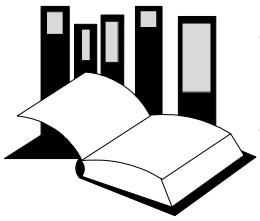
Ordenação por Fusão



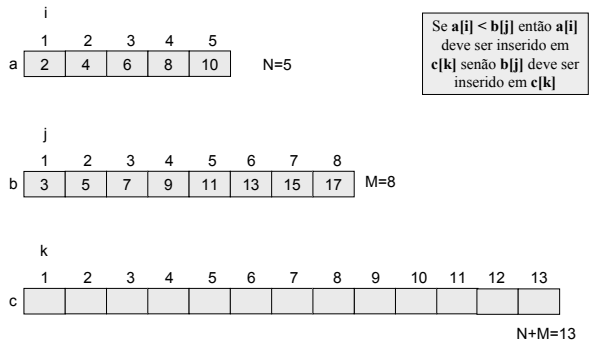
- Esta aula introduz métodos de ordenação por fusão
- A fusão é utilizada quando duas ou mais seqüências encontram-se ordenadas
- O objetivo é intercalar as seqüências ordenadas em uma seqüência ordenada única

Prof. Dr. José Augusto Baranauskas
DFM-FFCLRP-USP

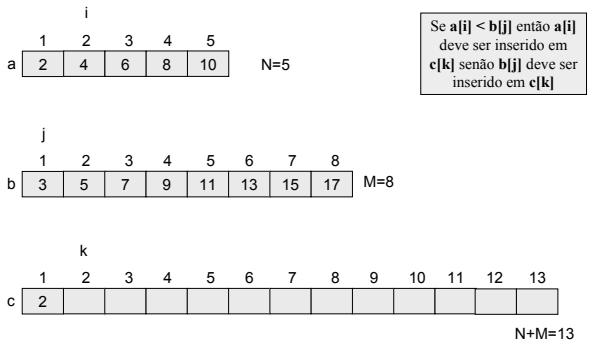
Fusão (Intercalação)

- A aplicação de algoritmos de ordenação é impraticável, se os dados não estiverem em memória principal, mas sim em discos ou fitas
- 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
- A operação de fusão combina duas ou mais seqüências ordenadas para formar uma única seqüência ordenada através da aplicação de repetidas seleções entre os elementos acessíveis a cada momento
- Assim sendo, a operação de fusão é muito mais simples que a de ordenação, sendo empregada como operação auxiliar no processo mais complexo de ordenação seqüencial.

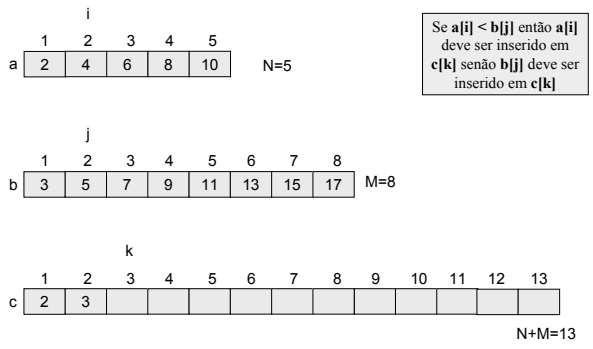
Exemplo



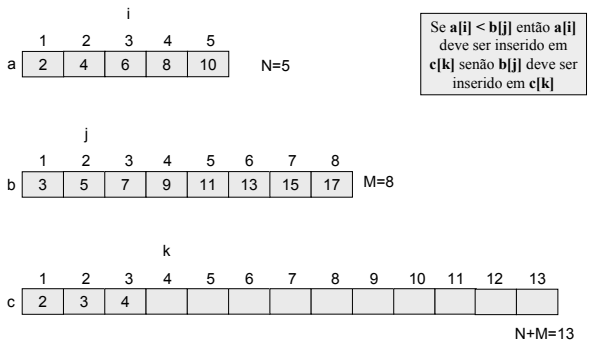
Exemplo



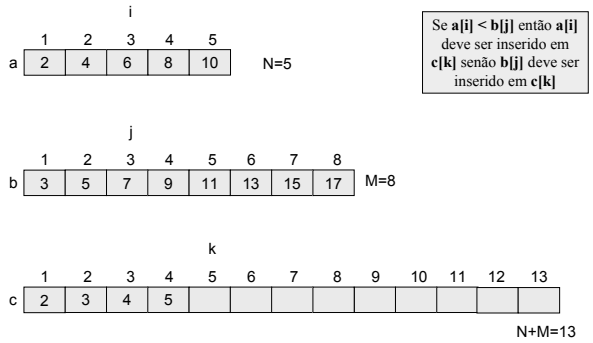
Exemplo



Exemplo



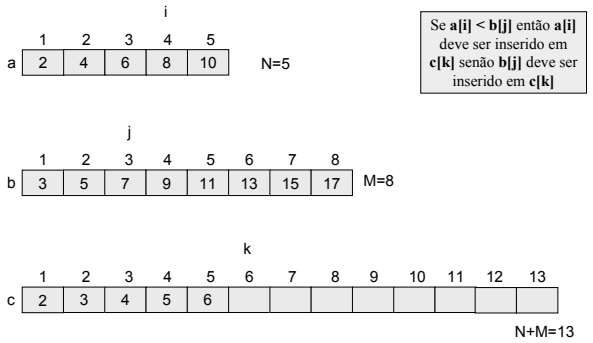
Exemplo



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

7

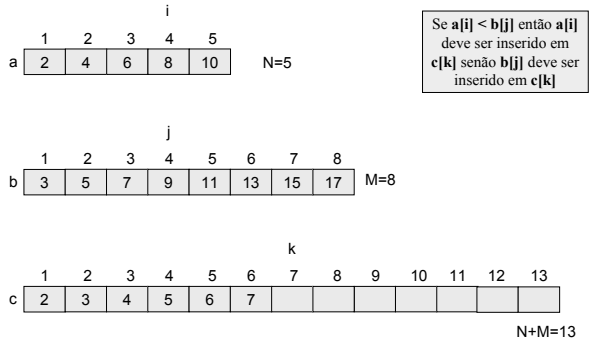
Exemplo



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

8

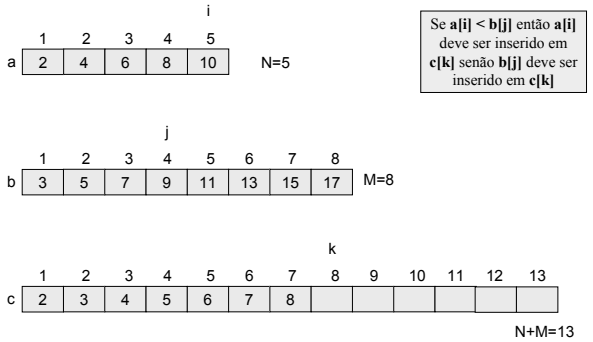
Exemplo



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

9

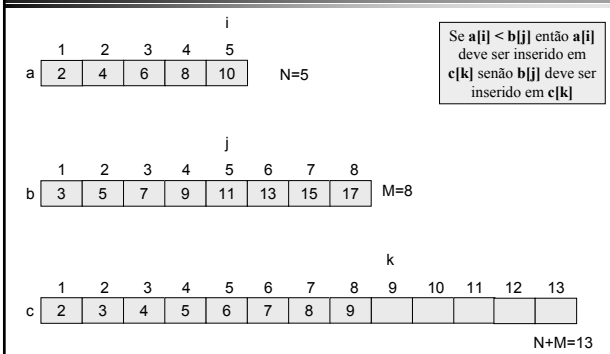
Exemplo



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

10

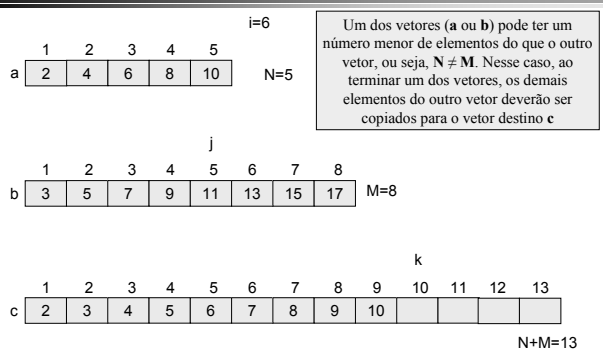
Exemplo



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

11

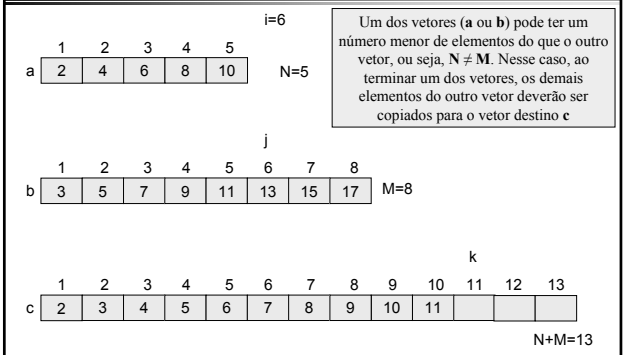
Exemplo



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

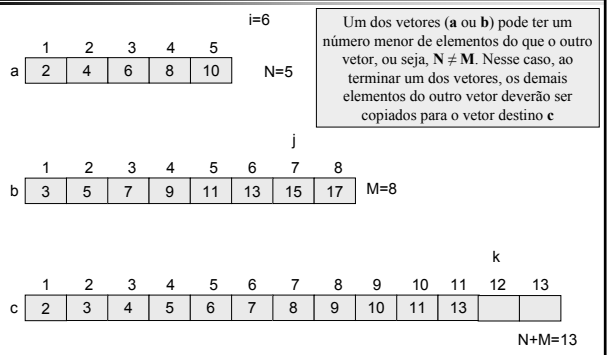
12

Exemplo



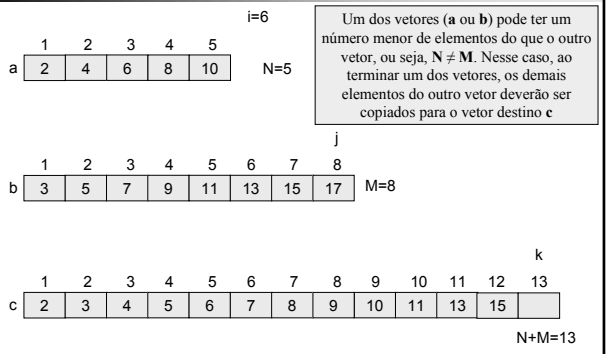
13

Exemplo



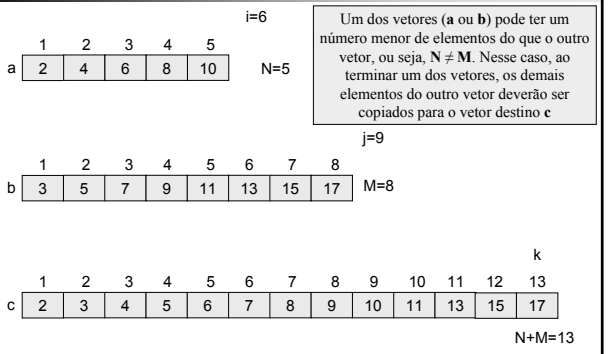
14

Exemplo



15

Exemplo



16

Algoritmo de Fusão (3 vetores)

```
int i,j,k;

i = 1;
j = 1;
k = 0;
while (i <= N && j <= M)
{ k++;
  if(a[i] < b[j])
  { c[k] = a[i];
    i++;
  }
  else
  { c[k] = b[j];
    j++;
  }
}
```

```
// #a > #b
// copiar resto de a p/ c
while (i <= N)
{ k++;
  c[k] = a[i];
  i++;
}
// #b > #a
// copiar resto de b p/ c
while (j <= M)
{ k++;
  c[k] = b[j];
  j++;
}
```

