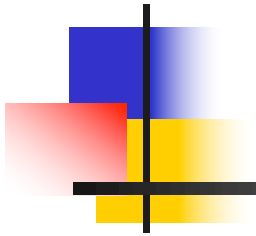
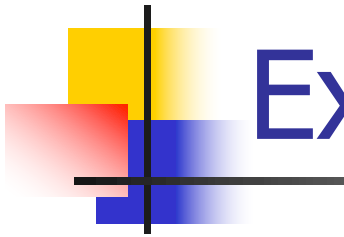


Ordenação
e
Busca em Arquivos

Cristina D. A. Ciferri



Exemplos de Busca

- Registros de tamanho fixo

M A R I A | R U A *b* 1 | S A O *b* C A R L O S | *b b b b b b b*

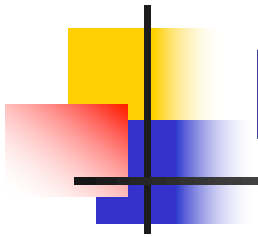
J O A O | R U A *b* A | R I O *b* C L A R O | *b b b b b b b b b*

P E D R O | R U A *b* X V | S A O *b* C A R L O S | *b b b b b b*

A N T O N I A | R U A *b* X V *b* D E *b* M A I O | I B A T E | *b*

A N A | R U A *b* A U G U S T O *b* P A I V A | I B A T E | *b b*

1. Recupere os dados do registro relativo ao João
2. Recupere os dados do registro relativo ao Pedro



Exemplos de Busca

- Registros de tamanho variável

M A R I A | R U A *b* 1 | S A O *b* C A R L O S | # J O A O | R

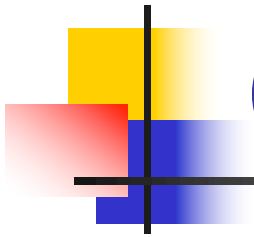
U A *b* A | R I O *b* C L A R O | # P E D R O | R U A *b* X V | S

A O *b* C A R L O S | # A N T O N I A | R U A *b* X V *b* D E *b* M

A I O | I B A T E | # A N A | R U A *b* A U G U S T O *b* P A I

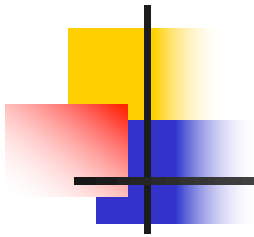
V A | I B A T E | #

3. Recupere os dados do registro relativo ao João
4. Recupere os dados do registro relativo ao Pedro



Ordenação

- Facilita a busca
- Pode ajudar a diminuir o número de acessos a disco



Busca Sequencial e Binária

- Busca sequencial
  - recupera cada registro do arquivo, verificando se os valores dos atributos satisfazem à condição de seleção
- Busca binária
  - recupera registros quando a condição de seleção envolve uma comparação de igualdade no atributo que determina a ordenação do arquivo

Custos: Comparações (RAM)

$$C_{busca\_sequencial} = n$$

- n: número de registros que são comparados
- todos os registros são varridos (pior caso)
- complexidade: O(n)

$$C_{busca\_binária} = \log_2(n) + 1$$

- n: número de registros que são comparados
- complexidade: O(log n)

