Processamento de Produtos (“X”)

- Alternativas e suas estimativas de custo
  - A1: laço aninhado *(“nested-loop”)*
  - A2: laço aninhado com índice *(“indexed nested-loop”)*
  - A3: merge-junção *(“balanced-line” ou “sort-merge”)*
  - A4: hash-junção *(“hash-join”)*

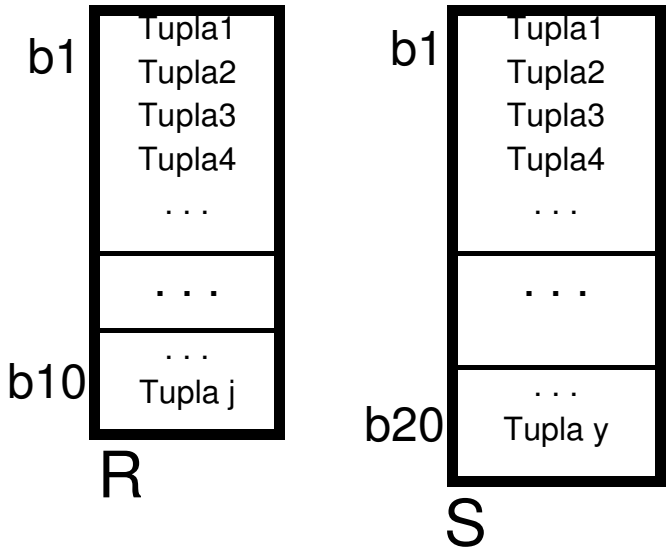
Laço Aninhado (A1)

- Dois laços para varredura de blocos das relações a serem combinadas

```
para cada bloco B_R de R faça
        para cada bloco B_S de S faça
        início
                se uma tupla t_R ∈ B_R satisfaz a condição de
                junção com uma tupla t_S ∈ B_S então
                        adicione t_R * t_S ao resultado
        fim
```

Laço Aninhado - Custo

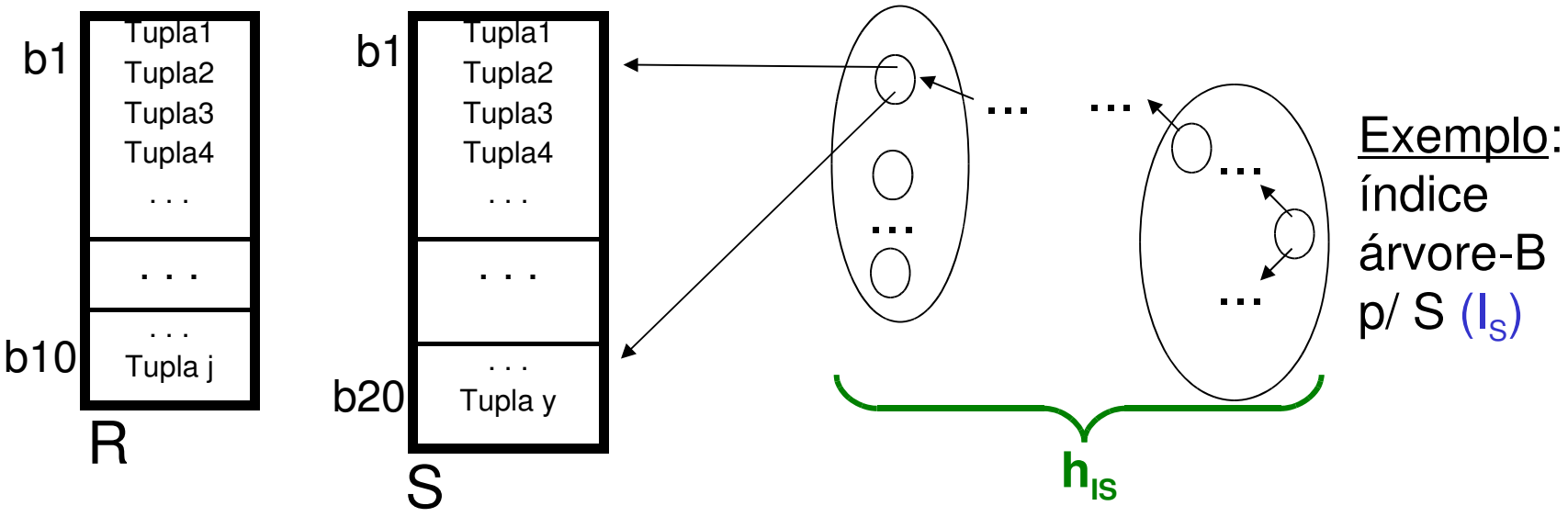
- Pior caso
  - apenas um bloco de cada relação cabe na memória
  - custo = MIN($b_R + b_R * b_S$, $b_S + b_S * b_R$)
- Exemplo



custo = MIN (10+10*20,
20+20*10)
= 210 acessos

Laço Aninhado com Índice (A2)

- Aplicado se existir um índice para o atributo de junção do laço interno
- Custo
  - para cada tupla externa de R, pesquisa-se o índice para buscar a tupla de S
  - custo diretamente associado ao tipo de índice