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

17

Algoritmo de Fusão (2 vetores)

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

18

Algoritmo de Fusão (2 vetores)

```
void merge(item a[], int L, int h, int R, item 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:
```

	L					h	h+1					R
a	2	4	6	8	10	12	1	3	5	7	13	15	17

```
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:
*/
```

	L												R
c	1	2	3	4	5	6	7	8	10	12	13	15	17

19

Algoritmo de Fusão (2 vetores)

```
void merge(item a[], int L,
   int h, int R, item 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]
*/
{ int i,j,k;

  i = L;
  j = h+1;
  k = L-1;

while (i <= h && j <= R)
 { k++;
   if(a[i] < a[j])
   { c[k] = a[i];
     i++;
   }
   else
   { c[k] = a[j];
     j++;
   }
 }
 while (i <= h)
 { k++;
   c[k] = a[i];
   i++;
 }
 while (j <= R)
 { k++;
   c[k] = a[j];
   j++;
 }
}
```

20

Fusão: Análise

- Em cada iteração do laço *while* **k** incrementa de 1
- O incremento total é **R-L+1**
- Portanto, o laço será incrementado no máximo **R-L+1** vezes
- O comando *if* move, no máximo, um elemento por iteração
- Por isso, o tempo total é **O(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 **O(N)**

21

Ordenação por Fusão Direta

1. Dividir a seqüência inicial **a** em duas metades, chamadas **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.

22

Exemplo (1º passo)

- Considere o vetor

a	45	56	12	43	95	19	8	67

- que é particionado em

b	45	56	12	43

c	95	19	8	67

- A fusão de elementos isolados em pares ordenados resulta em

a	45	95	19	56	8	12	43	67

23

Exemplo (2º passo)

- Dividindo-se ao meio

a	45	95	19	56	8	12	43	67

- obtém-se

b	45	95	19	56

c	8	12	43	67

- A fusão de pares ordenados em quádruplas resulta em

a	8	12	45	95	19	43	56	67

24

Exemplo (3º passo)

- Uma terceira divisão ao meio de

a	8	12	45	95	19	43	56	67

- resulta em

b	8	12	45	95

c	19	43	56	67

- A fusão de quáduplas ordenadas em óctuplas resulta em

a	8	12	19	43	45	56	67	95

25

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*, e 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 partição e uma fase de fusão
- Como pode ser observado, para que seja possível a realização da ordenação são necessárias três seqüências

26

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
- Em contraste com a ordenação descrita acima por *fusão de duas fases*, este novo método é denominado *fusão de fase única* ou *fusão balanceada*
- Tal método é superior ao anterior, uma vez que são necessárias somente a metade das operações de movimentações exigidas no outro caso
- O preço desta vantagem é uma quarta seqüência

27

Ordenação por Fusão (2 vetores)

- A classificação por intercalação consiste em várias passagens sobre os elementos a serem classificados
- 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

28

Passagem de Ordenação

```
void mpass(item a[], int N, int p, item 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
*/
{ int i,j;

  i = 1;
  while (i <= N-2*p+1)
  { merge(a,i,i+p-1,i+2*p-1,c);
    i = i + 2*p;
  }
  // intercarlar restante de comprimento < 2*p
  if(i+p-1 < N)
    merge(a,i,i+p-1,N,c);
  else
    for(j=i; j <= N; j++)
      c[j] = a[j];
}
```

29

Passagem de Ordenação

```
void mpass(item a[], int N, int p, item c[])
{ int i,j;
  i = 1;
  while (i <= N-2*p+1)
  { merge(a,i,i+p-1,i+2*p-1,c);
    i = i + 2*p;
  }
  if(i+p-1 < N)
    merge(a,i,i+p-1,N,c);
  else
    for(j=i; j <= N; j++)
      c[j] = a[j];
}
```

p=2
N=13
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													

30

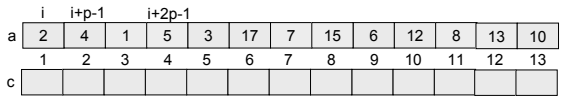
Passagem de Ordenação

```
void mpass(item a[], int N, int p, item c[])
{ int i,j;
  i = 1;
  while (i <= N-2*p+1)
  { merge(a,i,i+p-1,i+2*p-1,c);
```

```
void merge(item a[], int L, int h, int R, item c[])
```

```
    if(i+p-1 < N)
      merge(a,i,i+p-1,N,c);
    else
      for(j=i; j <= N; j++)
        c[j] = a[j];
}
```

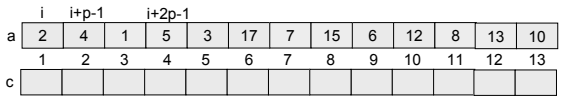
p=2
N=13
N-2p+1



Passagem de Ordenação

```
void mpass(item a[], int N, int p, item c[])
{ int i,j;
  i = 1;
  while (i <= N-2*p+1)
  { merge(a,i,i+p-1,i+2*p-1,c);
    i = i + 2*p;
  }
  if(i+p-1 < N)
    merge(a,i,i+p-1,N,c);
  else
    for(j=i; j <= N; j++)
      c[j] = a[j];
}
```

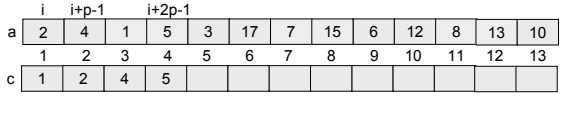
p=2
N=13
L h h+1 R
N-2p+1



Passagem de Ordenação

```
void mpass(item a[], int N, int p, item c[])
{ int i,j;
  i = 1;
  while (i <= N-2*p+1)
  { merge(a,i,i+p-1,i+2*p-1,c);
    i = i + 2*p;
  }
  if(i+p-1 < N)
    merge(a,i,i+p-1,N,c);
  else
    for(j=i; j <= N; j++)
      c[j] = a[j];
}
```

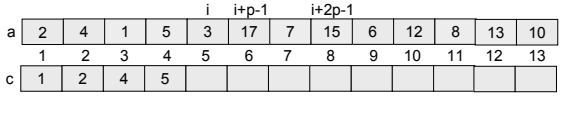
p=2
N=13
L h h+1 R
N-2p+1



Passagem de Ordenação

```
void mpass(item a[], int N, int p, item c[])
{ int i,j;
  i = 1;
  while (i <= N-2*p+1)
  { merge(a,i,i+p-1,i+2*p-1,c);
    i = i + 2*p;
  }
  if(i+p-1 < N)
    merge(a,i,i+p-1,N,c);
  else
    for(j=i; j <= N; j++)
      c[j] = a[j];
}
```

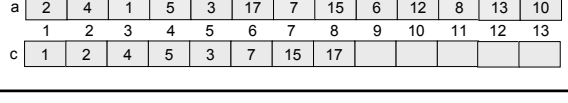
p=2
N=13
L h h+1 R
N-2p+1



Passagem de Ordenação

```
void mpass(item a[], int N, int p, item c[])
{ int i,j;
  i = 1;
  while (i <= N-2*p+1)
  { merge(a,i,i+p-1,i+2*p-1,c);
    i = i + 2*p;
  }
  if(i+p-1 < N)
    merge(a,i,i+p-1,N,c);
  else
    for(j=i; j <= N; j++)
      c[j] = a[j];
}
```

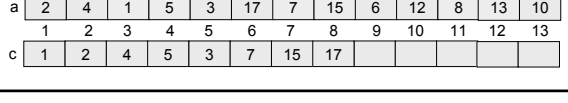
p=2
N=13
L h h+1 R
N-2p+1



Passagem de Ordenação

```
void mpass(item a[], int N, int p, item c[])
{ int i,j;
  i = 1;
  while (i <= N-2*p+1)
  { merge(a,i,i+p-1,i+2*p-1,c);
    i = i + 2*p;
  }
  if(i+p-1 < N)
    merge(a,i,i+p-1,N,c);
  else
    for(j=i; j <= N; j++)
      c[j] = a[j];
}
```

p=2
N=13
L h h+1 R
N-2p+1



Passagem de Ordenação

```
void mpass(item a[], int N, int p, item c[])
{ int i,j;
  i = 1;
  while (i <= N-2*p+1)
  { merge(a,i,i+p-1,i+2*p-1,c);
    i = i + 2*p;
  }
  if(i+p-1 < N)
    merge(a,i,i+p-1,N,c);
  else
    for(j=i; j <= N; j++)
      c[j] = a[j];
}
```

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	6	8	12	13	

37

Passagem de Ordenação

```
void mpass(item a[], int N, int p, item c[])
{ int i,j;
  i = 1;
  while (i <= N-2*p+1)
  { merge(a,i,i+p-1,i+2*p-1,c);
    i = i + 2*p;
  }
  if(i+p-1 < N)
    merge(a,i,i+p-1,N,c);
  else
    for(j=i; j <= N; j++)
      c[j] = a[j];
}
```

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	

38

Passagem de Ordenação

```
void mpass(item a[], int N, int p, item c[])
{ int i,j;
  i = 1;
  while (i <= N-2*p+1)
  { merge(a,i,i+p-1,i+2*p-1,c);
    i = i + 2*p;
  }
  if(i+p-1 < N)
    merge(a,i,i+p-1,N,c);
  else
    for(j=i; j <= N; j++)
      c[j] = a[j];
}
```

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	10

39

Passagem de Ordenação

```
void mpass(item a[], int N, int p, item c[])
{ int i,j;
  i = 1;
  while (i <= N-2*p+1)
  { merge(a,i,i+p-1,i+2*p-1,c);
    i = i + 2*p;
  }
  if(i+p-1 < N)
    merge(a,i,i+p-1,N,c);
  else
    for(j=i; j <= N; j++)
      c[j] = a[j];
}
```

p=2
N=13

L h h+1 R
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	10

40

Passagem de Ordenação

```
void mpass(item a[], int N, int p, item c[])
{ int i,j;
  i = 1;
  while (i <= N-2*p+1)
  { merge(a,i,i+p-1,i+2*p-1,c);
    i = i + 2*p;
  }
  if(i+p-1 < N)
    merge(a,i,i+p-1,N,c);
  else
    for(j=i; j <= N; j++)
      c[j] = a[j];
}
```

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

41

Ordenação por Fusão

```
void Mergesort(item a[], int N)
{ int i,p;
  item c[Max];
  // c é um vetor auxiliar e Max é o tamanho máximo de a

  p = 1;
  while (p < N)
  { mpass(a,N,p,c);
    p = 2*p;
    mpass(c,N,p,a);
    p = 2*p;
  }
}
```

