GBC053–Gerenciamento de Banco de Dados
Ordenação Externa

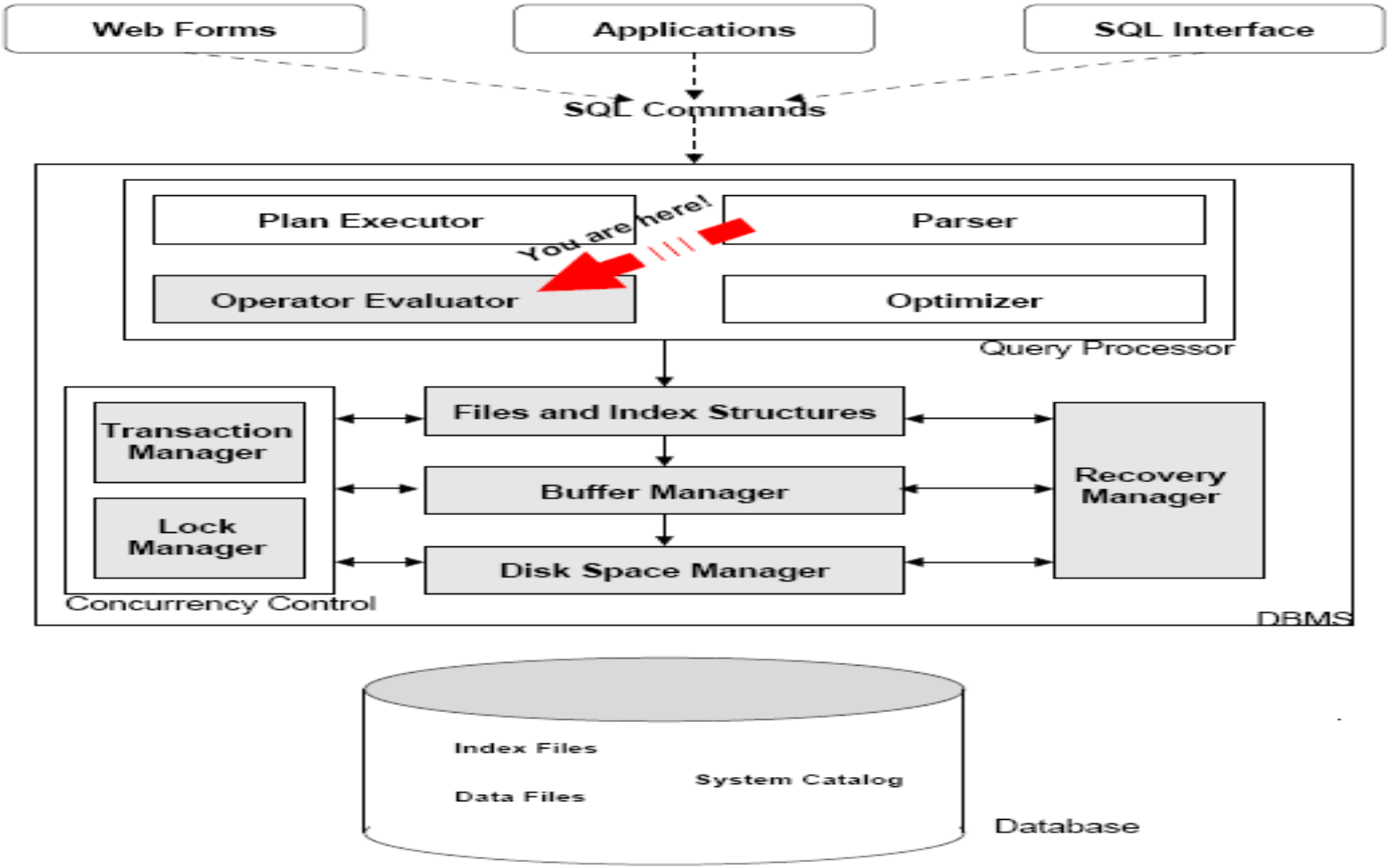
Ilmério Reis da Silva
ilmerio@facom.ufu.br
www.facom.ufu.br/~ilmerio/gbd
UFU/FACOM/BCC

Roteiro

- ***Fundamentos***
- ***Two-way Merge Sort***
- ***External Merge Sort***
- ***Arvore-B+ e Sort***

Fundamentos

Contexto


Web Forms
Applications
SQL Interface
SQL Commands
Plan Executor
You are here!
Parser
Operator Evaluator
Optimizer
Query Processor
Transaction Manager
Lock Manager
Concurrency Control
Files and Index Structures
Buffer Manager
Disk Space Manager
Recovery Manager
DBMS
Index Files
Data Files
System Catalog
Database

Sorting é parte do executor de consultas

- ***Estudamos arquivos, buffers e índices***
- ***Vamos estudar execução de consultas***
- ***Antes de estudar os vários operadores da álgebra relacional***
- ***Antes de estudar como esses operadores são combinados em um plano de consulta***
- ***Um operador básico de grande importância é:***

EXTERNAL SORT

Aplicações do operador SORT

- ***Uso explícito:***

  SELECT a, b, c

  FROM r

  ORDER BY a, b

- ***Bulk-Loading (construção bootom-up de Árvore B+)***

- ***Eliminação de duplicatas***

  SELECT DISTINCT a, b, c

  FROM r

- ***Melhorias no desempenho de alguns operadores, por exemplo, junção***

Definição Arquivo Ordenado

Um arquivo de registros está ordenado (sorted) com respeito a uma chave de ordenação (sort key) e uma ordem θ, se para todo par de registros r_1, r_2 com r_1 precedendo r_2 no arquivo, então suas chaves correspondentes estão na ordem θ

