5. Algoritmos de Ordenação

Prof. Renato Tinós

Local: Depto. de Computação e Matemática
(FFCLRP/USP)

Principais Tópicos

5.1. Ordenação por Inserção
5.2. Ordenação por Seleção
5.3. Método da Bolha
5.4. Ordenação por Fusão
    5.4.1. Operação de Fusão
    5.4.2. Ordenação por Fusão Direta
5.5. Heapsort
5.6. Quicksort
5.7. Considerações sobre o Problema de Ordenação
5.8. Ordenação em Tempo Linear

5.4. Ordenação por Fusão

- A aplicação de algoritmos de ordenação vistos até aqui é interessante quando os dados estão em memória principal
  - Nesse caso, os dados são descritos na forma de arquivos seqüenciais cuja característica é que, a cada instante, é possível o acesso (direto) a um e somente um dos seus componentes

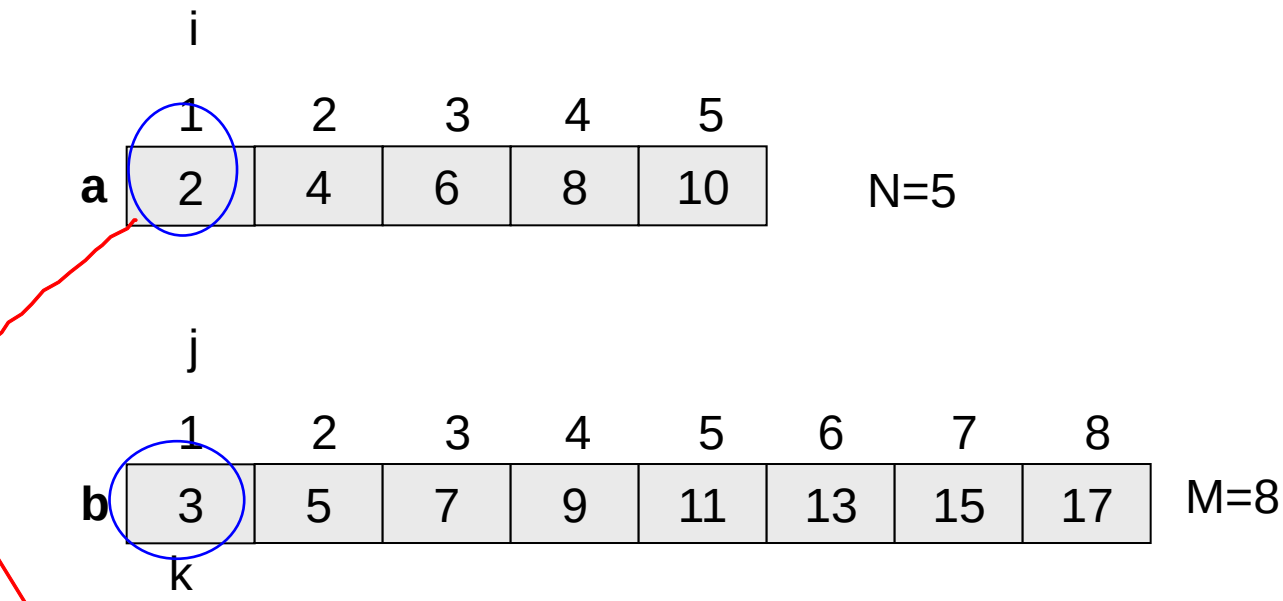
- Isso é uma restrição severa, se comparada com as possibilidades oferecidas pela estrutura de vetor

5.4.1. Operação de Fusão

- A **operação de fusão**
  - combina duas ou mais seqüências ordenadas para formar uma única seqüência ordenada
  - utiliza repetidas seleções entre os elementos acessíveis a cada momento
  - é mais simples que a de ordenação, sendo empregada como operação auxiliar no processo mais complexo de ordenação seqüencial.

5.4.1. Operação de Fusão

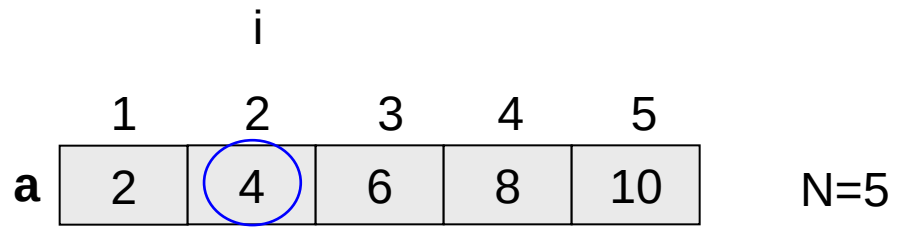
Exemplo 5.4.1.



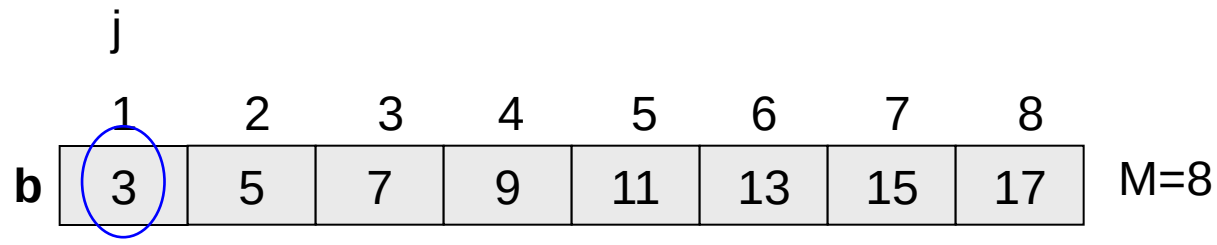
Se *a[i]* < *b[j]* então *a[i]* deve ser inserido em *c[k]* senão *b[j]* deve ser inserido em *c[k]*

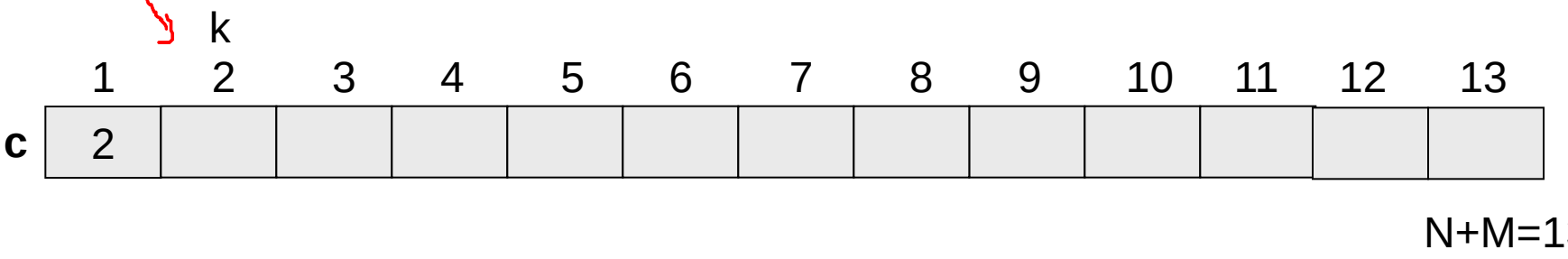
5.4.1. Operação de Fusão

Exemplo 5.4.1.



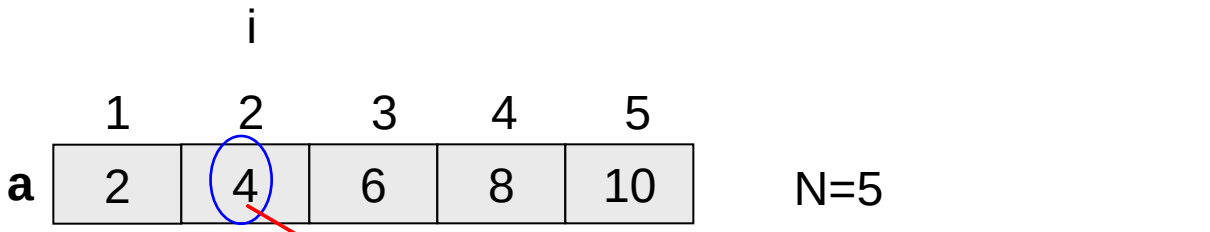
Se *a[i] < b[j]* então *a[i]* deve ser inserido em *c[k]* senão *b[j]* deve ser inserido em *c[k]*



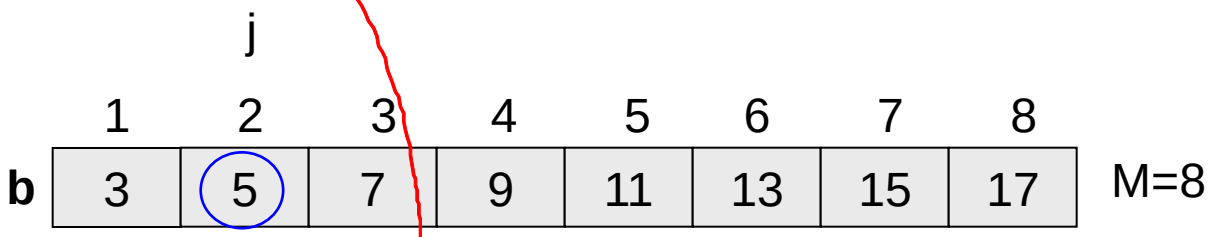


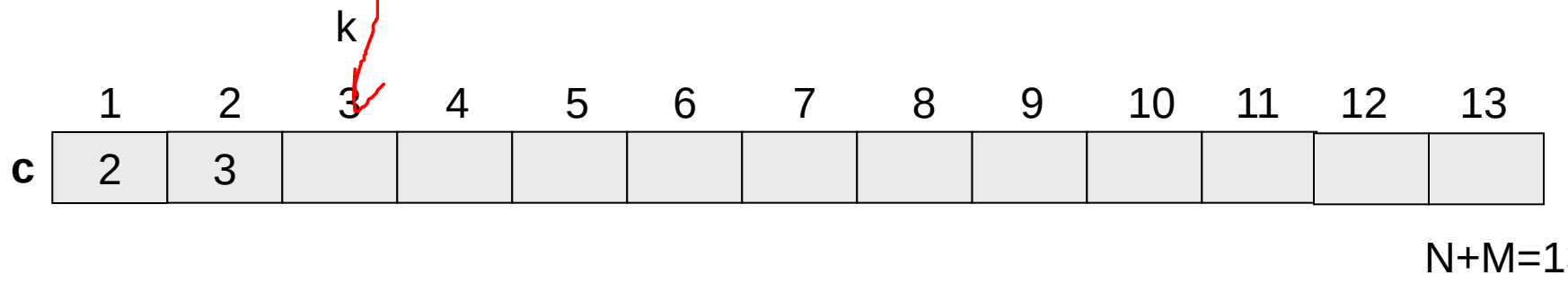
5.4.1. Operação de Fusão

Exemplo 5.4.1.



