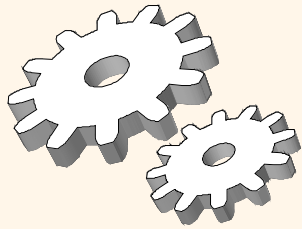
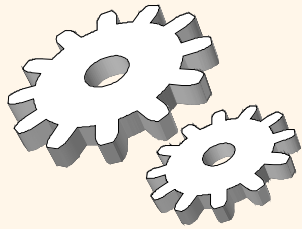


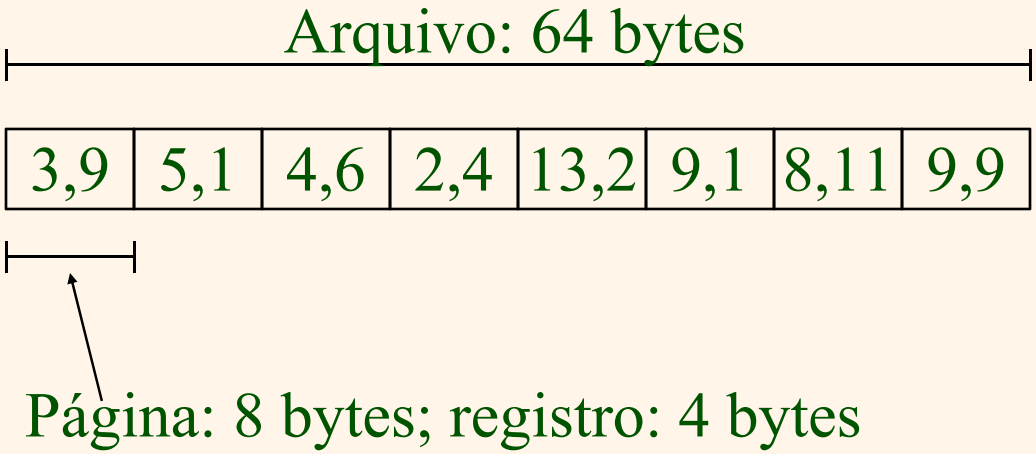
# *Classificação Externa*

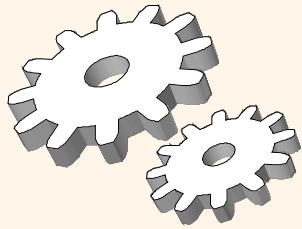
## Capítulo 13



# *Arquivos no SGBD*

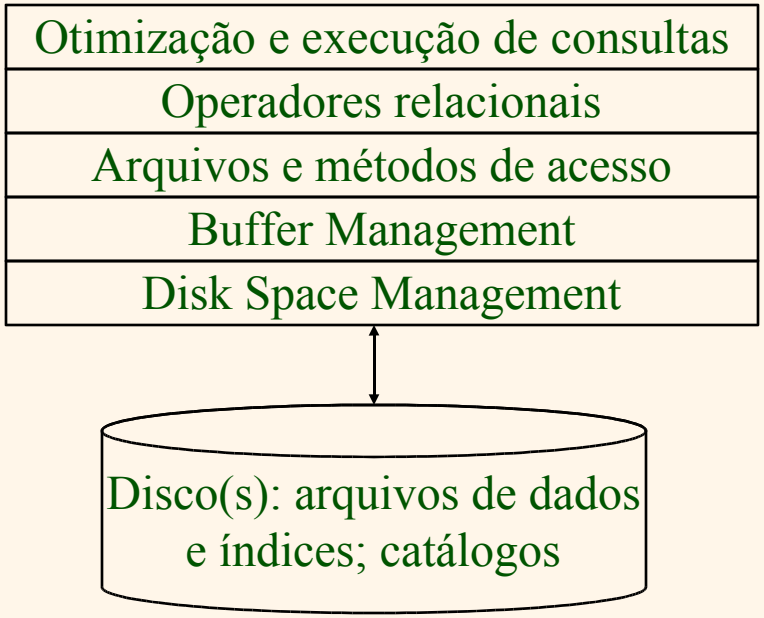
- Arquivos no sistema operacional: seqüência de bytes
- S.O. pode não disponibilizar operações necessárias
- Abstração para o SGBD: arquivos são seqüências de registros, organizados em páginas

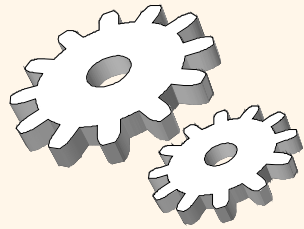




# *Arquivos no SGBD*

- E/S em unidades de página
- Custo de E/S tende a dominar sobre CPU
- Modelo de custo é o número de operações de E/S
- Arquitetura do SGBD



