Merge sort

Ordenação por Intercalação (Merge sort)

- A ideia básica desse método é intercalar as duas metades de uma lista desejada quando estas já se encontram ordenadas.

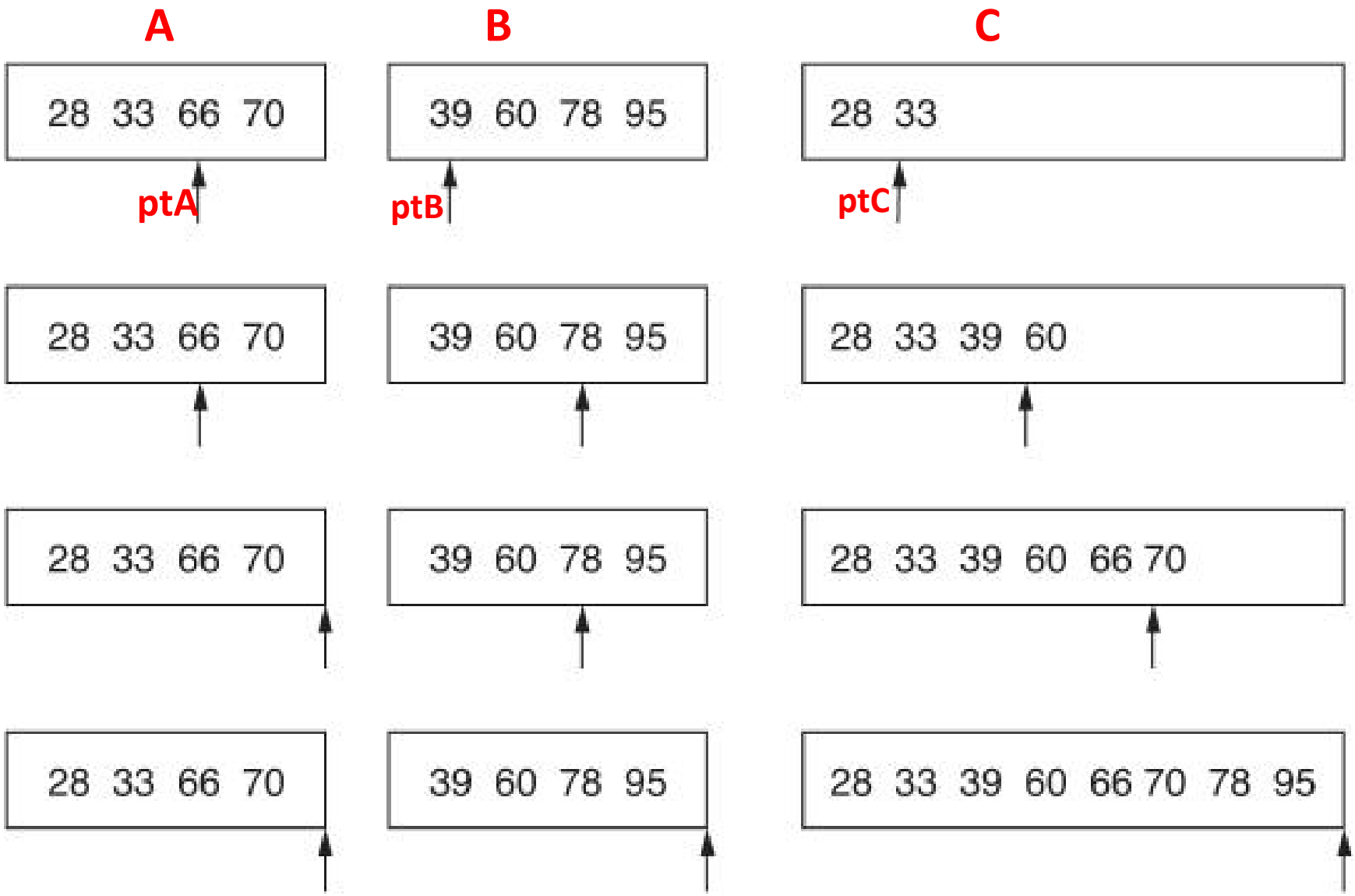
- Na realidade, deseja-se então ordenar primeiramente as duas metades, o que pode ser feito utilizando recursivamente o mesmo conceito.