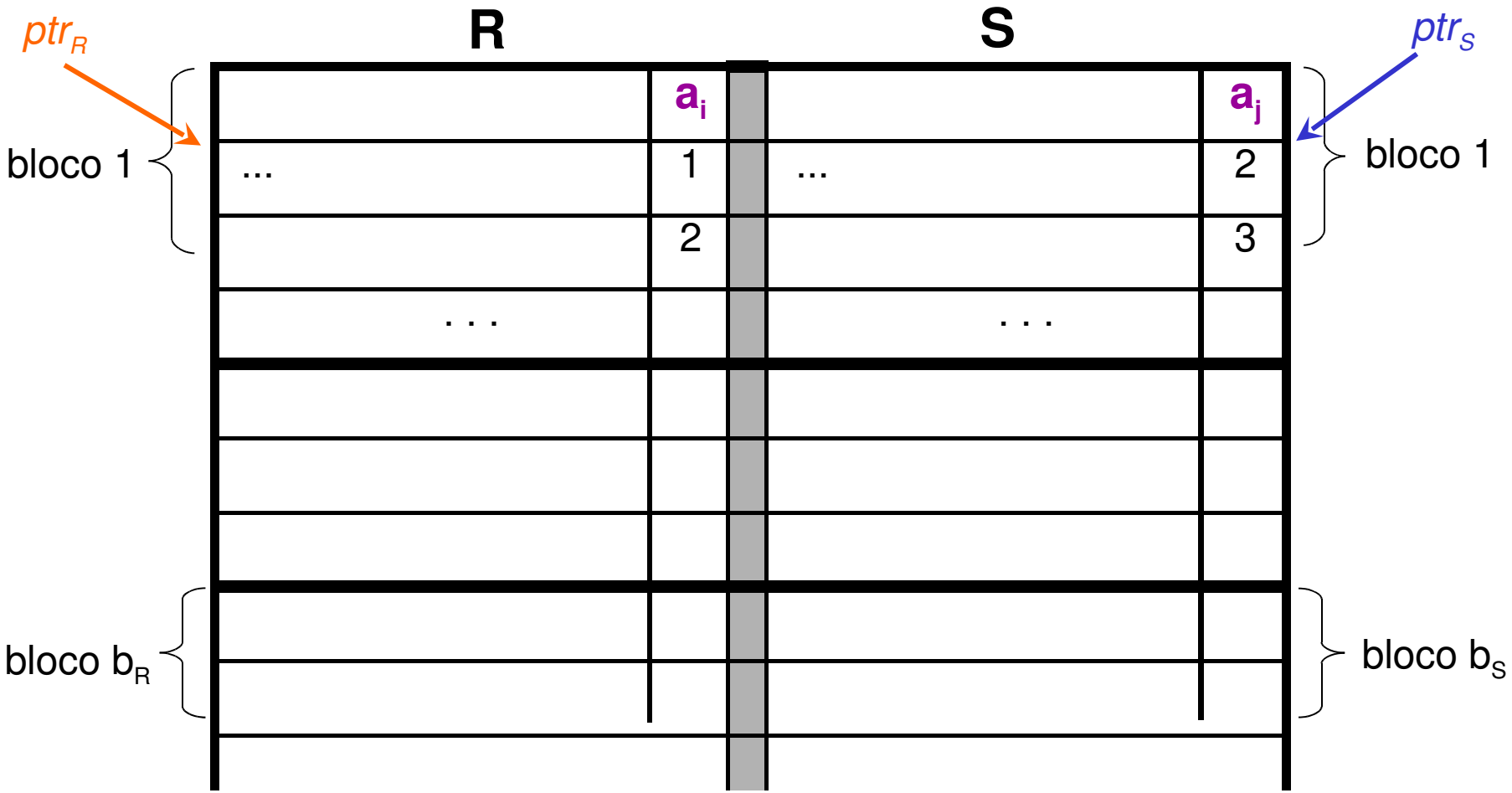


Laço Aninhado com Índice (A2)

- Exemplos
  - índice primário árvore-B para atributo chave (ver caso A3 da Seleção) em S
    - custo = $\mathbf{b_R + n_R * (h_{Is} + 1)}$
  - índice primário árvore-B para atributo não-chave (ver caso A4 da Seleção) em S
    - custo = $\mathbf{b_R + n_R * (h_{Is} + \lceil (C_s(a_i) / f_s) \rceil)}$
  - índice secundário árvore-B para atributo não-chave (ver caso A6 da Seleção) em S
    - custo = $\mathbf{b_R + n_R * (h_{Is} + 1 + \lceil C_s(a_i) \rceil)}$

Merge-Junção (A3)

