INSTITUTO METRÓPOLE DIGITAL

UFRN UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

Estrutura de Dados Básicas I.

Aula 10 – Algoritmos de ordenação IV

Prof. Eiji Adachi M. Barbosa

Avisos

- Prova será no dia 28/03/2017 ou 30/03/2017 (Ainda vou alinhar com LP1)

Objetivos

- Apresentar e implementar algoritmo de ordenação por intercalação (*Merge sort*)

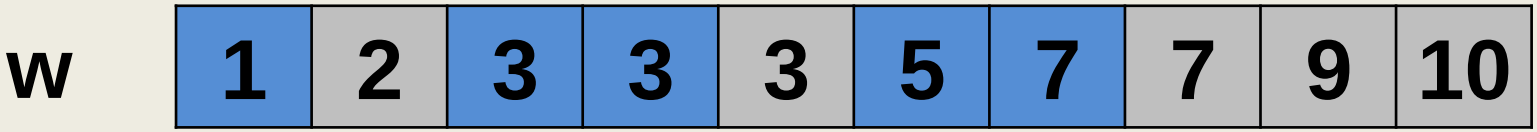
Referência extra

- Notas de aula “Análise de algoritmos”, prof. Paulo Feofiloff, IME-USP:
  - http://www.ime.usp.br/~pf/algoritmos/aulas/mrgsrt.html

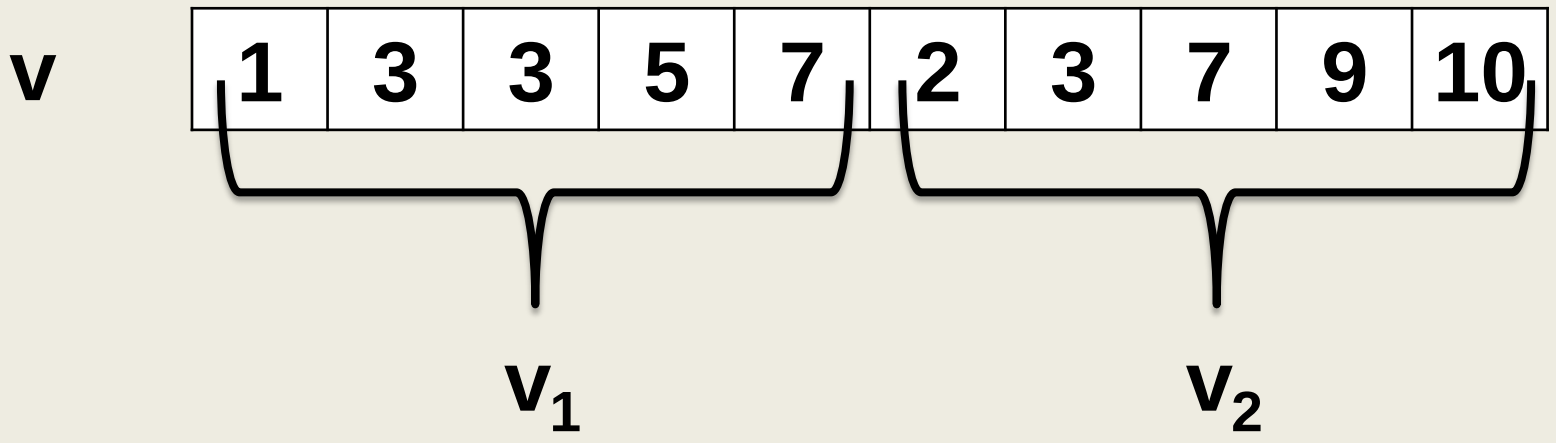
ORDENAÇÃO POR INTERCALAÇÃO

Ideia geral

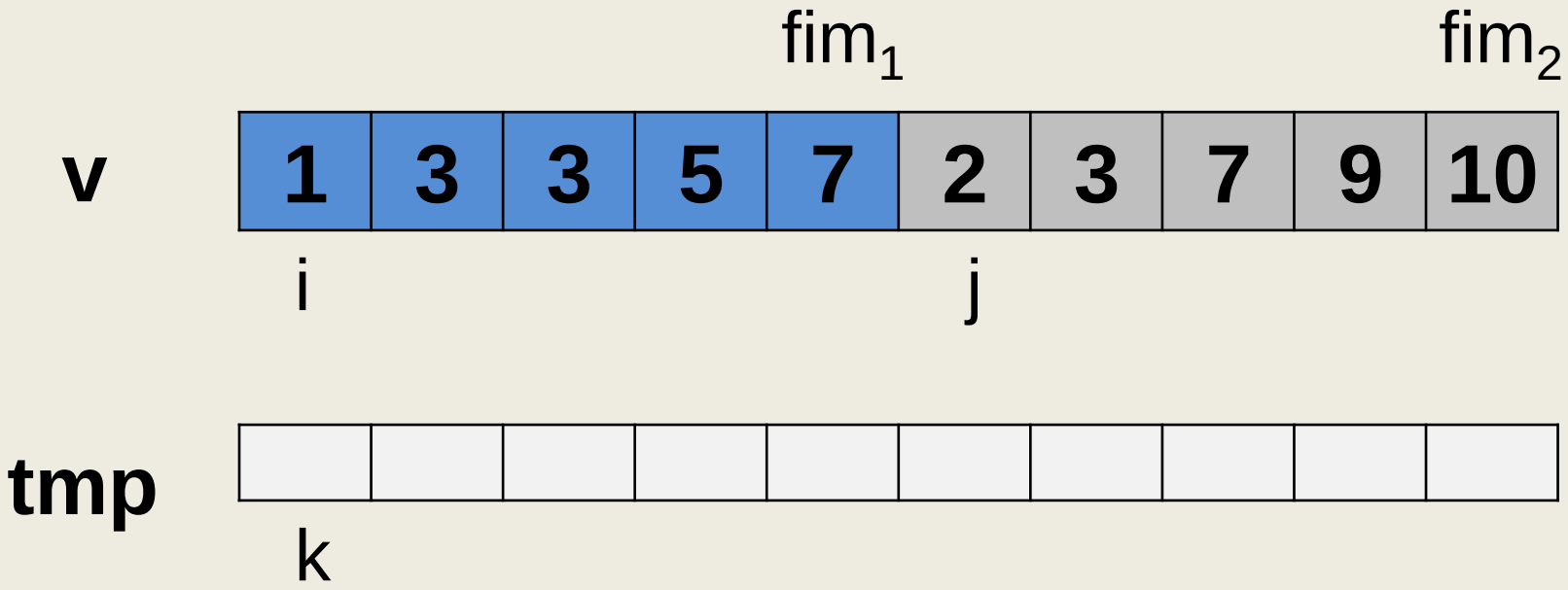
- A partir de dois vetores ordenados, posso construir um outro também ordenado