Como funciona um processo de intercalação?

- Um processo de intercalação funciona assim:

- Sejam duas listas A e B, ordenadas, com respectivamente n e m elementos.
- As duas listas são percorridas por ponteiros ptA e ptB, armazenando o resultado da intercalação na lista C, apontada pelo ponteiro ptC.
- O primeiro elemento de A é comparado com o primeiro elemento de B; o menor valor é colocado em C.
  - O ponteiro da lista onde se encontra o menor valor é incrementado, assim como o ponteiro da lista resultado;
  - o processo se repete até que uma das listas seja esgotada.

Exemplo de uma intercalação



Como funciona o Mergesort?

Problema: Seja L a lista que se deseja ordenar.

Solução: (dividir para conquistar) O método de ordenação por intercalação consiste em dividir a lista original em duas metades e ordená-las.

- O resultado são duas listas ordenadas que podem ser intercaladas.
- Para ordenar cada uma das metades o processo considerado é o mesmo, sendo o problema dividido em problemas menores, que são sucessivamente solucionados.

Como funciona o Mergesort?

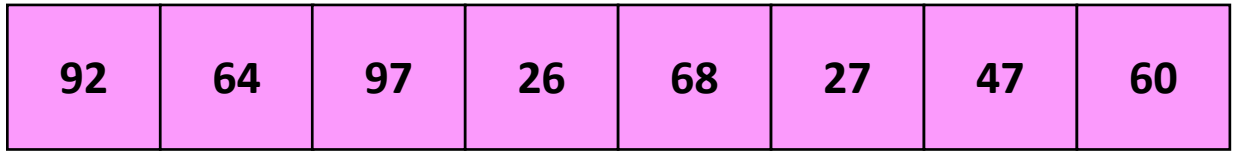
Solução:

- divide recursivamente o conjunto de dados até que cada subconjunto possua 1 elemento.

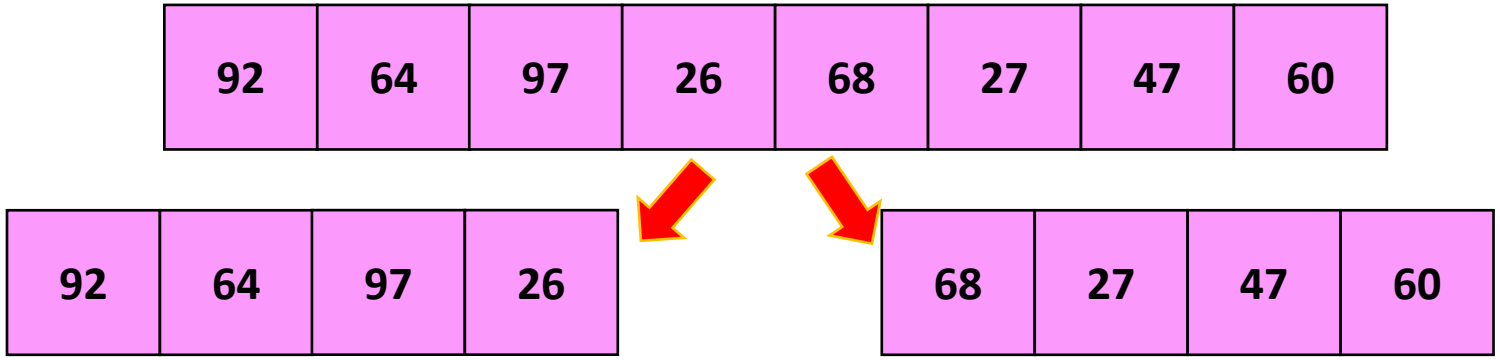
- combina 2 subconjuntos para obter 1 conjunto maior e ordenado.

- repete o processo até obter 1 conjunto ordenado.