- Aplicada se R e S estiverem fisicamente ordenadas pelos atributos de junção



Merge-Junção - Funcionamento

R

ptr_R →

ID-R	. . .	a_i
200	. . .	1
100	. . .	2
300	. . .	5

S

ID-S	. . .	a_j
C	. . .	2
A	. . .	3
E	. . .	5
B	. . .	5
D	. . .	7
F	. . .	8

← ptr_S

Resultado

ID-R	ID-S	. . .	a_i	a_j

Merge-Junção - Funcionamento

R

ptr_R →

ID-R	. . .	a_i
200	. . .	1
100	. . .	2
300	. . .	5

S

ID-S	. . .	a_j
C	. . .	2
A	. . .	3
E	. . .	5
B	. . .	5
D	. . .	7
F	. . .	8

← ptr_S

$ptr_R.a_i = ptr_S.a_j$? **NÃO**

Resultado

ID-R	ID-S	. . .	a_i	a_j

Merge-Junção - Funcionamento

R

ID-R	. . .	a_i
200	. . .	1
100	. . .	2
300	. . .	5

ptr_R →

S

ID-S	. . .	a_j
C	. . .	2
A	. . .	3
E	. . .	5
B	. . .	5
D	. . .	7
F	. . .	8

← ptr_S

$ptr_R.a_i = ptr_S.a_j$? **SIM**

Resultado

ID-R	ID-S	. . .	a_i	a_j
100	C	. . .	2	2

Merge-Junção - Funcionamento

R

ID-R	. . .	a_i
200	. . .	1
100	. . .	2
300	. . .	5

ptr_R →

S

ID-S	. . .	a_j
C	. . .	2
A	. . .	3
E	. . .	5
B	. . .	5
D	. . .	7
F	. . .	8

← ptr_S

$ptr_R.a_i = ptr_S.a_j$? **NÃO**

Resultado

ID-R	ID-S	. . .	a_i	a_j
100	C	. . .	2	2

Merge-Junção - Funcionamento

R

ID-R	. . .	a_i
200	. . .	1
100	. . .	2
300	. . .	5

ptr_R →

S

ID-S	. . .	a_j
C	. . .	2
A	. . .	3
E	. . .	5
B	. . .	5
D	. . .	7
F	. . .	8

← ptr_S

$ptr_R.a_i = ptr_S.a_j$? **SIM**

Resultado

ID-R	ID-S	. . .	a_i	a_j
100	C	. . .	2	2
300	E	. . .	5	5

Merge-Junção - Funcionamento

R

ID-R	. . .	a_i
200	. . .	1
100	. . .	2
300	. . .	5

ptr_R →

S

ID-S	. . .	a_j
C	. . .	2
A	. . .	3
E	. . .	5
B	. . .	5
D	. . .	7
F	. . .	8

← ptr_S

$ptr_R.a_i = ptr_S.a_j$? **SIM**

Resultado

ID-R	ID-S	. . .	a_i	a_j
100	C	. . .	2	2
300	E	. . .	5	5
300	B	. . .	5	5

Merge-Junção - Funcionamento

R

ID-R	. . .	a_i
200	. . .	1
100	. . .	2
300	. . .	5

ptr_R →

S

ID-S	. . .	a_j
C	. . .	2
A	. . .	3
E	. . .	5
B	. . .	5
D	. . .	7
F	. . .	8

← ptr_S

Resultado Final

ID-R	ID-S	. . .	a_i	a_j
100	C	. . .	2	2
300	E	. . .	5	5
300	B	. . .	5	5

Merge-Junção - Custo

- Pressupõe que pelo menos um bloco de cada relação cabe na memória
  - geralmente isso é possível
  - exige uma única leitura de cada relação
  - $\text{custo}_{\text{M-J}} = b_R + b_S$
- Se R e/ou S não estiverem ordenadas, elas podem ser ordenadas
  - custo = custo ordenação R e/ou S + $\text{custo}_{\text{M-J}}$