Ideia geral – Intercalação

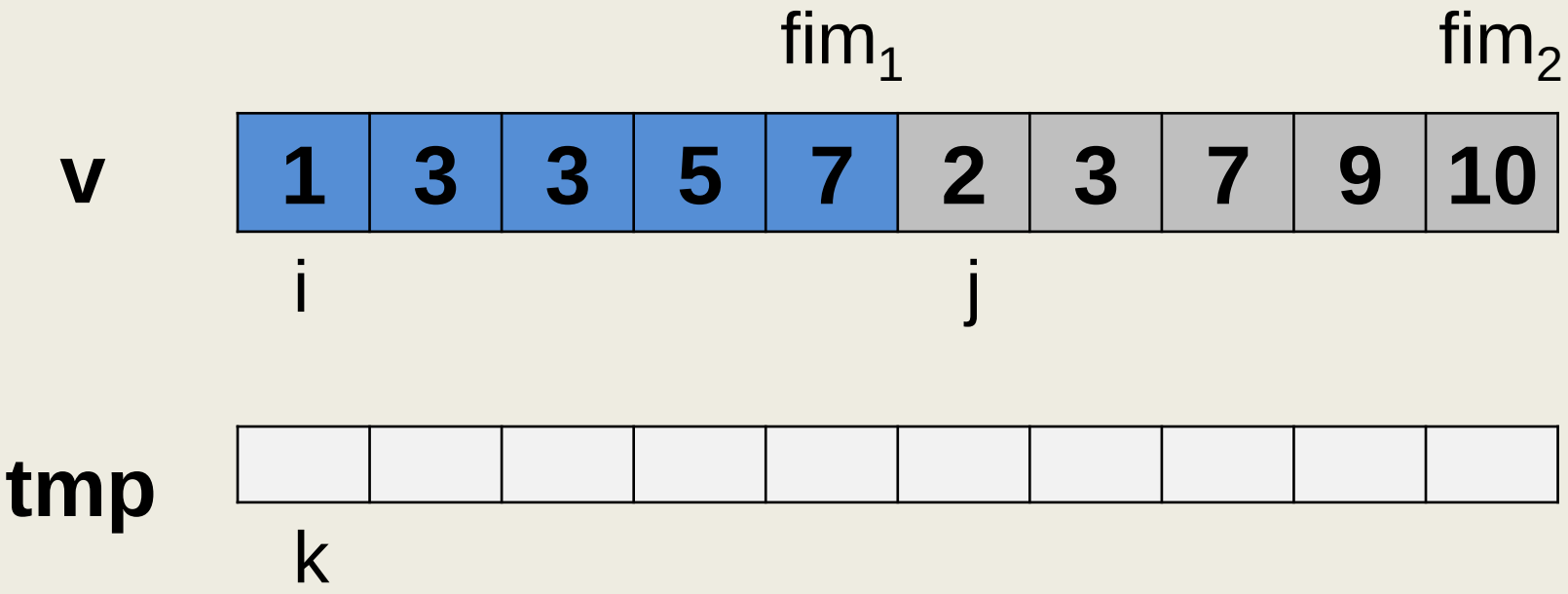


Ideia geral – Intercalação



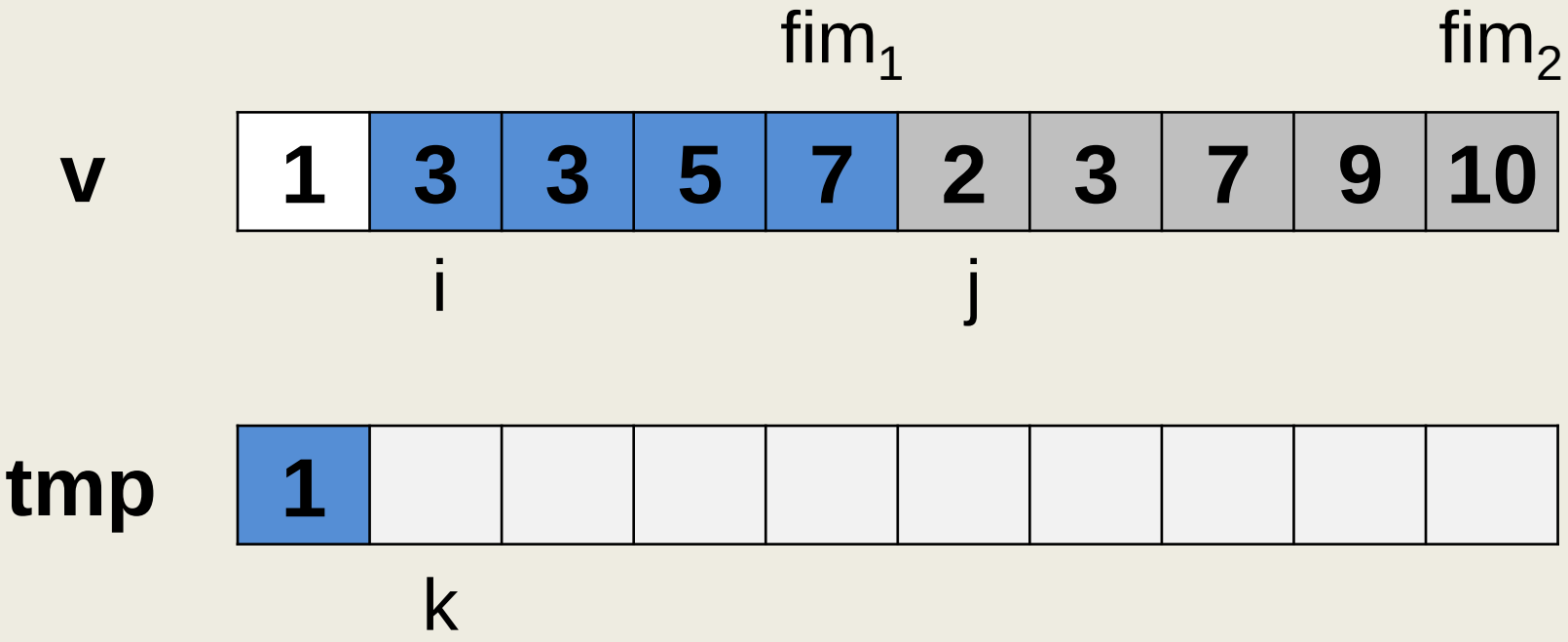
i: elemento atual de v_1
j: elemento atual de v_2
k: 1ª posição livre de tmp