Custos: Acessos a Disco

$$C_{busca\_sequencial} = b$$

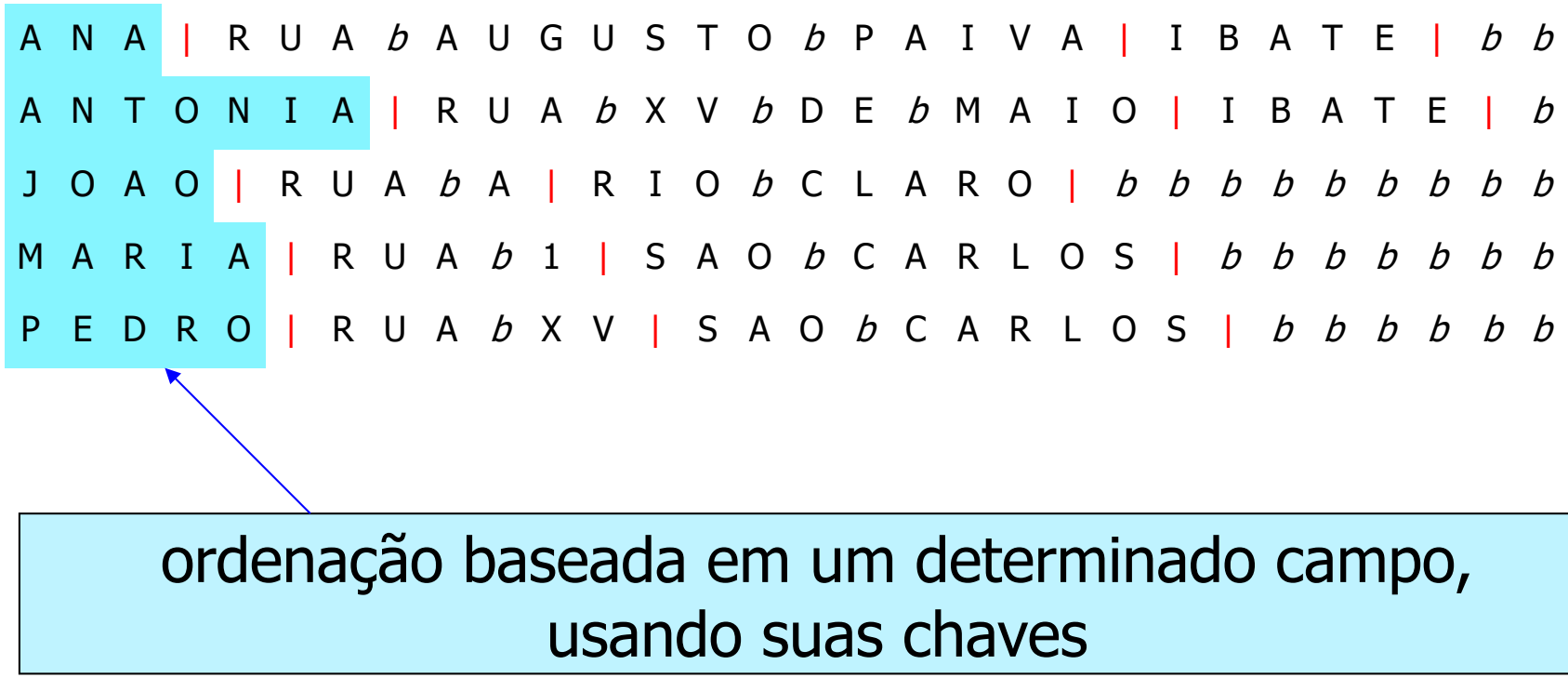
- b: número de blocos que contêm os registros
- todos os blocos são varridos

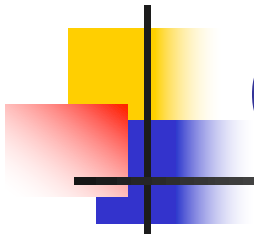
$$C_{busca\_binária} = \log_2(b) + \lceil s/bfr \rceil - 1$$

- log2(b): custo para localizar o primeiro registro
- $\lceil s/bfr \rceil$: blocos ocupados pelos registros que satisfazem à condição de seleção
- 1: custo para recuperar o primeiro registro

Arquivo Ordenado

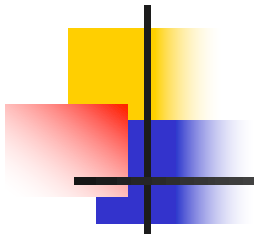
- Registros de tamanho fixo





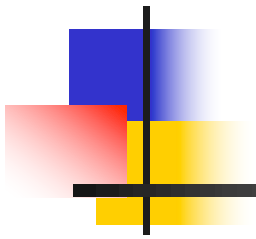
Chave (KEY)