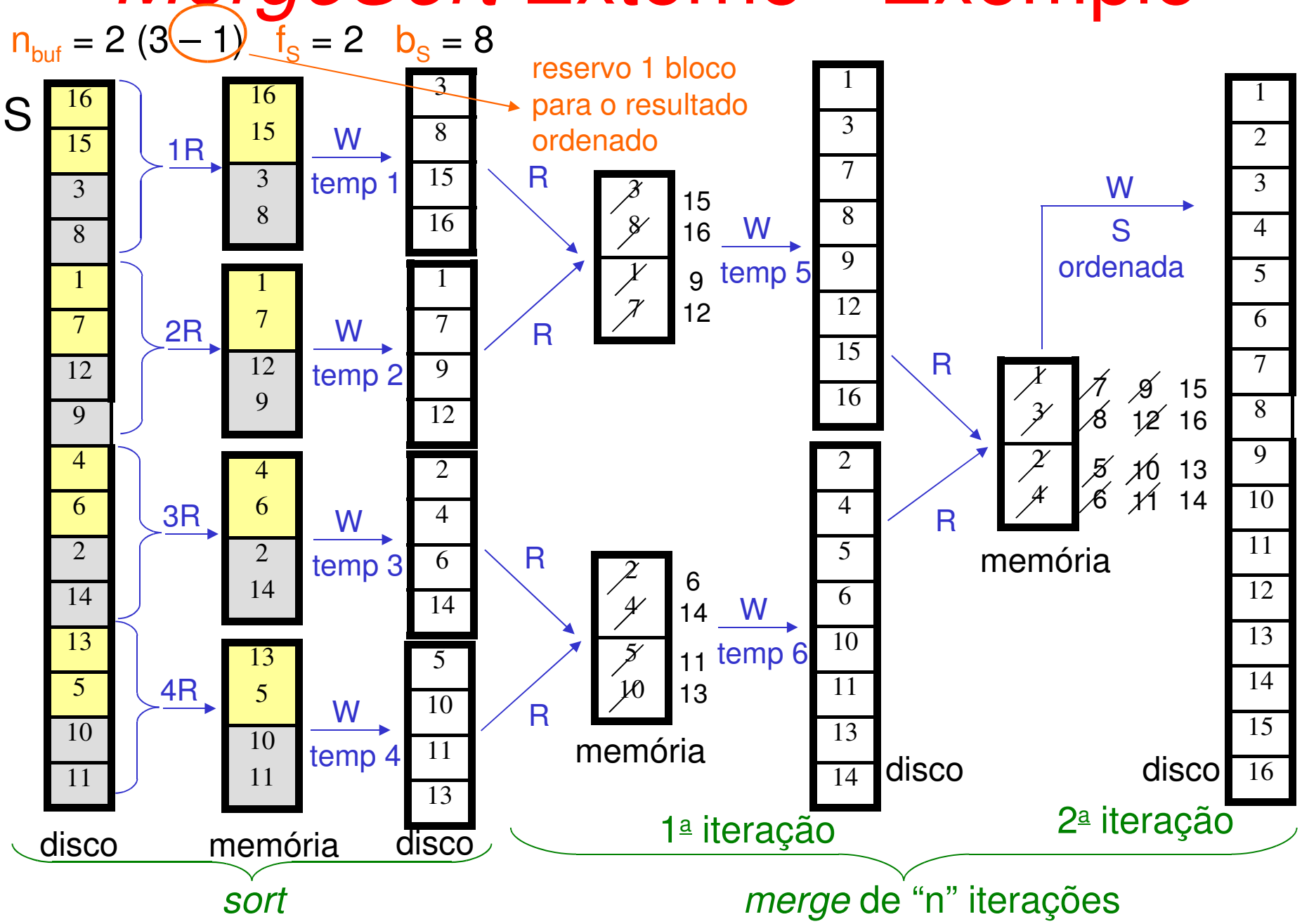
Ordenação Externa

- Ordenação interna
  - ordenação feita totalmente em memória
- Ordenação externa
  - ordenação na qual os dados não cabem inteiramente na memória
  - útil no processamento de consultas
    - exibição ordenada de dados (ORDER BY)
    - avaliação de planos de execução
  - técnica mais utilizada para ordenação de relações
    - ***MergeSort* Externo**

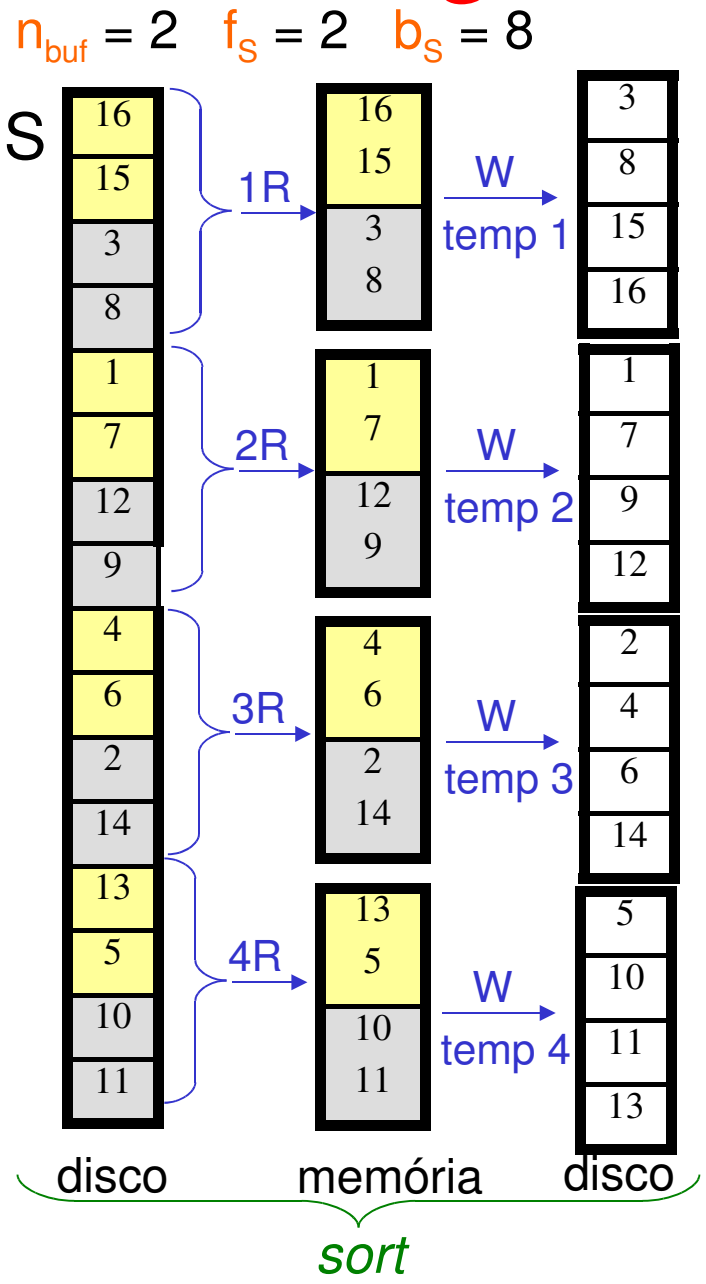
MergeSort Externo

- Executa em 2 etapas
- Etapa 1 – *Sort*
  - ordena partições da relação em memória
    - tamanho da partição depende da disponibilidade de blocos na memória (n_{buf} = nº de *buffers* disponíveis)
    - gera um arquivo temporário ordenado para cada partição
- Etapa 2 – *Merge* de "n" iterações
  - ordena um conjunto de temporários a cada iteração
    - gera um novo temporário resultante da ordenação
    - ordenação termina quando existir somente um temporário que mantém a relação inteira ordenada