Ideia geral – Intercalação



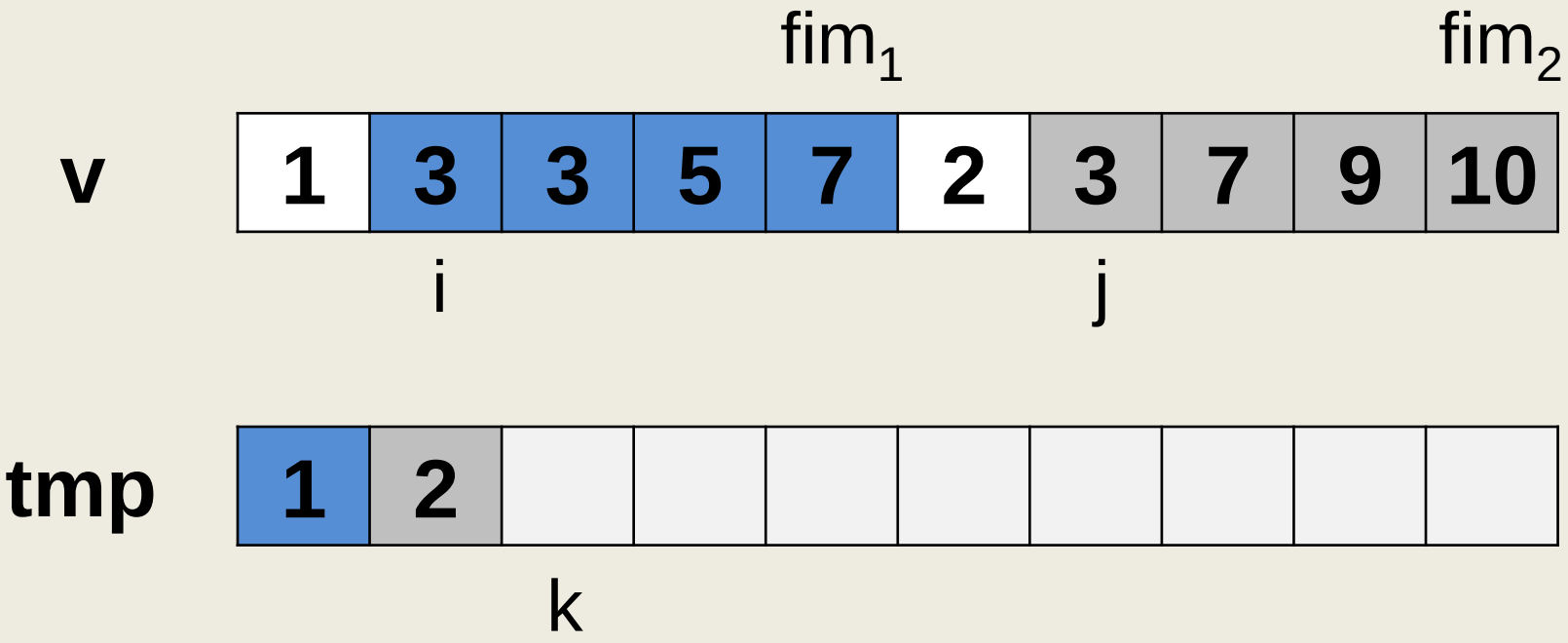
SE v[i] ≤ v[j] ENTÃO ?

Ideia geral – Intercalação



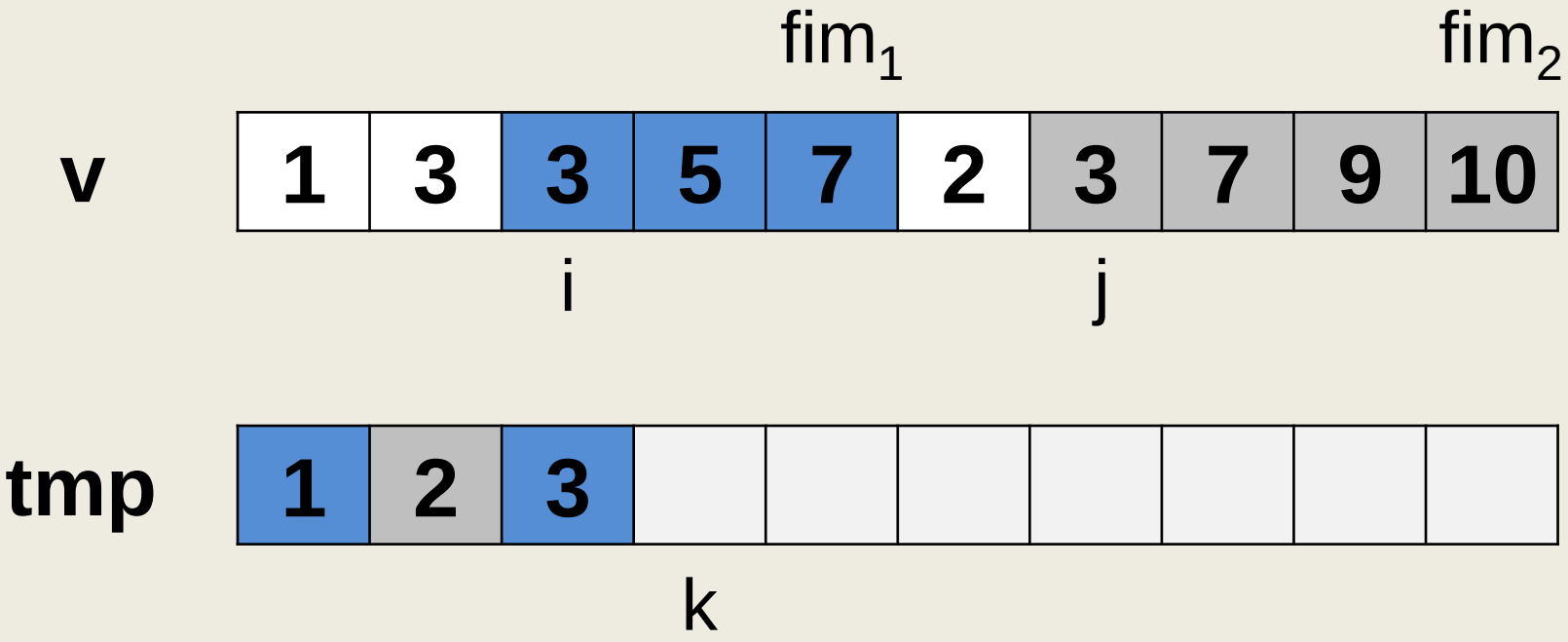
SE v[i] ≤ v[j] ENTÃO :
tmp[k] = v[i], i = i +1, k = k+1

Ideia geral – Intercalação



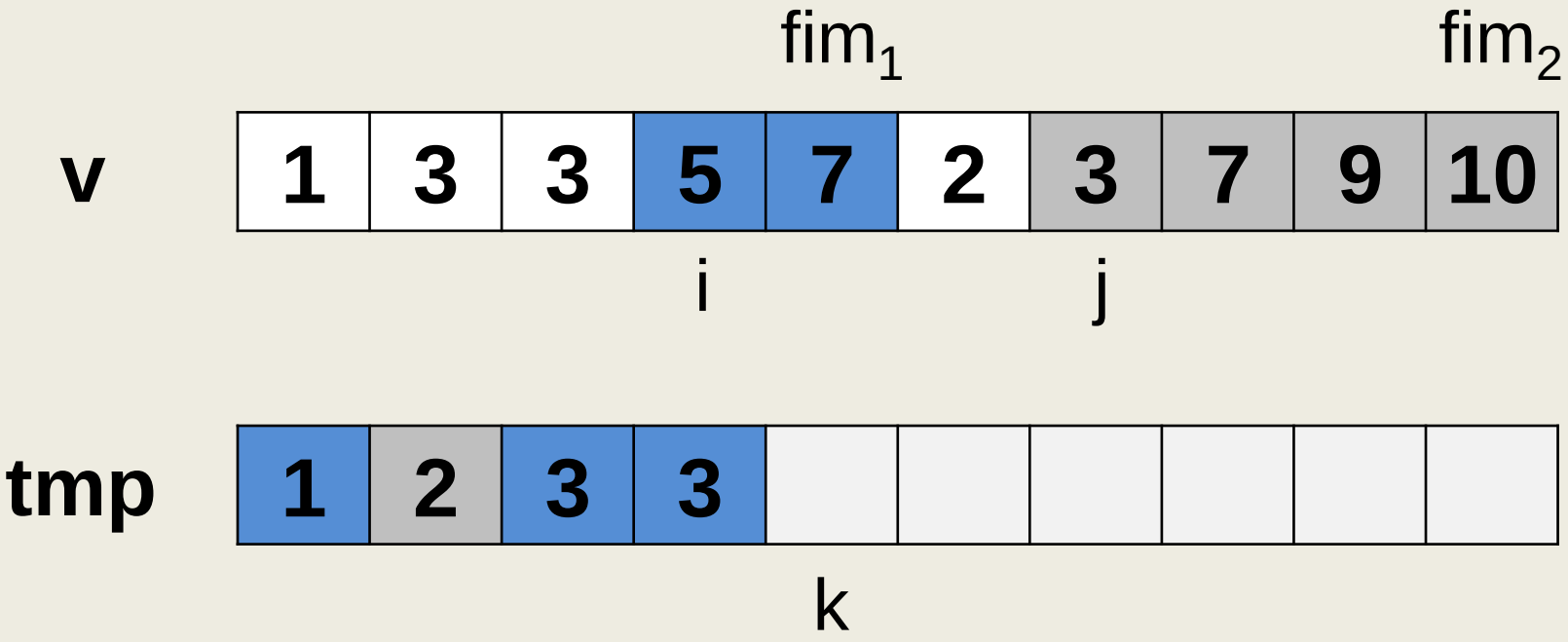

```
SE v[i] ≤ v[j] ENTÃO :
    tmp[k] = v[i], i = i +1, k = k+1
SENÃO :
    tmp[k] = v[j], j = j+1, k = k+1
```