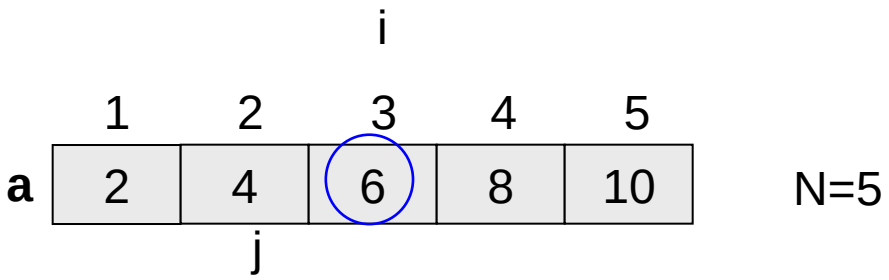
Se *a[i] < b[j]* então *a[i]* deve ser inserido em *c[k]* senão *b[j]* deve ser inserido em *c[k]*





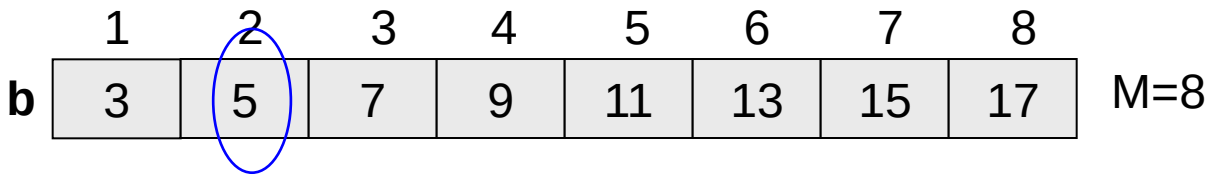
5.4.1. Operação de Fusão

Exemplo 5.4.1.



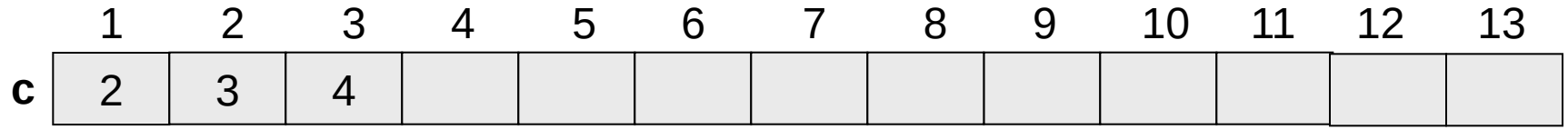
N=5

Se *a[i] < b[j]* então *a[i]* deve ser inserido em *c[k]* senão *b[j]* deve ser inserido em *c[k]*



M=8

k

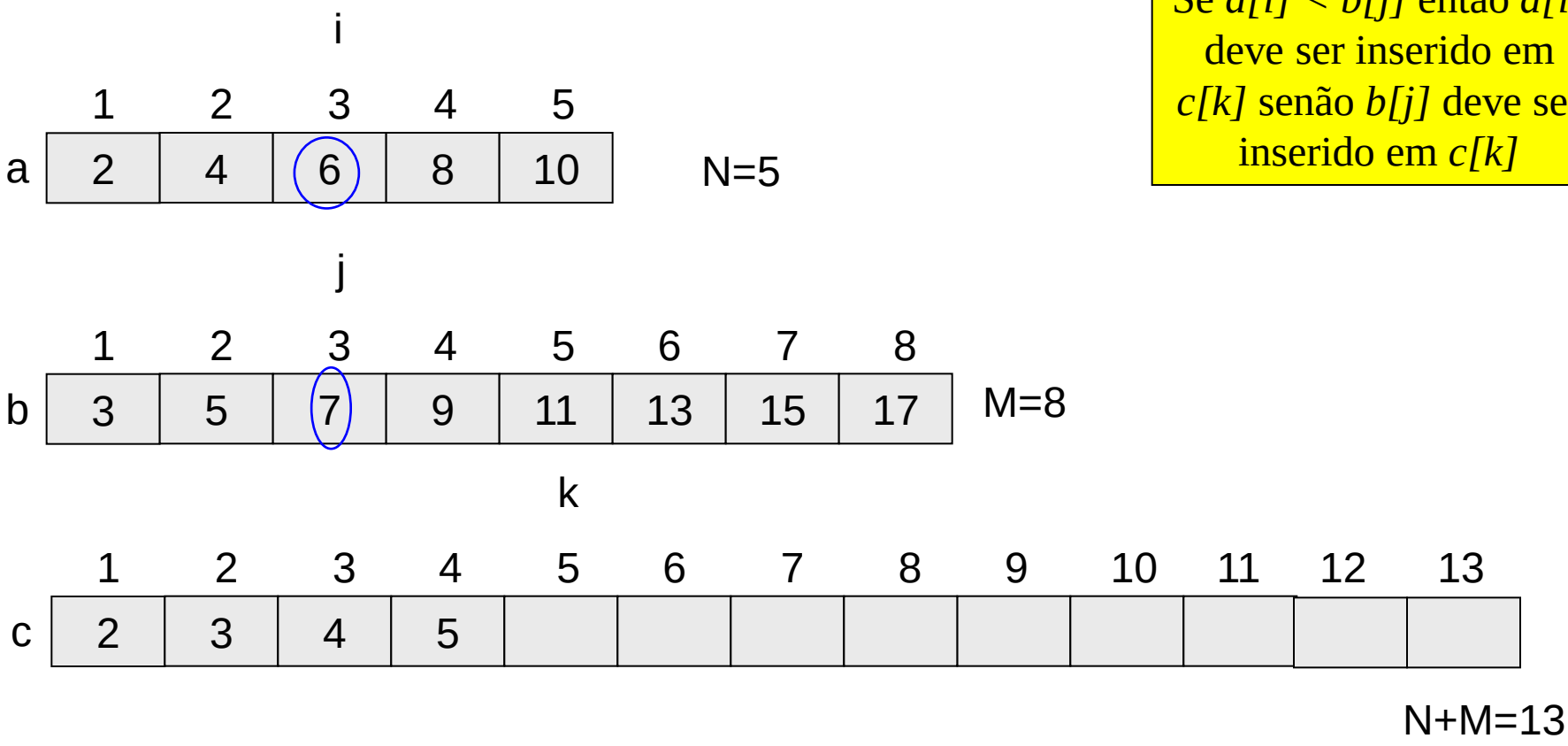


N+M=13

5.4.1. Operação de Fusão

Exemplo 5.4.1.

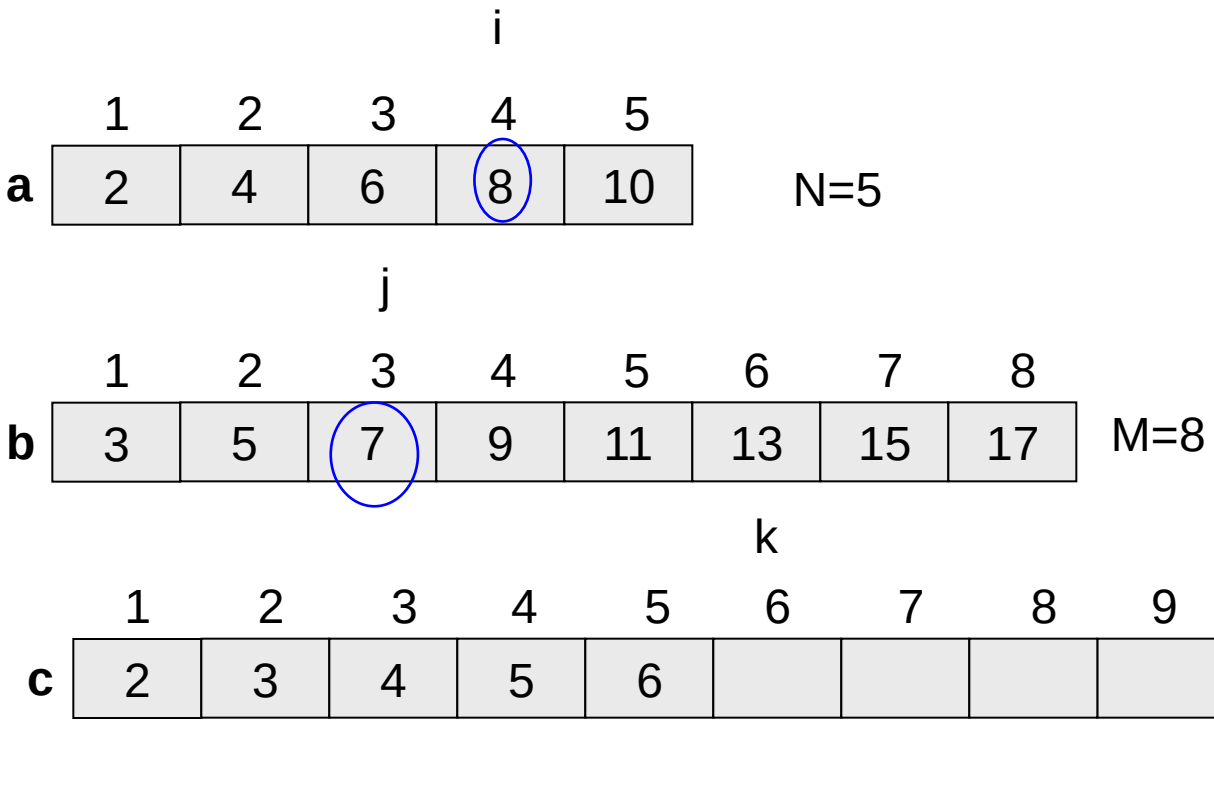
Se *a[i]* < *b[j]* então *a[i]* deve ser inserido em *c[k]* senão *b[j]* deve ser inserido em *c[k]*



5.4.1. Operação de Fusão

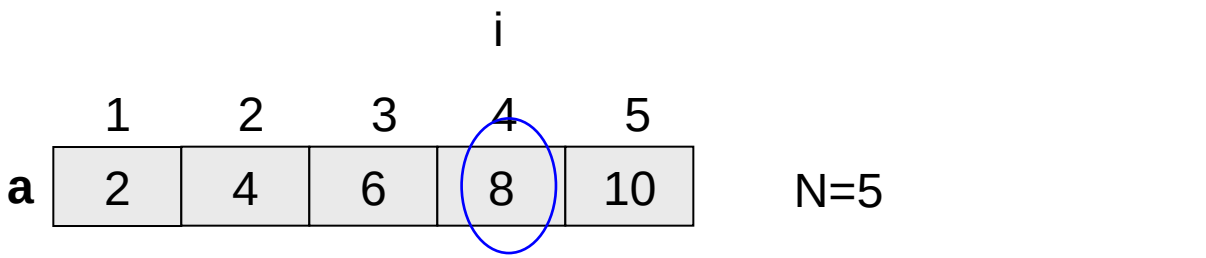
Exemplo 5.4.1.