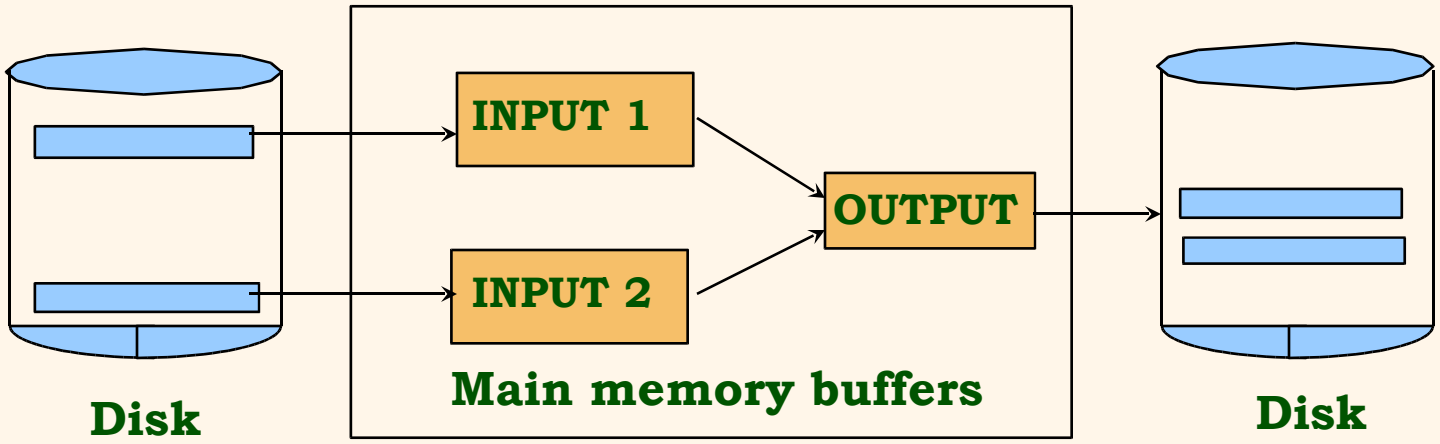


# *Por que Classificar?*

- Um problema clássico na ciência da computação!
- Dados são solicitados  em ordem
  - Por exemplo, procurar estudantes em ordem crescente de gpa
- Classificar é o primeiro passo para *bulk loading de Árvore* B+.
- Classificação é útil para eliminar  *cópias duplicadas* em um conjunto de registros (Por que?)
- *Sort-merge* algoritmo de junção requer classificação.
- Problema: Classificar 1Gb de dados com 1Mb de RAM.
  - Por que não usar memória virtual?

# *2- Way Sort: Requer 3 Buffers*

- Passo 1: Leia uma página, classifique-a e a escreva.
  - Somente uma página do  buffer é utilizada
- Passo 2, 3, …, etc.:
  - três páginas de buffer são utilizadas



# 2-Way External Merge Sort

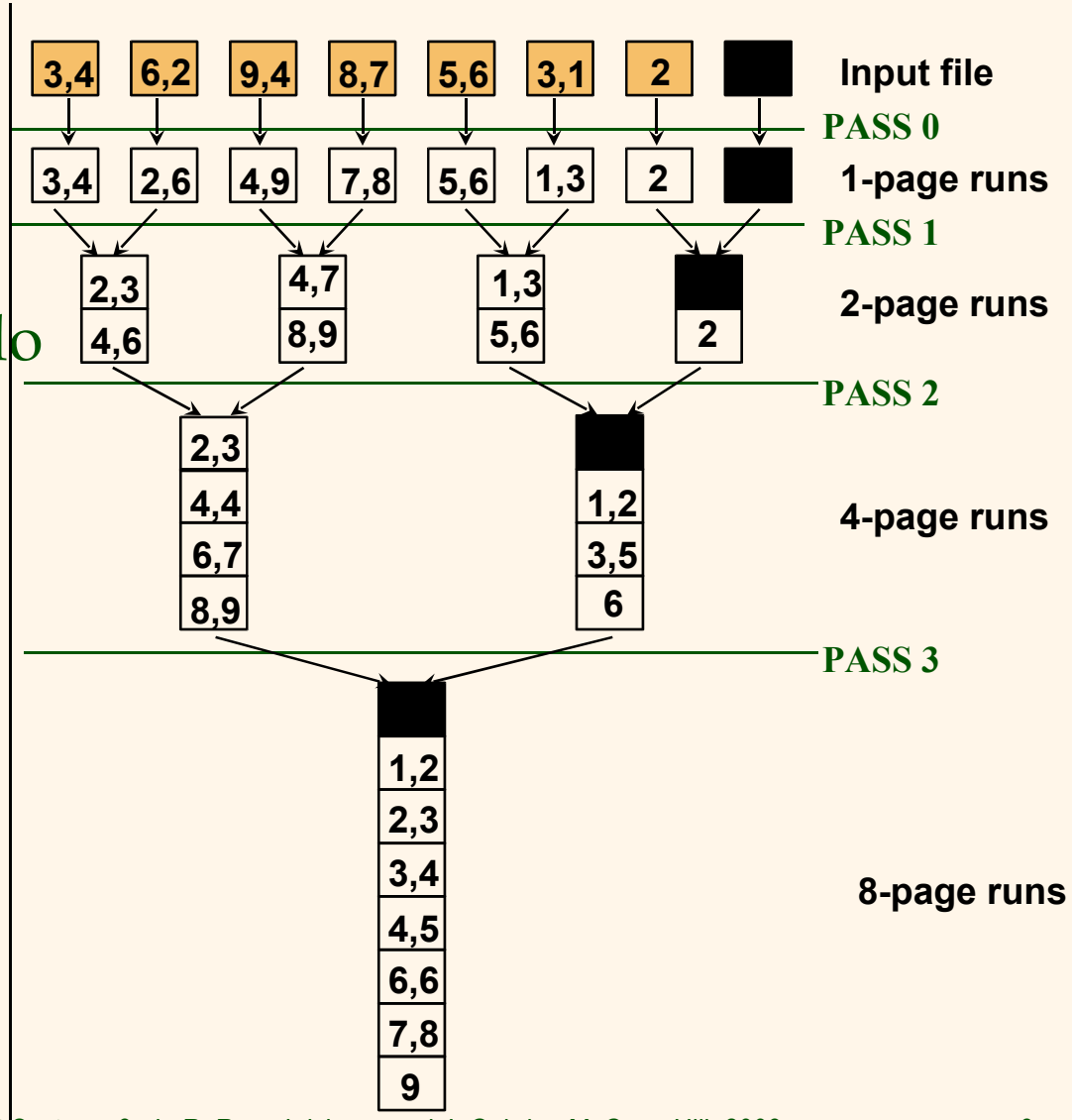
- Cada passo fazemos uma leitura + escrita de cada página no arquivo.
- N é o número de páginas do arquivo => o numero de passos