Ideia geral – Intercalação



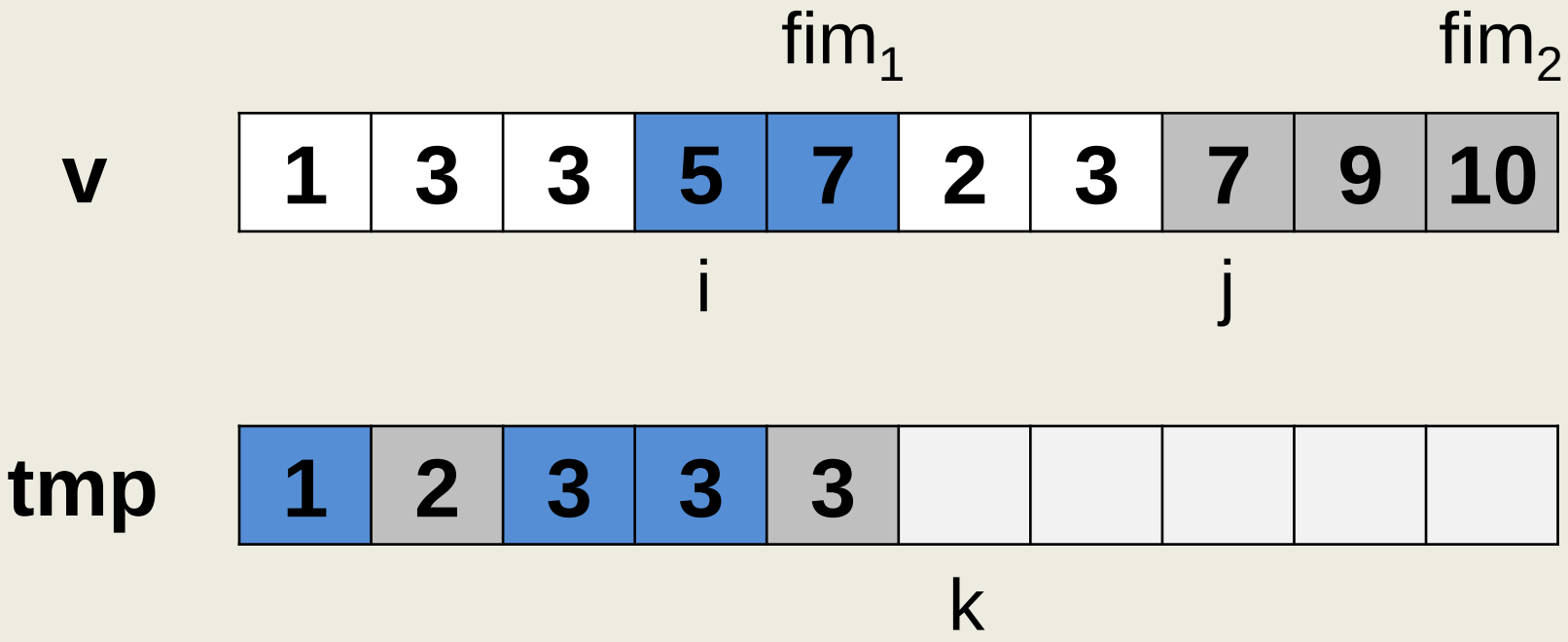

```
SE v[i] ≤ v[j] ENTÃO :
    tmp[k] = v[i], i = i +1, k = k+1
SENÃO :
    tmp[k] = v[j], j = j+1, k = k+1
```

Ideia geral – Intercalação



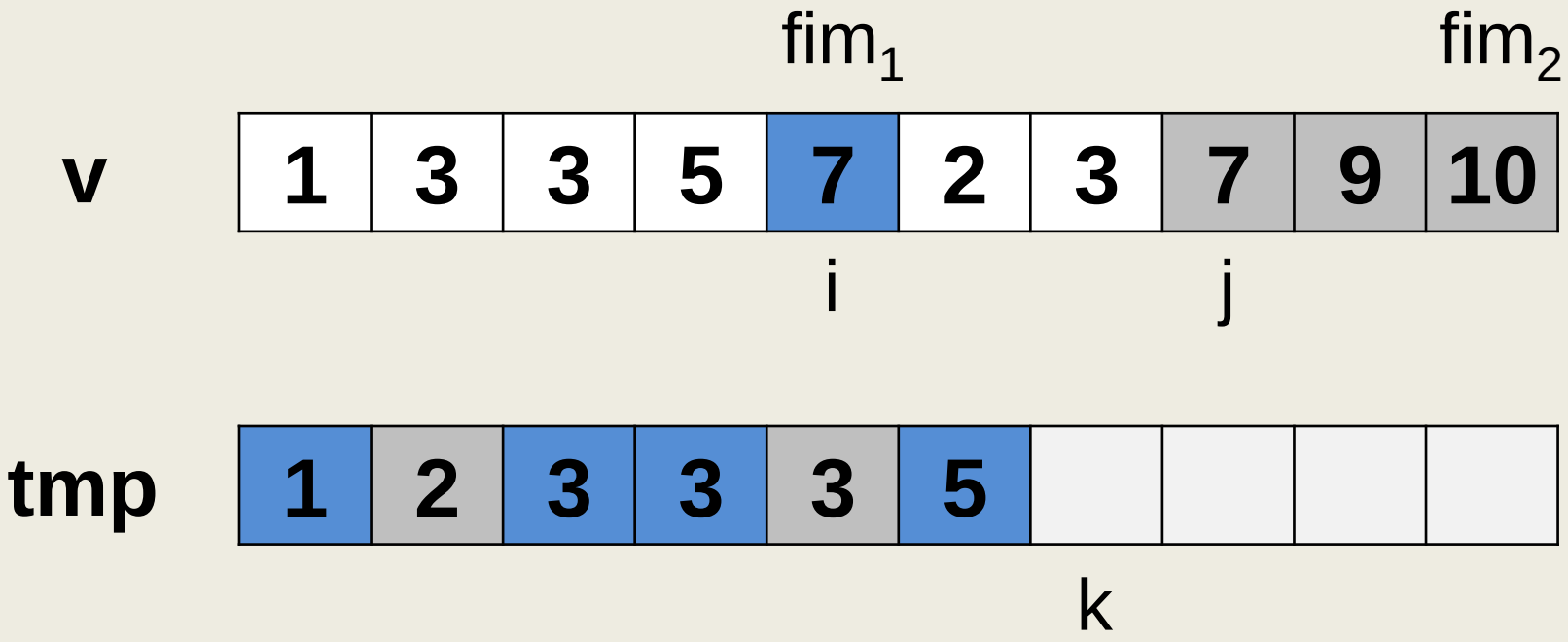

```
SE v[i] ≤ v[j] ENTÃO :
    tmp[k] = v[i], i = i +1, k = k+1
SENÃO :
    tmp[k] = v[j], j = j+1, k = k+1
```