Se *a[i]* < *b[j]* então *a[i]* deve ser inserido em *c[k]* senão *b[j]* deve ser inserido em *c[k]*

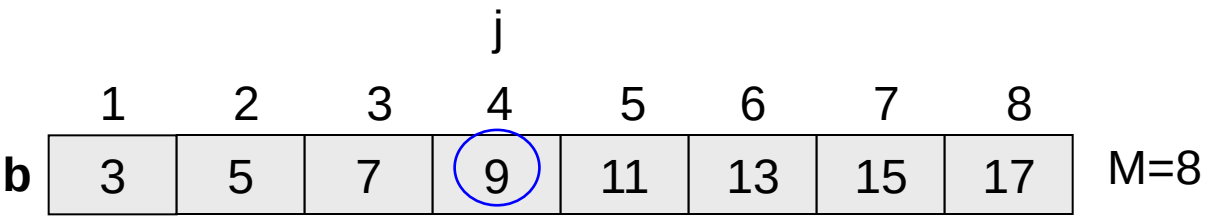


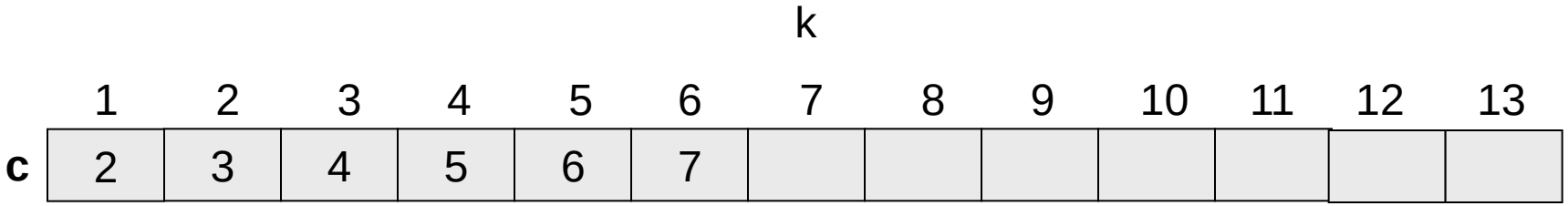
5.4.1. Operação de Fusão

Exemplo 5.4.1.



Se *a[i]* < *b[j]* então *a[i]* deve ser inserido em *c[k]* senão *b[j]* deve ser inserido em *c[k]*



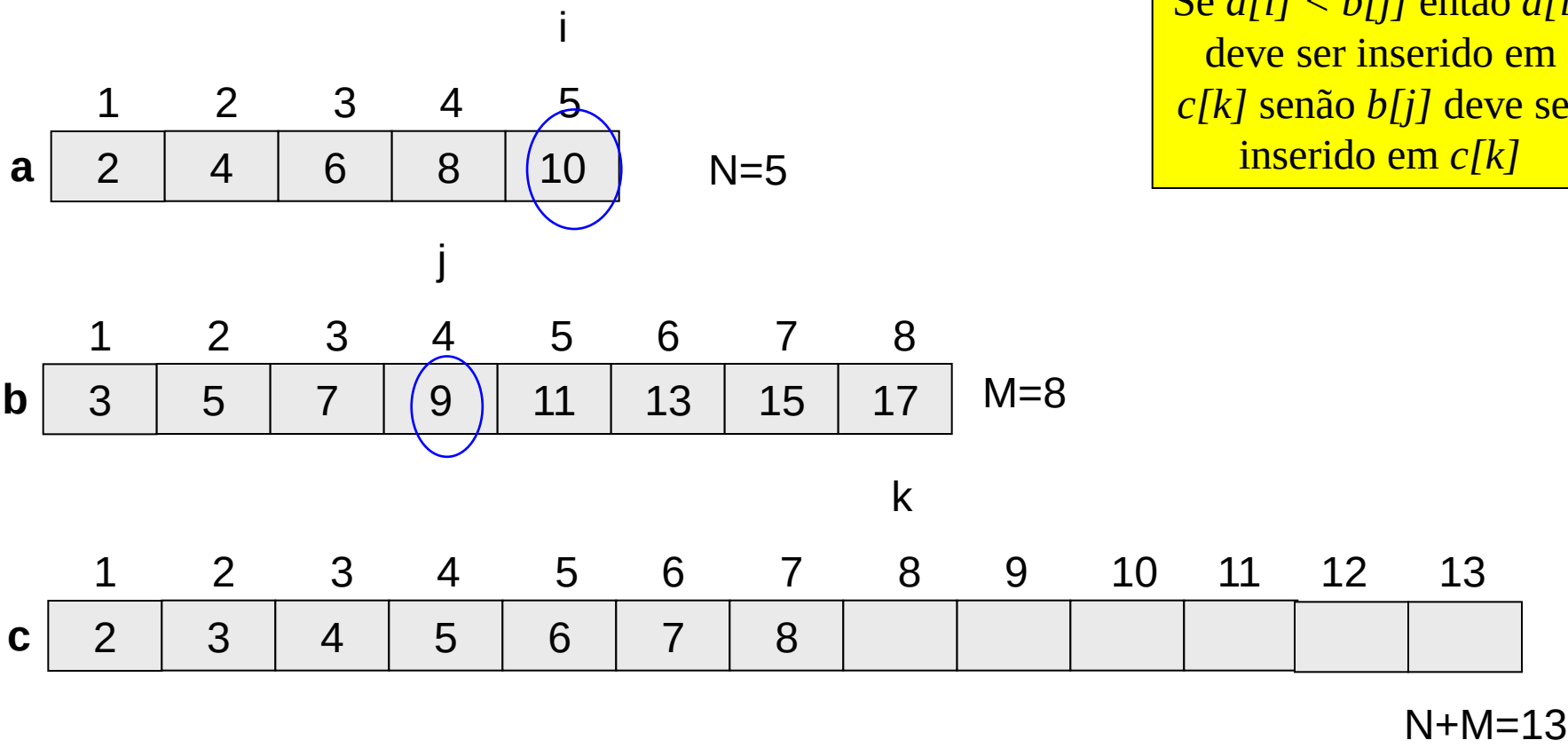


N+M=13

5.4.1. Operação de Fusão

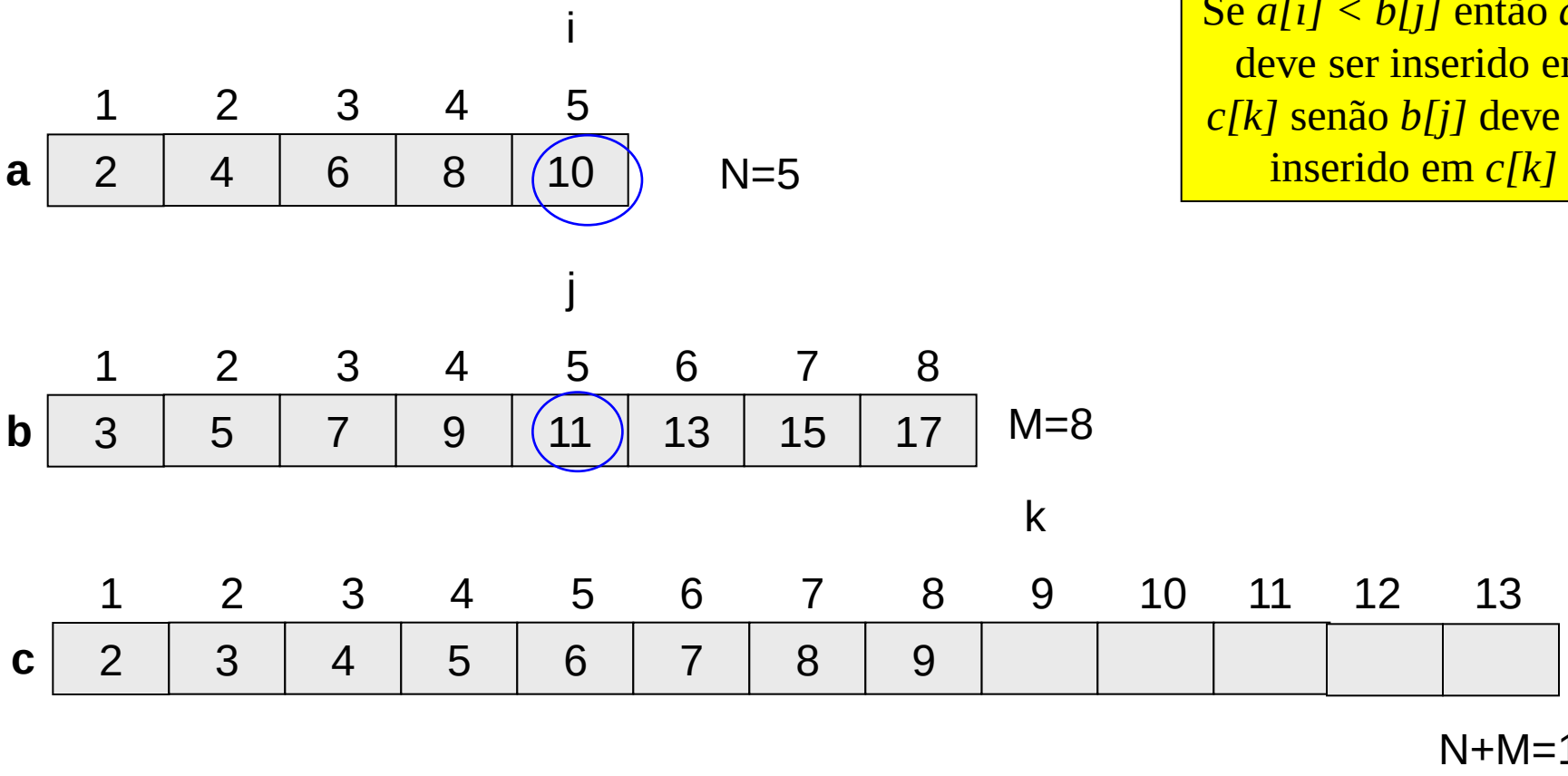
Exemplo 5.4.1.