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

42

Ordenação por Fusão

```
void Mergesort(item a[], int N)
{ int i,p;
  item c[Max];
  // c é um vetor auxiliar e Max é o tamanho máximo de a

  p = 1;
  while (p < N)
  { mpass(a,N,p,c);
    p = 2*p;
    mpass(c,N,p,a);
    p = 2*p;
  }
}
```

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

43

Ordenação por Fusão

```
void Mergesort(item a[], int N)
{ int i,p;
  item c[Max];
  // c é um vetor auxiliar e Max é o tamanho máximo de a

  p = 1;
  while (p < N)
  { mpass(a,N,p,c);
    p = 2*p;
    mpass(c,N,p,a);
    p = 2*p;
  }
}
```

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

44

Ordenação por Fusão

```
void Mergesort(item a[], int N)
{ int i,p;
  item c[Max];
  // c é um vetor auxiliar e Max é o tamanho máximo de a

  p = 1;
  while (p < N)
  { mpass(a,N,p,c);
    p = 2*p;
    mpass(c,N,p,a);
    p = 2*p;
  }
}
```

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

45

Exercício

```
void Mergesort(item a[], int N)
{ int i,p;
  item c[Max];

  p = 1;
  while (p < N)
  { mpass(a,N,p,c);
    p = 2*p;
    mpass(c,N,p,a);
    p = 2*p;
  }
}
```

- Utilizando o algoritmo Mergesort, obtenha o número de comparações e movimentações em cada passo (**p**) 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

46

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								

47

Ordenação por Fusão: Análise

- 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 $O(N \log_2 N)$
- Entretanto, existem chamadas de procedimento envolvidas no processo, o que, normalmente, acarreta um esforço computacional maior
- O algoritmo seguinte, além de implementar a ordenação por intercalação utilizando um único vetor, o faz em um único algoritmo (sem chamadas de procedimento)

48

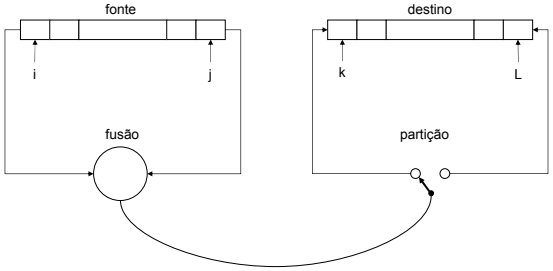
Ordenação por Fusão (1 vetor)

- Um único vetor pode ser facilmente utilizado para representar duas seqüências, se for interpretado como uma estrutura de dados com duas extremidades
- Ao invés de se efetuar a fusão a partir dos dados provenientes de dois conjuntos-fontes, os elementos são extraídos das duas extremidades do vetor

49

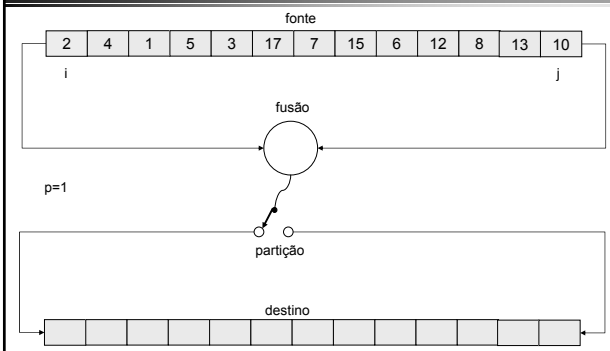
Ordenação por Fusão (1 vetor)

- Assim, a forma geral da fase combinada de partição-fusão pode ser ilustrada pela figura seguinte



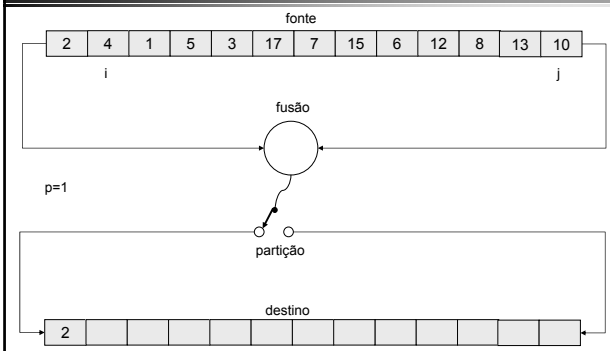
50

Ordenação por Fusão (1 vetor)



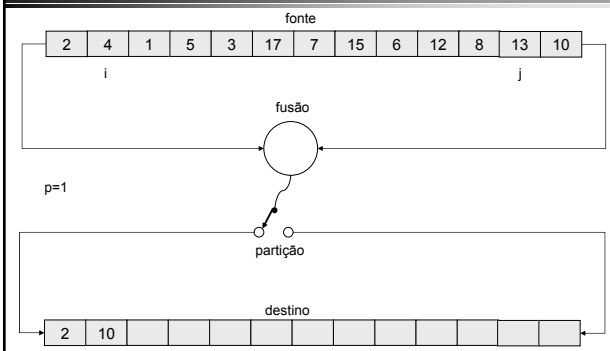
51

Ordenação por Fusão (1 vetor)



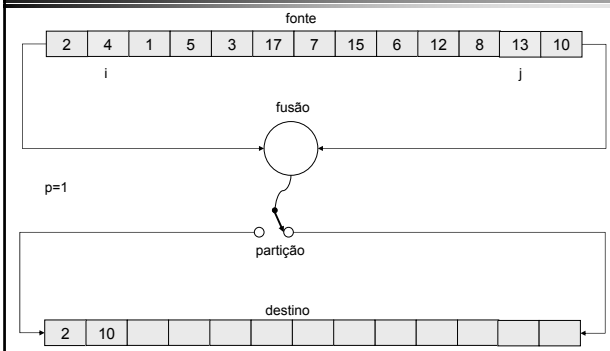
52

Ordenação por Fusão (1 vetor)



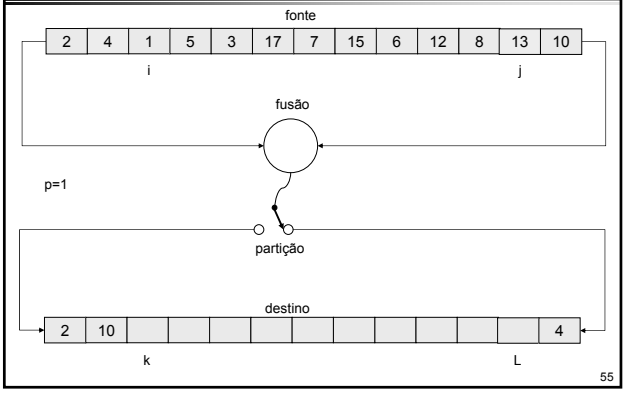
53

Ordenação por Fusão (1 vetor)



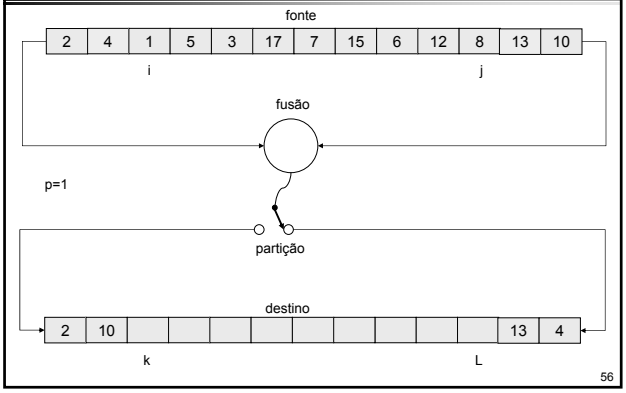
54

Ordenação por Fusão (1 vetor)



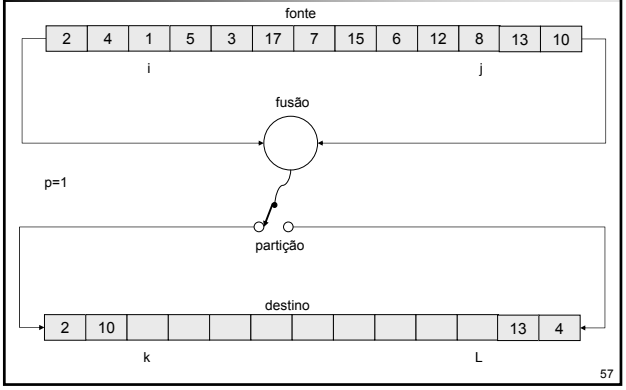
55

Ordenação por Fusão (1 vetor)



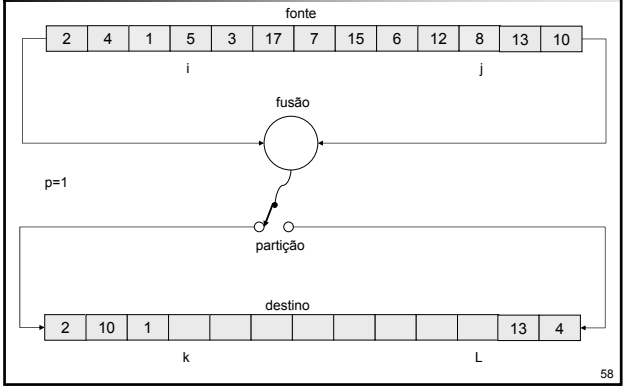
56

Ordenação por Fusão (1 vetor)



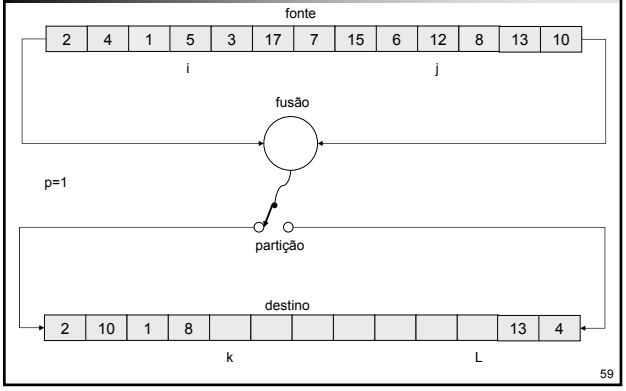
57

Ordenação por Fusão (1 vetor)



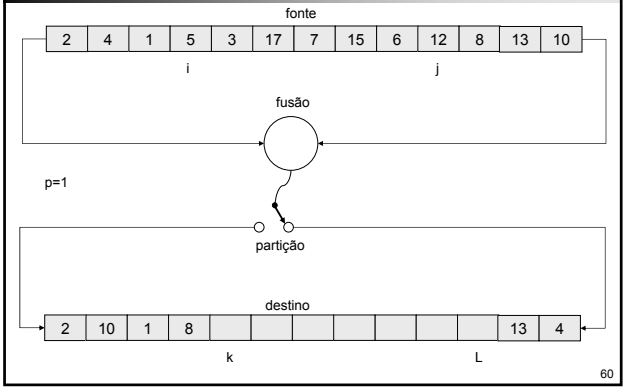
58

Ordenação por Fusão (1 vetor)



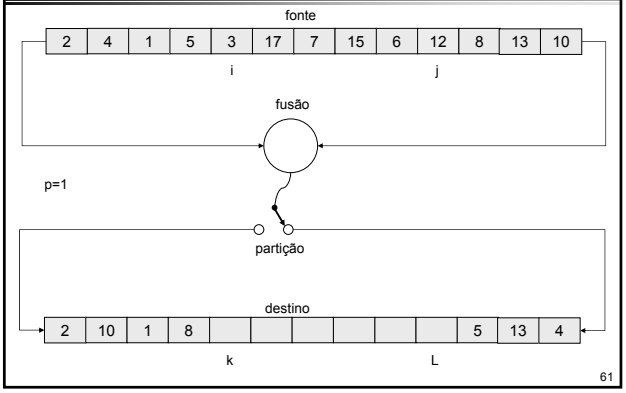
59

Ordenação por Fusão (1 vetor)