Ideia geral – Intercalação



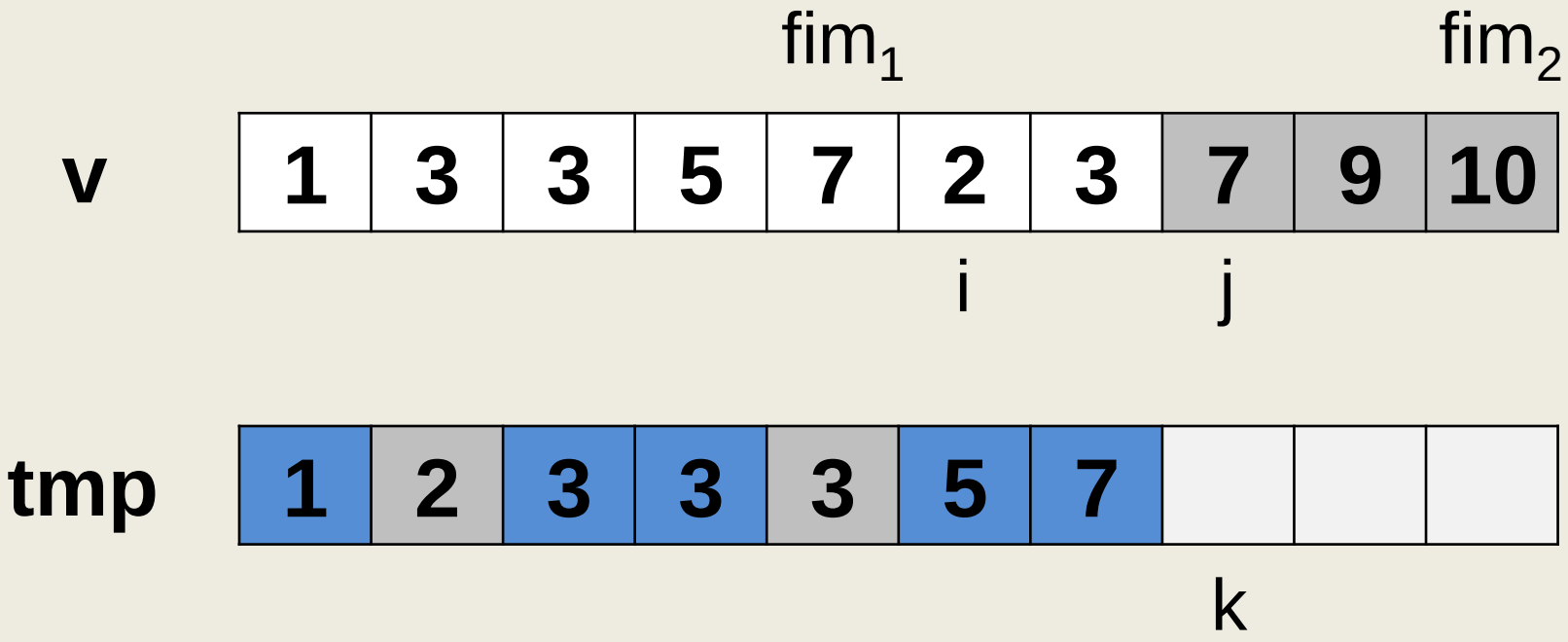

```
SE v[i] ≤ v[j] ENTÃO :
    tmp[k] = v[i], i = i +1, k = k+1
SENÃO :
    tmp[k] = v[j], j = j+1, k = k+1
```

Ideia geral – Intercalação



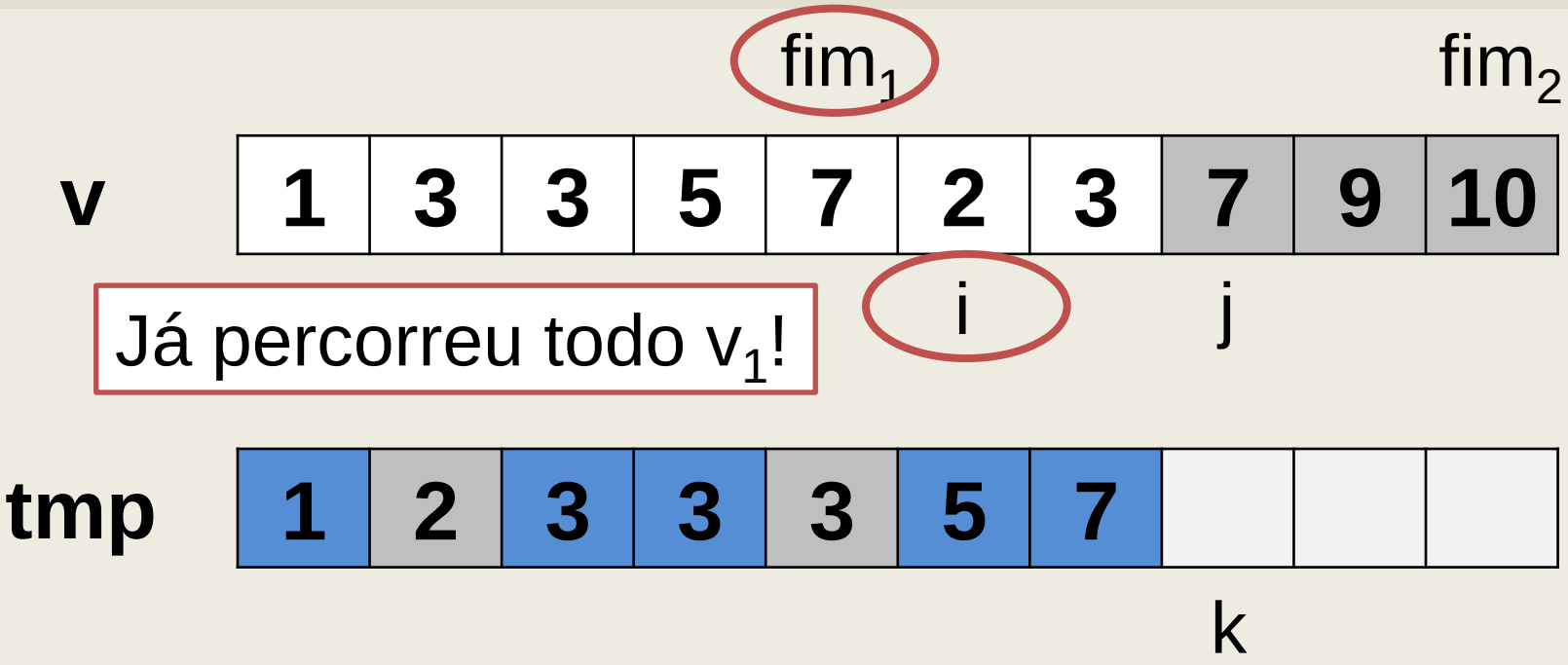

```
SE v[i] ≤ v[j] ENTÃO :
    tmp[k] = v[i], i = i +1, k = k+1
SENÃO :
    tmp[k] = v[j], j = j+1, k = k+1
```