Se *a[i]* < *b[j]* então *a[i]* deve ser inserido em *c[k]* senão *b[j]* deve ser inserido em *c[k]*



5.4.1. Operação de Fusão

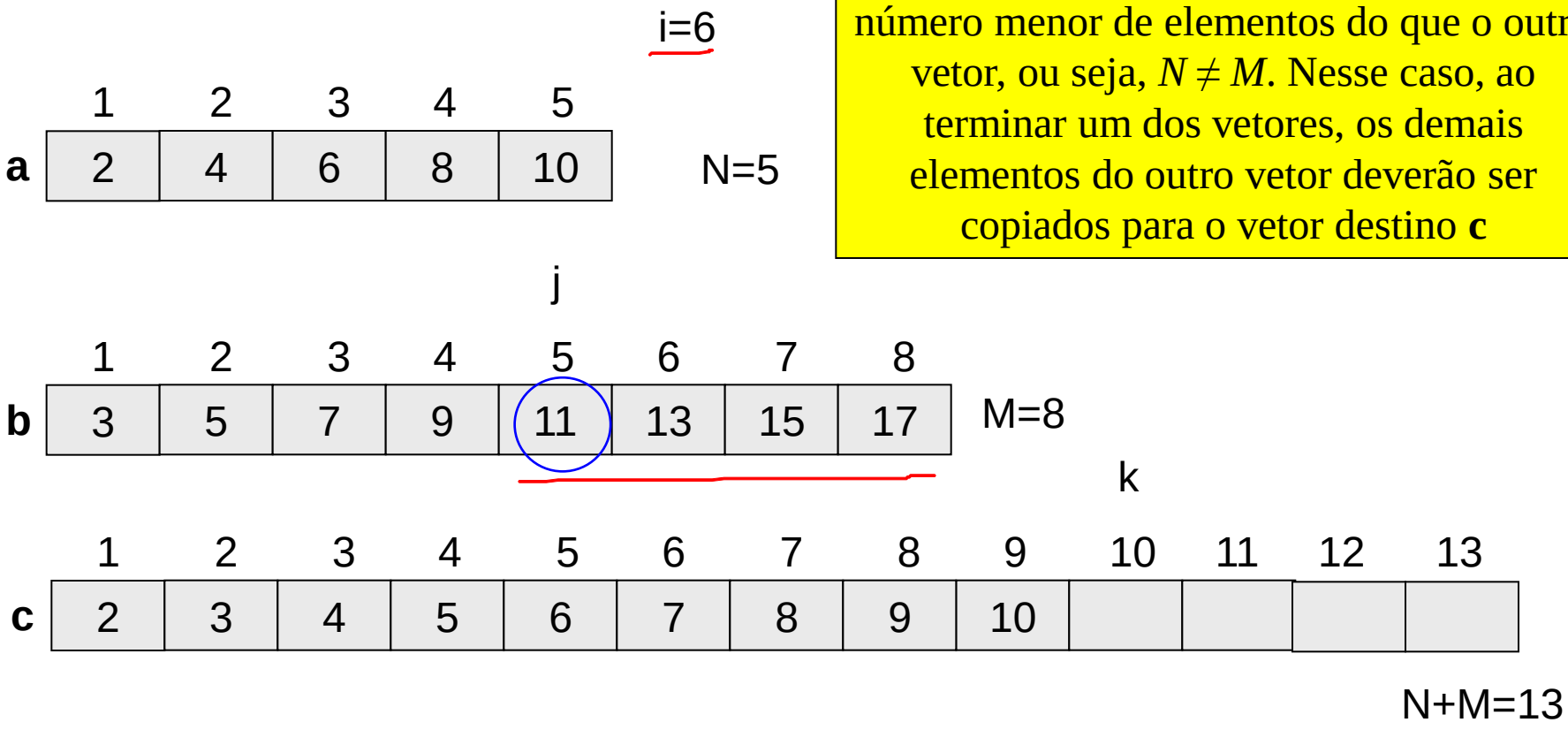
Exemplo 5.4.1.



Se *a[i]* < *b[j]* então *a[i]* deve ser inserido em *c[k]* senão *b[j]* deve ser inserido em *c[k]*

5.4.1. Operação de Fusão

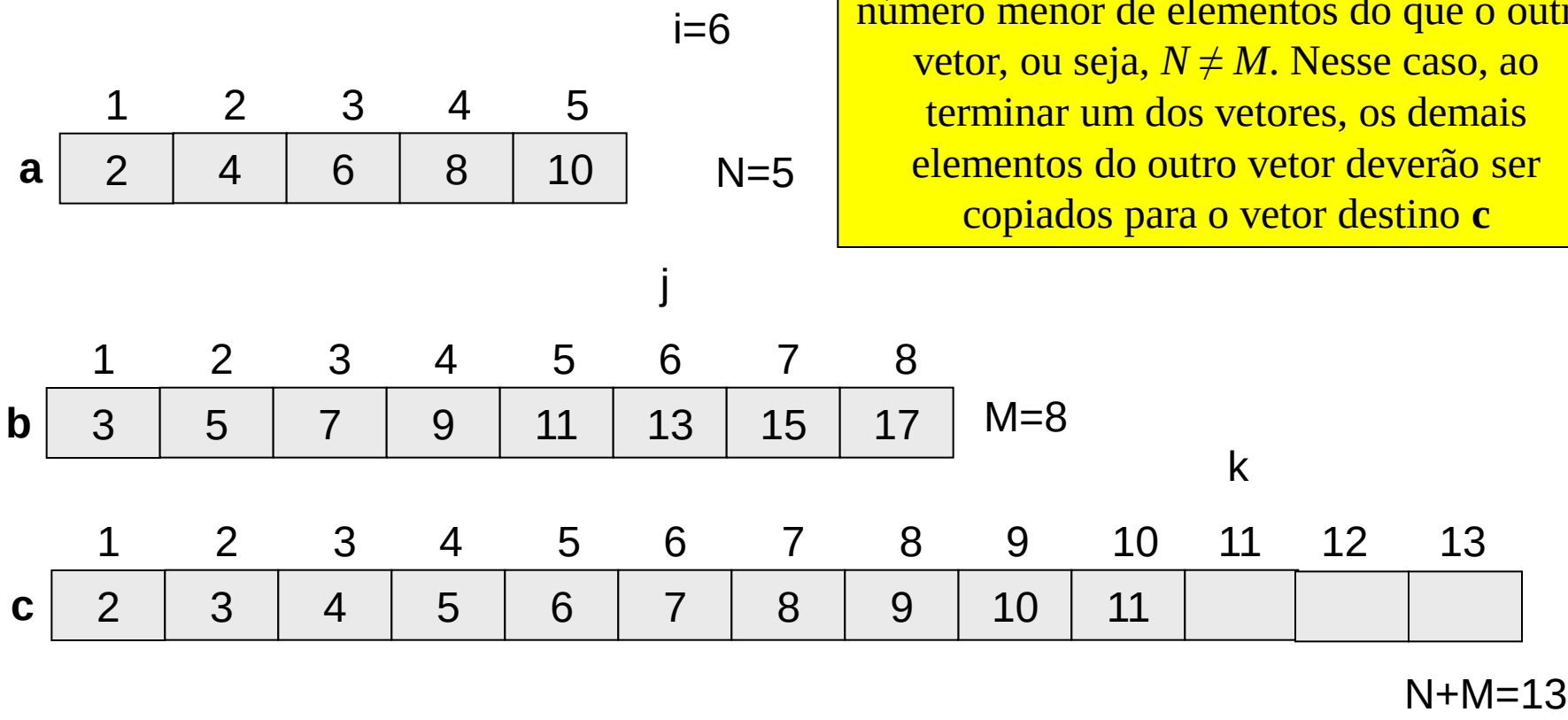
Exemplo 5.4.1.



Um dos vetores (**a** ou **b**) pode ter um número menor de elementos do que o outro vetor, ou seja, $N \neq M$. Nesse caso, ao terminar um dos vetores, os demais elementos do outro vetor deverão ser copiados para o vetor destino **c**

5.4.1. Operação de Fusão

Exemplo 5.4.1.

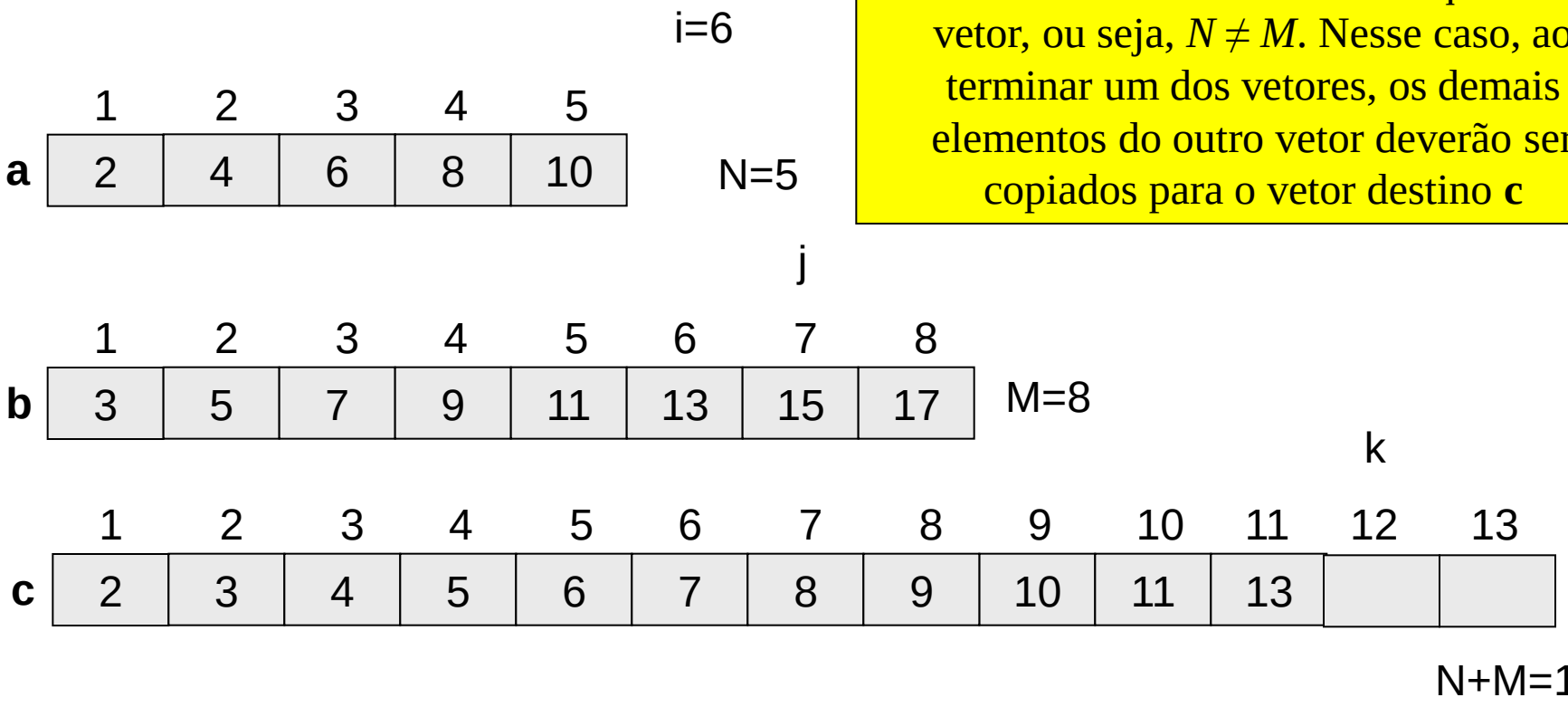


Um dos vetores (**a** ou **b**) pode ter um número menor de elementos do que o outro vetor, ou seja, $N \neq M$. Nesse caso, ao terminar um dos vetores, os demais elementos do outro vetor deverão ser copiados para o vetor destino **c**