$$= \lceil \log_2 N \rceil + 1$$

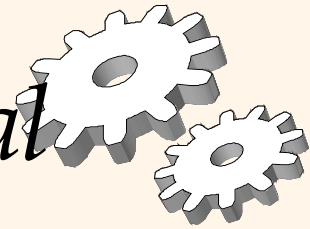
Assim o custo total é:

$$2N\left(\lceil \log_2 N \rceil + 1\right)$$

- *Idéia:* ***Dividir e conquistar:*** Classificar arquivos e intercalar

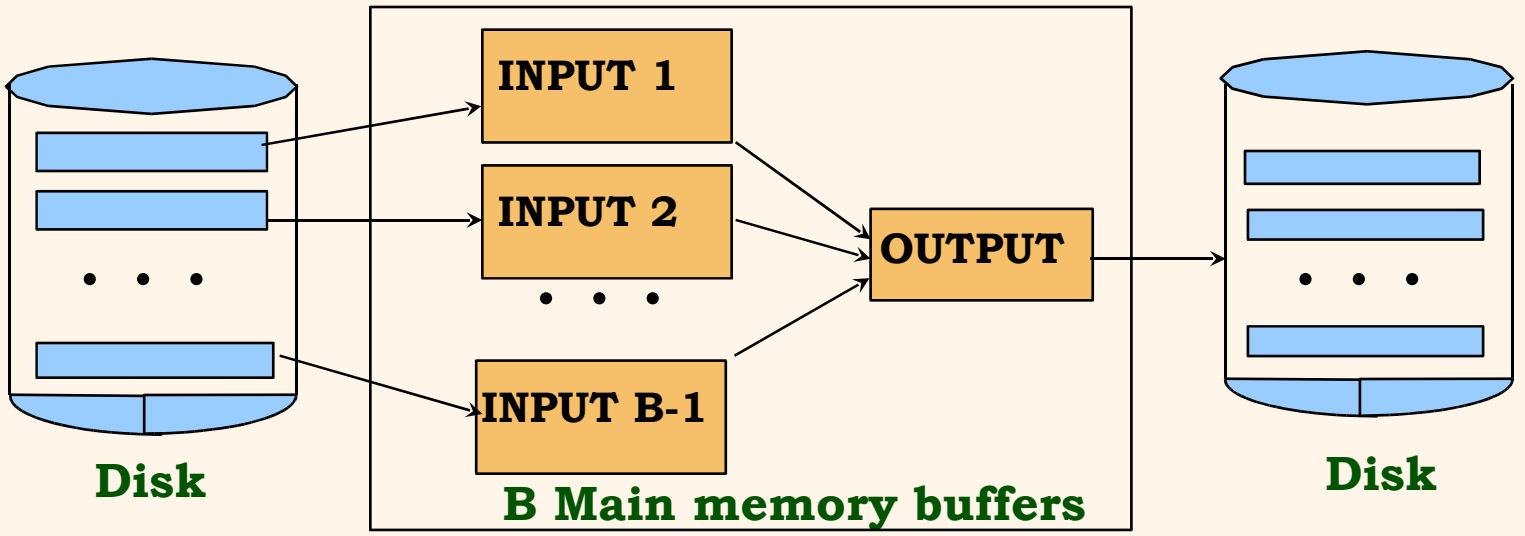


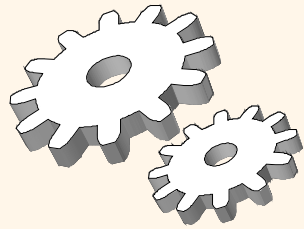
# External Merge Sort – Caso Geral



**Mais de 3 páginas no Buffer. Como nós podemos utiliza-las ?**

- Para classificar um arquivo com *N* páginas usando um buffer de B páginas:
  - Passo 0: Use B páginas do buffer. Produzimos $\lceil N / B \rceil$ runs ordenados de *B* páginas cada.
  - Passo 2, …, etc.: Merge *B-1 runs.*