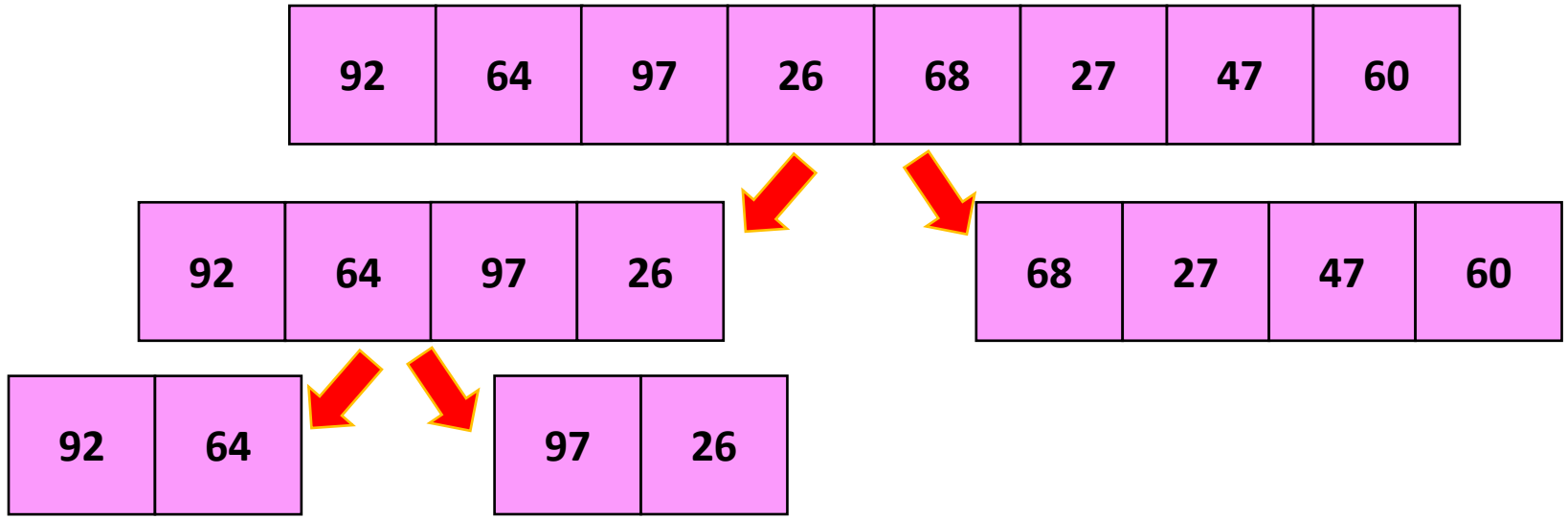
Exemplificando...



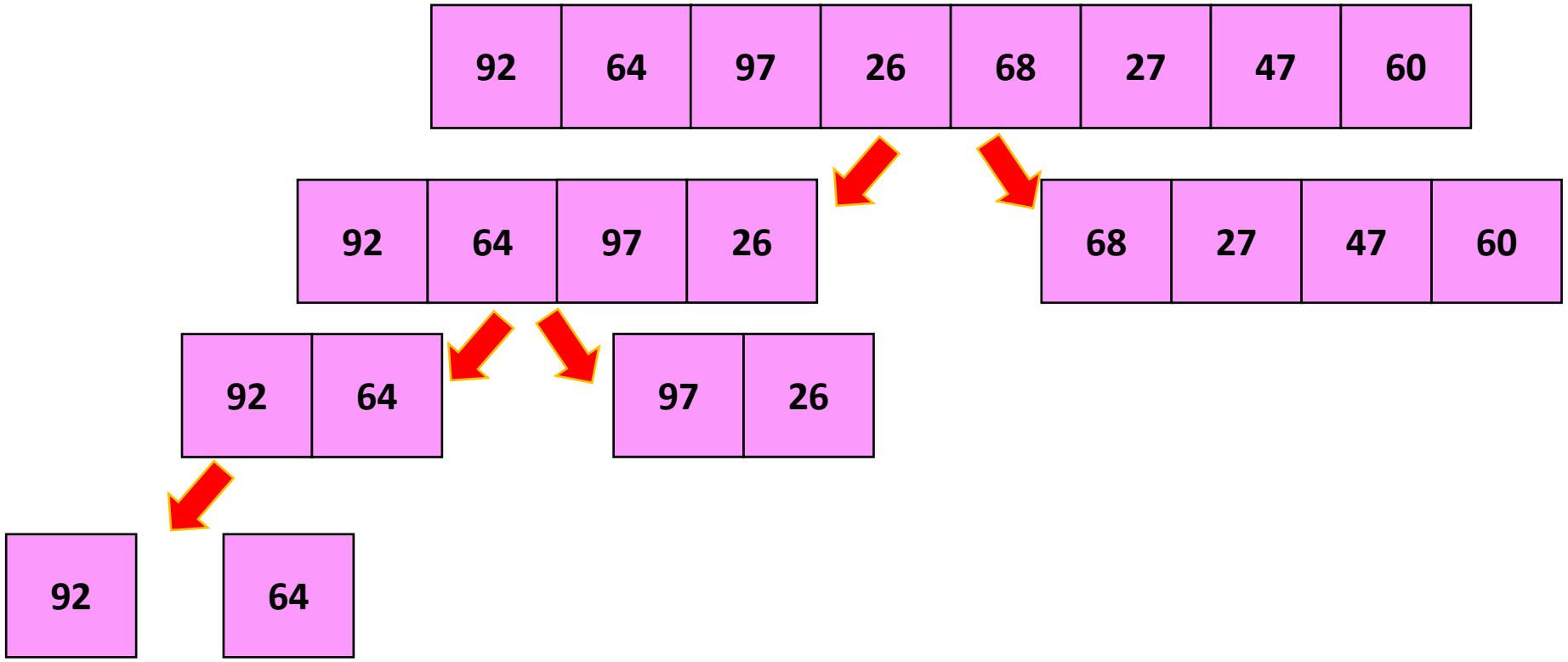
Exemplificando...