- Está associada a um registro e permite a sua recuperação
- Chave primária
  - identifica univocamente um registro
  - não tem repetição
- Chave secundária
  - não identifica univocamente um registro
  - tem repetição



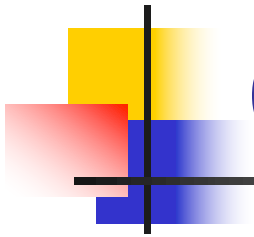
Forma Canônica da Chave

- Uma única representação para uma determinada chave
- Exemplo
  - "Ana", "ANA", ou "ana" devem indicar o mesmo registro
  - Forma canônica: todos os caracteres em letras maiúsculas → ANA

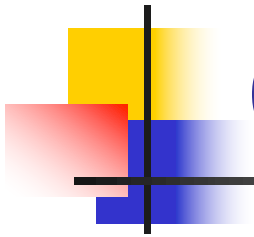


Ordenação Interna

Arquivo Completo Cabe em RAM

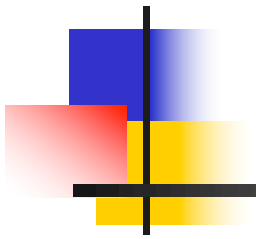


- Etapas
  1. leitura de todos os registros armazenados em disco para a RAM
  2. ordenação dos registros em RAM
     - escolha do campo base para ordenação
     - uso de um método de ordenação
  3. escrita de todos os registros armazenados em RAM para o disco



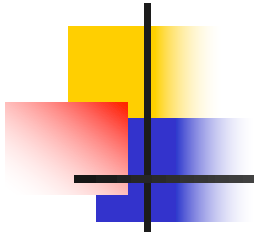
Custo

- Soma dos custos das 3 etapas
  - custo para ler o arquivo do disco para a RAM
  - custo para escrever o arquivo da RAM para o disco
  - custo do método de ordenação escolhido
    - Quicksort: O(n lg(n))
    - Heapsort: O(n lg(n))
      - *n: número de registros que são comparados*
- Importante
  - leitura e escrita sequenciais minimizam acessos a disco



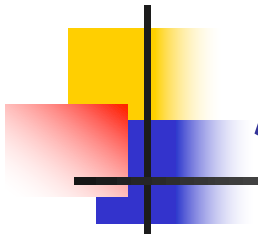
Ordenação Externa

Arquivo Completo não Cabe em RAM



- Funcionalidade
  - classifica registros de arquivos armazenados em disco
- Restrição
  - tamanho do arquivo é maior do que o tamanho da memória principal disponível
- Algoritmo típico
  - *sort-merge* externo