MergeSort Externo - Exemplo



MergeSort Externo - Custo



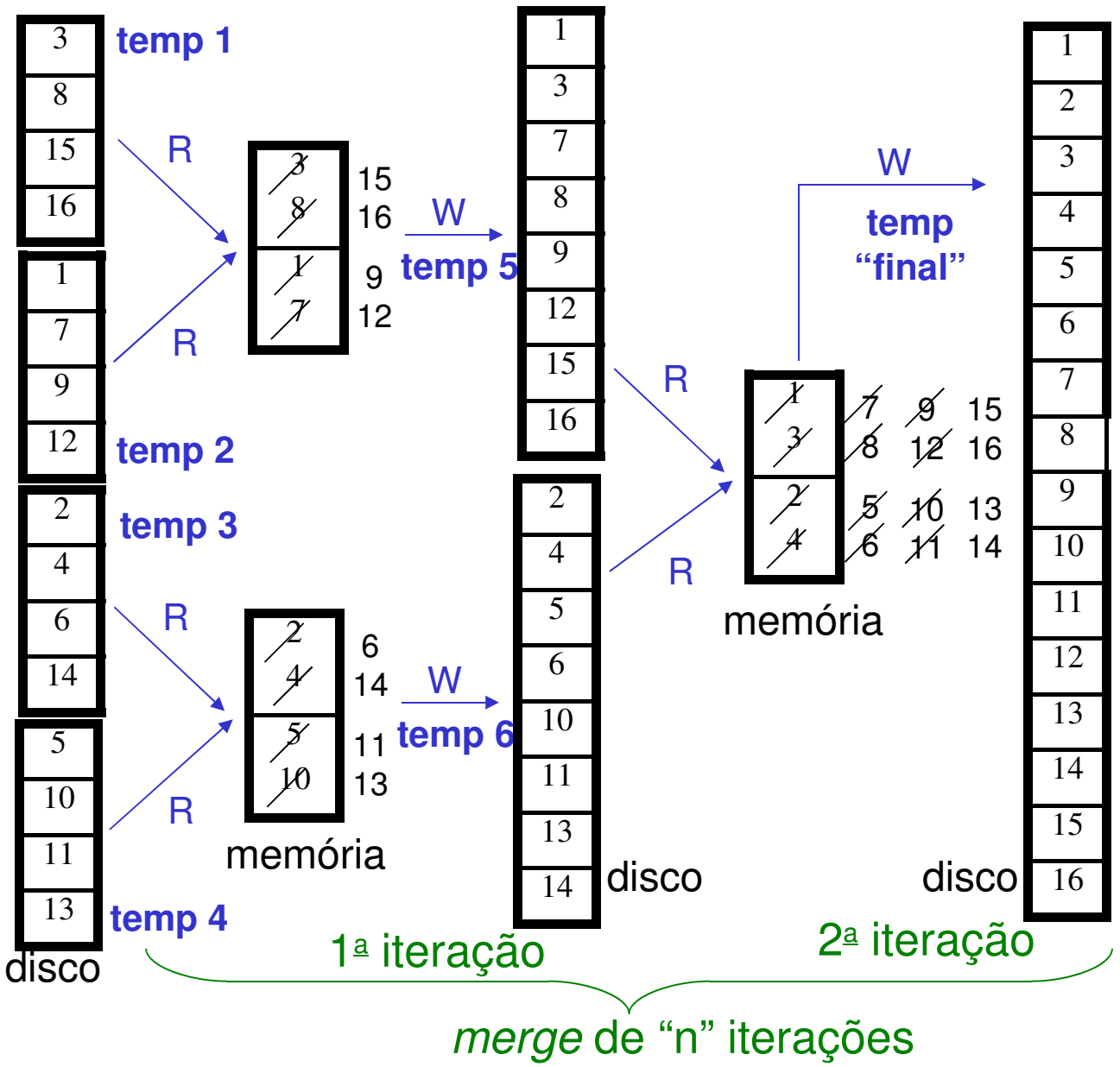
- 1 R + 1 W de todos os blocos da relação S
- custo = **2 * b_s**