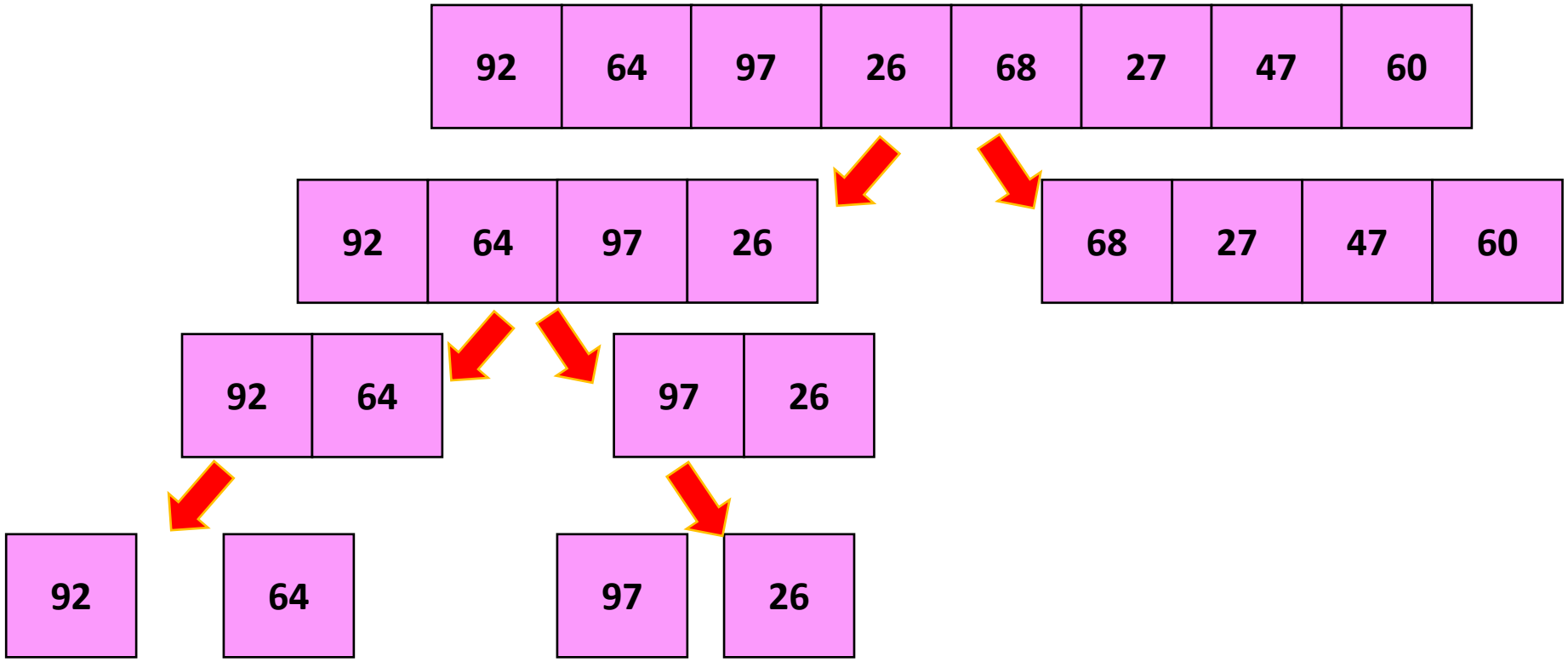
Exemplificando...