Ideia geral – Intercalação



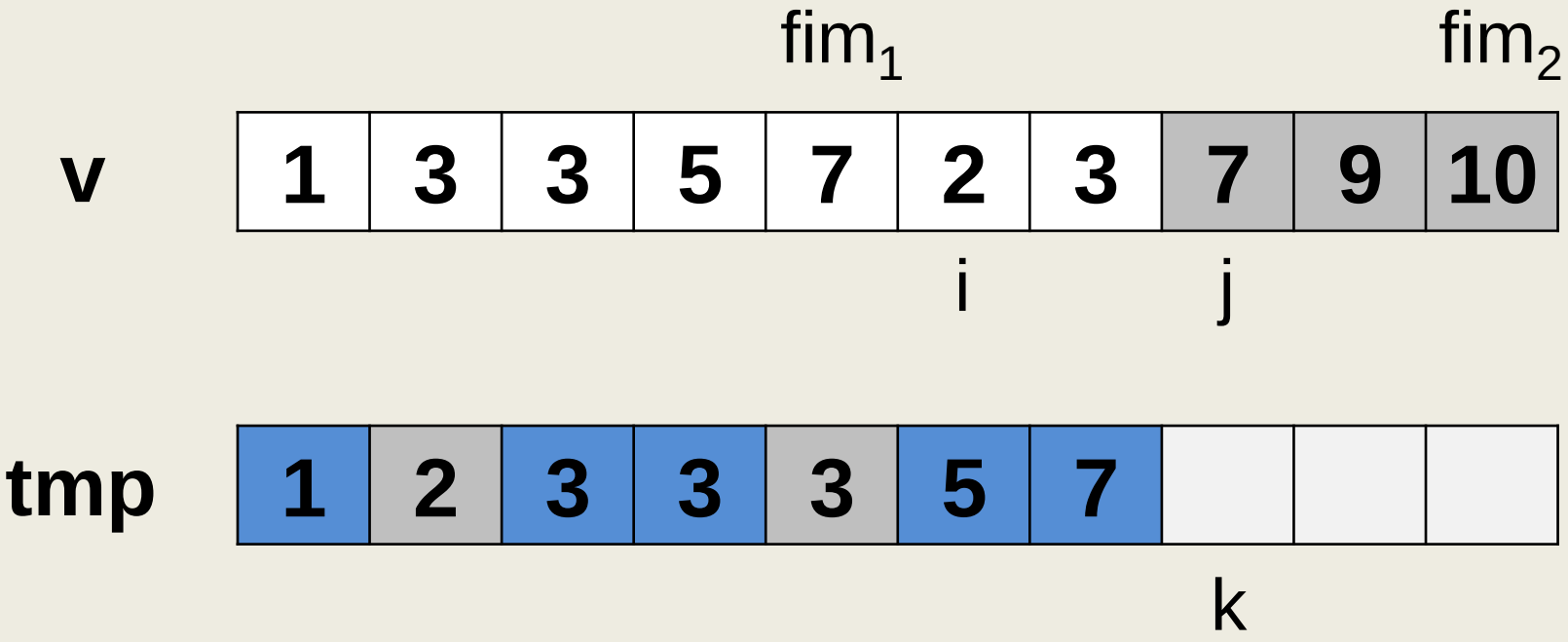

```
SE v[i] ≤ v[j] ENTÃO :
    tmp[k] = v[i], i = i +1, k = k+1
SENÃO :
    tmp[k] = v[j], j = j+1, k = k+1
```

Ideia geral – Intercalação




```
SE v[i] ≤ v[j] ENTÃO :
    tmp[k] = v[i], i = i +1, k = k+1
SENÃO :
    tmp[k] = v[j], j = j+1, k = k+1
```

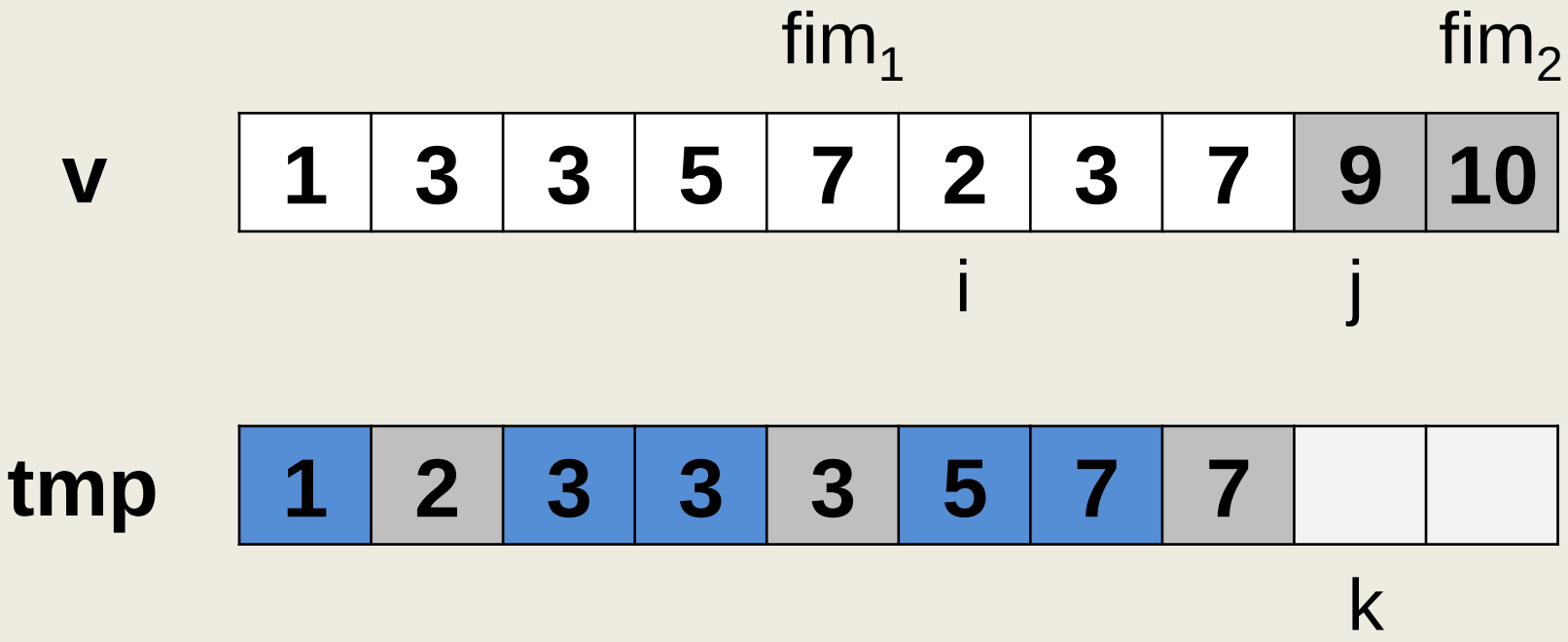
Ideia geral – Intercalação



SE v[i] ≤ v[j] ENTÃO :
    tmp[k] = v[i], i = i +1, k = k+1
SENÃO :
    tmp[k] = v[j], j = j+1, k = k+1