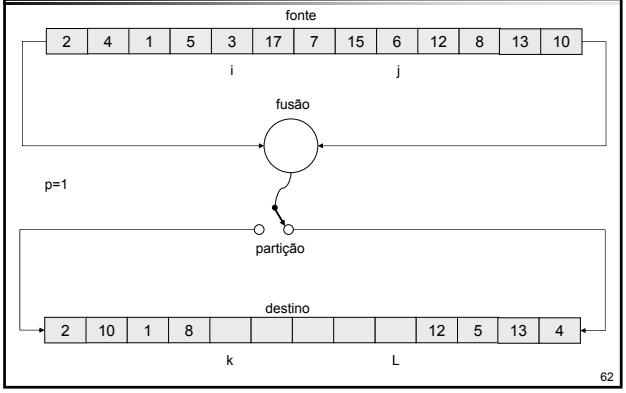
60

Ordenação por Fusão (1 vetor)



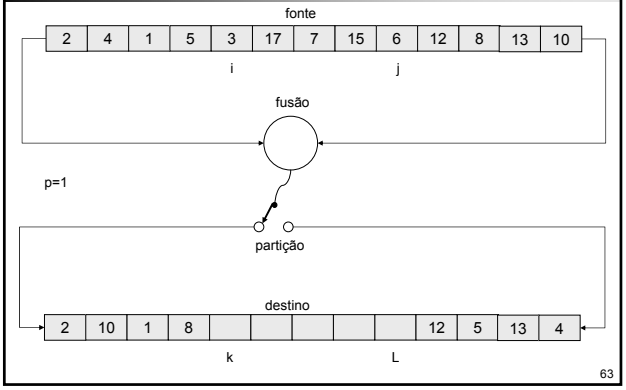
61

Ordenação por Fusão (1 vetor)



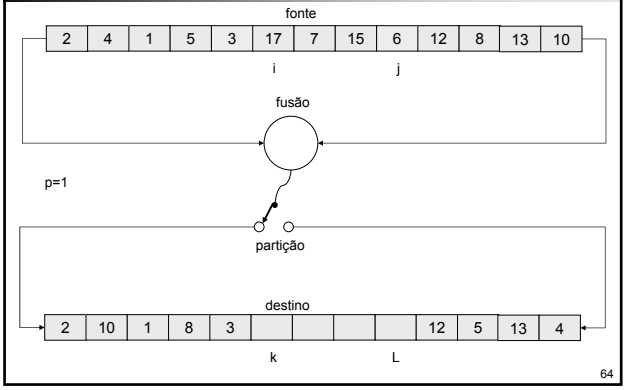
62

Ordenação por Fusão (1 vetor)



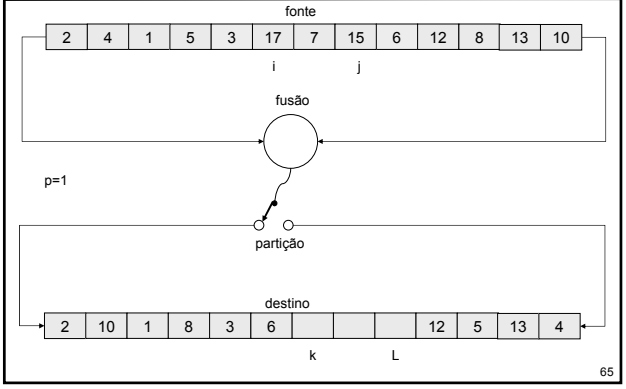
63

Ordenação por Fusão (1 vetor)



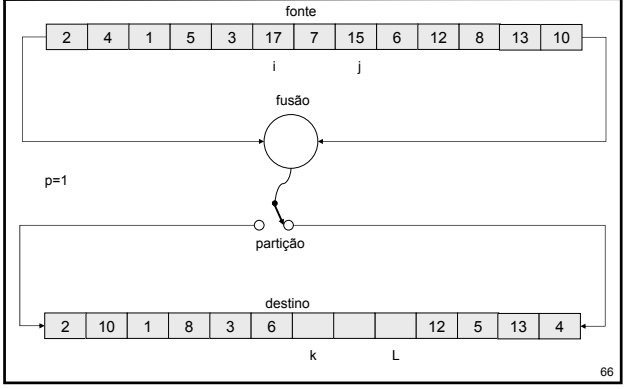
64

Ordenação por Fusão (1 vetor)



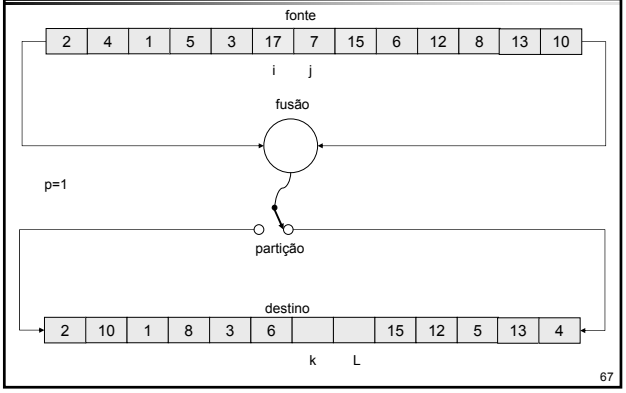
65

Ordenação por Fusão (1 vetor)



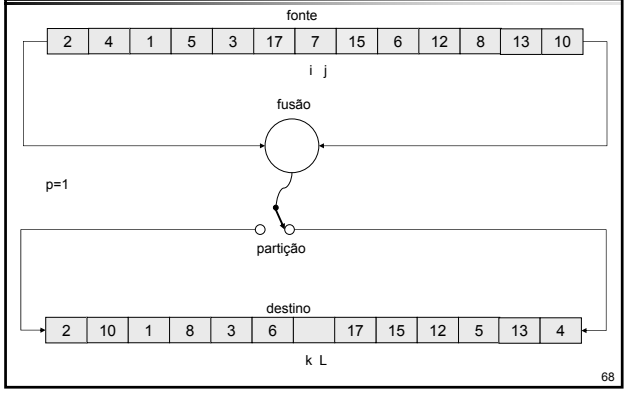
66

Ordenação por Fusão (1 vetor)



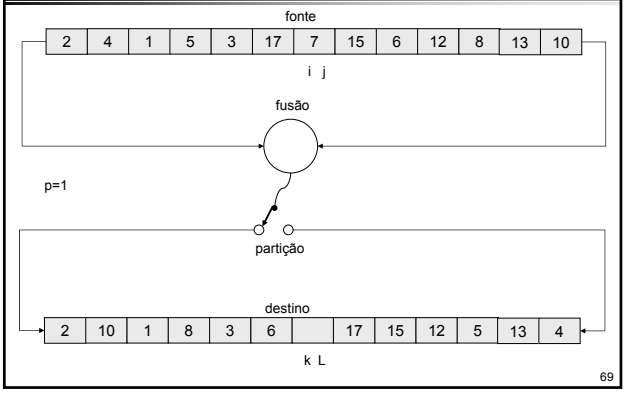
67

Ordenação por Fusão (1 vetor)



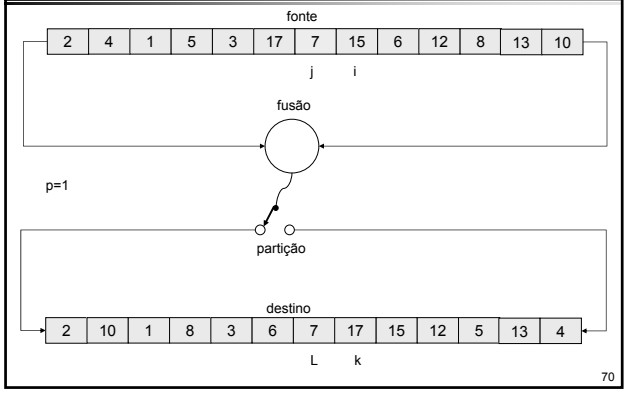
68

Ordenação por Fusão (1 vetor)



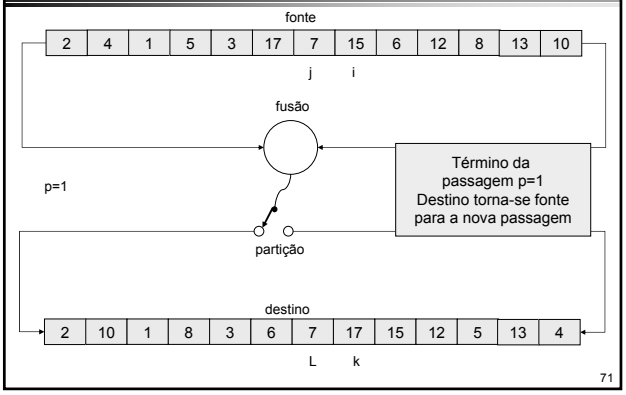
69

Ordenação por Fusão (1 vetor)



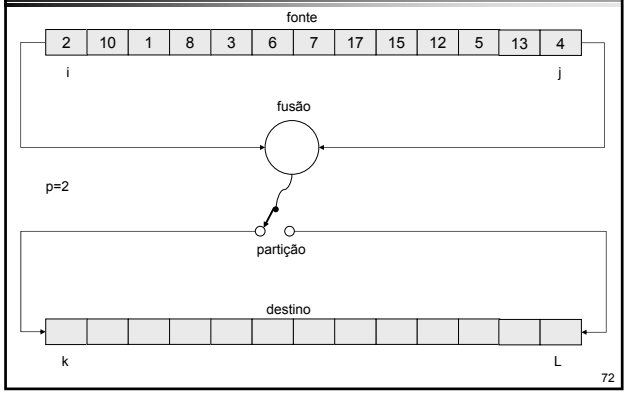
70

Ordenação por Fusão (1 vetor)



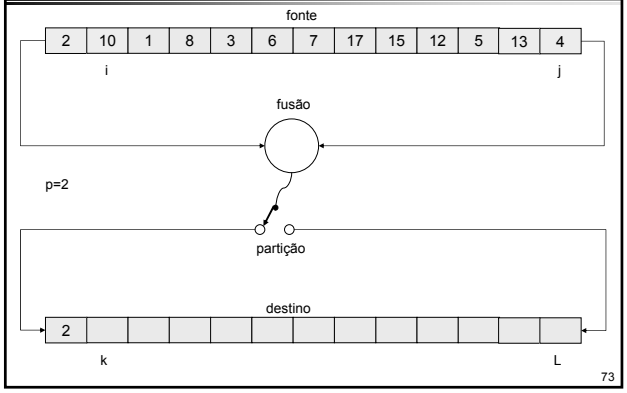
71

Ordenação por Fusão (1 vetor)