5.4.1. Operação de Fusão

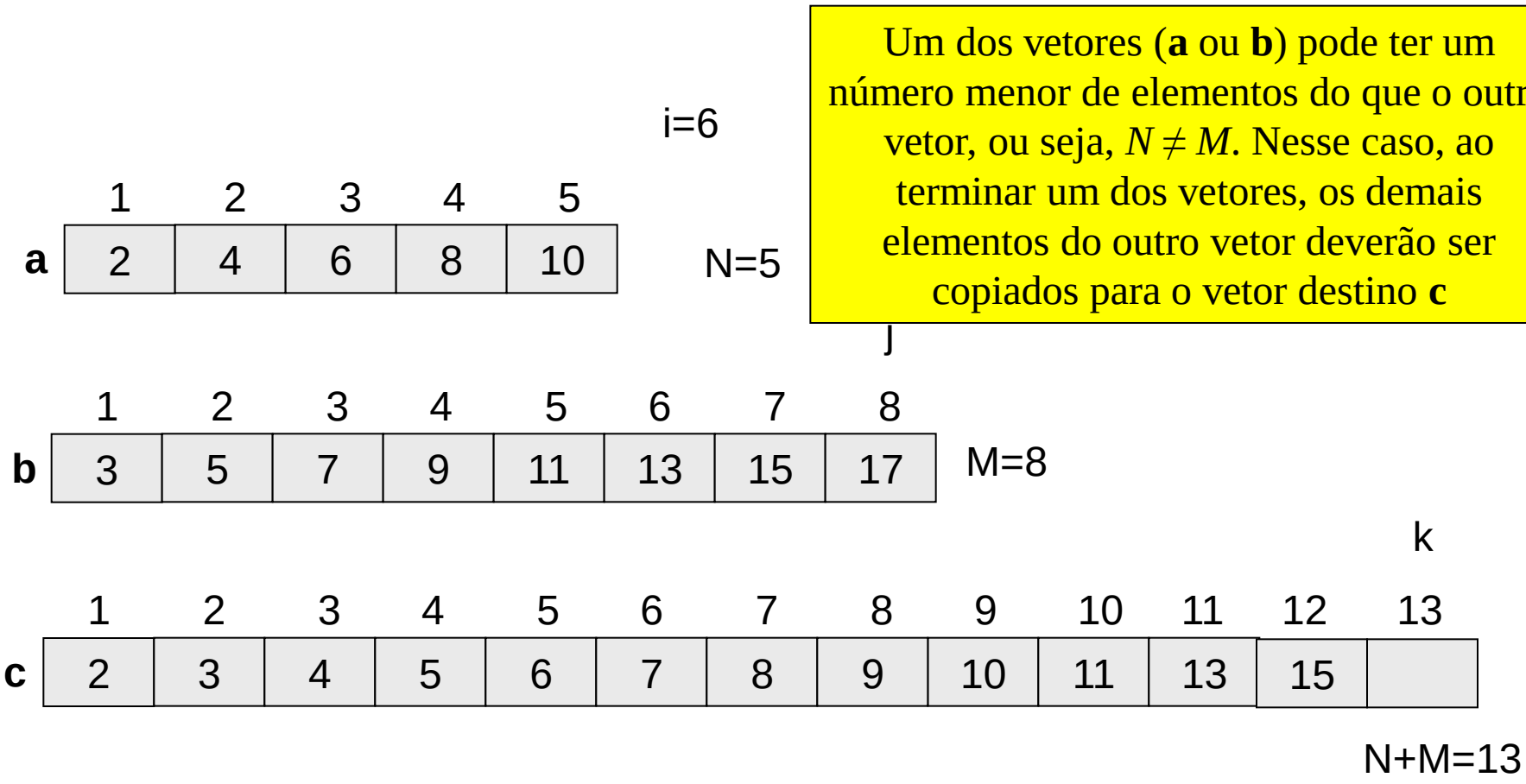
Exemplo 5.4.1.

Um dos vetores (**a** ou **b**) pode ter um número menor de elementos do que o outro vetor, ou seja, $N \neq M$. Nesse caso, ao terminar um dos vetores, os demais elementos do outro vetor deverão ser copiados para o vetor destino **c**



5.4.1. Operação de Fusão

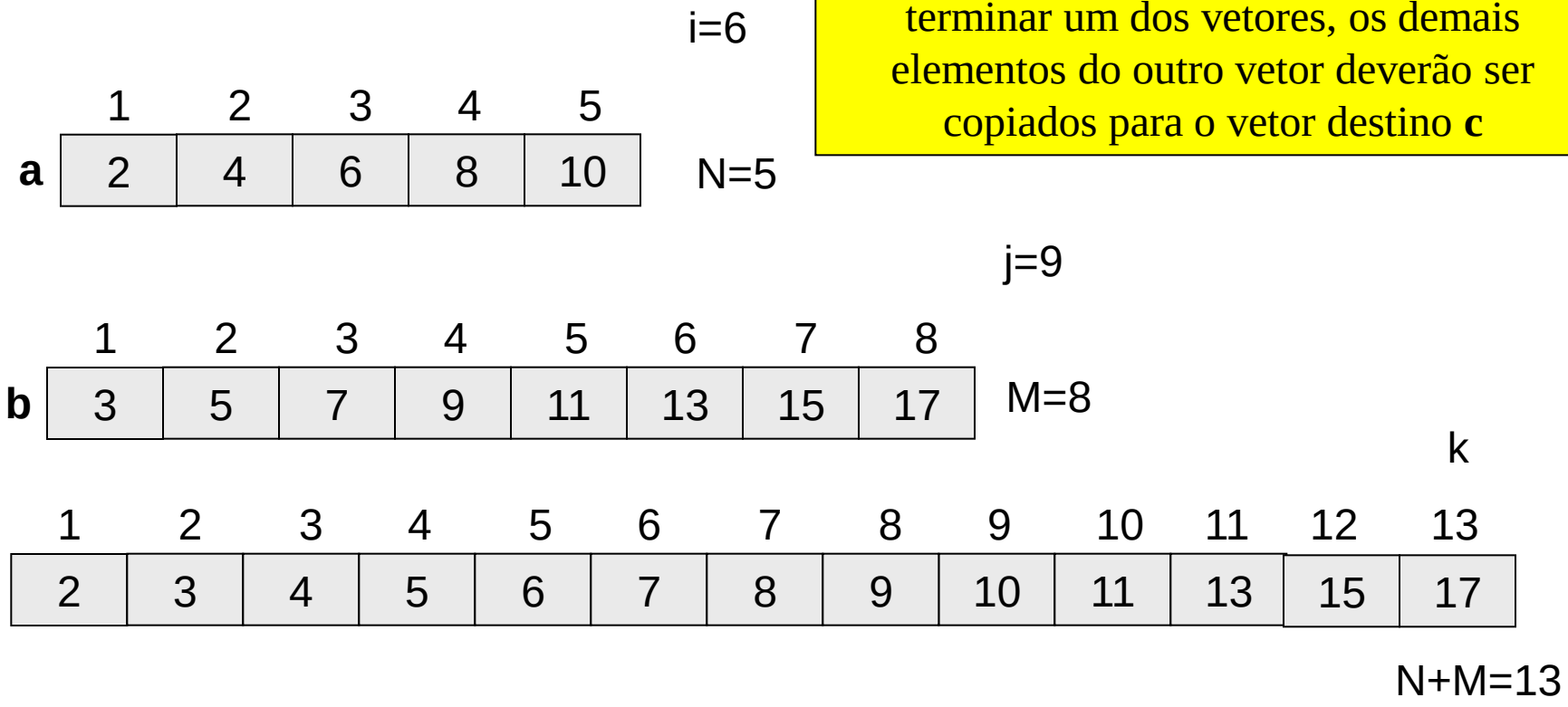
Exemplo 5.4.1.



5.4.1. Operação de Fusão

Exemplo 5.4.1.

Um dos vetores (**a** ou **b**) pode ter um número menor de elementos do que o outro vetor, ou seja, $N \neq M$. Nesse caso, ao terminar um dos vetores, os demais elementos do outro vetor deverão ser copiados para o vetor destino **c**



5.4.1. Operação de Fusão

Exercício 5.4.1. Desenvolva um algoritmo para realizar a operação de fusão de dois vetores descrita no exemplo anterior.