ENQUANTO j ≤ fim_2 FAÇA:
    tmp[k] = j
    k = k+1, j = j+1
FIM_ENQUANTO

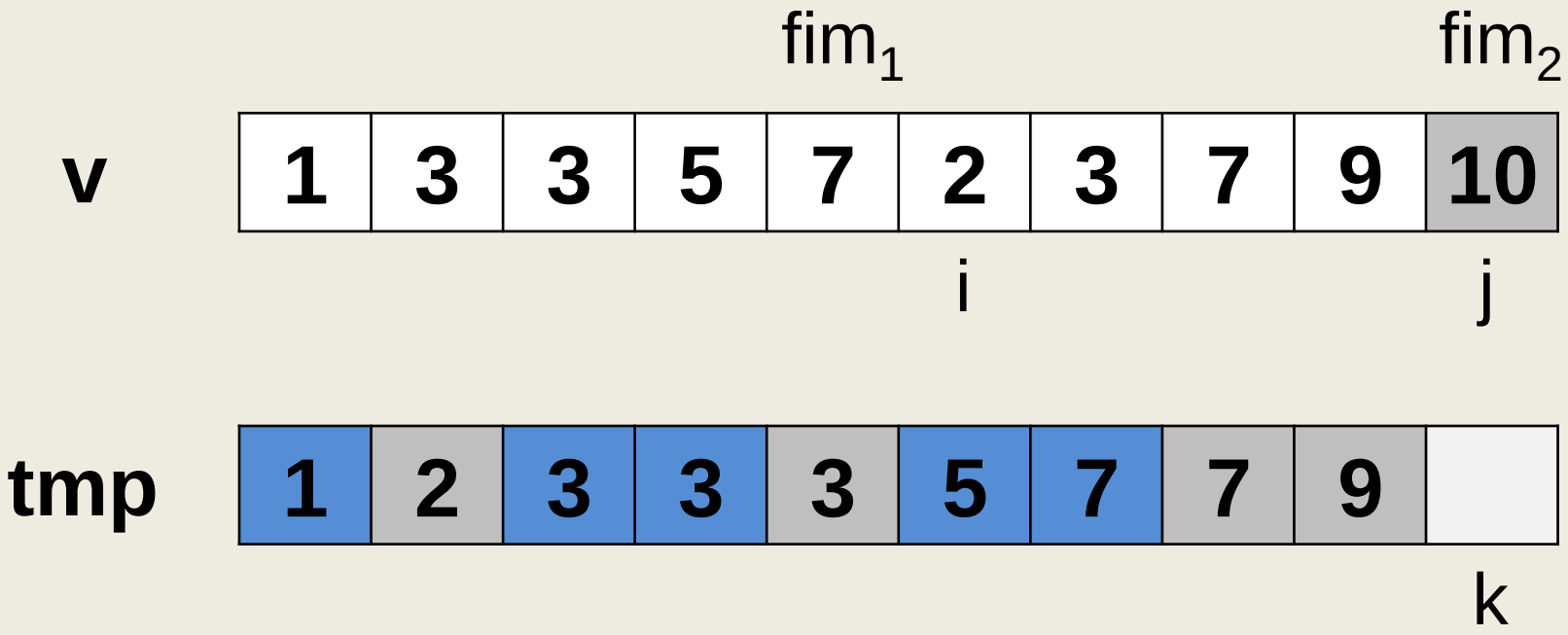
Ideia geral – Intercalação



SE v[i] ≤ v[j] ENTÃO :
    tmp[k] = v[i], i = i +1, k = k+1
SENÃO :
    tmp[k] = v[j], j = j+1, k = k+1

ENQUANTO j ≤ fim_2 FAÇA:
    tmp[k] = j
    k = k+1, j = j+1
FIM_ENQUANTO

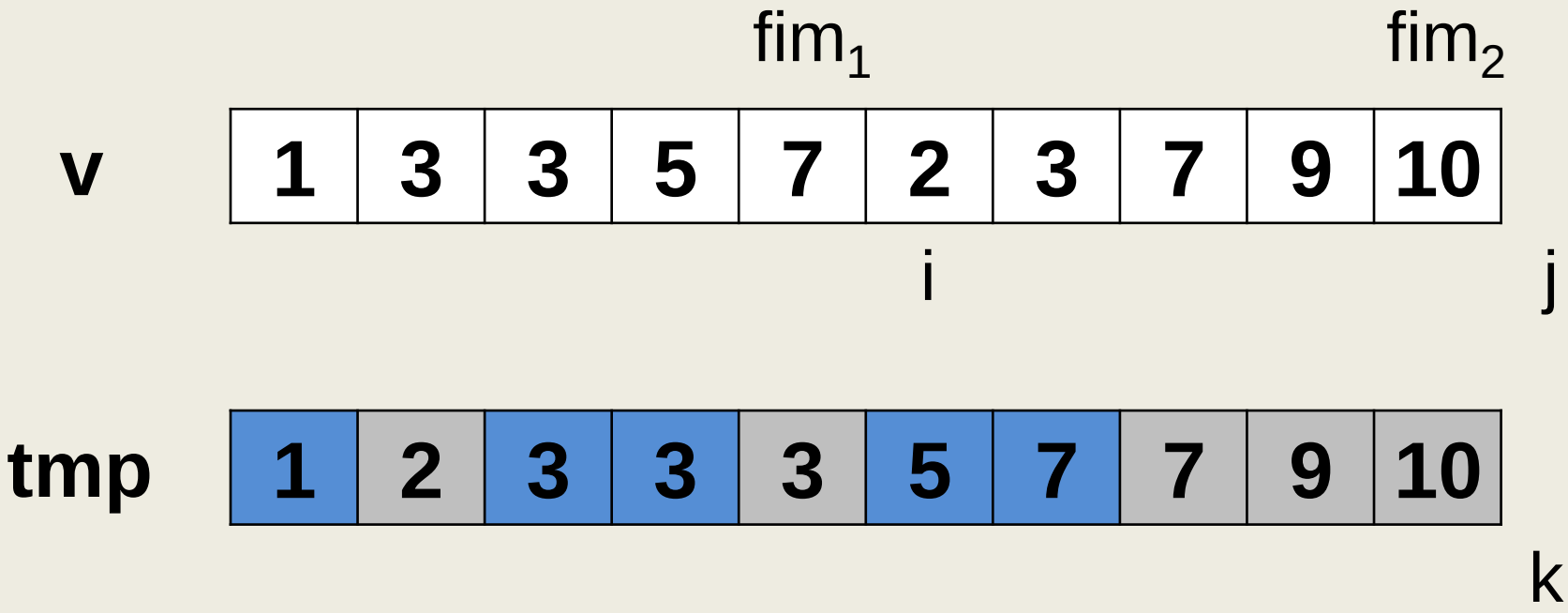
Ideia geral – Intercalação



SE v[i] ≤ v[j] ENTÃO :
    tmp[k] = v[i], i = i +1, k = k+1
SENÃO :
    tmp[k] = v[j], j = j+1, k = k+1

ENQUANTO j ≤ fim_2 FAÇA:
    tmp[k] = j
    k = k+1, j = j+1
FIM_ENQUANTO

Ideia geral – Intercalação



SE v[i] ≤ v[j] ENTÃO :
    tmp[k] = v[i], i = i +1, k = k+1
SENÃO :
    tmp[k] = v[j], j = j+1, k = k+1

ENQUANTO j ≤ fim_2 FAÇA:
    tmp[k] = j
    k = k+1, j = j+1
FIM_ENQUANTO

```
Intercalar( v[n], inicio1, inicio2, fim2 ) :
   fim1 = inicio2-1, i = inicio1, j=inicio2, k=0
   WHILE i ≤ fim1 && j ≤ fim2 :
       IF v[i] ≤ v[j] :
             tmp[k] = v[i], i = i+1
       ELSE:
             tmp[k] = v[j], j = j+1
       END_IF
       k = k+1
   END_WHILE
   WHILE i ≤ fim1:
       tmp[k] = v[i], i = i+1, k=k+1
   END_WHILE
   WHILE j ≤ fim2 :
        tmp[k] = v[j], j = j+1, k=k
   END_WHILE
   COPY tmp IN v
FIM
```

Qual a complexidade (pior e melhor caso)?