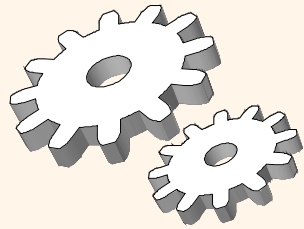




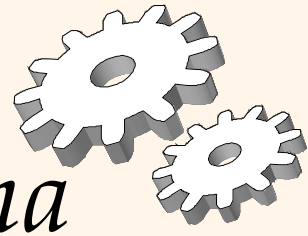
# *Custo do External Merge Sort*

- Número de Passos: $1 + \lceil \log_{B-1} \lceil N / B \rceil \rceil$
- Custo = 2N * (# de passos)
- Por exemplo., com 5 páginas no buffer, para classificar 108 páginas de arquivo:
  - Passo 0: $\lceil 108 / 5 \rceil$ - 21 runs ordenados de 5 páginas cada e 1 run com 3 páginas
  - Passo 1: $\lceil 22 / 4 \rceil$ - 5 runs ordenados de 20 páginas cada e 1 run com 8 páginas
  - Pass 2: 2 runs ordenados, 80 págs e 28 págs
  - Pass 3: Arquivo ordenado de 108 páginas

# *Número de passos do External Sort*



| 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 |
| 1,000,000,000 | 30 | 15 | 10 | 8 | 5 | 4 |



# *Algoritmo de Classificação Interna*

- Gerãção dos primeiros runs
- Quicksort é um algoritmo rápido para classificar em memória.
- Uma alternativa é "tournament sort" (a.k.a. "heapsort")
  - **`Topo:`** Leia ***B*** páginas
  - **`Saída:`** move o menor registro para um buffer de saída
  - Leia um novo registro *r*
  - Insira *r no heap*
  - Se *r* não for o menor, então **`GOTO Saída`**
  - senão remove *r* do heap
  - Gere como saída o heap em ordem; **`GOTO Topo`**

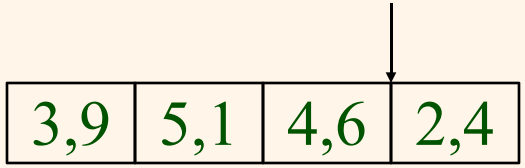
# *Exemplo Heapsort*

- B = 4 (2 páginas "atuais", 1 entrada, 1 saída); N=4

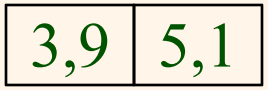
| 3,9 | 5,1 | 4,6 | 2,4 |
|---|---|---|---|

- Passo 1:

Ler até

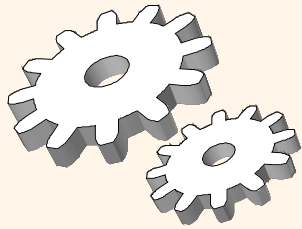



Heap



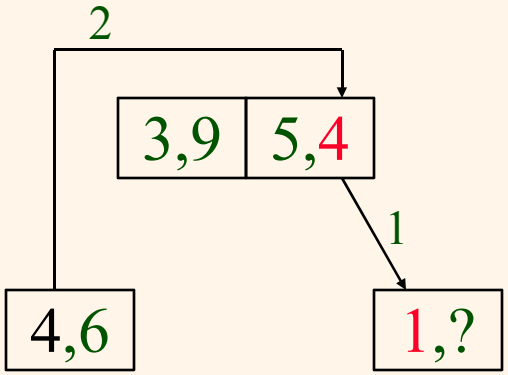

Entrada/Saída

# *Exemplo Heapsort*



Heap

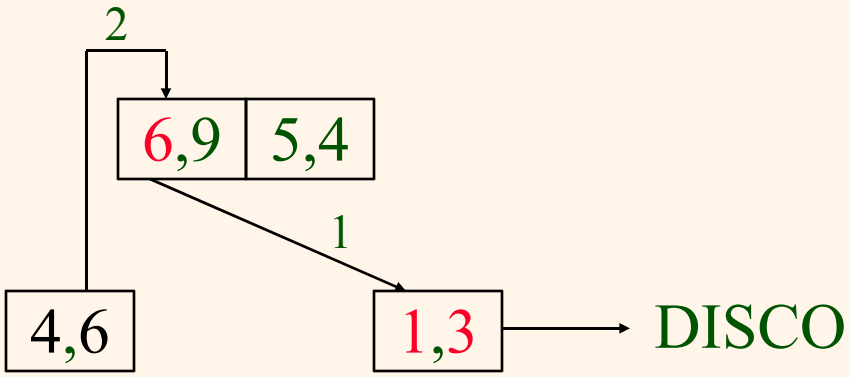
Entrada/Saída



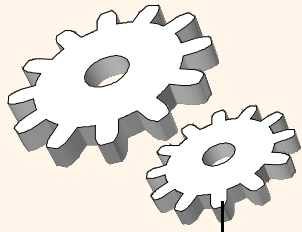

---

Heap

Entrada/Saída

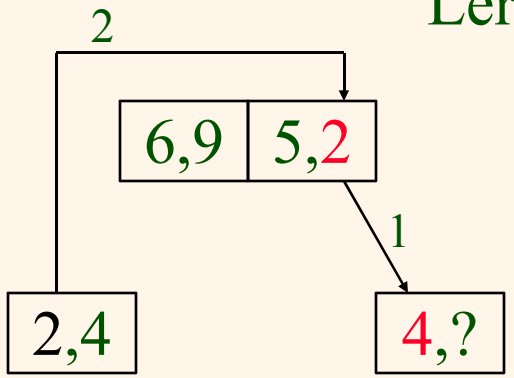


# *Exemplo Heapsort*



Ler até | 3,9 | 5,1 | 4,6 | 2,4

Heap



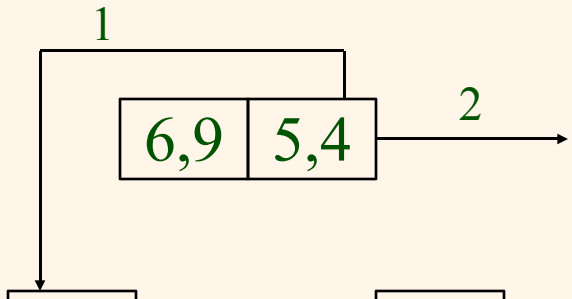

Entrada/Saída



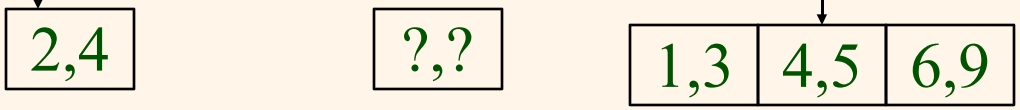

2 é o menor!

---

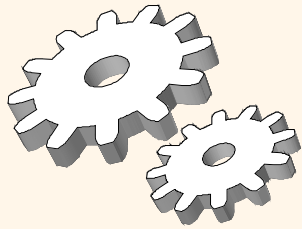
Heap



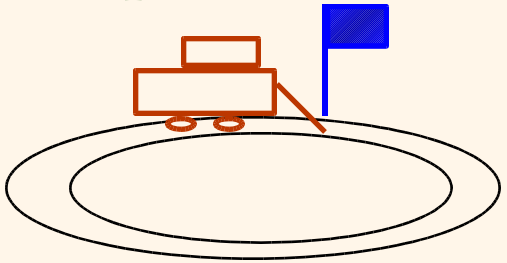

Entrada/Saída



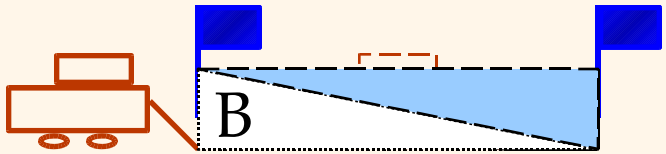
# *Mais sobre HeapSort*



- Fato: tamanho médio de um run no heapsort é *2B*
  - "snowplow" análogo a uma máquina de limpar neve (2 heaps??)

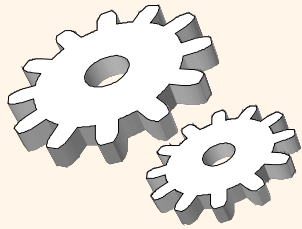


- Pior caso:
  - Qual é o tamanho mínimo de um run?
  - Quando isso acontece?
- Melhor caso:
  - Qual é o tamanho máximo de um run?
  - Quando isso acontece?



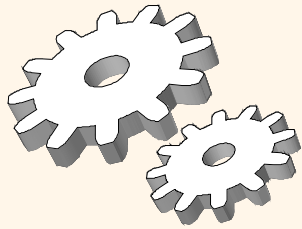

- Quicksort é mais rápido, mas ...

# *E/S para o External Merge Sort*



- … runs maiores leva a menos passos!
- Atualmente, fazemos E/S de uma página de cada vez
- Melhor, ler um *bloco de* páginas seqüencialmente!
- Sugere que nós devemos fazer com que cada buffer (entrada/saída) seja um *bloco de* páginas.
  - Mas isto irá reduzir #runs lidos por vez durante os passos de merge!
  - Na prática, a maioria dos arquivos ainda serão classificados em 2-3 passos.