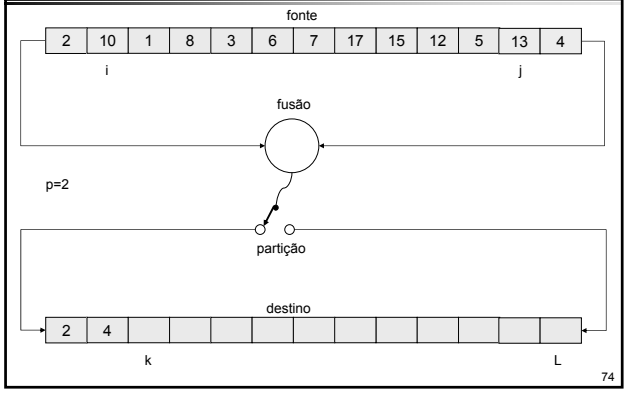
72

Ordenação por Fusão (1 vetor)



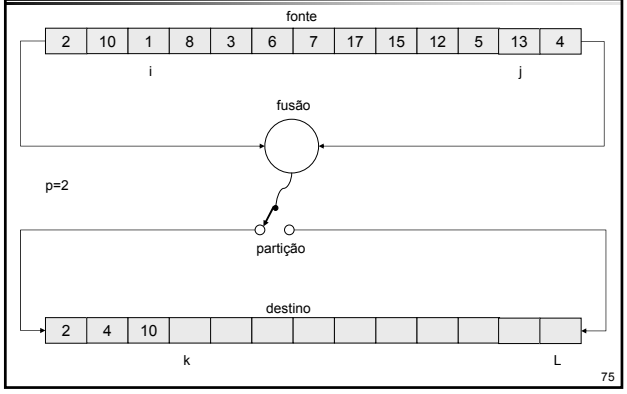
73

Ordenação por Fusão (1 vetor)



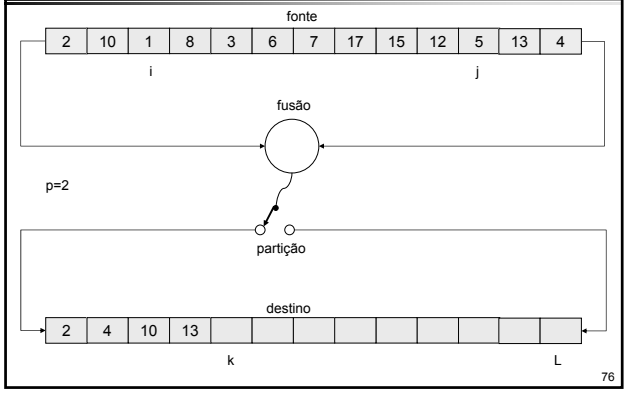
74

Ordenação por Fusão (1 vetor)



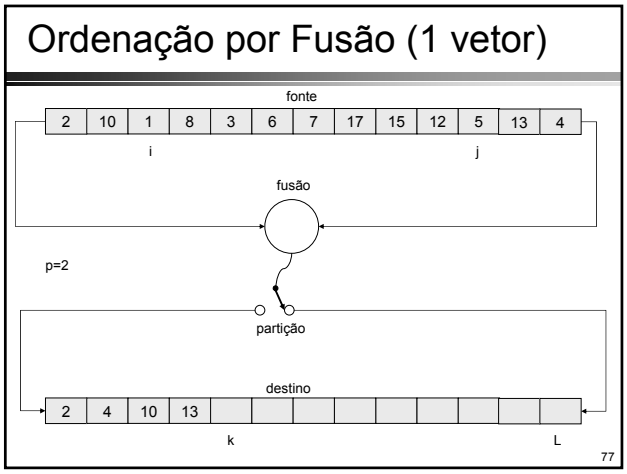
75

Ordenação por Fusão (1 vetor)



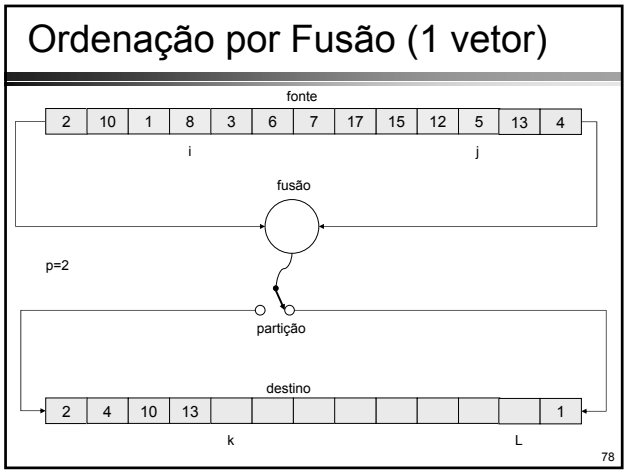
76

Ordenação por Fusão (1 vetor)



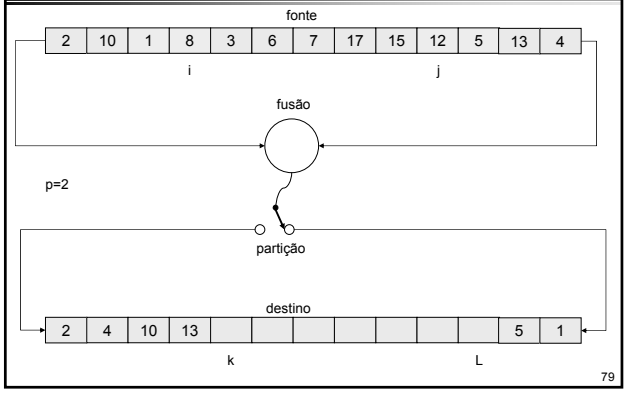
77

Ordenação por Fusão (1 vetor)



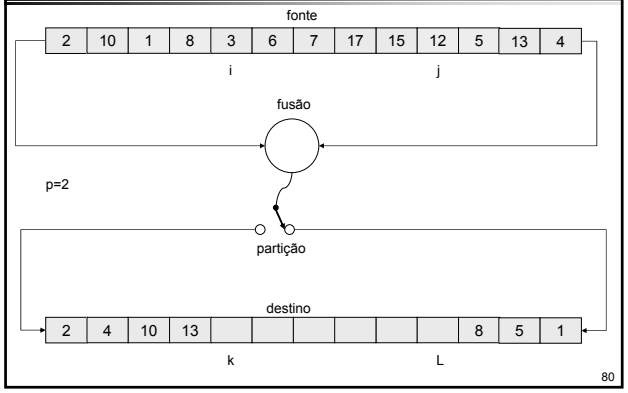
78

Ordenação por Fusão (1 vetor)



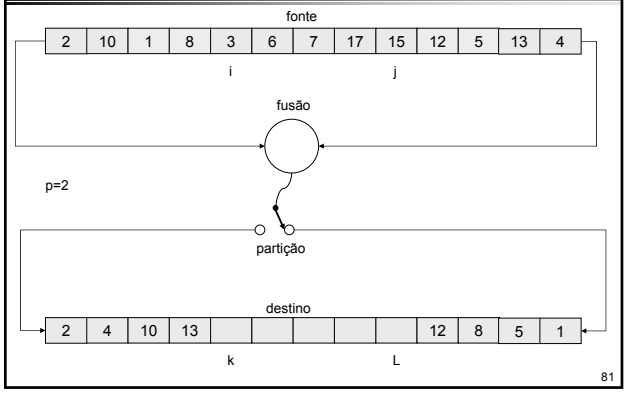
79

Ordenação por Fusão (1 vetor)



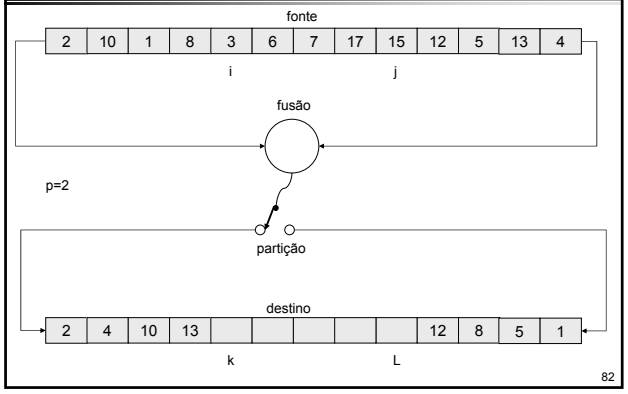
80

Ordenação por Fusão (1 vetor)



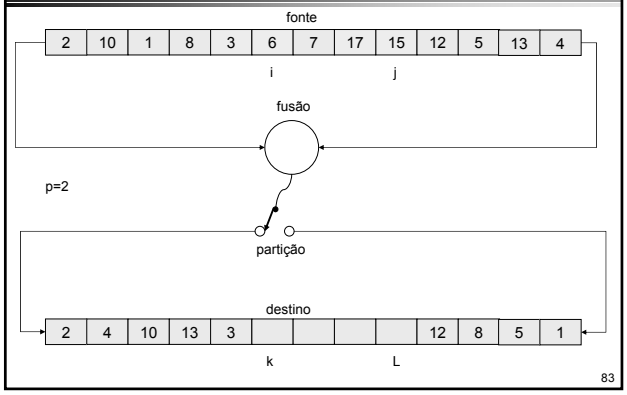
81

Ordenação por Fusão (1 vetor)



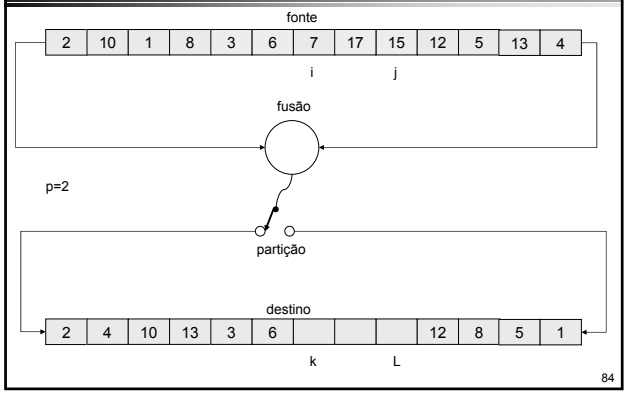
82

Ordenação por Fusão (1 vetor)



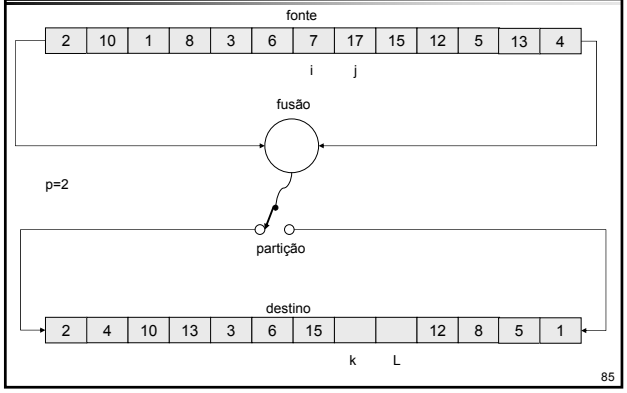
83

Ordenação por Fusão (1 vetor)