$$r_1 \text{ precede } r_2 \leftrightarrow r_1.k \ \theta \ r_2.k$$

Abordagem para o Problema

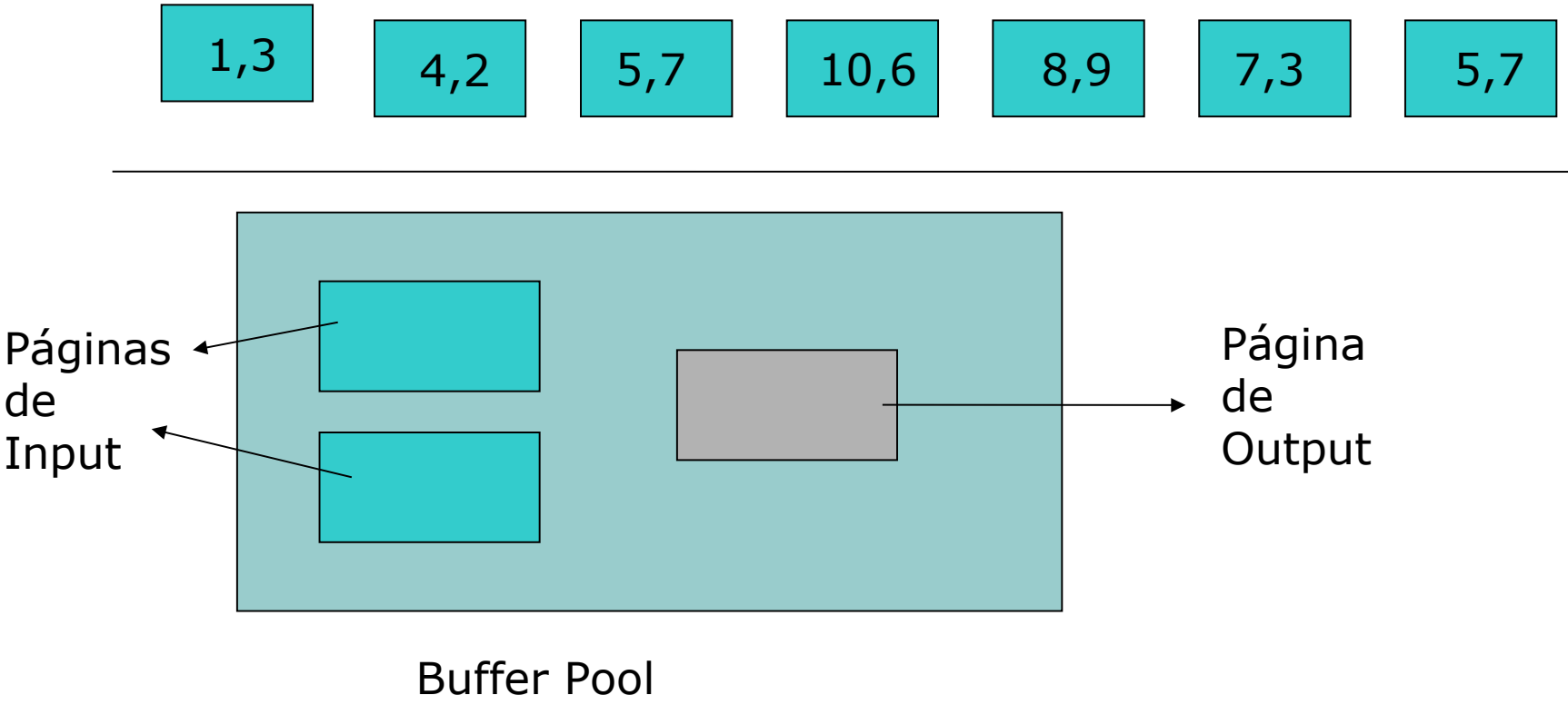
- ***O Problema:***

o arquivo não cabe na memória

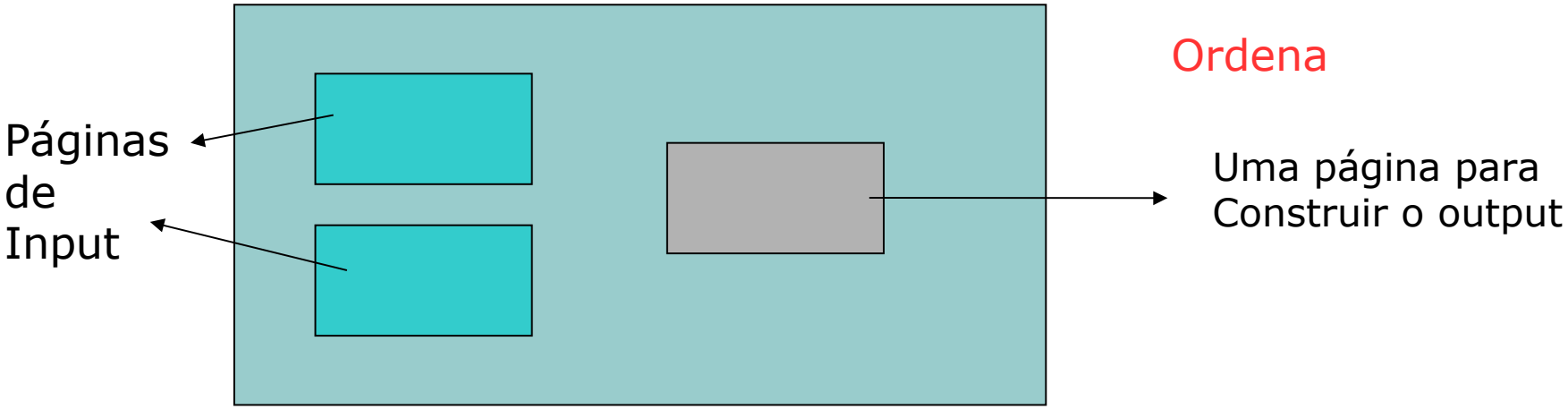
- ***Abordagem***
  - Ordenar um arquivo de tamanho arbitrário considerando três páginas disponíveis no buffer *(Two-way Merge Sort)*
  - Refinar o algoritmo com um uso mais efetivo do buffer *(External Merge Sort)*
  - *Melhorias (Replacement Sort)*

