada vez maiores, até que todo o arquivo esteja ordenado.



---

## Ordenação Externa

---

- Os algoritmos para ordenação externa devem reduzir o número de passadas sobre o arquivo.
- Uma boa medida de complexidade de um algoritmo de ordenação por intercalação é o número de vezes que um item é lido ou escrito na memória auxiliar.
- Os bons métodos de ordenação geralmente envolvem no total menos do que dez passadas sobre o arquivo.

---

## Intercalação Balanceada de Vários Caminhos

---

- Considere um arquivo armazenado em uma fita de entrada:

<i>I N T E R C A L A C A O B A L A N C E A D A</i>
--

- Objetivo:
  - Ordenar os 22 registros e colocá-los em uma fita de saída.
- Os registros são lidos um após o outro.
- Considere uma memória interna com capacidade para para três registros.
- Considere que esteja disponível seis unidades de fita magnética.

---

## Intercalação Balanceada de Vários Caminhos

---

- Fase de criação dos blocos ordenados:

fitas 1: *I N T   A C O   A D E*

fitas 2: *C E R   A B L   A*

fitas 3: *A A L   A C N*

---

## Intercalação Balanceada de Vários Caminhos

---

- Fase de intercalação - Primeira passada:
  1. O primeiro registro de cada fita é lido.
  2. Retire o registro contendo a menor chave.
  3. Armazene-o em uma fita de saída.
  4. Leia um novo registro da fita de onde o registro retirado é proveniente.
  5. Ao ler o terceiro registro de um dos blocos, sua fita fica inativa.
  6. A fita é reativada quando o terceiro registro das outras fitas forem lidos.

---

## Intercalação Balanceada de Vários Caminhos

---

- Fase de intercalação - Primeira passada:
  7. Neste instante um bloco de nove registros ordenados foi formado na fita de saída.
  8. Repita o processo para os blocos restantes.
- Resultado da primeira passada da segunda etapa:

fita 4: *A A C E I L N R T*

fita 5: *A A A B C C L N O*

fita 6: *A A D E*

---

## Intercalação Balanceada de Vários Caminhos

---

- Quantas passadas são necessárias para ordenar um arquivo de tamanho arbitrário?
  - Seja  $n$ , o número de registros do arquivo.
  - Suponha que cada registro ocupa  $m$  palavras na memória interna.
  - A primeira etapa produz  $n/m$  blocos ordenados.
  - Seja  $P(n)$  o número de passadas para a fase de intercalação.
  - Seja  $f$  o número de caminhos intercalados.
  - Assim:

$$P(n) = \log_f \frac{n}{m}.$$

No exemplo acima,  $n=22$ ,  $m=3$  e  $f=3$  temos:

$$P(n) = \log_3 \frac{22}{3} = 2.$$

---

## Intercalação Balanceada de Vários Caminhos

---

- No exemplo foram utilizadas  $2f$  fitas para uma intercalação-de- $f$ -caminhos.
- É possível usar apenas  $f + 1$  fitas:
  - Encaminhe todos os blocos para uma única fita.
  - Redistribua estes blocos entre as fitas de onde eles foram lidos.
  - O custo envolvido é uma passada a mais em cada intercalação.
- No caso do exemplo de 22 registros, apenas quatro fitas seriam suficientes:
  - A intercalação dos blocos a partir das fitas 1, 2 e 3 seria toda dirigida para a fita 4.
  - Ao final, o segundo e o terceiro blocos ordenados de nove registros seriam transferidos de volta para as fitas 1 e 2.

---

## Implementação por meio de Seleção por Substituição

---

- A implementação do método de intercalação balanceada pode ser realizada utilizando filas de prioridades.
- As duas fases do método podem ser implementadas de forma eficiente e elegante.
- Operações básicas para formar blocos ordenados:
  - Obter o menor dentre os registros presentes na memória interna.
  - Substituí-lo pelo próximo registro da fita de entrada.
- Estrutura ideal para implementar as operações: *heap*.
- Operação de substituição:
  - Retirar o menor item da fila de prioridades.
  - Colocar um novo item no seu lugar.
  - Reconstituir a propriedade do *heap*.



---

## Implementação por meio de Seleção por Substituição

---

Algoritmo:

1. Inserir  $m$  elementos do arquivo na fila de prioridades.
2. Substituir o menor item da fila de prioridades pelo próximo item do arquivo.
3. Se o próximo item é menor do que o que saiu, então:
  - Considere-o membro do próximo bloco.
  - Trate-o como sendo maior do que todos os itens do bloco corrente.
4. Se um item marcado vai para o topo da fila de prioridades então:
  - O bloco corrente é encerrado.
  - Um novo bloco ordenado é iniciado.