# *Número de Passos para Otimizar Classificação*

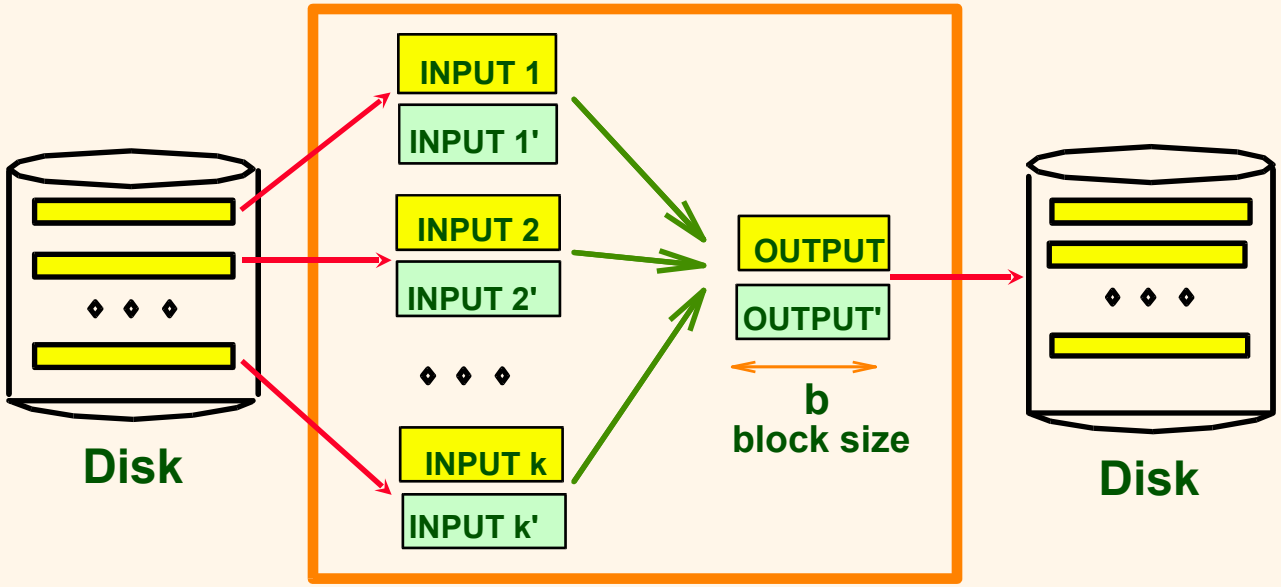


| N | B=1,000 | B=5,000 | B=10,000 |
|---|---|---|---|
| 100 | 1 | 1 | 1 |
| 1,000 | 1 | 1 | 1 |
| 10,000 | 2 | 2 | 1 |
| 100,000 | 3 | 2 | 2 |
| 1,000,000 | 3 | 2 | 2 |
| 10,000,000 | 4 | 3 | 3 |
| 100,000,000 | 5 | 3 | 3 |
| 1,000,000,000 | 5 | 4 | 3 |

- *Tamanho do Bloco = 32, passos iniciais produzem runs de tamanho 2B.*

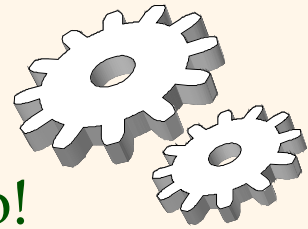
# *Double Buffering*

- Para reduzir tempo de espera de E/S para um pedido ser completado, pode-se usar "shadow block".
  - Potencialmente, mais passos; na prática, a maioria dos arquivos *ainda* serão classificados em 2-3 passos.



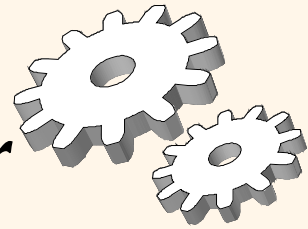

**B main memory buffers, k-way merge**

# *Classificando registros!*

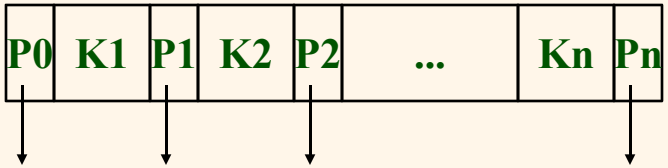


- Classificação tornou-se um esporte competitivo!
  - Classificação paralela é o nome do jogo ...
- Tarefa: Classificar 1 Milhão de registros com 100 bytes de tamanho cada
  - DBMS típico: 15 minutos
  - Recorde Mundial: 3.5 ***segundos***
    - 12-CPU SGI machine, 96 discos, 2GB de RAM
- Novos benchmarks propostos:
  - Minute Sort: Quanto você pode classificar em um minuto?
  - Dollar Sort: Quanto você pode classificar por $1.00?

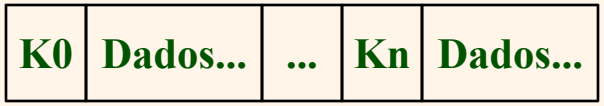
# *Usando Arvorés B+ para classificar*



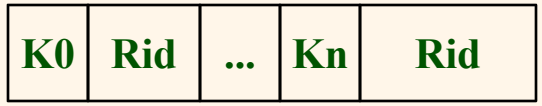
- **Árvore B+: balanceada, utilizando listas duplamente encadeadas para ligar nós "vizinhos"**
- **Nós não-folha contém chaves e ponteiros**
- **Nós folha contém entradas de dados**
- **Formato do nó não folha**