Two-Way Merge Sort

Two-way Merge Sort

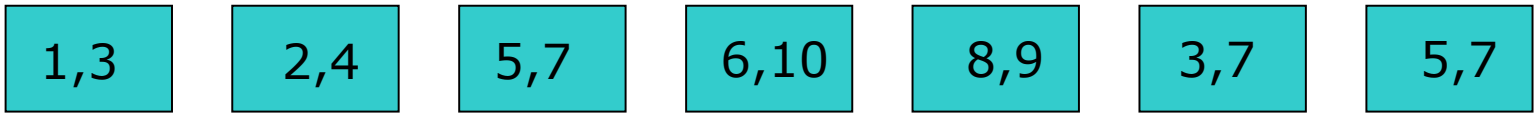


Etapa 0 – Ordena registros em cada página

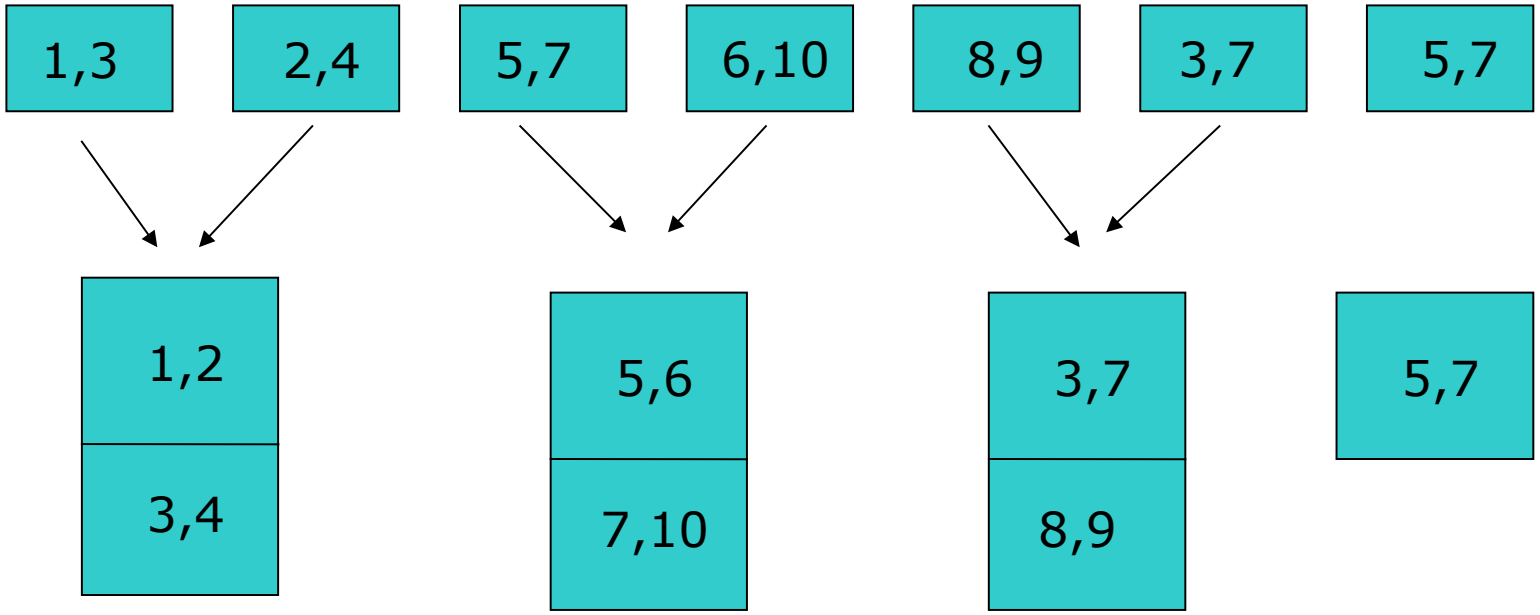


Produz 7 subarquivos ordenados

/ 7*2 I/O

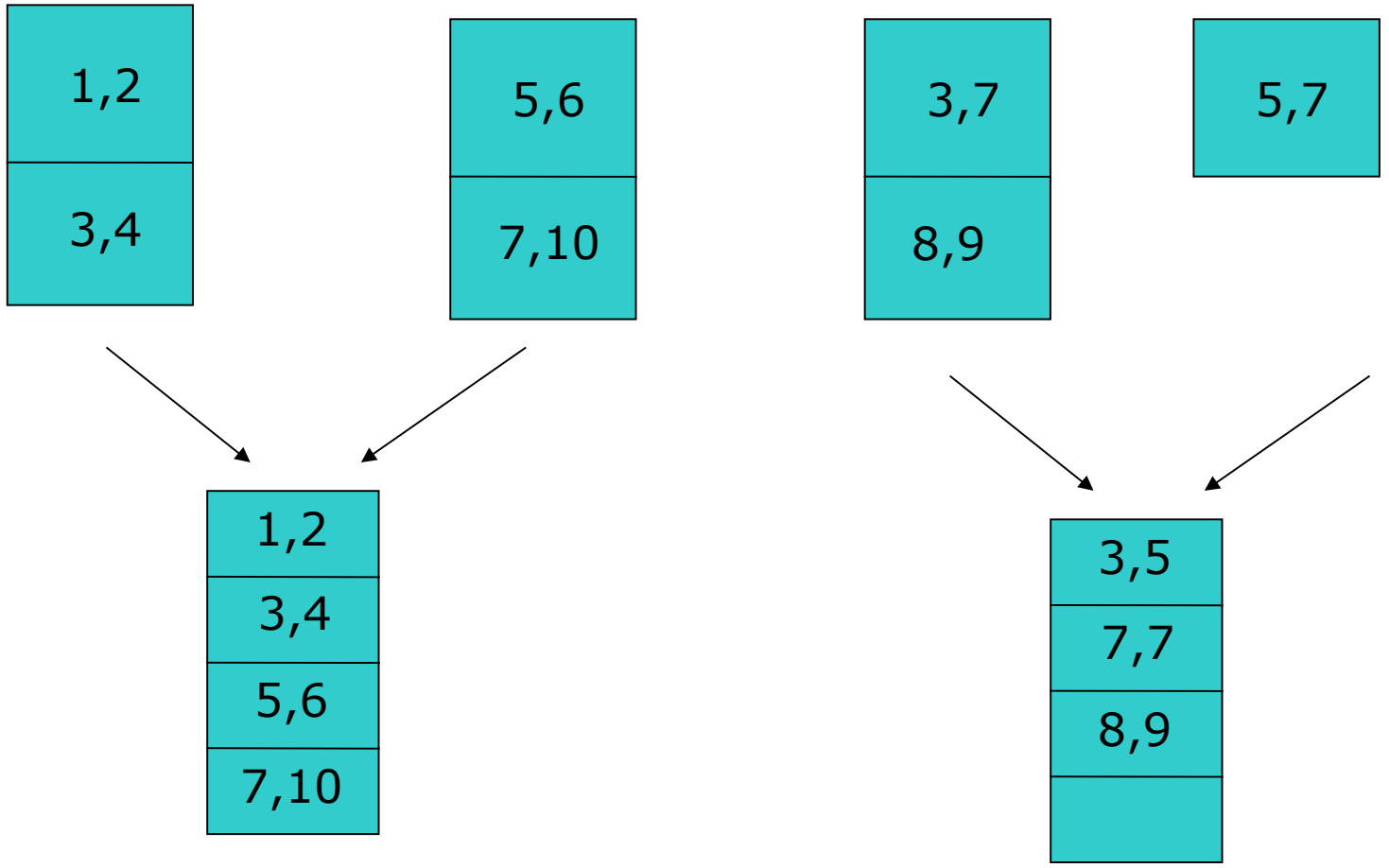


Etapa 1 – Merge de pares de páginas



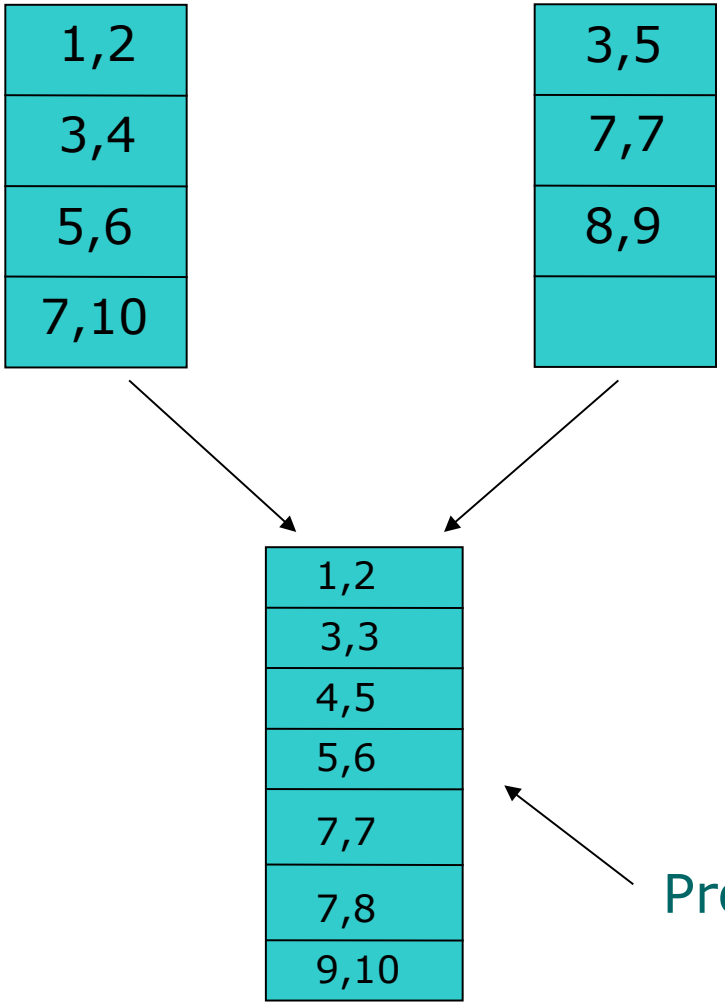
Produz 4 subarquivos ordenados **7 x 2 I/O**