## Implementação por meio de Seleção por Substituição

Entra	1	2	3
<i>E</i>	<i>I</i>	<i>N</i>	<i>T</i>
<i>R</i>	<i>N</i>	<i>E*</i>	<i>T</i>
<i>C</i>	<i>R</i>	<i>E*</i>	<i>T</i>
<i>A</i>	<i>T</i>	<i>E*</i>	<i>C*</i>
<i>L</i>	<i>A*</i>	<i>E*</i>	<i>C*</i>
<i>A</i>	<i>C*</i>	<i>E*</i>	<i>L*</i>
<i>C</i>	<i>E*</i>	<i>A</i>	<i>L*</i>
<i>A</i>	<i>L*</i>	<i>A</i>	<i>C</i>
<i>O</i>	<i>A</i>	<i>A</i>	<i>C</i>
<i>B</i>	<i>A</i>	<i>O</i>	<i>C</i>
<i>A</i>	<i>B</i>	<i>O</i>	<i>C</i>

Entra	1	2	3
<i>L</i>	<i>C</i>	<i>O</i>	<i>A*</i>
<i>A</i>	<i>L</i>	<i>O</i>	<i>A*</i>
<i>N</i>	<i>O</i>	<i>A*</i>	<i>A*</i>
<i>C</i>	<i>A*</i>	<i>N*</i>	<i>A*</i>
<i>E</i>	<i>A*</i>	<i>N*</i>	<i>C*</i>
<i>A</i>	<i>C*</i>	<i>N*</i>	<i>E*</i>
<i>D</i>	<i>E*</i>	<i>N*</i>	<i>A</i>
<i>A</i>	<i>N*</i>	<i>D</i>	<i>A</i>
	<i>A</i>	<i>D</i>	<i>A</i>
	<i>A</i>	<i>D</i>	
	<i>D</i>		

- Primeira passada sobre o arquivo exemplo.
- Os asteriscos indicam quais chaves pertencem a blocos diferentes.

---

## Implementação por meio de Seleção por Substituição

---

- Fase de intercalação dos blocos ordenados obtidos na primeira fase:
  - Operação básica: obter o menor item dentre os ainda não retirados dos  $f$  blocos a serem intercalados.

Algoritmo:

- Monte uma fila de prioridades de tamanho  $f$ .
- A partir de cada uma das  $f$  entradas:
  - Substitua o item no topo da fila de prioridades pelo próximo item do mesmo bloco do item que está sendo substituído.
  - Imprima em outra fita o elemento substituído.

## Implementação por meio de Seleção por Substituição

- Exemplo:

Entra	1	2	3
<i>A</i>	<i>A</i>	<i>C</i>	<i>I</i>
<i>L</i>	<i>A</i>	<i>C</i>	<i>I</i>
<i>E</i>	<i>C</i>	<i>L</i>	<i>I</i>
<i>R</i>	<i>E</i>	<i>L</i>	<i>I</i>
<i>N</i>	<i>I</i>	<i>L</i>	<i>R</i>
	<i>L</i>	<i>N</i>	<i>R</i>
<i>T</i>	<i>N</i>	<i>R</i>	
	<i>R</i>	<i>T</i>	
	<i>T</i>		

- Para  $f$  pequeno não é vantajoso utilizar seleção por substituição para intercalar blocos:
  - Obtém-se o menor item fazendo  $f - 1$  comparações.
- Quando  $f$  é 8 ou mais, o método é adequado:
  - Obtém-se o menor item fazendo  $\log_2 f$  comparações.

---

## Considerações Práticas

---

- As operações de entrada e saída de dados devem ser implementadas eficientemente.
- Deve-se procurar realizar a leitura, a escrita e o processamento interno dos dados de forma simultânea.
- Os computadores de maior porte possuem uma ou mais unidades independentes para processamento de entrada e saída.
- Assim, pode-se realizar processamento e operações de E/S simultaneamente.

---

## Considerações Práticas

---

- Técnica para obter superposição de E/S e processamento interno:
  - Utilize  $2f$  áreas de entrada e  $2f$  de saída.
  - Para cada unidade de entrada ou saída, utiliza-se duas áreas de armazenamento:
    1. Uma para uso do processador central
    2. Outra para uso do processador de entrada ou saída.
  - Para entrada, o processador central usa uma das duas áreas enquanto a unidade de entrada está preenchendo a outra área.
  - Depois a utilização das áreas é invertida entre o processador de entrada e o processador central.
  - Para saída, a mesma técnica é utilizada.