MergeSort Externo - Exemplo

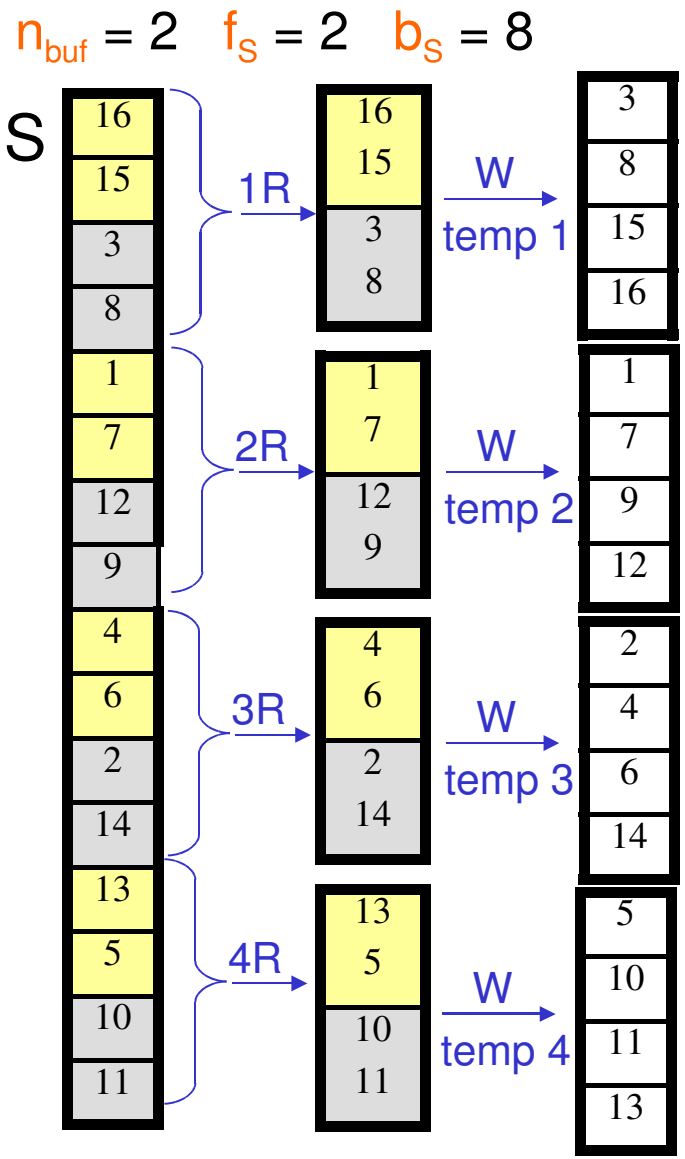
$n_{buf} = 2$ $f_S = 2$ $b_S = 8$

- Nº de iterações é dependente do nº de temporários a ordenar
- A cada iteração, o nº de temporários se reduz a um fator de n_{buf}
  - nº iterações: $\log_{n_{buf}} (b_S / n_{buf})$
  - 1 R + 1 W a cada iteração: $2 * b_S$
- custo = **$2 * b_S * \log_{n_{buf}} (b_S / n_{buf})$**

nº inicial de temporários



MergeSort Externo - Custo



Custo Total = $2 * b_S +$

$2 * b_S * \log n_{buf} (b_S / n_{buf})$

$= \mathbf{2 * b_S (\log n_{buf} (b_S / n_{buf}) + 1)}$

Exemplo = $2 . 8 (\log_2 (8 / 2) + 1)$

$= 16 (2 + 1) = 48$ acessos

Merge-Junção - Custo

- Se ambas as relações (R e S) estão ordenadas
  - custo = $b_R + b_S$
- Se uma delas (R) não está ordenada
  - custo = $2 * b_R \ (\log n_{buf} \ (b_R / n_{buf}) + 1) + b_R + b_S$
- Se ambas as relações não estão ordenadas
  - custo = $2 * b_R \ (\log n_{buf} \ (b_R / n_{buf}) + 1) +$

$$2 * b_S \ (\log n_{buf} \ (b_S / n_{buf}) + 1) +$$

$$b_R + b_S$$

Exercício 4

Estime o custo das operações produtórias abaixo, considerando as estimativas sobre *Pac* dadas no Exercício 1, além de $n_{buf} = 5$, $n_{Cons} = 1000$ tuplas, $t_{Cons} = 20$ bytes e $V_{Cons}(codp) = 500$. A relação *Cons* está ordenada por *codm* e possui um índice secundário hash para *codp*:

a) $\rho_{P1}(Pac) \; X \; \theta = \sigma_{P1.idade = P2.idade} \; \rho_{P2}(Pac)$

b) $Pac \bowtie Cons$

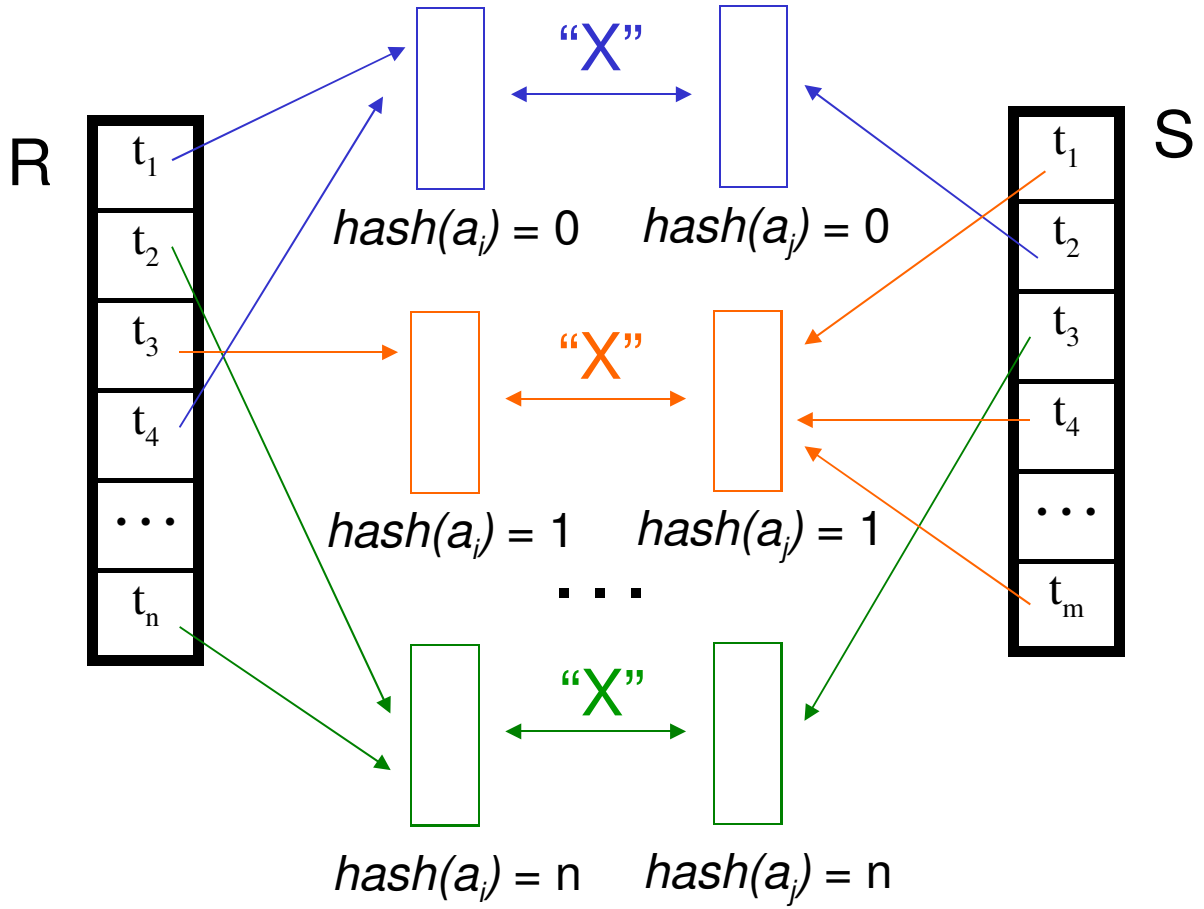
Hash-Junção (A4)