Etapa 2 – Ordena pares de subarquivos ordenados



Produz 2 subarquivos ordenados **7 x 2 I/O**

Etapa 3 – Ordena par de subarquivos ordenados



7 x 2 I/O

Análise – número de etapas

- ***Seja N o número de páginas do arquivo***
- ***Considere que $N = 2^s$***
  - Etapa 0 : 2^s subarquivos ordenados
  - Etapa 1 : 2^{s-1} subarquivos ordenados
  - Etapa 2 : 2^{s-2} subarquivos ordenados
  - ...
  - Etapa s : 1 arquivo ordenado
  - Total de etapas = $s+1 = \log_2 N + 1$
- ***Genericamente***

$$\underbrace{1}_{\text{Pass 0}} + \underbrace{\lceil \log_2 N \rceil}_{\text{Passes } 1 \ldots s}.$$

Análise – Custo em IOs

- ***Número de etapas = $log_2 N + 1$***
- ***Número de I/O por etapa = 2N***
- ***Total de I/O = 2N(func_teto($log_2 N$) + 1)***
- ***O(N log N)***

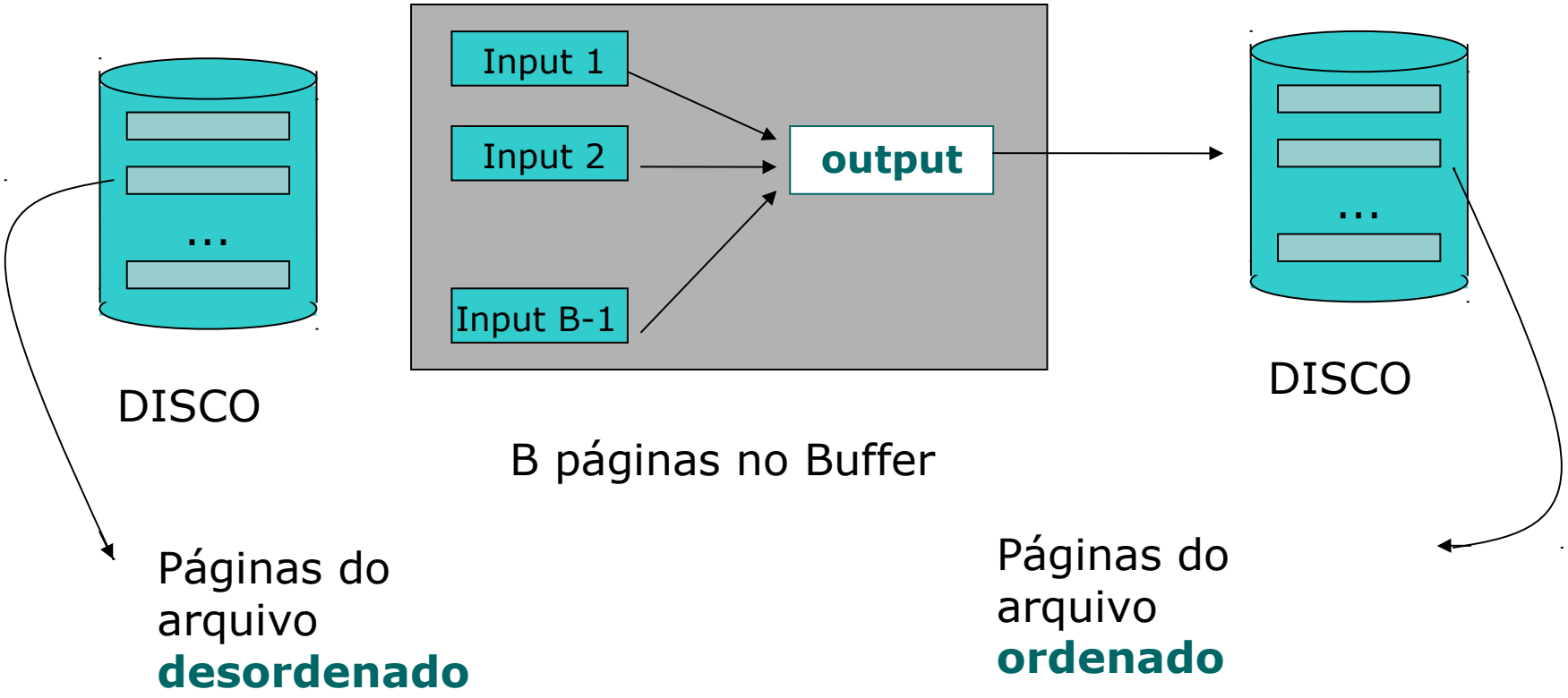
External Merge Sort

External Merge Sort

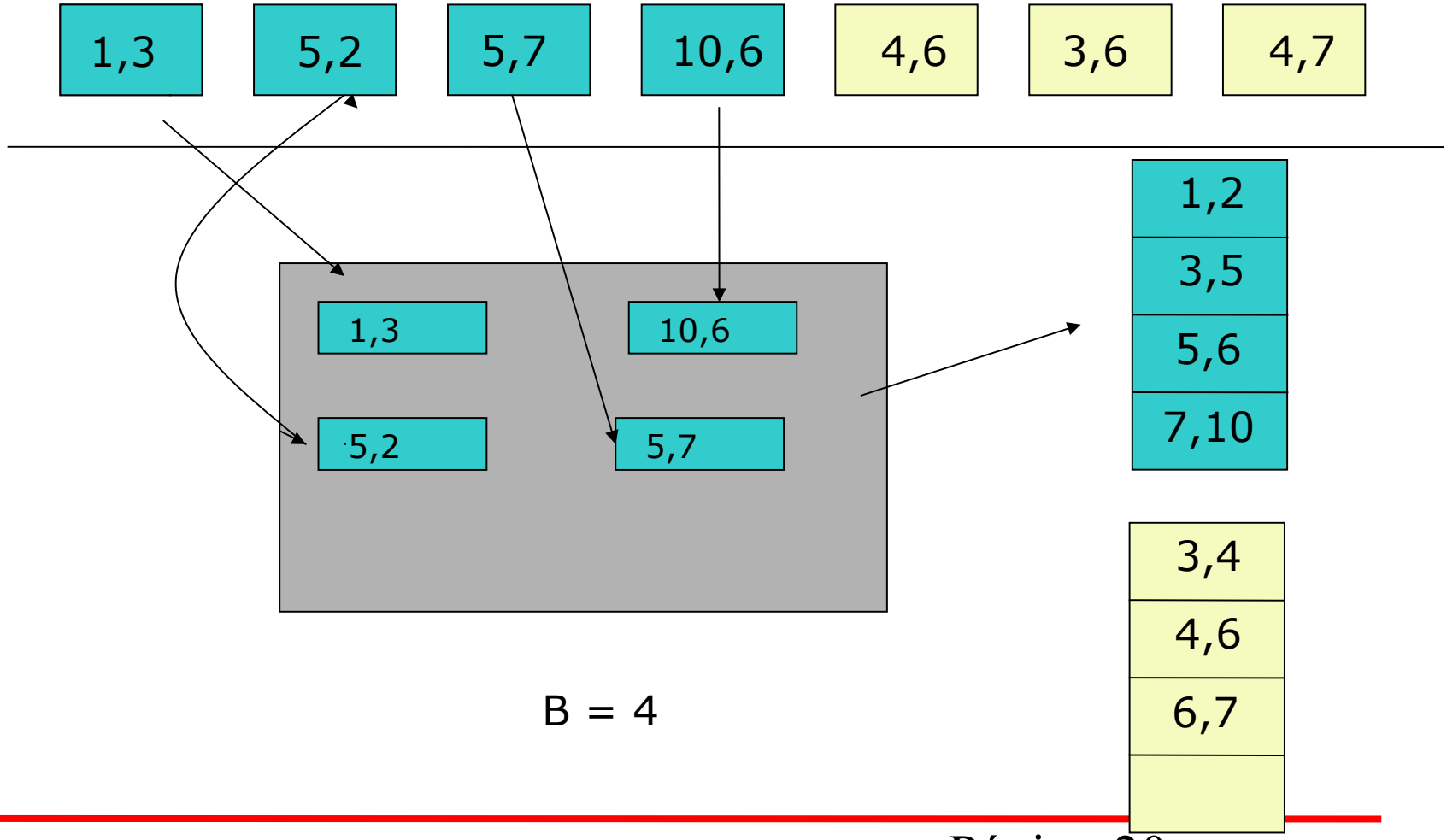
External Merge Sort

- ***Buffer com B páginas***
- ***Etapa 0 :***
  - B páginas são carregadas no buffer, ao invés de uma a uma.
  - B páginas são ordenadas e são criados N/B arquivos ordenados.
- ***Etapas i=1,2,...***
  - B-1 páginas são utilizadas no buffer
  - 1 página é usada para construir o output.