---

## Considerações Práticas

---

- Problemas com a técnica:
  - Apenas metade da memória disponível é utilizada.
  - Isso pode levar a uma ineficiência se o número de áreas for grande.  
Ex: Intercalação-de- $f$ -caminhos para  $f$  grande.
  - Todas as  $f$  áreas de entrada em uma intercalação-de- $f$ -caminhos se esvaziando aproximadamente ao mesmo tempo.

---

## Considerações Práticas

---

- Solução para os problemas:
  - Técnica de previsão:
    - \* Requer a utilização de uma única área extra de armazenamento durante a intercalação.
    - \* Superpõe a entrada da próxima área que precisa ser preenchida com a parte de processamento interno do algoritmo.
    - \* É fácil saber qual área ficará vazia primeiro.
    - \* Basta olhar para o último registro de cada área.
    - \* A área cujo último registro é o menor, será a primeira a se esvaziar.



---

## Considerações Práticas

---

- Escolha da ordem de intercalação  $f$ :
  - Para fitas magnéticas:
    - \*  $f$  deve ser igual ao número de unidades de fita disponíveis menos um.
    - \* A fase de intercalação usa  $f$  fitas de entrada e uma fita de saída.
    - \* O número de fitas de entrada deve ser no mínimo dois.
  - Para discos magnéticos:
    - \* O mesmo raciocínio acima é válido.
    - \* O acesso seqüencial é mais eficiente.
  - Sedegwick (1988) sugere considerar  $f$  grande o suficiente para completar a ordenação em poucos passos.
  - Porém, a melhor escolha para  $f$  depende de vários parâmetros relacionados com o sistema de computação disponível.

# Ordenação Externa

- Notação:
  - $n$ : número de registros a serem ordenados,
  - $m$ : tamanho da memória principal,
  - $f$ : número de caminhos intercalados
- Estratégia geral:
  1. Quebrar o arquivo em blocos menores ordenados
    - Intercalação balanceada: criar  $n/m$  blocos de tamanho  $m$
    - Seleção por substituição: usar um heap de tamanho  $m$  para criar blocos de tamanho médio  $2m$
  2. Iterativamente intercalar blocos ordenados, criando blocos ordenados cada vez maiores

# Ordenação Externa

- Notação:
  - n: número e registros a serem ordenados,
  - m: tamanho da memória principal,
  - f: número de caminhos intercalados
  - b: número inicial de blocos ordenados
  - $P(n)$ : número de passadas na fase de intercalação

$$P(n) = \log_{\{f\}} (b)$$

- Intercalação balanceada:  $P(n) = \log_f (\text{teto}(n/m))$ 
  - Ex:  $n=22, m=3, f=3, P(n)=2$
- Seleção por substituição:  $P(n) = \log_f (\text{teto}(n/2m))$ 
  - Ex:  $n=22, m=3, f=3, P(n)=2$
- Tem que considerar tb o custo de uma passada adicional para criar os blocos ordenados iniciais

# Ordenação Externa

- Como escolher  $f$  (número de caminhos)?
  - Ex:  $n=200$ milhões,  $m=1$ milhão,  $P(n)=2$ ?
$$f^2 \geq \lceil n/2m \rceil \Rightarrow f \geq \sqrt{100} \Rightarrow f \geq 10$$
- E o número de fitas?
  - Intercalação balanceada:  $2f$  (muito grande)
  - Alternativa:  $f+1$ 
    - Usar uma única fita de saída
    - Custo: uma cópia adicional a cada passada da fase de intercalação!
    - Como eliminar este custo adicional?

---

## Intercalação Polifásica

---

- Problema com a intercalação balanceada de vários caminhos:
  - Necessita de um grande número de fitas.
  - Faz várias leituras e escritas entre as fitas envolvidas.
  - Para uma intercalação balanceada de  $f$  caminhos são necessárias  $2f$  fitas.
  - Alternativamente, pode-se copiar o arquivo quase todo de uma única fita de saída para  $f$  fitas de entrada.
  - Isso reduz o número de fitas para  $f + 1$ .
  - Porém, há um custo de uma cópia adicional do arquivo.
- Solução:
  - **Intercalação polifásica.**

---

## 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 esvazie.
- Neste ponto, uma das fitas de saída troca de papel com a fita de entrada.