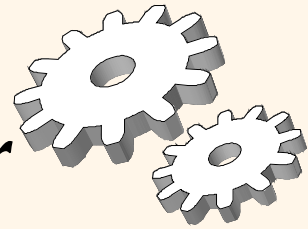

- **Nó folha**



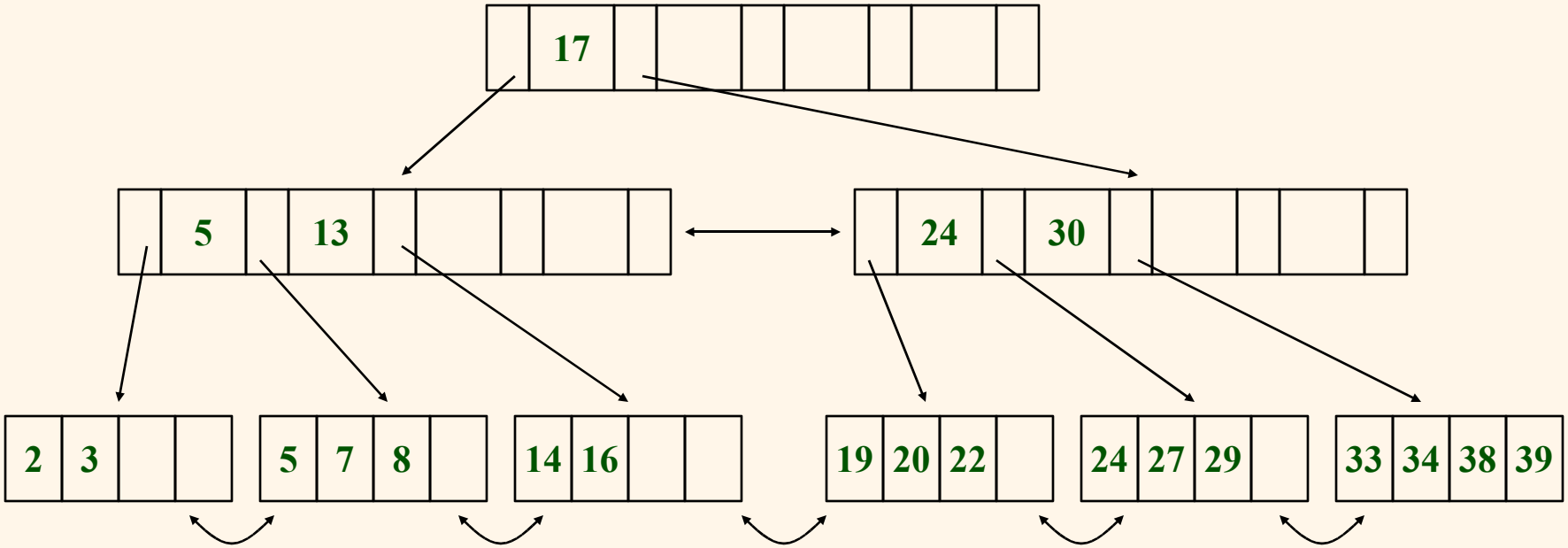

**ou**



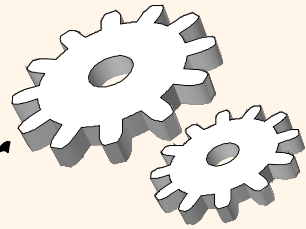
# *Usando Arvorés B+ para classificar*



- **Exemplo de índice com árvore B+**



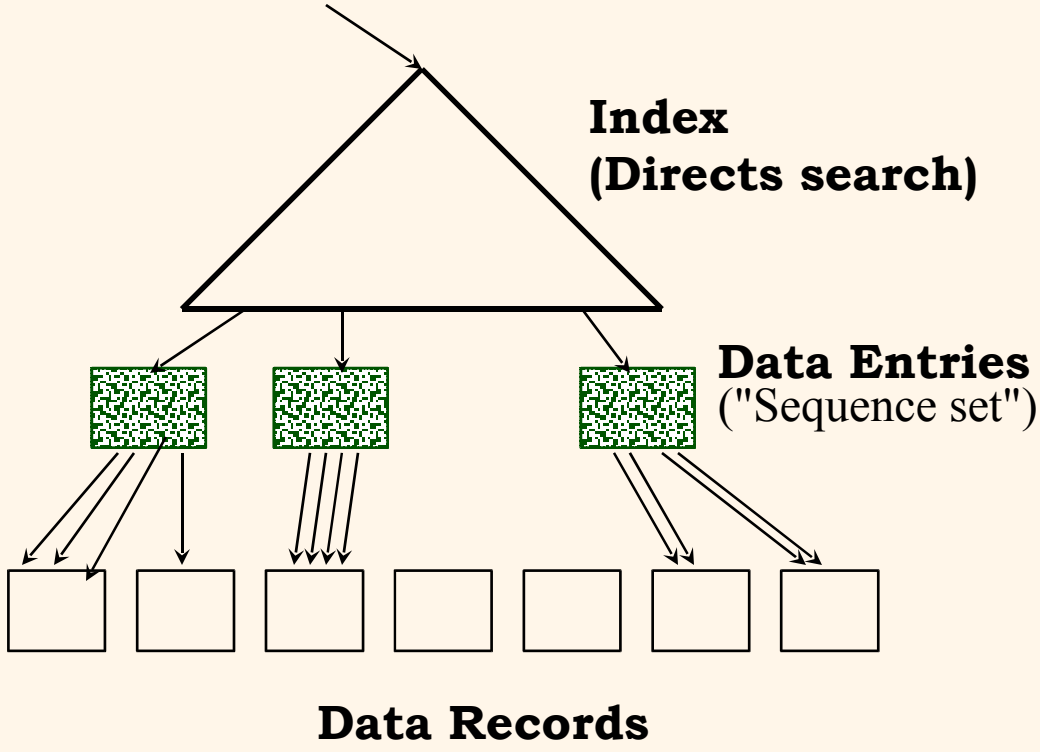
# *Usando Arvorés B+ para classificar*



- Cenário: Tabela a ser classificada tem uma árvore B+ como índice das coluna(s) de classificação.
- Idéia: Pode-se recuperar registros em ordem percorrendo as páginas folhas.
- ***Esta é uma boa idéia?***
- Casos para considerar:
  - Árvore B+ é agrupada — ***Boa idéia!***
  - Árvore B+ não é agrupada — ***Pode ser uma idéia muito ruim!***

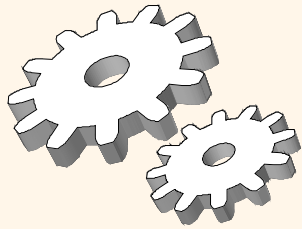
# *Usando Árvore B+ Agrupada*

- 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.