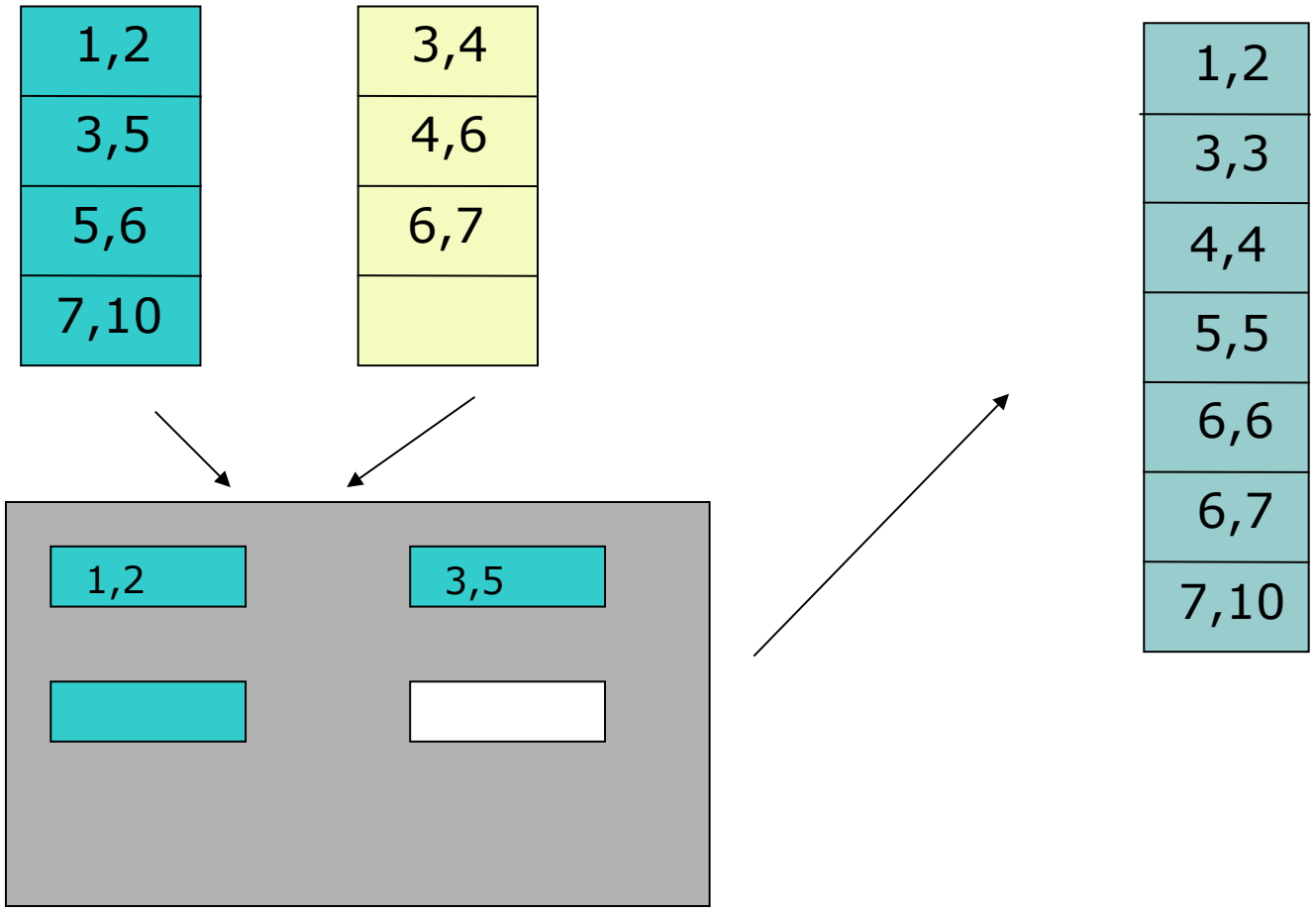
Esquema de utilização do buffer no External Merge Sort – Etapas 1, 2, ...



Etapa 0 – External Merge Sort



Etapa 1 – External Merge Sort



Análise Custo External Merge Sort

- ***Número de arquivos produzidos na etapa 0 = N/B = N1***
- ***Número de etapas = $\log_{B-1} N1 + 1$***
- ***Número de I/O por etapa = 2N***