pode ordenar arquivos realmente grandes

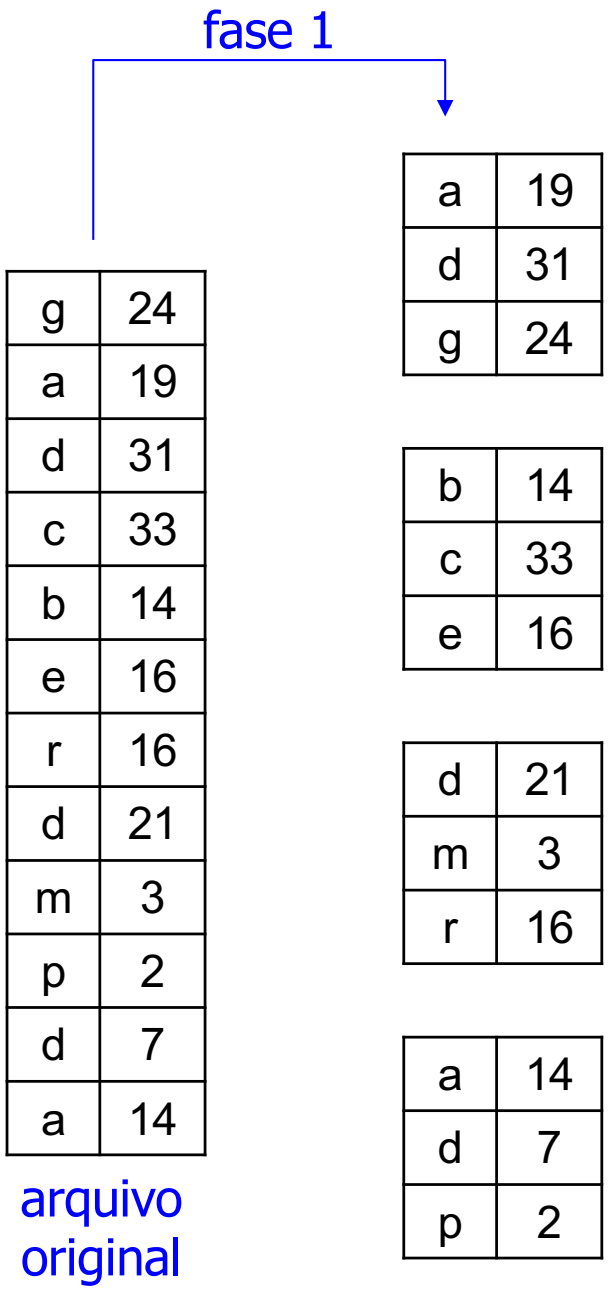


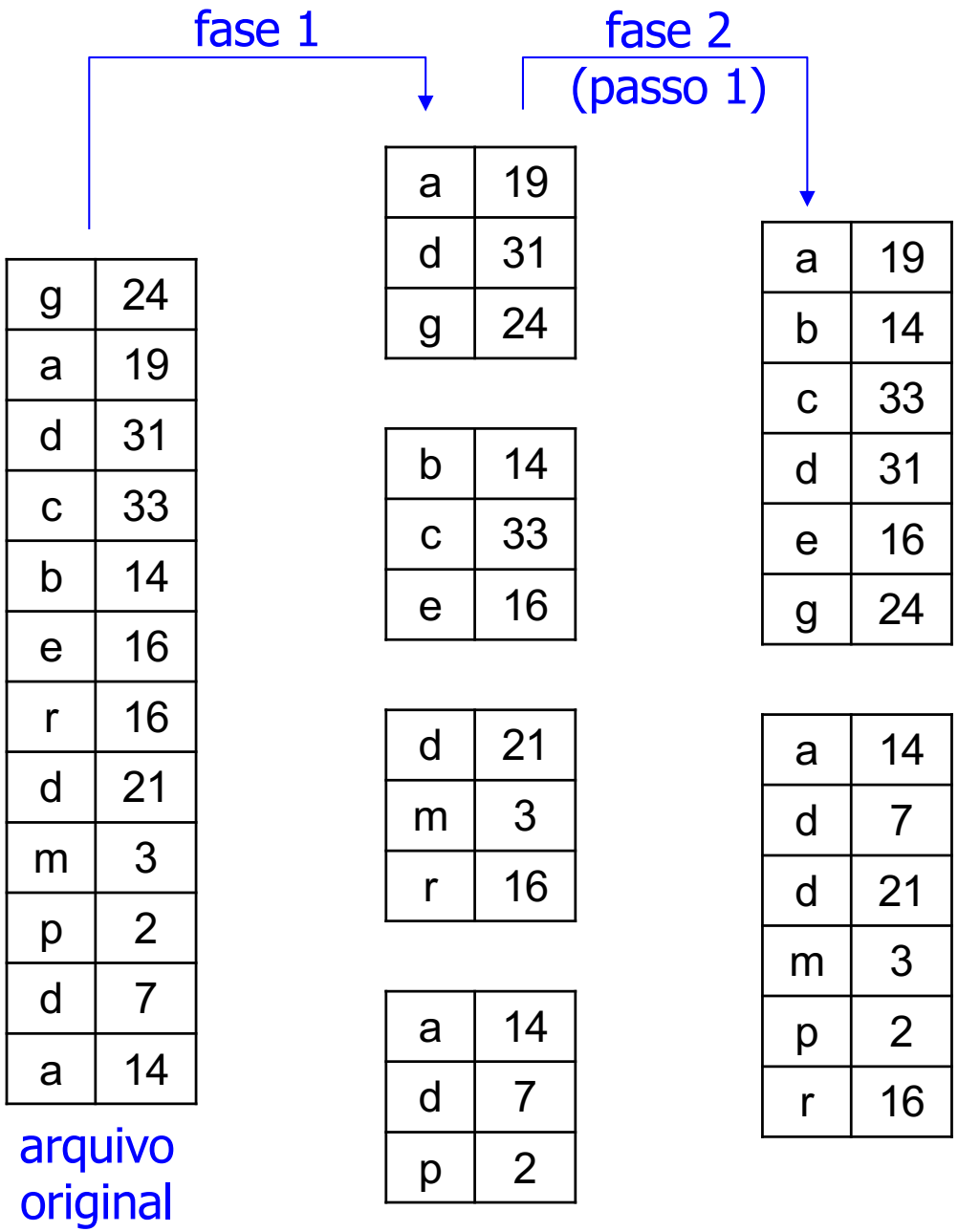
Sort-Merge Externo

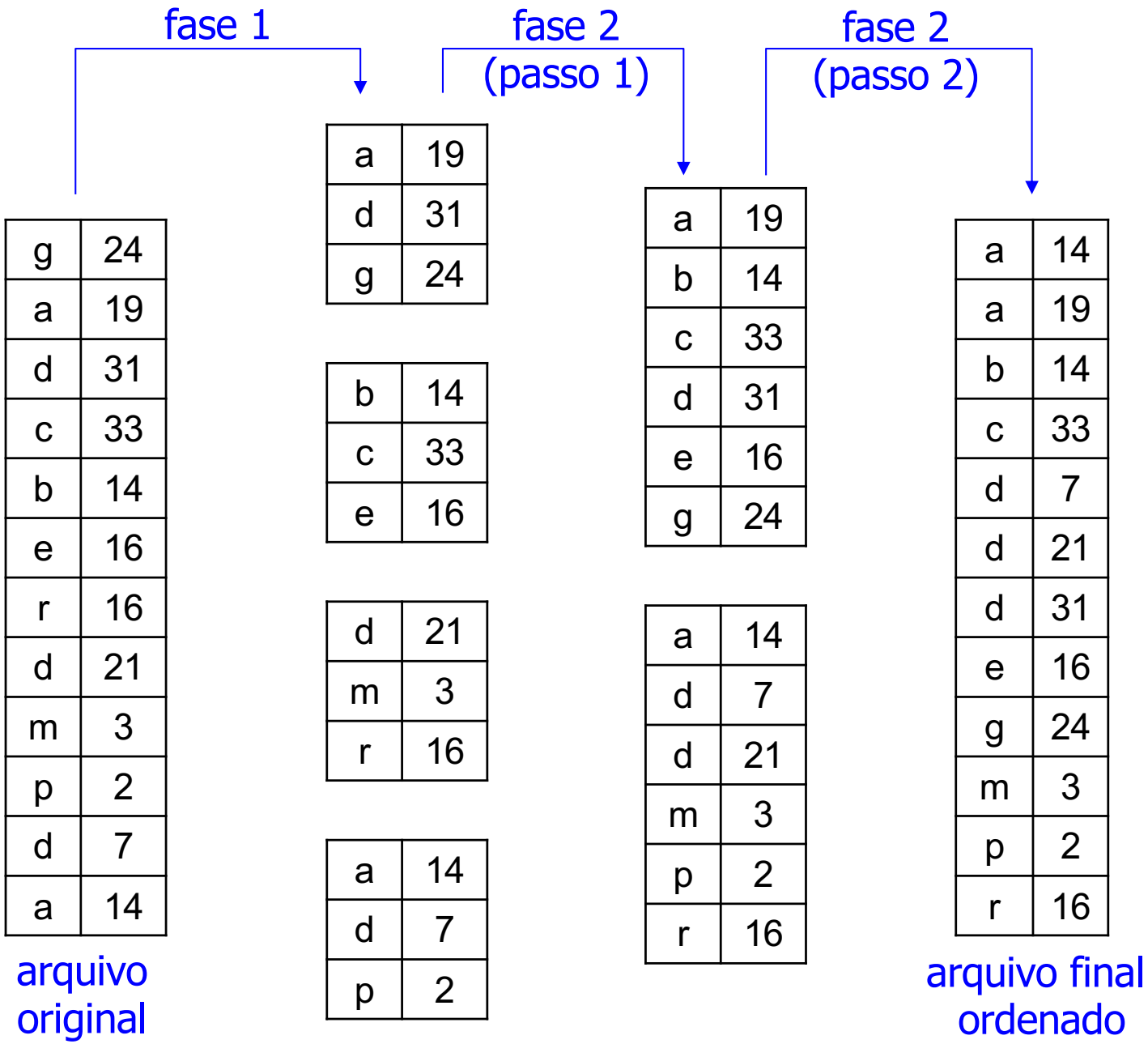
- Fase 1
  - cria subarquivos ordenados (i.e., *runs*) a partir do arquivo original
- Fase 2
  - combina os subarquivos ordenados em subarquivos ordenados maiores até que o arquivo completo esteja ordenado