- Aplicada se existir um índice *hash* com a mesma função definido para os atributos de junção das relações R e S
- Executa em 2 etapas

1. Particionamento
   - separa em partições (conjunto de 1 ou mais blocos) as tuplas de R e S que possuem o mesmo valor para a função de *hash*

1. Junção
   - analisa e combina as tuplas de uma mesma partição

Hash-Junção - Funcionamento



Hash-Junção - Custo

- Fase de Particionamento
  - lê as relações R e S e as reescreve, organizadas em partições
    - sempre que um conjunto de tuplas com o mesmo valor de *hash* adquire o tamanho de um bloco, este bloco é anexado a um arquivo temporário para a partição
    - para fins de simplificação de cálculo, considera-se que a função *hash* distribui uniformemente os valores das tuplas
      - evita escrita de muitas pequenas partições (difícil estimar na prática). Assim, assume-se custo "W" = custo "R" e não custo "W" > custo "R"
    - custo = $2 * b_R + 2 * b_S = \mathbf{2 * (b_R + b_S)}$

Hash-Junção - Custo

- Fase de Junção
  - lê as partições de mesmo *hash* e combina as tuplas
    - novamente, para fins de simplificação de cálculos, considera-se uma nova leitura de todos os blocos de R e S organizados nas partições
      - na prática, é possível que alguns blocos de R ou de S sejam lidos mais de uma vez para serem combinados com outros blocos, porém, tal situação é muito dinâmica, dependendo dos dados existentes em R e S
    - custo = $\mathbf{(b_R + b_S)}$
- Custo Total
  - custo = $2 * (b_R + b_S) + (b_R + b_S) = \mathbf{3 * (b_R + b_S)}$

Estimativa de Tamanho de Produtos

- Produto cartesiano (R X S)
  - tamanho = $n_R * n_S$

- Junção por igualdade (*"equi-join"* – natural ou theta)
  - junção natural sem atributo em comum
    - tamanho = $n_R * n_S$

Estimativa de Tamanho de Produtos

- Junção por igualdade (*"equi-join"*)
  - junção por referência (*fk(R) = pk(S)*)
    - tamanho estimado <= n_R ("<" devido a eventuais nulos)
      - na prática, assume-se **tamanho = n_R**

R

ID-R	. . .	ID-S
100	. . .	1
200	. . .	2
300	. . .	2

S

ID-S	. . .
1	. . .
2	. . .
3	. . .
4	. . .
5	. . .
. . .	. . .

R "X" S (supondo join theta)

ID-R	R.ID-S	. . .	S.ID-S	…
100	1	. . .	1	. . .
200	2	. . .	2	. . .
300	2	. . .	2	. . .

Estimativa de Tamanho de Produtos

- Junção por igualdade (*"equi-join"*)
  - junção entre chaves candidatas (atributos *unique*)
    - tamanho <= MIN (n_R , n_S) ("<": nem todos podem casar)
      - na prática, assume-se **tamanho = MIN(n_R , n_S)**

R

ID-R	. . .	CPF
100	. . .	1
200	. . .	2
300	. . .	5

S

. . .	CPF
. . .	1
. . .	2
. . .	3
. . .	4
. . .	5
. . .	. . .