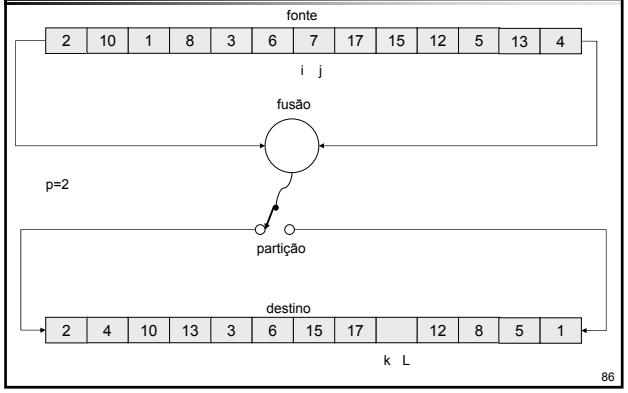
84

Ordenação por Fusão (1 vetor)



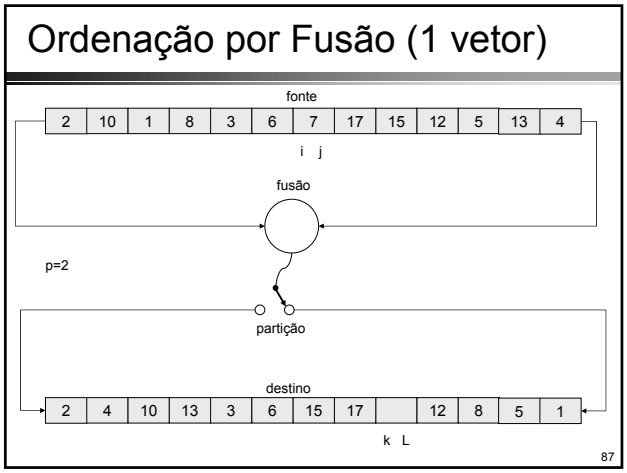
85

Ordenação por Fusão (1 vetor)



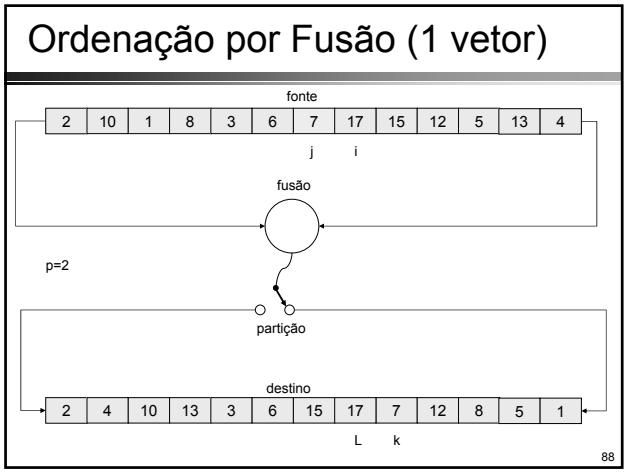
86

Ordenação por Fusão (1 vetor)



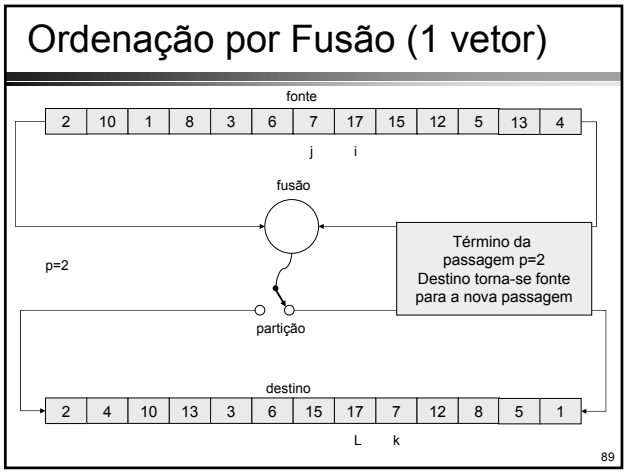
87

Ordenação por Fusão (1 vetor)



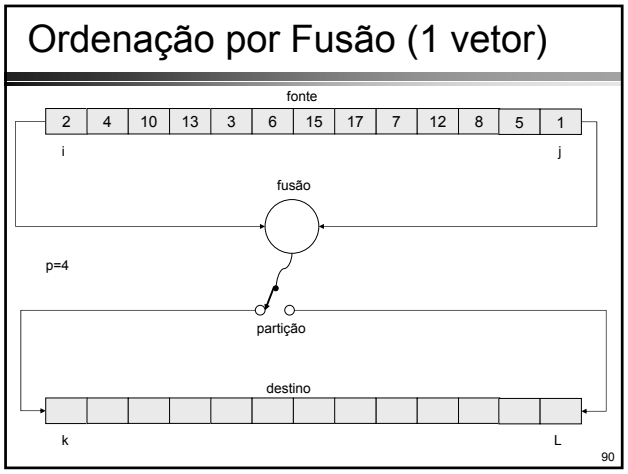
88

Ordenação por Fusão (1 vetor)



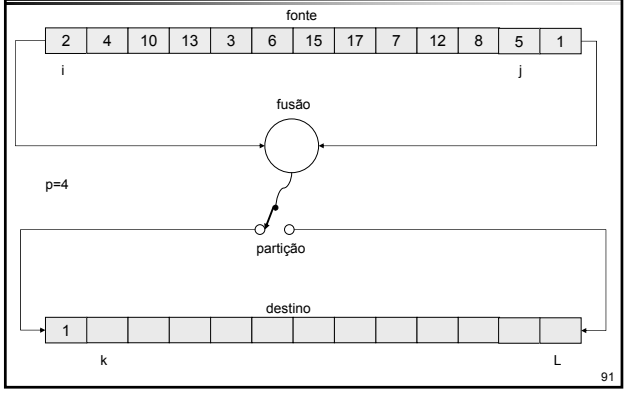
89

Ordenação por Fusão (1 vetor)



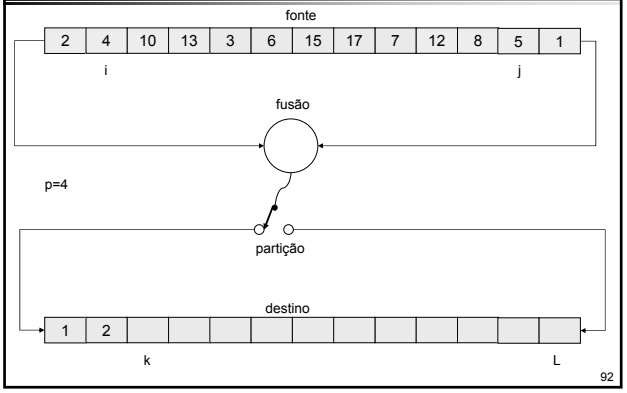
90

Ordenação por Fusão (1 vetor)



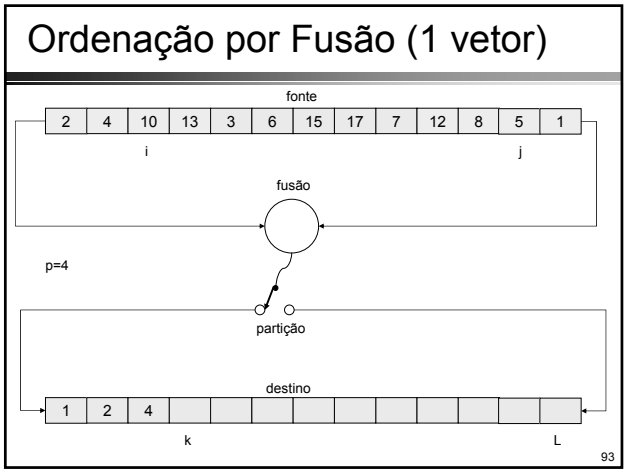
91

Ordenação por Fusão (1 vetor)



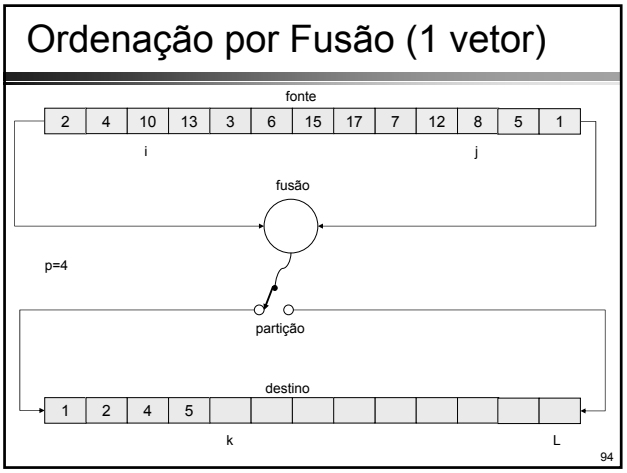
92

Ordenação por Fusão (1 vetor)



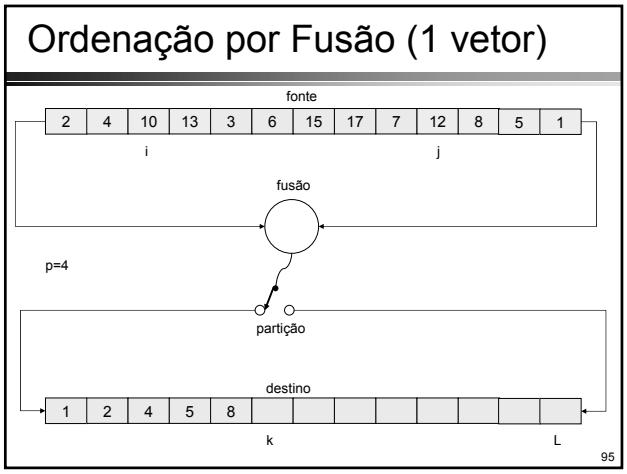
93

Ordenação por Fusão (1 vetor)



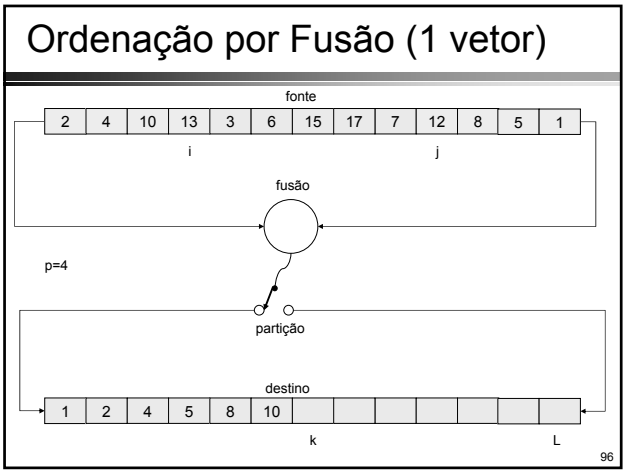
94

Ordenação por Fusão (1 vetor)



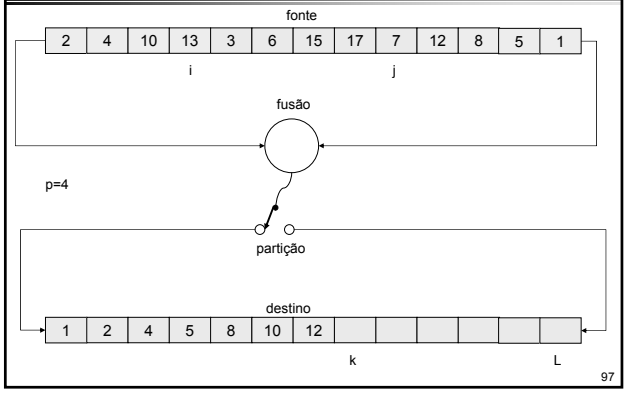
95

Ordenação por Fusão (1 vetor)