g	24
a	19
d	31
c	33
b	14
e	16
r	16
d	21
m	3
p	2
d	7
a	14

arquivo original


fase 1
g 24
a 19
d 31
c 33
b 14
e 16
r 16
d 21
m 3
p 2
d 7
a 14
arquivo original
a 19
d 31
g 24
b 14
c 33
e 16
d 21
m 3
r 16
a 14
d 7
p 2


fase 1
fase 2
(passo 1)
g 24
a 19
d 31
c 33
b 14
e 16
r 16
d 21
m 3
p 2
d 7
a 14
arquivo
original
a 19
d 31
g 24
b 14
c 33
e 16
d 21
m 3
r 16
a 14
d 7
p 2
a 19
b 14
c 33
d 31
e 16
g 24
a 14
d 7
d 21
m 3
p 2
r 16


fase 1
fase 2
(passo 1)
fase 2
(passo 2)
g 24
a 19
d 31
c 33
b 14
e 16
r 16
d 21
m 3
p 2
d 7
a 14
arquivo
original
a 19
d 31
g 24
b 14
c 33
e 16
d 21
m 3
r 16
a 14
d 7
p 2
a 19
b 14
c 33
d 31
e 16
g 24
a 14
d 7
d 21
m 3
p 2
r 16
a 14
a 19
b 14
c 33
d 7
d 21
d 31
e 16
g 24
m 3
p 2
r 16
arquivo final
ordenado