Já sei intercalar, mas como conseguir dois vetores ordenados?

Qual o caso trivial do vetor ordenado?

Ordenação por intercalação

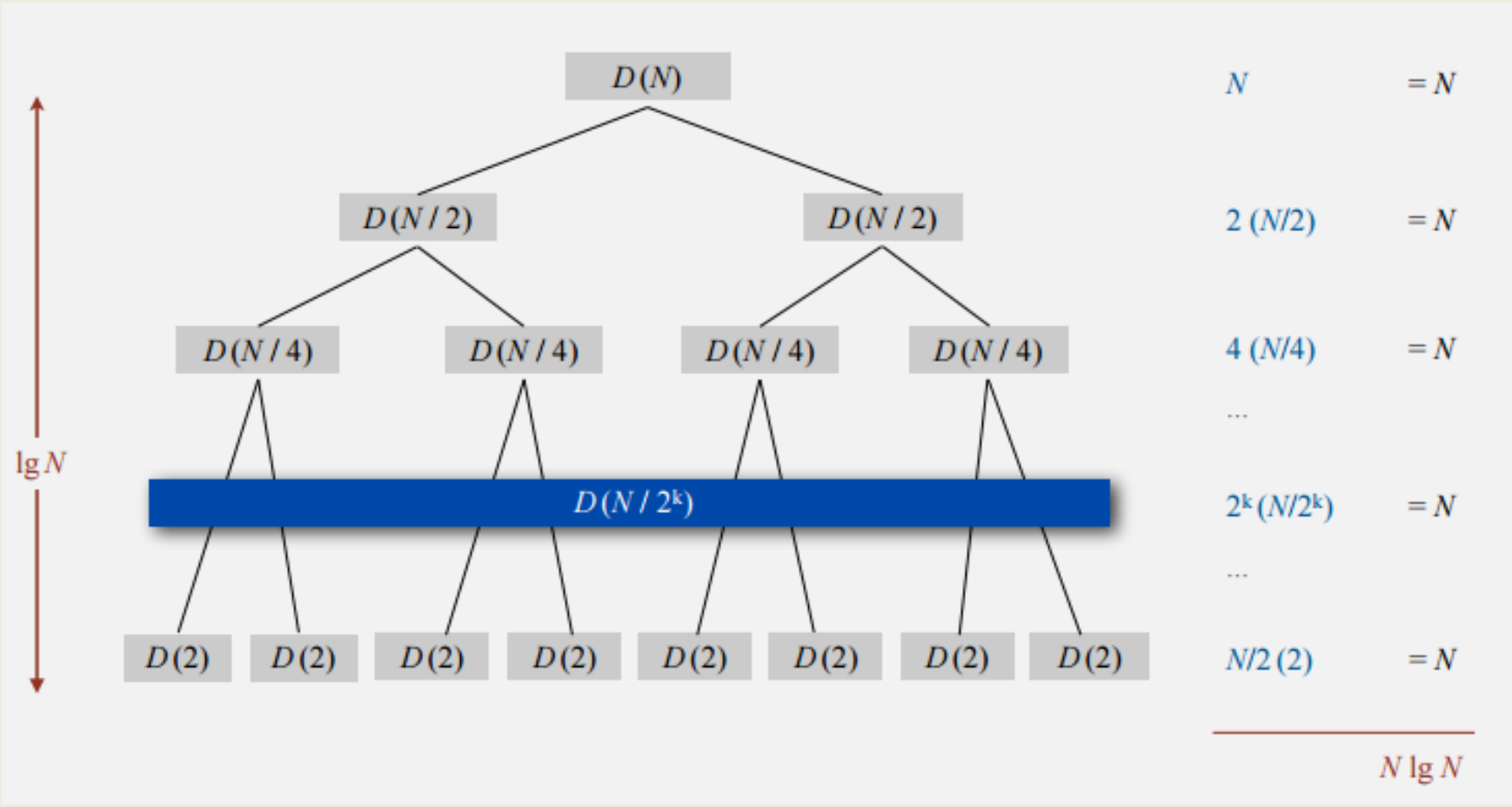
- Estratégia “Dividir para conquistar”:
  - Dividir:
    - Dividir o vetor ao meio sucessivamente, até chegar a um vetor de tamanho 1
  - Conquistar:
    - Intercalar vetores de tamanho 1, formando vetores de tamanho 2
    - Intercalar vetores de tamanho 2, formando vetores de tamanho 4
    - Intercalar vetores de tamanho 4, formando vetores de tamanho 8
    - ...
    - Intercalar vetores de tamanho n/2, formando vetor de tamanho n

Ordenação por intercalação

```
OrdenarIntercalacao( v[1...n], esquerda, direita ) :
   SE esquerda < direita ENTÃO :
       meio = (direita+esquerda)/2
       OrdenarIntercalacao( v, esquerda, meio )
       OrdenarIntercalacao( v, meio+1, direita )
       Intercalar(v, esquerda, meio+1, direita)
   FIM_SE
FIM
```

Qual a complexidade (pior e melhor caso)?

Ordenação por intercalação



Ordenação por intercalação

- Custo total:
  - $T(n) = 2*T(n/2) + n$
- Dividindo por n:
  - $T(n) / n = T(n/2) / (n/2) + 1$
  - ...
  - $T(n)/n = T(1)/1 + \log_2 n$
  - $T(n) = n * (1 + \log_2 n)$
  - $T(n) = n + n*\log_2 n$

Ordenação por intercalação

- Pior e melhor caso: $\Theta(n^*\log_2 n)$

Observação sobre *copy* do algoritmo de intercalação

- Na última linha do algoritmo de intercalação, é feita a cópia dos elementos do array auxiliar *tmp* para o array de entrada *v.* Deve-se atentar ao fato de que são manipulados apenas os elementos de v que estão compreendidos entre [inicio1, fim2] e que este intervalo não é necessariamente entre [0,n-1] ao longo das chamadas recursivas do merge sort. Isto quer dizer que se deve tomar cuidado com a cópia dos elementos de tmp para v. A cópia deve ser feita da seguinte forma:

```
for(i = inicio1, j = 0; i <= fim2; i++, j++)
{
        v[i] = tmp[j];
}
```

INSTITUTO METRÓPOLE DIGITAL

UFRN UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

Estrutura de Dados Básicas I.

Aula 9 – Algoritmos de ordenação II

Prof. Eiji Adachi M. Barbosa