5.4.1. Operação de Fusão

enquanto não chegou
no fim de um vetor

```
...
i ← 1
j ← 1
k ← 0
enquanto (i ≤ N ) e ( j ≤ M)
  k ← k + 1
  se ( a[i] < b[j] )
            c[k] ← a[i]
            i ← i + 1
  senão
            c[k] ← b[j]
            j ← j + 1
  fim se
fim enquanto
```

```
enquanto (i ≤ N )
  k ← k + 1
  c[k] ← a[i]
  i ← i + 1
}
```

chegou no fim do vetor b

```
enquanto (j ≤ M )
  k ← k + 1
  c[k] ← b[j];
  j ← j + 1
}
...
```

chegou no fim do vetor a

Ao término, $k=N+M$ que é o número de elementos do vetor **c**

5.4.1. Operação de Fusão

- Este mesmo algoritmo pode ser facilmente alterado para a situação na qual os elementos a serem intercalados encontram-se em um mesmo vetor

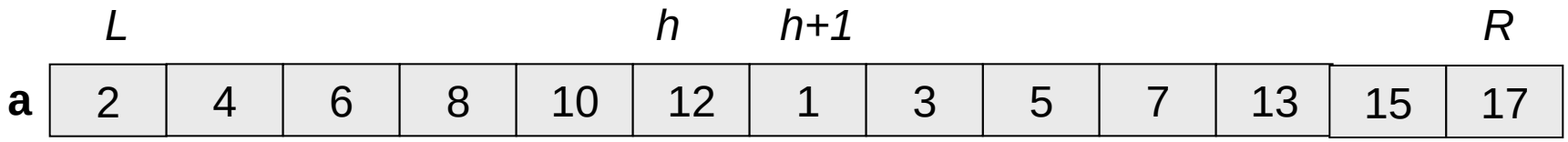
- Nesse caso, deseja-se intercalar as seqüências ordenadas *a[L],..., a[h]* e *a[h+1],..., a[R]* para obter o vetor *c[L], ..., c[R]* , também ordenado

5.4.1. Operação de Fusão

Algoritmo merge(*a[], L , h , R , c[]*)

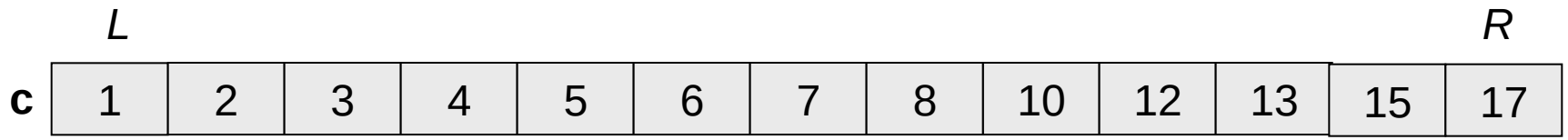
pré:(*a[L],...,a[h]*) e (*a[h+1],...,a[R]*) são duas seqüências ordenadas com chaves *a[L]≤ ... ≤ a[h]* e *a[h+1] ≤ ... ≤ a[R]*

Por exemplo:



pós:(*a[L],...,a[h]*) e (*a[h+1],...,a[R]*) são intercaladas para obter a seqüência *(c[L],...,c[R]) tal que c[L]≤ ... ≤ c[R]*

Por exemplo:



5.4.1. Operação de Fusão

Exercício 5.4.2. Desenvolva um algoritmo para realizar a operação de fusão em um único vetor descrita no exemplo anterior.

5.4.1. Operação de Fusão

primeira das 3 funções para a ordenação por fusão

Algoritmo merge(*a[ ] ,L ,h ,R ,c[ ]*)

pré:(*a[L],...,a[h]*) e (*a[h+1],...,a[R]*) são duas seqüências ordenadas com chaves *a[L]≤ ... ≤ a[h]* e *a[h+1] ≤ ... ≤ a[R]*

pós:(*a[L],...,a[h]*) e (*a[h+1],...,a[R]*) são intercaladas para obter a seqüência *(c[L],...,c[R]) tal que c[L]≤ ... ≤ c[R]*

```
i ← L
j ← h + 1
k ← L - 1
```

```
enquanto ( i ≤ h  e  j ≤ R )
    k ← k + 1
    se ( a[ i ] < a[ j ] )
            c[ k ] ← a[ i ]
            i ← i + 1
    senão
            c[k] ← a[j]
            j ← j + 1
    fim se
fim enquanto
enquanto ( i ≤ h )
    k ← k + 1
    c[ k ] ← a[ i ]
    i ← i + 1
fim enquanto
enquanto ( j ≤ R )
    k ← k + 1
    c[ k ] ← a[ j ]
    j ← j + 1
fim enquanto
```

5.4.1. Operação de Fusão

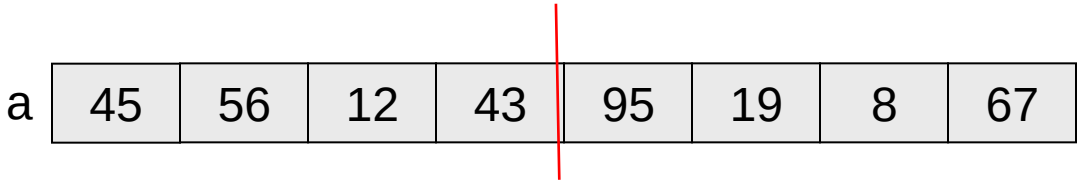
- Análise
  - Em cada iteração do laço **enquanto**, *k* incrementa de 1
  - O laço será executado no máximo *R-L+1* vezes
  - O comando **se** move, no máximo, um elemento por iteração
  - Por isso, o tempo é proporcional a (*R-L+1*)
  - Repare que *R-L+1* é a quantidade de elementos a serem intercalados. No caso de todo o vetor (*L=1* e *R=N*), temos complexidade *O(N)*

5.4.2. Ordenação por Fusão Direta

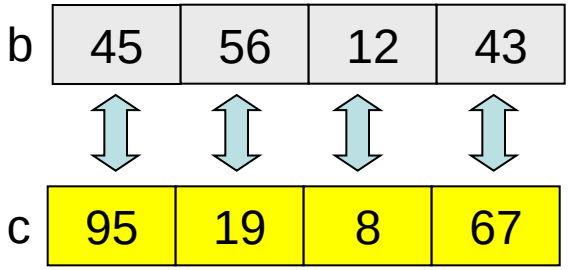
1. Dividir a seqüência inicial **a** em duas seqüências, **b** e **c**;
2. Fundir **b** e **c** por meio da combinação de elementos isolados para formarem pares ordenados;
3. Denominar **a** a seqüência assim obtida e repetir os passos 1 e 2, desta vez efetuando a fusão de pares ordenados em quádruplas ordenadas;
4. Repetir os passos anteriores, executando a fusão de quádruplas em óctuplas, e assim prosseguindo duplicando a cada vez o comprimento das subseqüências envolvidas no processo de fusão, até que toda seqüência esteja ordenada.

5.4.2. Ordenação por Fusão Direta

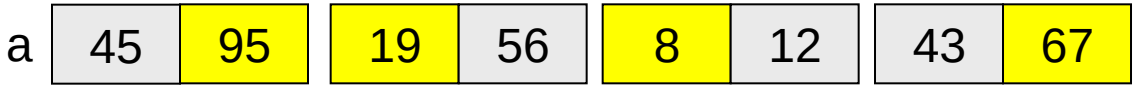
Exemplo 5.4.2. Considere o vetor



- que é particionado em

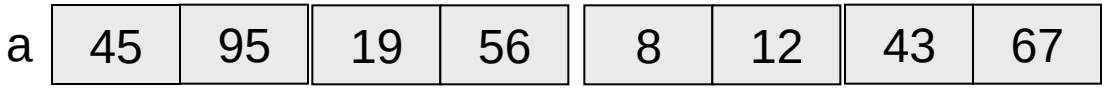


- A fusão de elementos isolados em pares ordenados resulta em

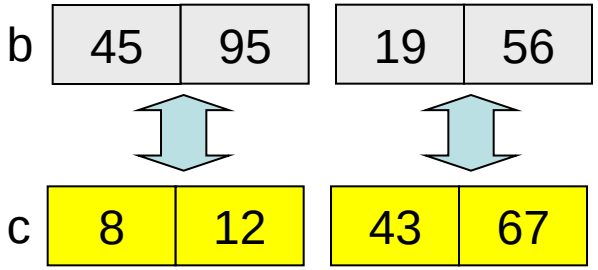


5.4.2. Ordenação por Fusão Direta

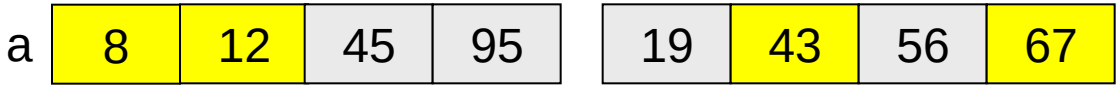
- Particionando-se novamente



- Obtém-se

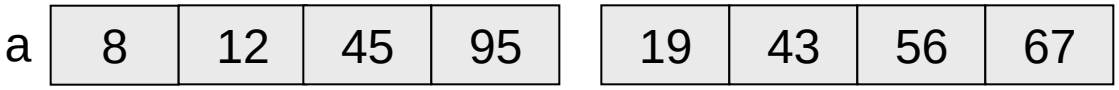


- A fusão de pares ordenados em quádruplas resulta em

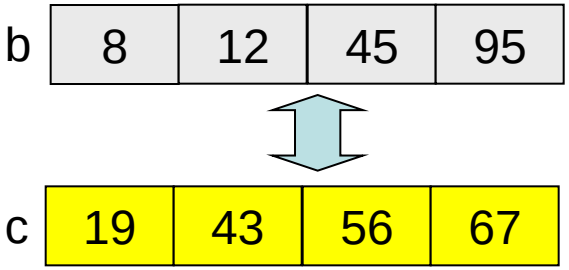


5.4.2. Ordenação por Fusão Direta

- Particionando o vetor obtido



- Obtém-se dois vetores ordenados



- A fusão de quáduplas ordenadas em óctuplas resulta em

5.4.2. Ordenação por Fusão Direta

- No contexto do algoritmo de fusão,
  - cada operação que trata de uma só vez todo o conjunto de dados é denominada **fase**
  - o menor subprocesso que, por sua repetição sucessiva, implementa o processo de ordenação propriamente dito é chamado **passo** ou **estágio**
- No exemplo anterior, a ordenação foi realizada em três passos, cada qual consistindo de uma **fase de particionamento** e uma **fase de fusão**
- Como pode ser observado, para que seja possível a realização da ordenação, são necessárias três seqüências

5.4.2. Ordenação por Fusão Direta

- Na realidade, as fases de particionamento não oferecem nenhuma contribuição ao processo de ordenação
  - Essas fases constituem a metade de todas as operações de movimentações e podem ser eliminadas através da combinação da fase de particionamento com a de fusão

- Ao invés de efetuar uma fusão para produzir uma seqüência única, o resultado do processo de fusão é imediatamente redistribuído em duas seqüências, as quais constituirão as fontes de dados que alimentarão os passo seguinte

5.4.2. Ordenação por Fusão Direta

- Em contraste com a ordenação descrita anteriormente e conhecida como **fusão de duas fases**, este novo método é denominado **fusão de fase única** ou **fusão balanceada**

- A fusão de fase única é mais interessante, uma vez que são necessárias somente a metade das operações de movimentações exigidas na fusão de duas fases

5.4.2. Ordenação por Fusão Direta

- A ideia básica é realizar várias passagens de fusão sobre os elementos do vetor

- Na primeira passagem, são intercaladas seqüências de tamanho 1, na segunda o tamanho das seqüências é 2 e na *i*-ésima passagem as seqüências intercaladas são de tamanho 2^{i-1}