- ***Total de I/O = $2N(\log_{B-1} N1 + 1)$ ou***

$$2 \cdot N \cdot \left(1 + \lceil \log_{B-1} \lceil N/B \rceil \rceil \right)$$

Exemplo

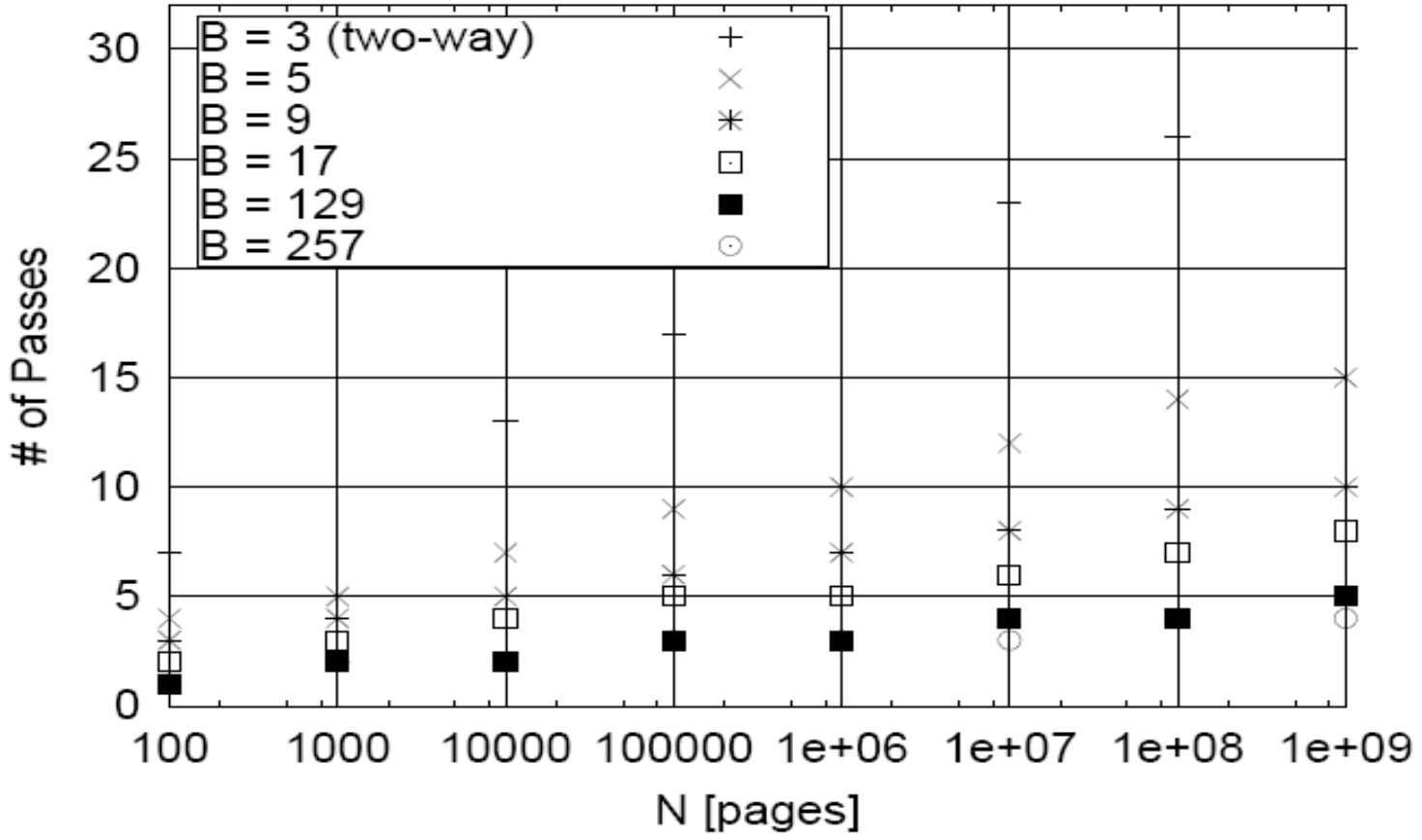
- ***B = 5***
- ***N = 108 páginas***
- ***Etapa 0 : 108/5 = 22 arquivos,***
  - 21 de 5 páginas e 1 de 3 páginas
- ***Etapa 1 : 22/4 = 6 arquivos***
  - 5 de 20 páginas e 1 de 8 páginas
- ***Etapa 2 : 6/4 = 2 arquivos***
  - 1 de 80 páginas e 1 de 28 páginas
- ***Etapa 3 : 1 arquivo ordenado de 108 pág***
- ***Total de I/O = 2*108*4 = 864***
- $2*108(\log_4 22 + 1) = 864$

Comparação de Custos : n° de etapas

N	B=3	B=5	B=9	B=17	B=129	B=257
100	7	4	3	2	1	1
1.000	10	5	4	3	2	2
10.000	13	7	5	4	2	2
100.000	17	9	6	5	3	3
1.000.000	20	10	7	5	3	3
10.000.000	23	12	8	6	4	3
100.000.000	26	14	9	7	4	4
1000.000.000	30	15	10	8	5	4

Numero de I/O = etapas * 2N

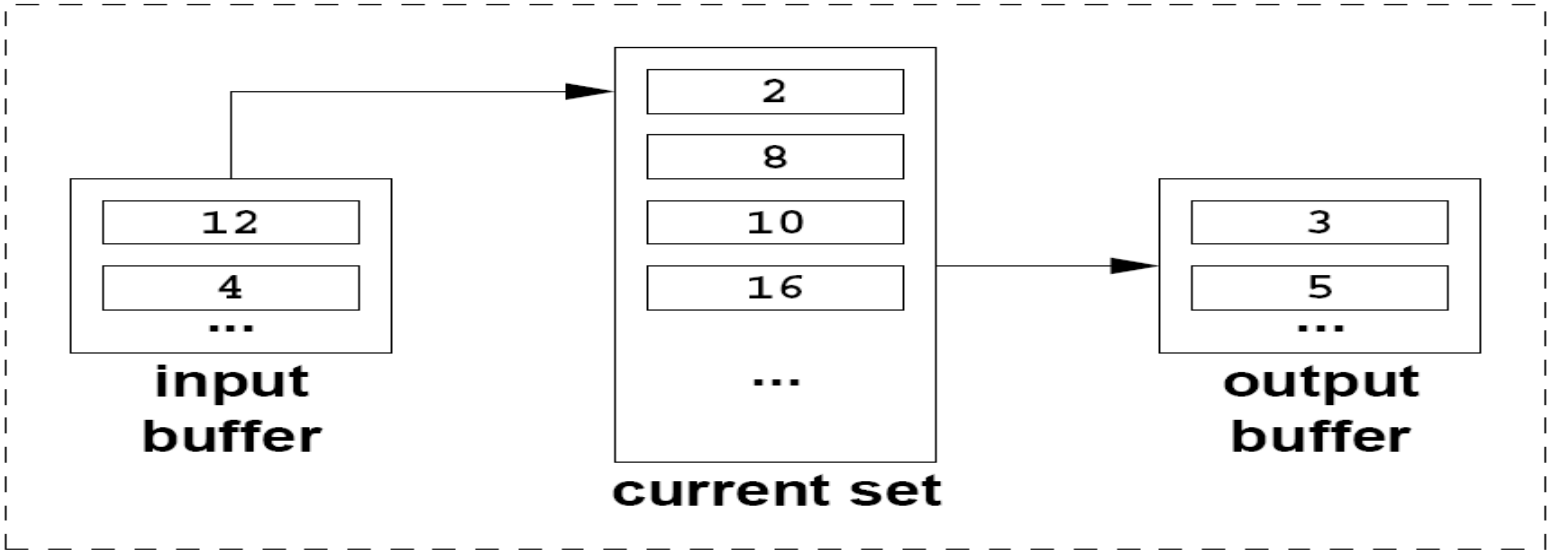
Comparação com Two-way



Melhorias : minimizando o número de subarquivos do passo 0

Replacement Sort

- ***Seja B o número de páginas do bufferpool***
- ***Reserve uma para entrada(Inp) e outra para saída(Out)***
- ***B-2 páginas formarão o current-set (CSet)***
- ***A cada rodada grava-se um current-run(CRun)***



Loop de uma rodada do Replacement Sort

SE $\exists$ *tupla* $t \in CSet \mid t.k \geq t'.k \ \forall \ t' \in CRun$

PEGUE $t \in CS$ *de menor*$(t.k) \mid t.k \geq t'.k \ \forall \ t' \in CRun$

APPEND t **em** Out

TRANSFIRA tupla de Inp **para** $CSet$

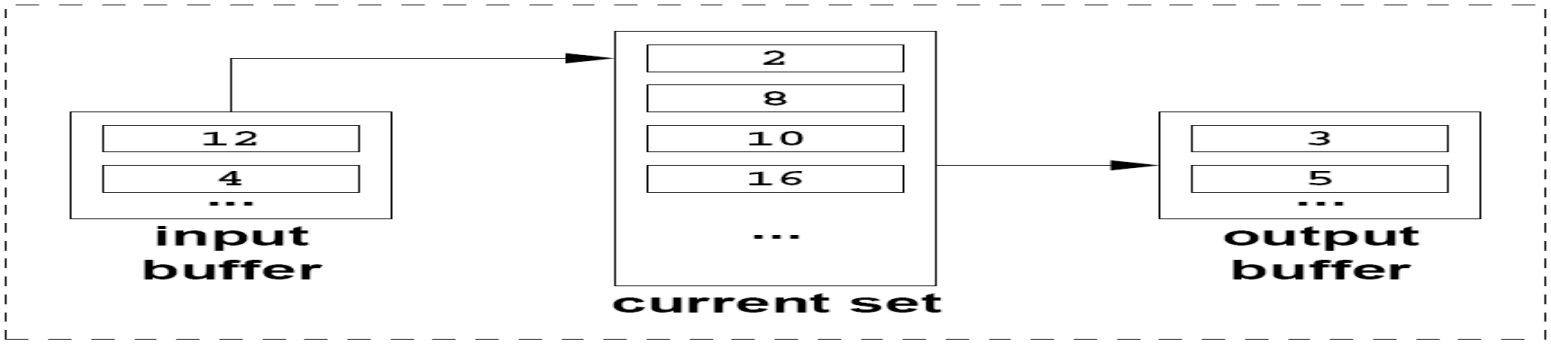
(Leia nova página em Inp **quando esvaziar)**