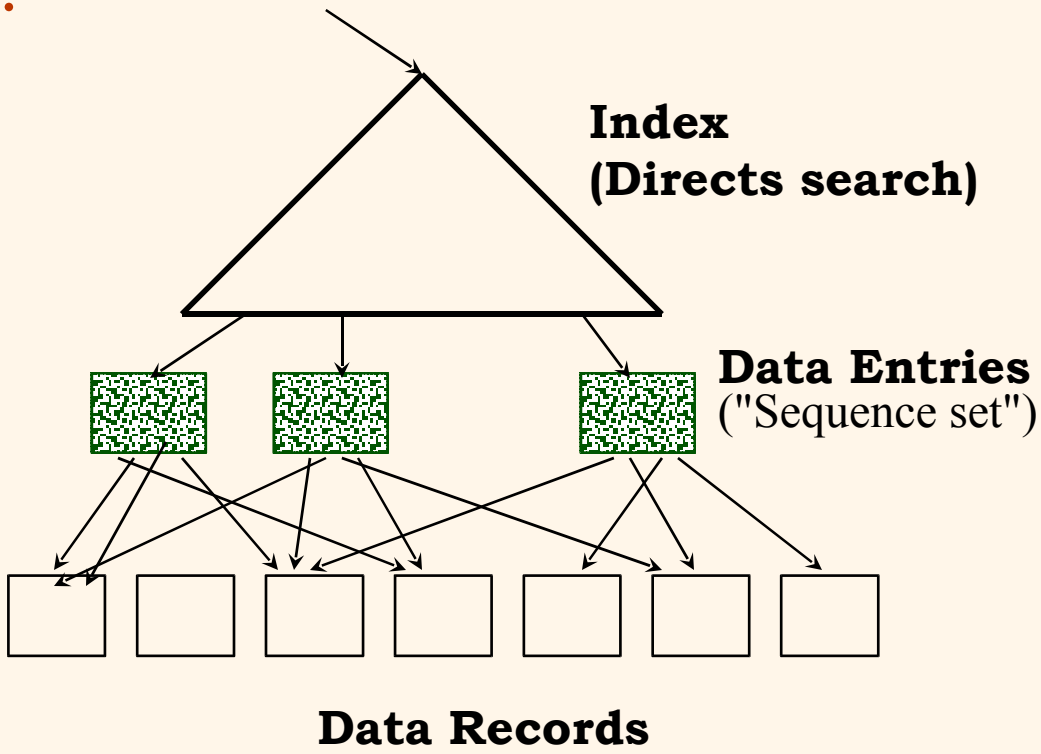



☛*Sempre melhor que classificação externa!*

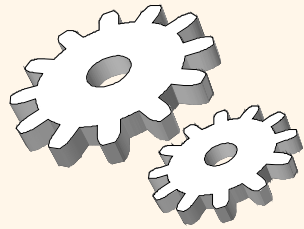
# *Usando Árvore B+ Não Agrupada*



- Alternativa (2) para entrada de dados; cada entrada de dados contém um *rid* local onde está o registro. No geral, uma E/S para cada registro!



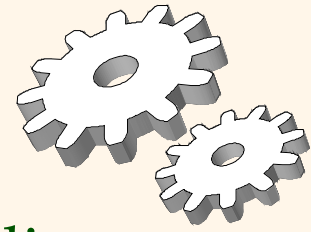
# *Classificação Externa vs. Índice Unclustered*



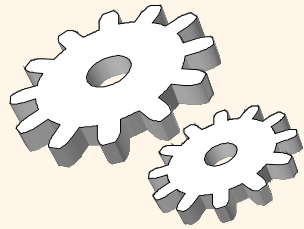
| N | Sorting | p=1 | p=10 | p=100 |
|---|---|---|---|---|
| 100 | 200 | 100 | 1,000 | 10,000 |
| 1,000 | 2,000 | 1,000 | 10,000 | 100,000 |
| 10,000 | 40,000 | 10,000 | 100,000 | 1,000,000 |
| 100,000 | 600,000 | 100,000 | 1,000,000 | 10,000,000 |
| 1,000,000 | 8,000,000 | 1,000,000 | 10,000,000 | 100,000,000 |
| 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.***

# *Resumo*



- Classificação externa é importante; DBMS deve dedicar parte do conjunto de buffers para classificação!
- External Merge Sort minimiza custo de E /S :
  - Passo 0: Produz runs ordenadas de tamanho ***B*** (# págs do buffer). Últimos passos: ***merge*** dos runs.
  - # de runs "merged" por vez depende de ***B***, e do ***tamanho do bloco.***
  - Blocos maiores significa menor custo de E/S por página.
  - Blocos maiores significa menor # de runs merged.
  - Na prática, # de runs raramente é maior que 2 ou 3.



# *Resumo, continuação.*

- Escolha de algoritmo de classificação interna é importante:
  - Quicksort: Rápido!
  - Heapsort/tournament sort: mais lento (2x), mas runs são maiores
- As melhores classificações são muito rápidas:
  - Apesar de ter mais de 40 anos de pesquisa, nós ainda estamos melhorando!
- Árvore B+ agrupada é boa para classificar; Árvore  não agrupada é muito ruim.