- O algoritmo ***mpass*** realiza uma passagem de ordenação por fusão

5.4.2. Ordenação por Fusão Direta

```
Algoritmo mpass( a[ ] , N , p ,c[ ]  )
/*
pré: N>0 é o número de elementos do vetor a e p é o tamanho das
subseqüências de a que serão intercaladas
pós: Intercala pares adjacentes de comprimento p da seqüência a para c
*/
 i ← 1
enquanto ( i ≤ N-2*p+1 )
     merge( a, i , i+p-1 , i+2*p-1 , c )
     i ← i + 2*p
fim enquanto
// intercalar restante de comprimento < 2*p
se ( i+p-1 < N )
    merge( a , i , i+p-1 , N , c )
senão
    para j ← i até j ← N
            c[ j ] ← a[ j ]
    fim para
fim se
```

segunda das 3 funções para a ordenação por fusão

5.4.2. Ordenação por Fusão Direta

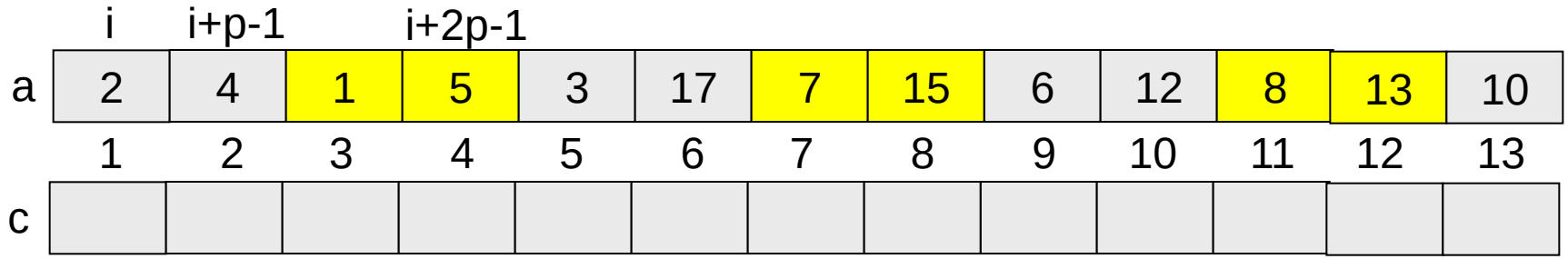
```
Algoritmo mpass( a[ ] , N , p ,c[ ]  )

i ← 1
enquanto ( i ≤ N-2*p+1 )
     merge( a, i , i+p-1 , i+2*p-1 , c )
     i ← i + 2*p
fim enquanto
se ( i+p-1 < N )
    merge( a , i , i+p-1 , N , c )
senão
    para j ← i até j ← N
            c[ j ] ← a[ j ]
    fim para
fim se
```

fundir de 2 em 2

p=2

N=13

N-2p+1



5.4.2. Ordenação por Fusão Direta

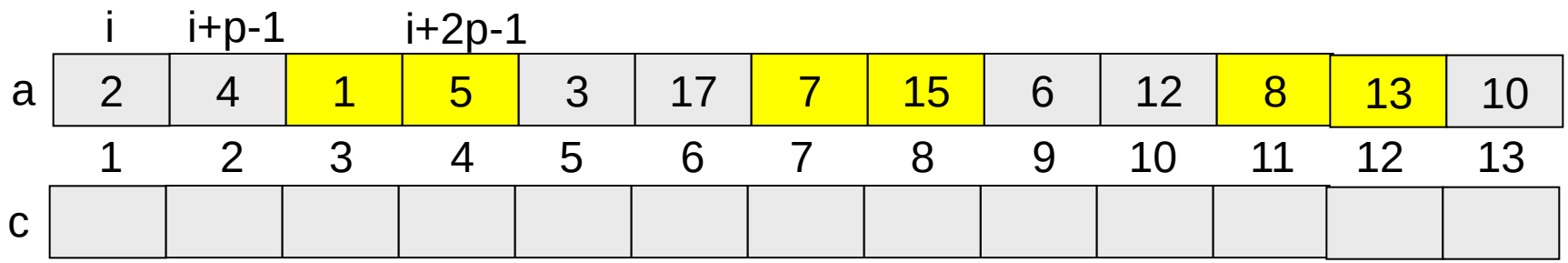
```
Algoritmo mpass( a[ ] , N , p ,c[ ]  )

 i ← 1
 enquanto ( i ≤ N-2*p+1 )
⇨  merge( a, i , i+p-1 , i+2*p-1 , c )
    i ← i + 2*p
 fim enquanto
 se ( i+p-1 < N )
    merge( a , i , i+p-1 , N , c )
 senão
    para j ← i até j ← N
            c[ j ] ← a[ j ]
    fim para
 fim se
```

```
merge( a , L , h , R , c )
```

p=2

N=13

N-2p+1



5.4.2. Ordenação por Fusão Direta

```
Algoritmo mpass( a[ ] , N , p ,c[ ]  )

 i ← 1
 enquanto ( i ≤ N-2*p+1 )
    merge( a, i , i+p-1 , i+2*p-1 , c )
    i ← i + 2*p
 fim enquanto
 se ( i+p-1 < N )
    merge( a , i , i+p-1 , N , c )
 senão
    para j ← i até j ← N
          c[ j ] ← a[ j ]
    fim para
 fim se
```

```
merge( a , L , h , R , c )
```

p=2

N=13

L h h+1 R N-2p+1

i i+p-1 i+2p-1

	1	2	3	4	5	6	7	8	9	10	11	12	13
a	2	4	1	5	3	17	7	15	6	12	8	13	10
c													

5.4.2. Ordenação por Fusão Direta

```
Algoritmo mpass( a[ ] , N , p ,c[ ]  )

 i ← 1
 enquanto ( i ≤ N-2*p+1 )
    merge( a, i , i+p-1 , i+2*p-1 , c )
    i ← i + 2*p
 fim enquanto
 se ( i+p-1 < N )
    merge( a , i , i+p-1 , N , c )
 senão
    para j ← i até j ← N
          c[ j ] ← a[ j ]
    fim para
 fim se
```

```
merge( a , L , h , R , c )
```

p=2

N=13

L h h+1 R N-2p+1

i i+p-1 i+2p-1

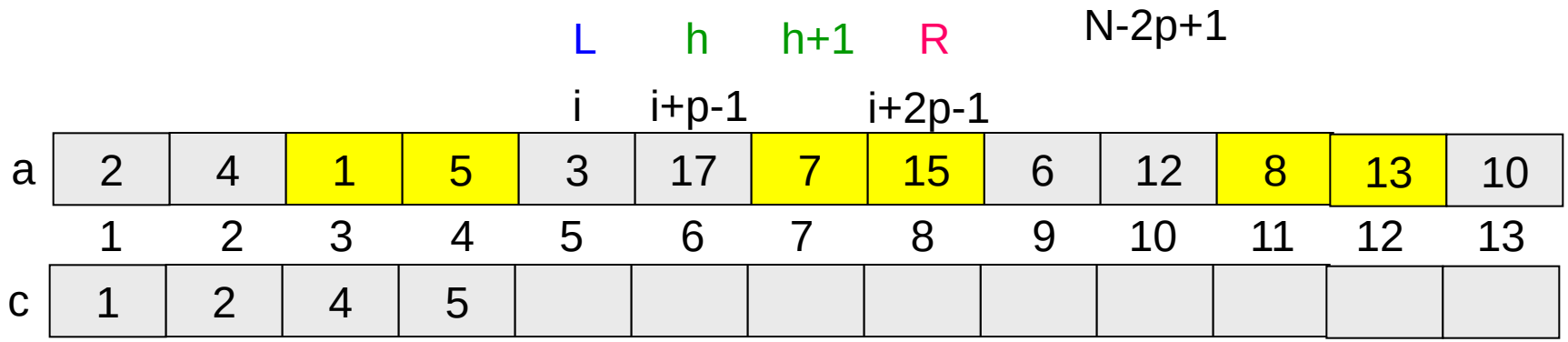
a	2	4	1	5	3	17	7	15	6	12	8	13	10
	1	2	3	4	5	6	7	8	9	10	11	12	13
c	1	2	4	5									

5.4.2. Ordenação por Fusão Direta

```
Algoritmo mpass( a[ ] , N , p ,c[ ]  )

i ← 1
enquanto ( i ≤ N-2*p+1 )
     merge( a, i , i+p-1 , i+2*p-1 , c )
     i ← i + 2*p
fim enquanto
se ( i+p-1 < N )
    merge( a , i , i+p-1 , N , c )
senão
    para j ← i até j ← N
            c[ j ] ← a[ j ]
    fim para
fim se
```

p=2

N=13



5.4.2. Ordenação por Fusão Direta

```
Algoritmo mpass( a[ ] , N , p ,c[ ]  )

i ← 1
enquanto ( i ≤ N-2*p+1 )
     merge( a, i , i+p-1 , i+2*p-1 , c )
     i ← i + 2*p
fim enquanto
se ( i+p-1 < N )
    merge( a , i , i+p-1 , N , c )
senão
    para j ← i até j ← N
            c[ j ] ← a[ j ]
    fim para
fim se
```

p=2

N=13

L h h+1 R N-2p+1

i i+p-1 i+2p-1

a	2	4	1	5	3	17	7	15	6	12	8	13	10
	1	2	3	4	5	6	7	8	9	10	11	12	13
c	1	2	4	5	3	7	15	17					