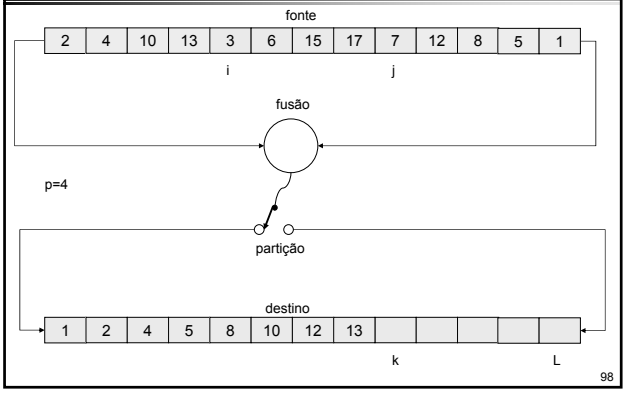
96

Ordenação por Fusão (1 vetor)



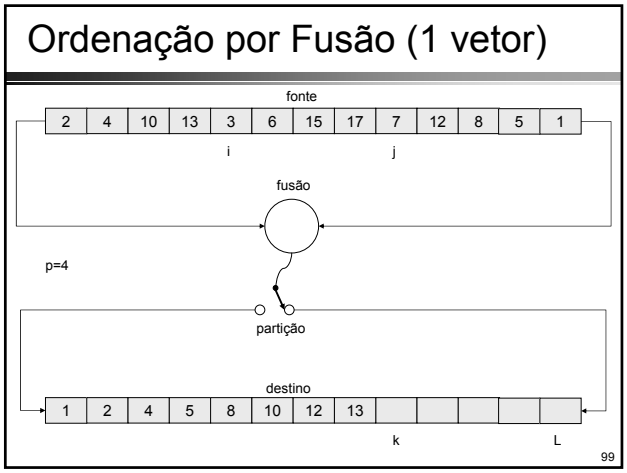
97

Ordenação por Fusão (1 vetor)



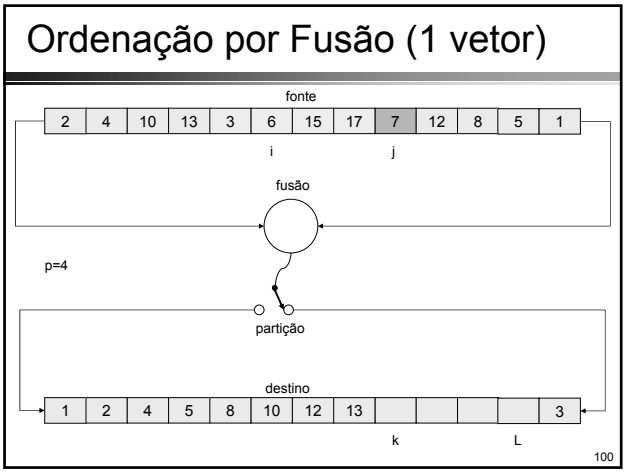
98

Ordenação por Fusão (1 vetor)



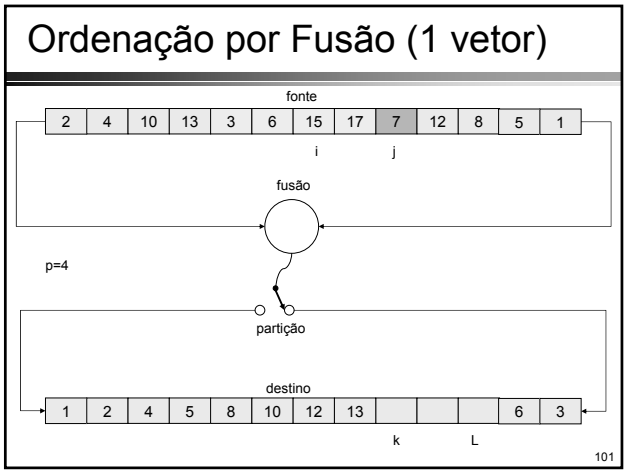
99

Ordenação por Fusão (1 vetor)



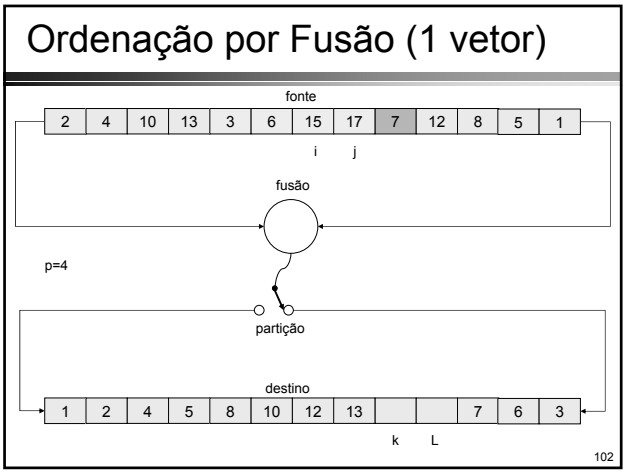
100

Ordenação por Fusão (1 vetor)



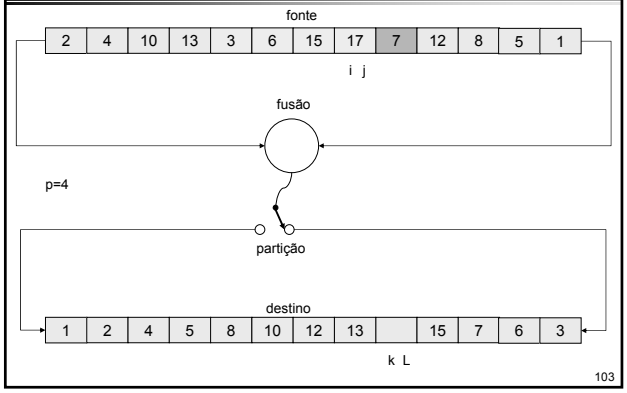
101

Ordenação por Fusão (1 vetor)



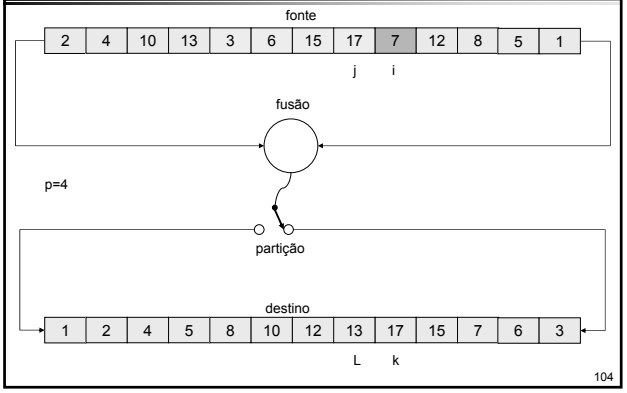
102

Ordenação por Fusão (1 vetor)



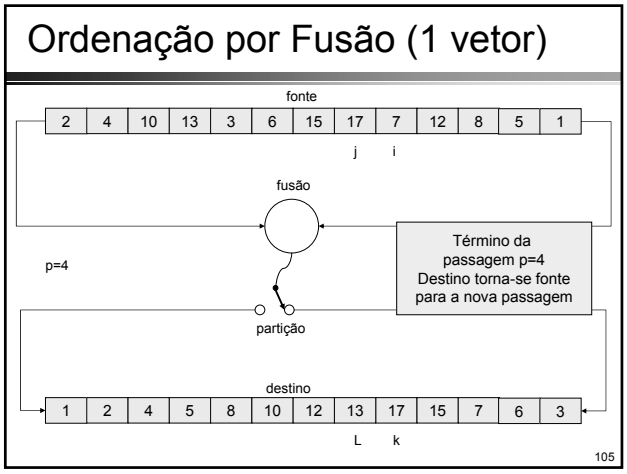
103

Ordenação por Fusão (1 vetor)



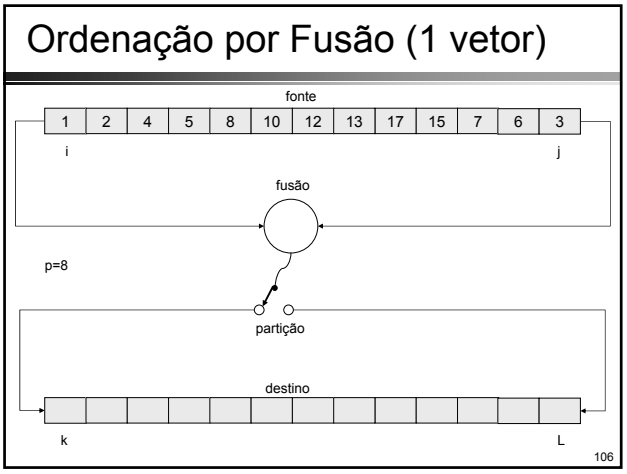
104

Ordenação por Fusão (1 vetor)



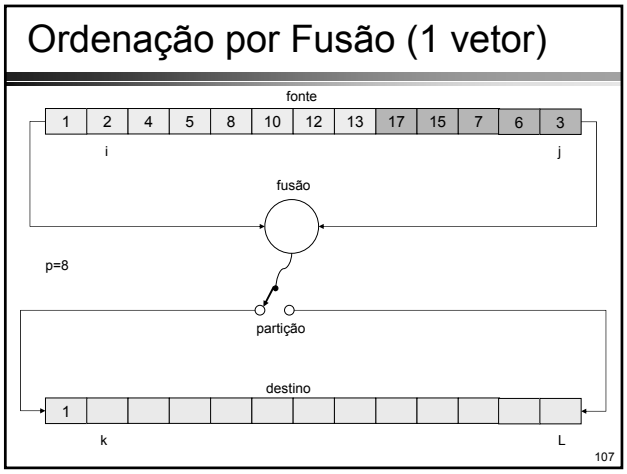
105

Ordenação por Fusão (1 vetor)



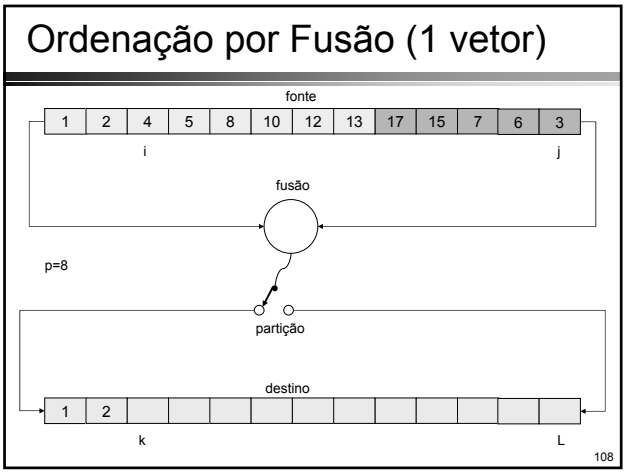
106

Ordenação por Fusão (1 vetor)