R "X" S (supondo join theta)

ID-R	. . .	R.CPF	…	S.CPF
100	. . .	1	. . .	1
200	. . .	2	. . .	2
300	. . .	5	. . .	5

Estimativa de Tamanho de Produtos

- Junção por igualdade (*"equi-join"*)
  - junção entre <u>atributos não-chave</u> ($a_i(R) = a_j(S)$)
    - cada tupla de *R* associa-se com $C_S(a_j)$ de *S*
      - se tenho n_R tuplas $\Rightarrow$ $n_R * C_S(a_j)$
    - idem para as tuplas de S: $n_S * C_R(a_i)$
    - tamanho estimado = MIN($\lceil n_R * C_S(a_j) \rceil$, $\lceil n_S * C_R(a_i) \rceil$)
      - menor estimativa geralmente está mais próxima do real

R

...	a_i
...	1
...	2
...	2

S

...	a_j
...	1
...	1
...	2
...	4
...	5

R "X" S (supondo join theta)

...	a_i	a_j
...	1	1
...	1	1
...	2	2
...	2	2

$C_R(a_i) = 3/2 = 1,5$

$C_s(a_j) = 5/4 = 1,3$

$\lceil n_R * C_S(a_j) \rceil = 4$

$\lceil n_S * C_R(a_i) \rceil = 8$

Tamanho estimado = 4

Estimativa de Tamanho de Produtos

- Junção theta <u>por desigualdade</u> (*a_i (R) > a_j (S), por exemplo*)
  - <u>estimativa</u>: cada tupla de R > n_s / 2 tuplas de S e vice-versa
  - tamanho estimado (caso médio) = $\lceil AVG(n_R * (n_s / 2) , n_S * (n_R / 2)) \rceil$ = $\lceil n_R * (n_s / 2) \rceil$ OU $\lceil n_S * (n_R / 2)) \rceil$

R

...	a_i
...	2
...	3

S

...	a_j
...	1
...	2
...	3
...	4
...	5

R "X" S (supondo join theta)

...	a_i	a_j
...	2	1
...	3	1
...	3	2

$n_R * (n_S / 2) = 2*2,5 = 5$

$n_S * (n_R / 2) = 5*1 = 5$

Tamanho estimado = 5

Junções Complexas - Custo

- Estimativas de custo apresentadas até agora consideram uma única condição de junção
  - na prática, várias condições podem ser definidas
- Dada uma junção complexa conjuntiva

  R "X" $_{\theta = \sigma\, c1 \wedge c2 \wedge \ldots \wedge cn}$ S
  - estima-se o custo de junção de cada condição c_i
    - R "X" $_{\theta = \sigma\, ci}$ S
  - escolhe-se a condição c_i de menor custo para ser implementada
    - as demais condições $c_1, c_2, \ldots, c_{i-1}, c_{i+1}, \ldots, c_n$ são verificadas a medida que as tuplas de R "X" $_{\theta = \sigma\, ci}$ S são geradas

Junções Complexas - Custo

- Dada uma junção complexa disjuntiva R “X”$_{\theta = \sigma\, c1 \vee c2 \vee \ldots \vee cn}$ S, tem-se as seguintes alternativas (escolhe-se a de menor custo)
  - aplica-se o algoritmo de laço aninhado
    - mais simples e independente de condição de junção
  - aplica-se (R “X”$_{\theta = \sigma\, c1}$S) $\cup$ (R “X”$_{\theta = \sigma\, c2}$S) $\cup \ldots \cup$ (R “X”$_{\theta = \sigma\, cn}$ S)
    - custo total é a soma dos menores custos de cada junção individual
    - tamanho do resultado é dado pela estimativa de tamanho de operações de união (a ser visto)

Junções Complexas - Tamanho

- Dada uma junção R "X"$_{\theta = \sigma\, c1 \wedge c2 \wedge \ldots \wedge cn}$ S

1. estima-se a cardinalidade (tamanho) de cada condição c_i
   - $C(c_i)$
1. aplica-se a fórmula adequada de cálculo de tamanho, dentre as definidas para seleção (σ)
   - considera-se $n_R * n_S$ o tamanho da relação de entrada (produto cartesiano de R e S)
     - para o exemplo acima:

$$\text{Tamanho} = \lceil (\mathbf{n_R * n_S}).\ (C(c_1).\ C(c_2).\ \ldots .\ C(c_n)) / (\mathbf{n_R * n_S})^n \rceil$$

Tamanho Junções Complexas - Exemplo

n_{Med} = 100 tuplas; V_{Med}(nome) = 100; V_{Med}(cidade) = 50; n_{Pac} = 500 tuplas; V_{Pac}(nome) = 500; V_{Pac}(cidade) = 150

e

Pac "X"$_{\theta = \sigma \text{ Pac.idade} > \text{Med.idade} \wedge \text{Pac.cidade} = \text{Med.cidade} \wedge \text{Pac.nome} = \text{Med.nome}}$ Med

C(Pac.idade > Med.idade) = n_{Pac} (n_{Med} /2) = 25000 tuplas

C(Pac.cidade = Med.cidade) = MIN(n_{Pac} . C_{Med}(cidade), n_{Med} . C_{Pac}(cidade)) = MIN(500.100/50,100.500/150) = 333,33 tuplas

C(Pac.nome = Med.nome) = MIN(n_{Pac} . C_{Med}(nome), n_{Med} . C_{Pac}(nome)) = MIN(500.100/100,100.500/500) = 100 tuplas

Tamanho = $\lceil$ **(500 * 100)**. (25000. 333,33. 100) / **(500 * 100)**^{3} $\rceil$ = 1 tupla

Exercício 5

Estime o tamanho das operações produtórias do Exercício 4