5.4.2. Ordenação por Fusão Direta

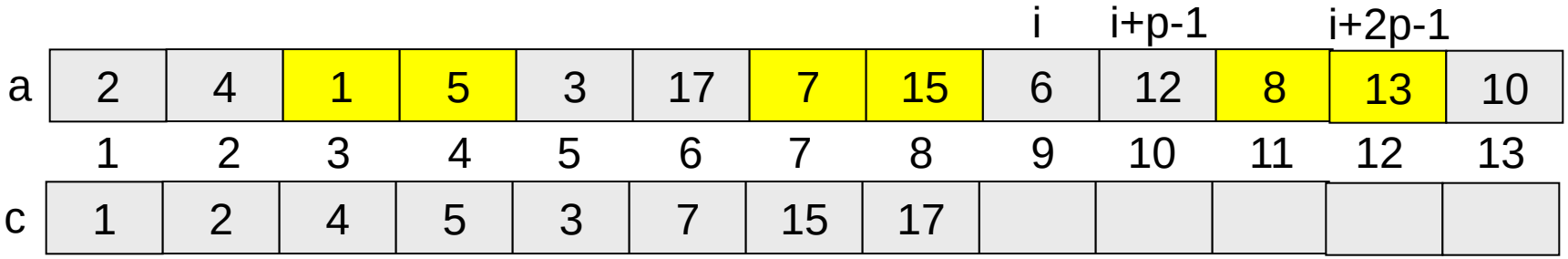
```
Algoritmo mpass( a[ ] , N , p ,c[ ]  )

i ← 1
enquanto ( i ≤ N-2*p+1 )
     merge( a, i , i+p-1 , i+2*p-1 , c )
     i ← i + 2*p
fim enquanto
se ( i+p-1 < N )
    merge( a , i , i+p-1 , N , c )
senão
    para j ← i até j ← N
            c[ j ] ← a[ j ]
    fim para
fim se
```

p=2

N=13

L h h+1 R

N-2p+1



5.4.2. Ordenação por Fusão Direta

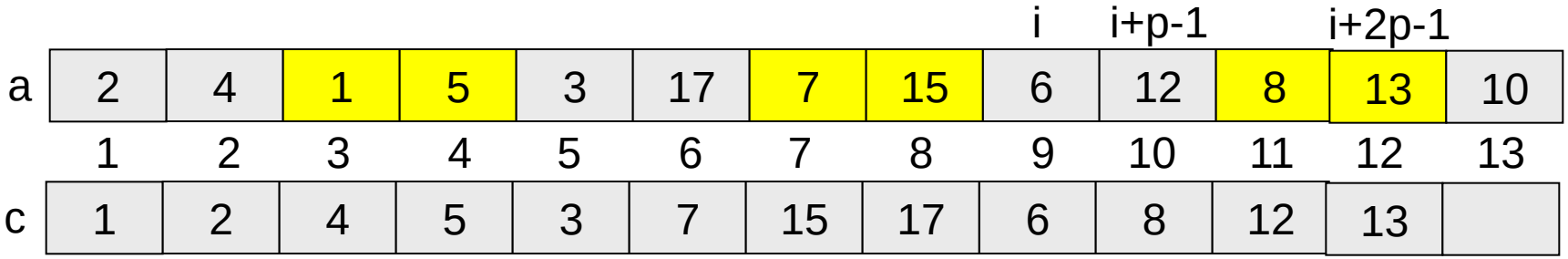
```
Algoritmo mpass( a[ ] , N , p ,c[ ]  )

i ← 1
enquanto ( i ≤ N-2*p+1 )
     merge( a, i , i+p-1 , i+2*p-1 , c )
     i ← i + 2*p
fim enquanto
se ( i+p-1 < N )
    merge( a , i , i+p-1 , N , c )
senão
    para j ← i até j ← N
            c[ j ] ← a[ j ]
    fim para
fim se
```

p=2

N=13

L h h+1 R

N-2p+1



5.4.2. Ordenação por Fusão Direta

```
Algoritmo mpass( a[ ] , N , p ,c[ ]  )

i ← 1
enquanto ( i ≤ N-2*p+1 )
     merge( a, i , i+p-1 , i+2*p-1 , c )
     i ← i + 2*p
fim enquanto
se ( i+p-1 < N )
    merge( a , i , i+p-1 , N , c )
senão
    para j ← i até j ← N
            c[ j ] ← a[ j ]
    fim para
fim se
```

caso que tenho 2p elementos

caso que tenho menos que 2p elementos, mas mais que p elementos

caso que tenho menos que p elementos

p=2

N=13

N-2p+1

i i+p-1

a	2	4	1	5	3	17	7	15	6	12	8	13	10
	1	2	3	4	5	6	7	8	9	10	11	12	13
c	1	2	4	5	3	7	15	17	6	8	12	13	

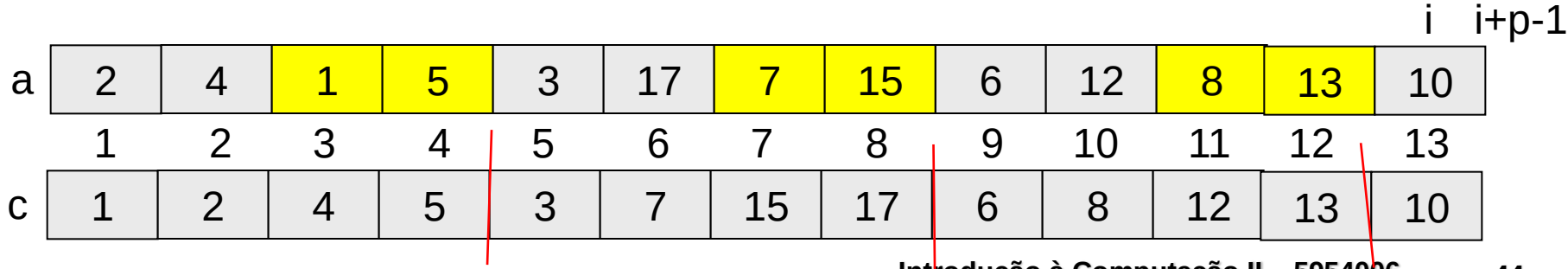
5.4.2. Ordenação por Fusão Direta

```
Algoritmo mpass( a[ ] , N , p ,c[ ]  )

i ← 1
enquanto ( i ≤ N-2*p+1 )
     merge( a, i , i+p-1 , i+2*p-1 , c )
     i ← i + 2*p
fim enquanto
se ( i+p-1 < N )
    merge( a , i , i+p-1 , N , c )
senão
    para j ← i até j ← N
            c[ j ] ← a[ j ]
    fim para
fim se
```

p=2

N=13

N-2p+1



5.4.2. Ordenação por Fusão Direta

```
Algoritmo mpass( a[ ] , N , p ,c[ ]  )

i ← 1
enquanto ( i ≤ N-2*p+1 )
     merge( a, i , i+p-1 , i+2*p-1 , c )
     i ← i + 2*p
fim enquanto
se ( i+p-1 < N )
    merge( a , i , i+p-1 , N , c )
senão
    para j ← i até j ← N
            c[ j ] ← a[ j ]
    fim para
fim se
```

Término da passagem para p=2

a	2	4	1	5	3	17	7	15	6	12	8	13	10
	1	2	3	4	5	6	7	8	9	10	11	12	13
c	1	2	4	5	3	7	15	17	6	8	12	13	10

5.4.2. Ordenação por Fusão Direta

terceira das 3 funções para a ordenação por fusão

```
Algoritmo mergesort( a[ ] , N )

p ← 1
 enquanto ( p < N )
     mpass( a, N , p , c )
     p ← 2*p
     mpass( c, N , p , a )
     p ← 2*p
 fim enquanto
```

p=1

N=13

a	4	2	5	1	17	3	7	15	12	6	13	8	10
	1	2	3	4	5	6	7	8	9	10	11	12	13
c	2	4	1	5	3	17	7	15	6	12	8	13	10

5.4.2. Ordenação por Fusão Direta

```
Algoritmo mergesort( a[ ] , N )

p ← 1
 enquanto ( p < N )
      mpass( a, N , p , c )
      p ← 2*p
      mpass( c, N , p , a )
      p ← 2*p
 fim enquanto
```

p=2

N=13

c	2	4	1	5	3	17	7	15	6	12	8	13	10
	1	2	3	4	5	6	7	8	9	10	11	12	13
a	1	2	4	5	3	7	15	17	6	8	12	13	10

5.4.2. Ordenação por Fusão Direta

```
Algoritmo mergesort( a[ ] , N )

p ← 1
 enquanto ( p < N )
     mpass( a, N , p , c )
     p ← 2*p
     mpass( c, N , p , a )
     p ← 2*p
 fim enquanto
```

p=4

N=13

a	1	2	4	5	3	7	15	17	6	8	12	13	10
	1	2	3	4	5	6	7	8	9	10	11	12	13
c	1	2	3	4	5	7	15	17	6	8	10	12	13

5.4.2. Ordenação por Fusão Direta

```
Algoritmo mergesort( a[ ] , N )

p ← 1
 enquanto ( p < N )
      mpass( a, N , p , c )
      p ← 2*p
      mpass( c, N , p , a )
      p ← 2*p
 fim enquanto
```

p=8

N=13

c	1	2	3	4	5	7	15	17	6	8	10	12	13
	1	2	3	4	5	6	7	8	9	10	11	12	13
a	1	2	3	4	5	6	7	8	10	12	13	15	17

5.4.2. Ordenação por Fusão Direta

Exercício 5.4.3. Utilizando ordenação pelo método mergesort, obtenha o número de comparações e movimentações (*p*) em cada passo para os seguintes vetores

- [45 56 12 43 95 19 8 67]
- [8 12 19 43 45 56 67 95]
- [95 67 56 45 43 19 12 8]
- [19 12 8 45 43 56 67 95]

5.4.2. Ordenação por Fusão Direta

Exercício 5.4.3. Solução

p	Ci	Mi	45	56	12	43	95	19	8	67
1	4	8	45	56	12	43	19	95	8	67
2	5	8	12	43	45	56	8	19	67	95
4	6	8	8	12	19	43	45	56	67	95
8	0	8	8	12	19	43	45	56	67	95
	15	32								

p	Ci	Mi	8	12	19	43	45	56	67	95
1	4	8	8	12	19	43	45	56	67	95
2	4	8	8	12	19	43	45	56	67	95
4	4	8	8	12	19	43	45	56	67	95
8	0	8	8	12	19	43	45	56	67	95
	12	32								

p	Ci	Mi	19	12	8	45	43	56	67	95
1	4	8	12	19	8	45	43	56	67	95
2	5	8	8	12	19	45	43	56	67	95
4	5	8	8	12	19	43	45	56	67	95
8	0	8	8	12	19	43	45	56	67	95
	14	32								

p	Ci	Mi	95	67	56	45	43	19	12	8
1	4	8	67	95	45	56	19	43	8	12
2	4	8	45	56	67	95	8	12	19	43
4	4	8	8	12	19	43	45	56	67	95
8	0	8	8	12	19	43	45	56	67	95
	12	32								

5.4.2. Ordenação por Fusão Direta

- Análise

merge: O(R-L-1)
mpass: O(N)

  - São efetuadas, no total, $\log_2 N$ passagens sobre os dados. Com dois arquivos podem ser intercalados em tempo linear (algoritmo **merge**), cada passagem de ordenação por intercalação toma o tempo **O(N)**

  - Como existem $\log_2 N$ passagens, o tempo total é

$$O(N \log_2 N)$$

  - Vale ressaltar, entretanto, que existem várias chamadas de procedimento envolvidas no processo, o que, normalmente, acarreta um grande número de operações

Comentários

- **Agradecimentos**

  - Parte do material desta apresentação foi obtida através de slides da disciplina de Introdução à Computação II ministrada pelo Prof. José Augusto Baranauskas