(Grave Out **em** $CRun$ **quando encher)**

SENÃO

Grave Out **como última página do** $CRun$

Inicie uma nova rodada i.e., novo $CRun$



Replacement Sort - Considerações finais

- ***Registros de tamanho variável dificultam o processo.***
- ***Seleção da menor chave do CSet pode usar uma estrutura de dados, por exemplo, um max Heap (árvore onde todo filho tem chave menor ou igual ao pai)***
- ***Em média o tamanho do subarquivo de cada rodada é igual a 2B***
- ***Exemplo: B = 6, 1 reg/pg. Logo temos 4 reg no Cset***

503 087 512 061 908 170 897 275 426 154 509 612 700

Outras melhorias no Sort Externo

Diminuindo do Custo Médio de Cada IO

- ***IO baseado em blocos de várias páginas pode melhorar o tempo médio de cada IO (diminui o número de seeks), mas pode aumentar número de passos***
- ***Double Buffering pode ser usado para paralelizar trabalho da CPU enquanto aguarda requisição de IO***

Sort Externo com IO baseado em Blocos

- ***unidade de leitura e gravação = bloco com b páginas***
- ***diminui número de seeks, portanto o custo médio do IO***
- ***b páginas para Out***
- ***merge de no máximo (B−b)/b = B/b -1***
- ***é o sort merge externo com b páginas de output e merge de (B−b)/b subarquivos por passo***
- ***Exemplo:B = 10***
  - *SE b = 1, merge de nove subarquivos por passo*
  - *SE b = 2, merge de 4 subarquivos por passo*
- ***aumenta-se o número de pasos, portanto o número de IOs***
- ***melhoria depende de b, tamanho do arquivo, características de desempenho do disco.***

Análise Sort Externo com IO baseado em Blocos

Análise incluindo Replacement Sort no Passo 0

- *Passo 0: produz N2 = N/(2B) subarquivos*
- *Passos 1, 2, ...: merge F = B/b -1 subarquivos por vez*
- *NroPassos = 1+ $\log_F N2$, sendo que originalmente seria $NroPasso_{Merge\ Sort\ Original} = 1 + \log_{B-1} N/B$*

N	B=1,000	B=5,000	B=10,000
100	1	1	1
1,000	1	1	1
10,000	2	2	1
100,000	3	2	2
1,000,000	3	2	2
10,000,000	4	3	3
100,000,000	5	3	3
1,000,000,000	5	4	3

Número de passos em função de N e B, usando $b = 32$

Exemplo Sort Externo com IO baseado em Blocos

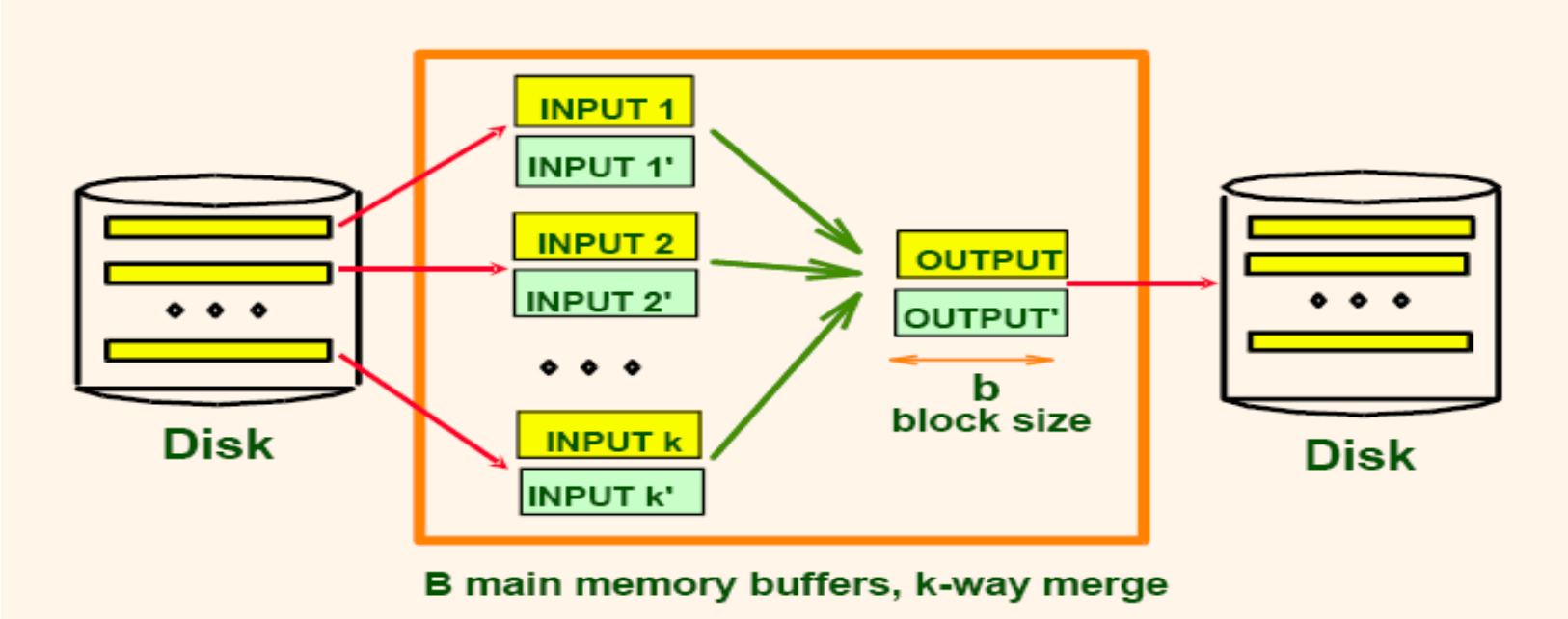
Exemplo: N = 1.000.000, B = 5000, b = 32,

- *N2 = piso(1.000.000/ (2×5.000)) = 100*
- *F = piso(5.000/32) – 1 = 155*
- *NroPassos = 1 + teto($\log_{155}$ 100) = 2*

- ***Considerações finais***
  - *Necessita maior área de bloco para manter número de passos semelhante ao algoritmo original*
  - *Logo pode haver aumento do número de passos e de I/O*
  - *Em cada I/O o custo em tempo será menor, devido ao tamanho dos blocos de leitura e gravação.*

Double Buffering

- ***considerando que a CPU é bem mais rápida que o IO***
- ***mantem CPU ocupada enquanto aguarda requisição de IO***
- ***para isso faz prefetch de páginas shadow, uma para cada página de input, além de página shadow para output (figura)***