# Intercalação Polifásica

- Blocos iniciais criados na Seleção por Substituição:
  - Arquivo: INTERCALACAOBALANCEADA
  - f1: INRT AACEN
  - f2: ACEL AAD
  - f3: AABCLO
- Blocos iniciais criados na Intercalação Polifásica:
  - f1: INRT ACEL AABCLO
  - f2: AACEN AAD
  - f3:
- Fase de intercalação polifásica (nenhuma cópia direta)
  - f1: AABCLO (alguns blocos não são intercalados)
  - f2:
  - f3: AACEINNRT AAACDEL

---

## Intercalação Polifásica

---

- Exemplo:
  - Depois da 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*

- Finalmente:

fita 1: *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:

- A intercalação é realizada em muitas fases.
- As fases não envolvem todos os blocos.
- Nenhuma cópia direta entre fitas é realizada.



---

## Intercalação Polifásica

---

- A implementação da intercalação polifásica é simples.
- A parte mais delicada está na distribuição inicial dos blocos ordenados entre as fitas.
- Distribuição dos blocos nas diversas etapas do exemplo:

fita 1	fita 2	fita 3	Total
3	2	0	5
1	0	2	3
0	1	1	2
1	0	0	1

# Intercalação Polifásica (4 fitas)

f1	f2	f3	f4	Total
0	13	11	7	31
7	6	4	0	17
3	2	0	4	9
1	0	2	2	5
0	1	1	1	3
1	0	0	0	1

Menor num.  
Fibonacci  
generalizado  
(Tribonacci)  
maior que  
num. inicial de  
blocos.

---

## Intercalação Polifásica

---

### Análise:

- A análise da intercalação polifásica é complicada.
- O que se sabe é que ela é ligeiramente melhor do que a intercalação balanceada para valores pequenos de  $f$ .
- Para valores de  $f > 8$ , a intercalação balanceada pode ser mais rápida.

# Quicksort Externo

- Quicksort interno (paradigma de divisão e conquista):
  - Escolher um elemento pivot  $p$  da lista (rand/mediana)
  - Particionamento: reordenar a lista, tal que todos os elementos  $<p$  fiquem antes de  $p$ , e todos os elementos  $>p$  fiquem depois de  $p$ . (se  $=p$ , qq lado)
  - Recursivamente ordenar as sub-listas  $<p$  e  $>p$ .
- Quicksort externo (mesmo paradigma)
  - Igual ao quicksort interno, so que o pivot é substituído por um buffer de tamanho  $m$  ( $m=O(\log n)$ ,  $m \geq 3$ )
  - É um algoritmo *in situ*, ou seja, não necessita de memória externa adicional.
  - Os  $n$  registros a serem ordenados estão em memória secundária de acesso randômico.

---

## Quicksort Externo

---

- Foi proposto por Monard em 1980.
- Utiliza o paradigma de **divisão e conquista**.
- O algoritmo ordena *in situ* um arquivo  $A = \{R_1, \dots, R_n\}$  de  $n$  registros.
- Os registros estão armazenados consecutivamente em memória secundária de acesso randômico.
- O algoritmo utiliza somente  $O(\log n)$  unidades de memória interna e não é necessária nenhuma memória externa adicional.

---

## Quicksort Externo

---

- Seja  $R_i$ ,  $1 \leq i \leq n$ , o registro que se encontra na  $i$ -ésima posição de  $A$ .
- Algoritmo:
  1. Particionar  $A$  da seguinte forma:
$$\{R_1, \dots, R_i\} \leq R_{i+1} \leq R_{i+2} \leq \dots \leq R_{j-2} \leq R_{j-1} \leq \{R_j, \dots, R_n\},$$
  2. chamar recursivamente o algoritmo em cada um dos subarquivos  $A_1 = \{R_1, \dots, R_i\}$  e  $A_2 = \{R_j, \dots, R_n\}$ .

---

## Quicksort Externo

---

- Para o partionamento é utilizada uma área de armazenamento na memória interna.
- Tamanho da área:  $\text{TamArea} = j - i - 1$ , com  $\text{TamArea} \geq 3$ .
- Nas chamadas recursivas deve-se considerar que:
  - Primeiro deve ser ordenado o subarquivo de menor tamanho.
  - Condição para que, na média,  $O(\log n)$  subarquivos tenham o processamento adiado.
  - Subarquivos vazios ou com um único registro são ignorados.
  - Caso o arquivo de entrada  $A$  possua no máximo  $\text{TamArea}$  registros, ele é ordenado em um único passo.