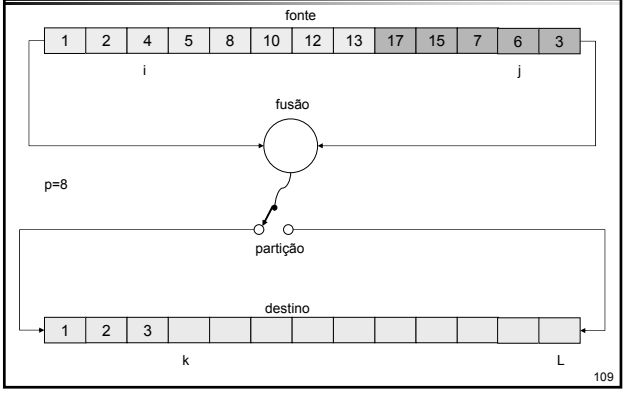
107

Ordenação por Fusão (1 vetor)



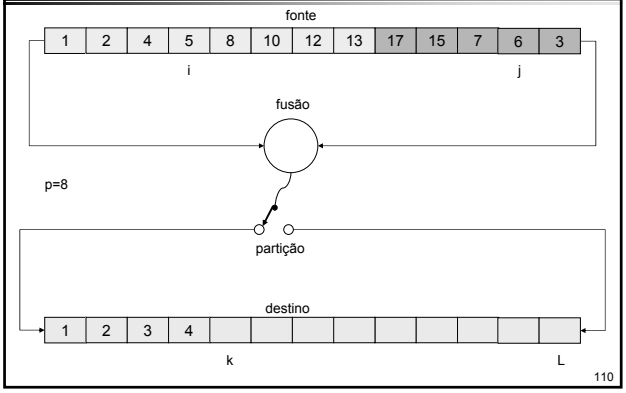
108

Ordenação por Fusão (1 vetor)



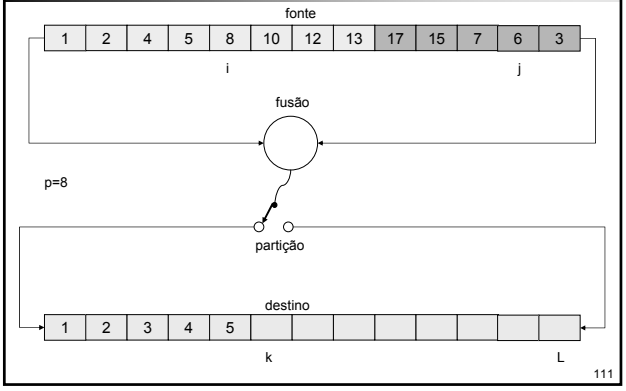
109

Ordenação por Fusão (1 vetor)



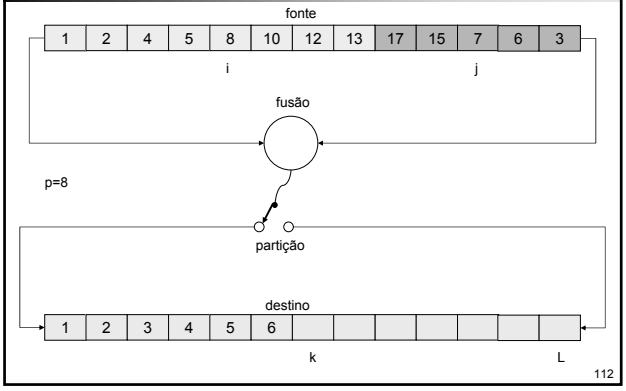
110

Ordenação por Fusão (1 vetor)



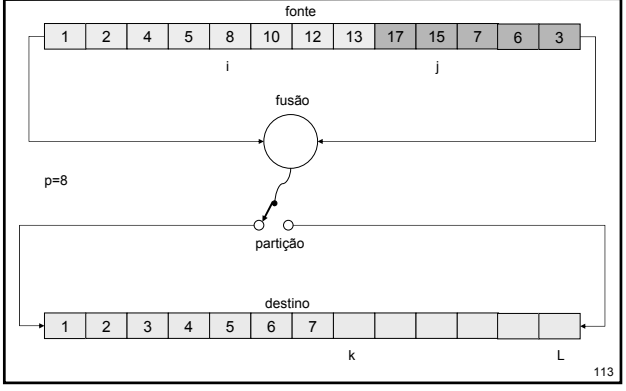
111

Ordenação por Fusão (1 vetor)



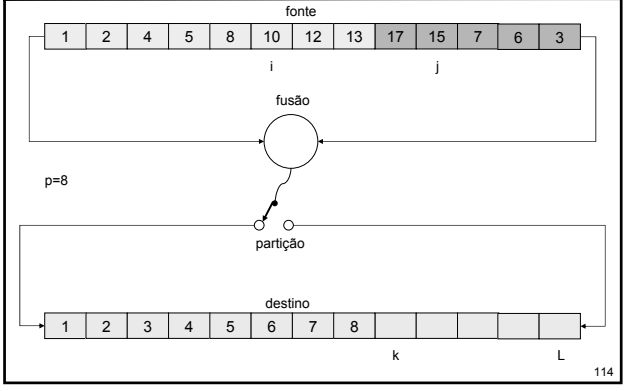
112

Ordenação por Fusão (1 vetor)



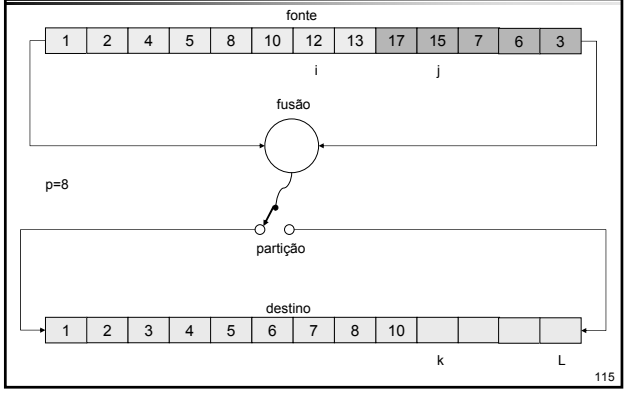
113

Ordenação por Fusão (1 vetor)



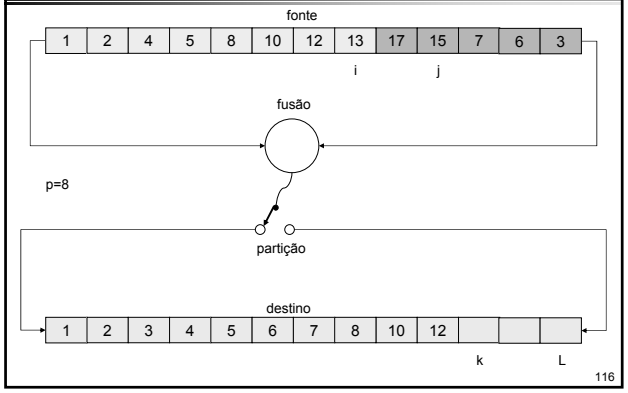
114

Ordenação por Fusão (1 vetor)



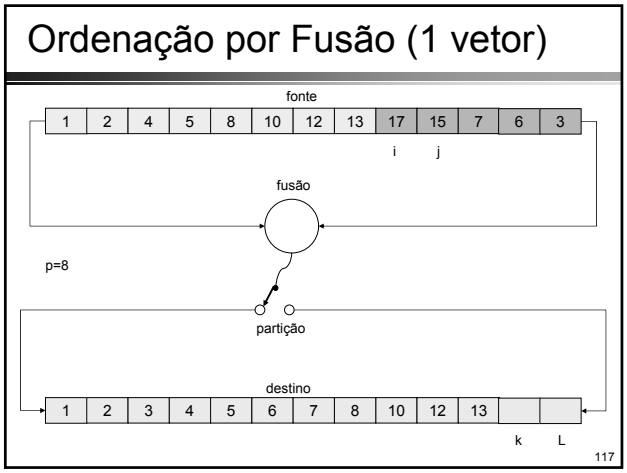
115

Ordenação por Fusão (1 vetor)



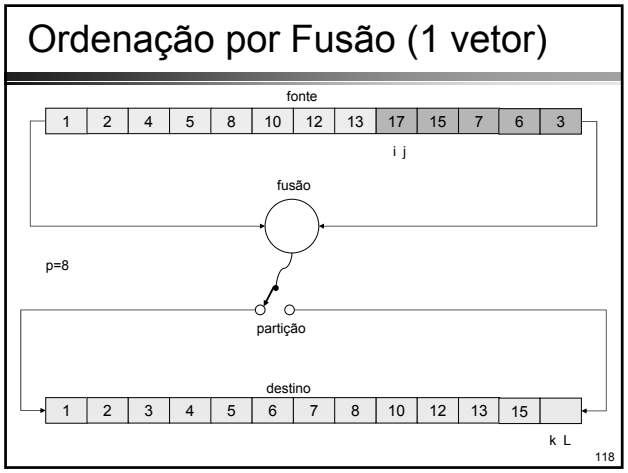
116

Ordenação por Fusão (1 vetor)



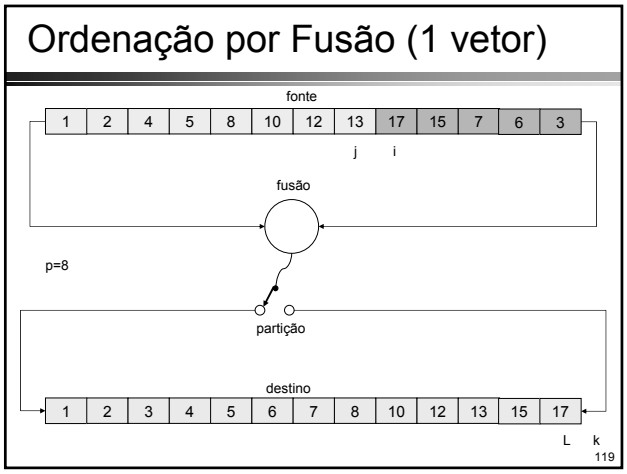
117

Ordenação por Fusão (1 vetor)



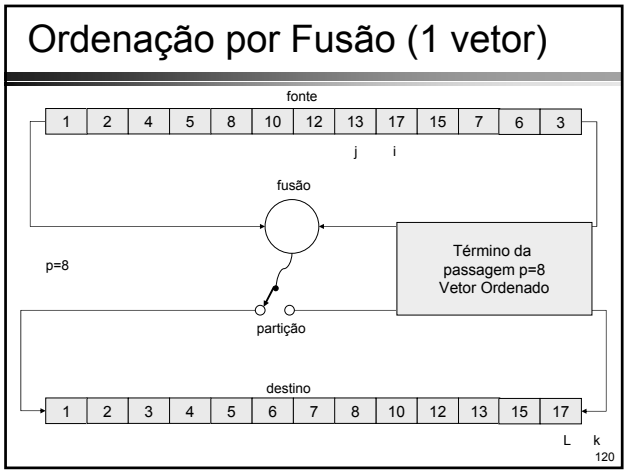
118

Ordenação por Fusão (1 vetor)



119

Ordenação por Fusão (1 vetor)



120

Resumo

- Aparentemente, o algoritmo de ordenação por fusão é bastante competitivo, mesmo quando comparado aos métodos mais sofisticados de ordenação vistos nas aulas anteriores
- Entretanto, a sobrecarga geral necessária à manipulação de índices é relativamente alta e uma desvantagem decisiva é a necessidade de uma área de armazenamento de **2N** elementos
- Esta é a razão pela qual a ordenação por fusão raramente é utilizada para vetores, isto é, para dados armazenados em memória principal

121