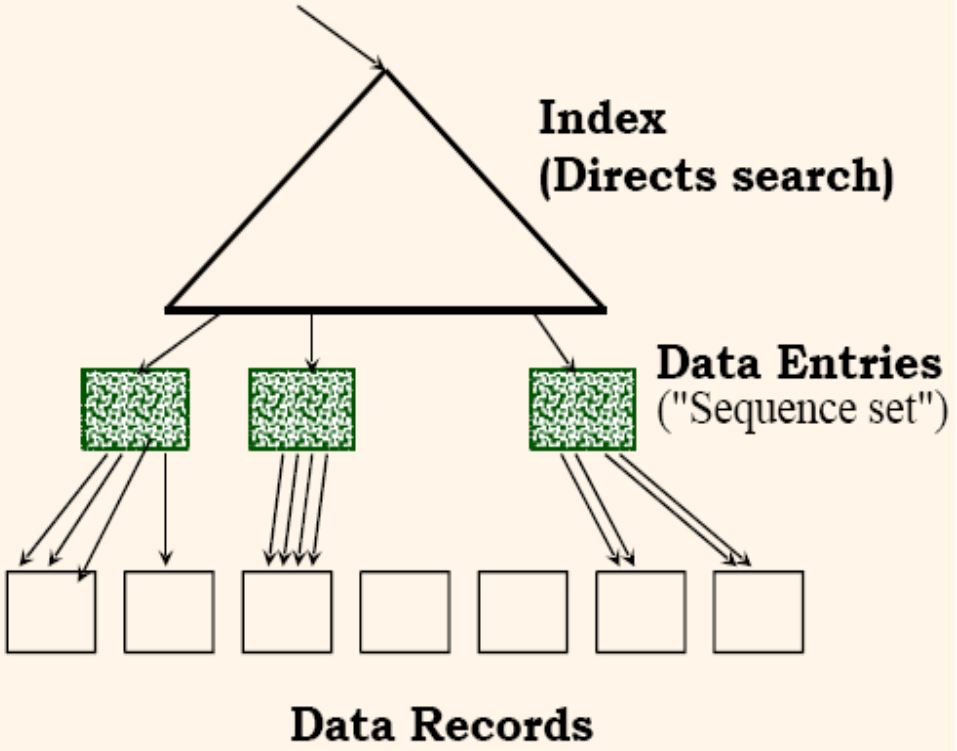
Árvore B+ como alternativa à ordenação

Árvore B+ como alternativa à ordenação

- *se existe índice baseado em árvore B+, considere utilizá-la em substituição à ordenação externa*
- *Idéia: recuperar registros percorrendo as folhas*
- *Considerações:*
  - *Se a Árvore B+ é agrupada trata-se de uma boa idéia*
  - *Senão é perigoso e pode ser uma idéia muito ruim*

Árvore B+ Agrupada como alternativa à ordenação

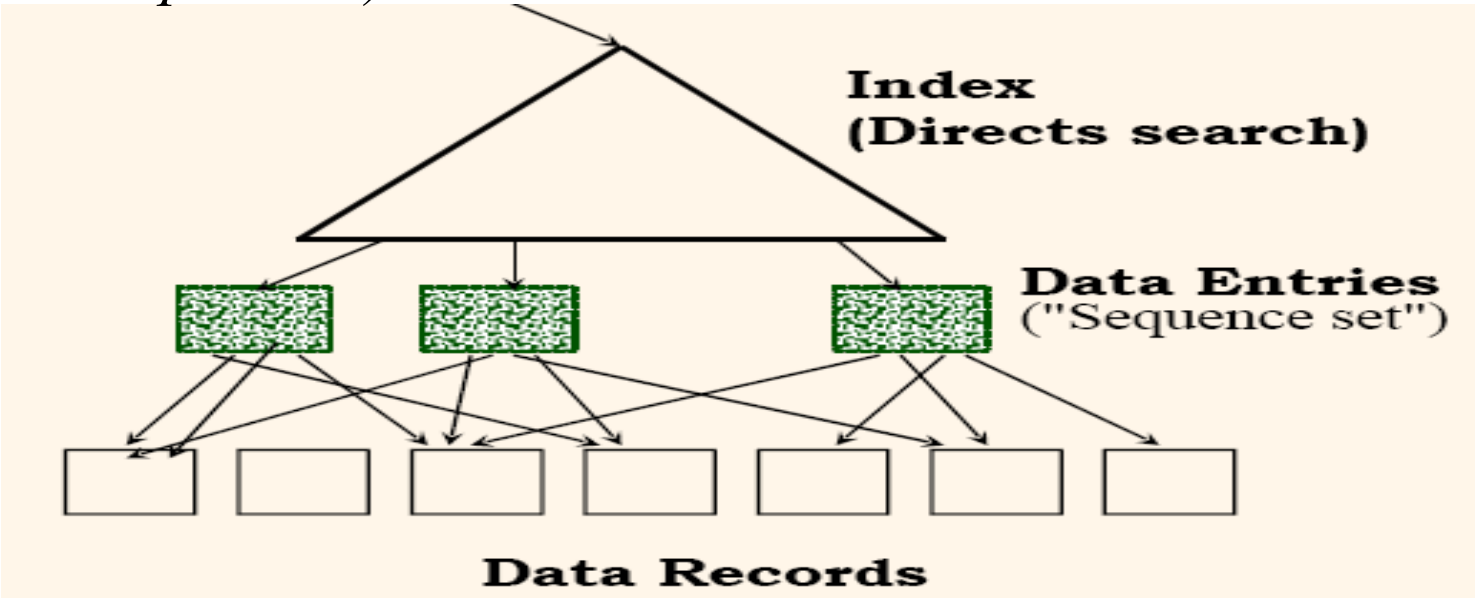
- Custo: vai da raiz para a folha mais a esquerda, então recupera todas páginas folhas (Alternativa 1)
- Se Alternativa 2 é usada? Custo adicional para recuperar registros de dados: Mas cada página é trazida somente uma vez.



☛*Sempre melhor que classificação externa!*

Árvore B+ Não Agrupada como alternativa à ordenação

- *Custo (alternativa 2)*
- *Raiz até a folha mais à esquerda*
- *Percorre sequencial set e para cada entrada*
  - *Busca página do data set (um IO por registro => pode ser péssimo)*



Comparação Árvore B+ Não Agrupada x ordenação

N	Sorting	p=1	p=10	p=100
100	200	100	1,000	10,000
1,000	2,000	1,000	10,000	100,000
10,000	40,000	10,000	100,000	1,000,000
100,000	600,000	100,000	1,000,000	10,000,000
1,000,000	8,000,000	1,000,000	10,000,000	100,000,000
10,000,000	80,000,000	10,000,000	100,000,000	1,000,000,000

- *p*: número médio de registros por página
- *B=1,000 e tamanho do bloco =32 para classificação*
- *p=100 é o valor mais real.*

Considerações finais sobre classificação externa

- *É importante no processamento de consultas*
- *External Merge Sort é a solução mais usada para minimizar custo de IO*
  - *Passo 0: subarquivos ordenados de tamanho B (ou 2B com replacement sort)*
  - *Passos seguinte: merge de subarquivos com número dependente de B e do tamanhos dos blocos*
    - ✓*Maiores blocos minimizam custo de cada IO*
    - ✓*Menores blocos minimizam número de passos*
- *Árvore B+ agrupada é uma boa alternativa, mas a não agrupada pode ser muito ruim.*

Exercícios – Ordenação Externa

EXERCÍCIOS CAP 13 LIVRO-TEXTO

*FIM - Ordenação Externa**

**** material baseado no livro-texto e páginas públicas da internet***