# Quicksort Externo

1. Lê os primeiros  $m/2$  e os últimos  $m/2$  elementos para o buffer e ordena
2. Guarda  $\max(\text{buffer})$  e  $\min(\text{buffer})$  para evitar reordenar elementos no meio q já estão escritos
3. Lê o próximo elemento  $x$  do começo ou do fim, alternadamente, de forma a balancear a escrita
4. Se  $x \leq \min(\text{buffer})$ , escreve  $x$  no espaço liberado no começo do arquivo de entrada  
Se  $x \geq \max(\text{buffer})$ , escreve  $x$  no fim do arquivo  
Senão, escreve  $\min(\text{buffer})$  ou  $\max(\text{buffer})$  e coloca  $x$  no buffer
5. Ao terminar, escrever o conteúdo do buffer
6. Recursivamente, ordena(partição menor); ordena(partição restante)



# Quicksort Externo

- Exemplo: ordenar  $A\{5, 3, 10, 6, 1, 7, 4\}$ ,  $m=3$ 
  1. Lê  $\{4,5,7\}$ , ordene em memória:  $B\{4,5,7\}$ ,  $\text{min}=4, \text{max}=7$
  2. Lê  $\{3\}$ : ( $3 \leq \text{min}$ )  $\Rightarrow$  escreve 3:  $A\{\textcolor{red}{3}, \textcolor{green}{3}, 10, 6, 1, \textcolor{green}{7}, \textcolor{green}{4}\}$
  3. Lê  $\{1\}$ : ( $1 \leq \text{min}$ )  $\Rightarrow$  escreve 1:  $A\{\textcolor{red}{3}, \textcolor{red}{1}, 10, 6, \textcolor{green}{1}, \textcolor{green}{7}, \textcolor{green}{4}\}$
  4. Lê  $\{10\}$ : ( $10 \geq \text{max}$ )  $\Rightarrow$  escreve 10:  $A\{\textcolor{red}{3}, \textcolor{red}{1}, \textcolor{green}{10}, 6, \textcolor{green}{1}, \textcolor{green}{7}, \textcolor{red}{10}\}$
  5. Lê  $\{6\}$ : ( $\text{min} < 6 < \text{max}$ )  $\Rightarrow$  escreve max (p/ balancear), coloca  $\{6\}$  no buffer:  
 $A\{\textcolor{red}{3}, \textcolor{red}{1}, \textcolor{green}{10}, \textcolor{green}{6}, \textcolor{green}{1}, \textcolor{red}{7}, \textcolor{red}{10}\}$ ,  $B\{4,5,6\}$ ,  $\text{min}=4, \text{max}=6$
  6. Escreve o buffer no arquivo:  $A\{3, 1, \textcolor{blue}{4}, \textcolor{blue}{5}, \textcolor{blue}{6}, 7, 10\}$
  7. Recursão: ordena  $A\{3, 1\}$ ; ordena  $A\{7,10\}$ ;

# Quicksort Externo

- Exemplo: ordenar  $A\{5, 3, 10, 6, 1, 7, 4\}$ ,  $m=3$ 
  1. Lê  $\{5,4,3\}$ , ordene em memória:  $B\{3,4,5\}$ ,  $\text{min}=3, \text{max}=5$
  2. Lê  $\{7\}$ :  $(7 \geq \text{max}) \Rightarrow$  escreve 7:  $A\{5, 3, 10, 6, 1, 7, 7\}$
  3. Lê  $\{1\}$ :  $(1 \leq \text{min}) \Rightarrow$  escreve 1:  $A\{1, 3, 10, 6, 1, 7, 7\}$
  4. Lê  $\{10\}$ :  $(10 \geq \text{max}) \Rightarrow$  escreve 10:  $A\{1, 3, 10, 6, 1, 10, 7\}$
  5. Lê  $\{6\}$ :  $(6 \geq \text{max}) \Rightarrow$  escreve 6:  $A\{1, 3, 10, 6, 6, 10, 7\}$
  11. Escreve o buffer no arquivo:  $A\{1, 3, 4, 5, 6, 10, 7\}$
  12. Recursão: ordena  $A\{1\}$ ; ordena  $A\{6,10,7\}$ ;

---

## Quicksort Externo: Análise

---

- Seja  $n$  o número de registros a serem ordenados.
- Seja  $e$  e  $b$  o tamanho do bloco de leitura ou gravação do Sistema operacional.
- Melhor caso:  $O(\frac{n}{b})$ 
  - Por exemplo, ocorre quando o arquivo de entrada já está ordenado.
- Pior caso:  $O(\frac{n^2}{\text{TamArea}})$ 
  - ocorre quando um dos arquivos retornados pelo procedimento Particao tem o maior tamanho possível e o outro é vazio.
  - A medida que  $n$  cresce, a probabilidade de ocorrência do pior caso tende a zero.
- Caso Médio:  $O(\frac{n}{b} \log(\frac{n}{\text{TamArea}}))$ 
  - É o que tem a maior probabilidade de ocorrer.