Custo do Algoritmo

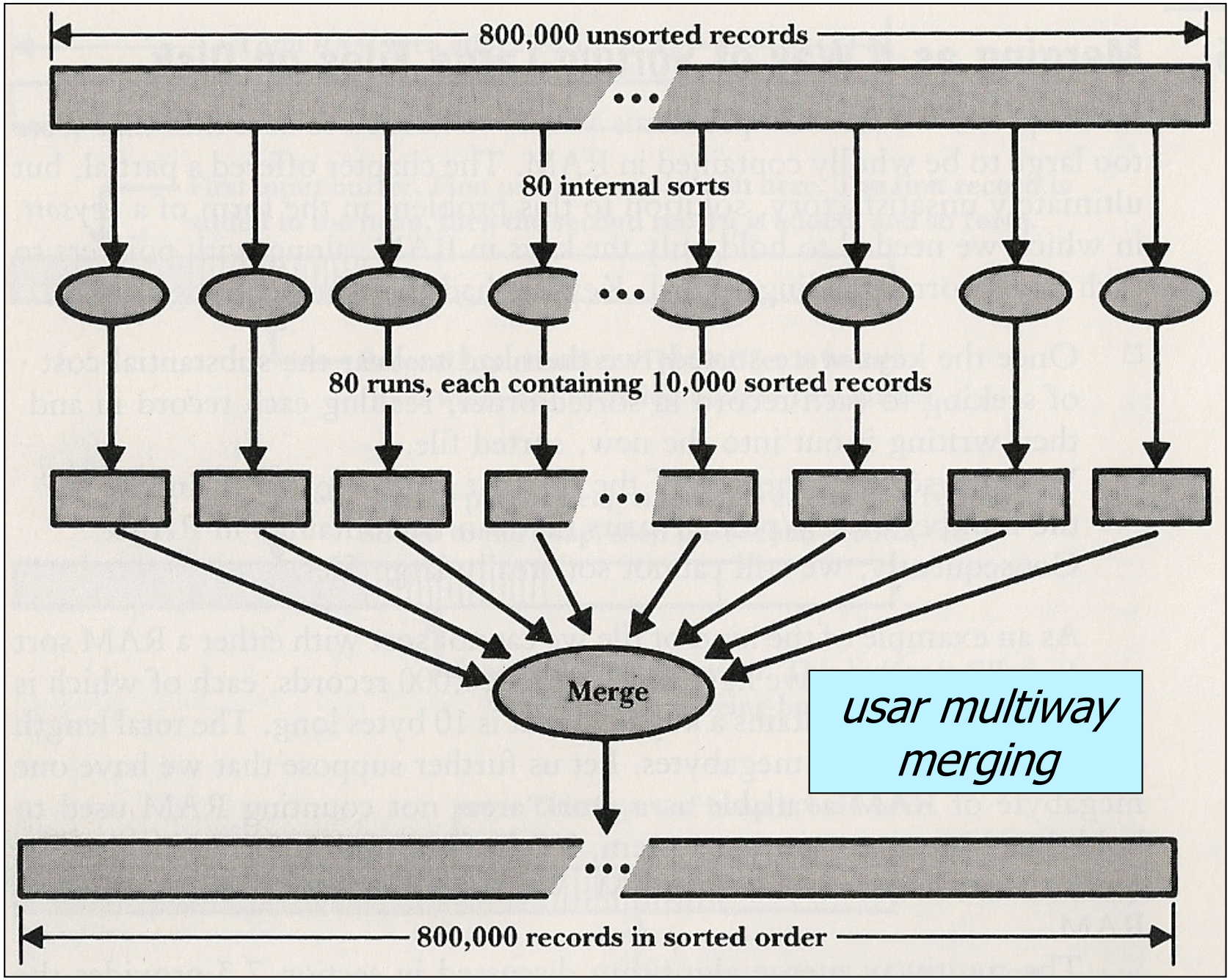
$$(2 * b) + (2 * (b * (\log_{d_m} b)))$$

- (2 * b)
  - número de acessos a disco na fase 1
- cada bloco do arquivo é acessado duas vezes
  - fase de leitura dos dados para a memória
  - fase de escrita dos dados ordenados no disco

Custo do Algoritmo

$$(2 * b) + (2 * (b * (\log_{d_m} b)))$$

- $(2 * (b * (\log_{d_m} b)))$
  - número de acessos a disco na fase 2
- cada bloco dos subarquivos é acessado várias vezes, dependendo do grau de combinação
  - fase de leitura dos dados para a memória
  - fase de escrita dos dados ordenados no disco



usar multiway merging