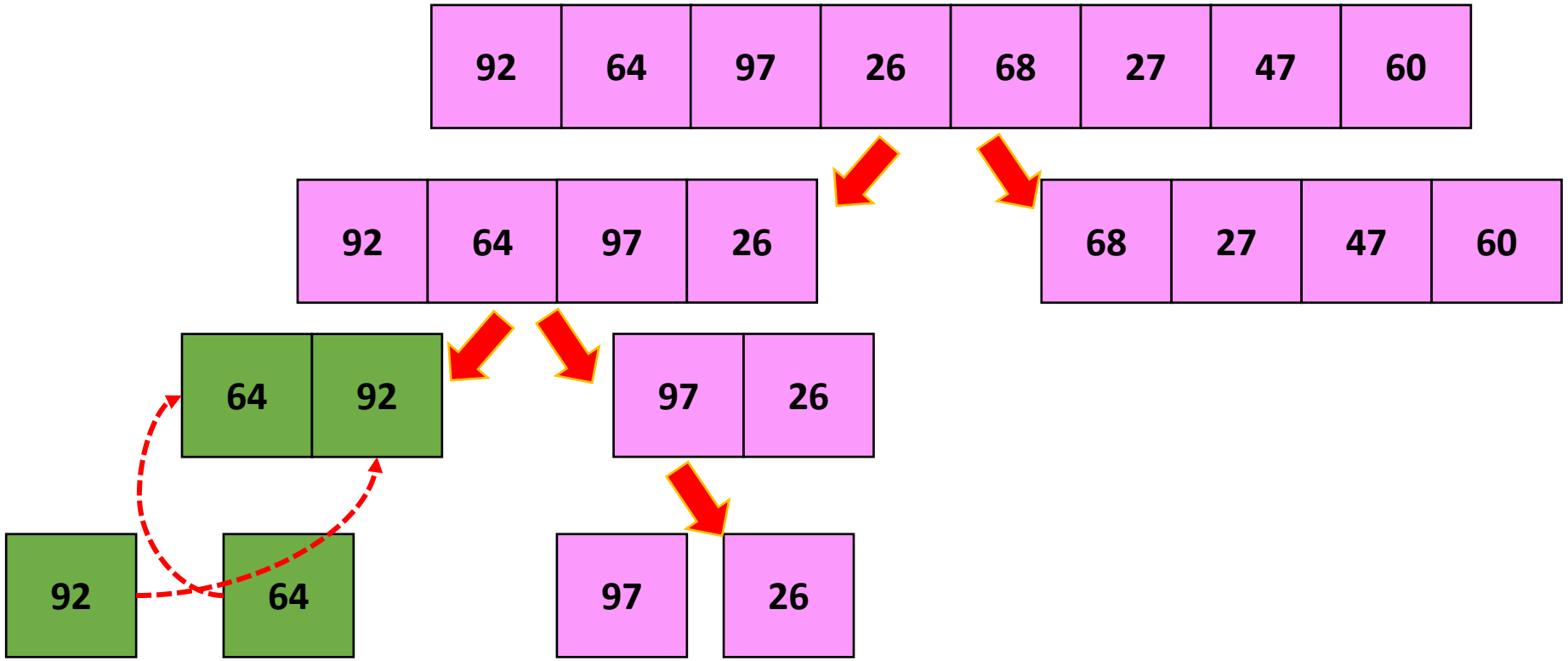
Exemplificando...



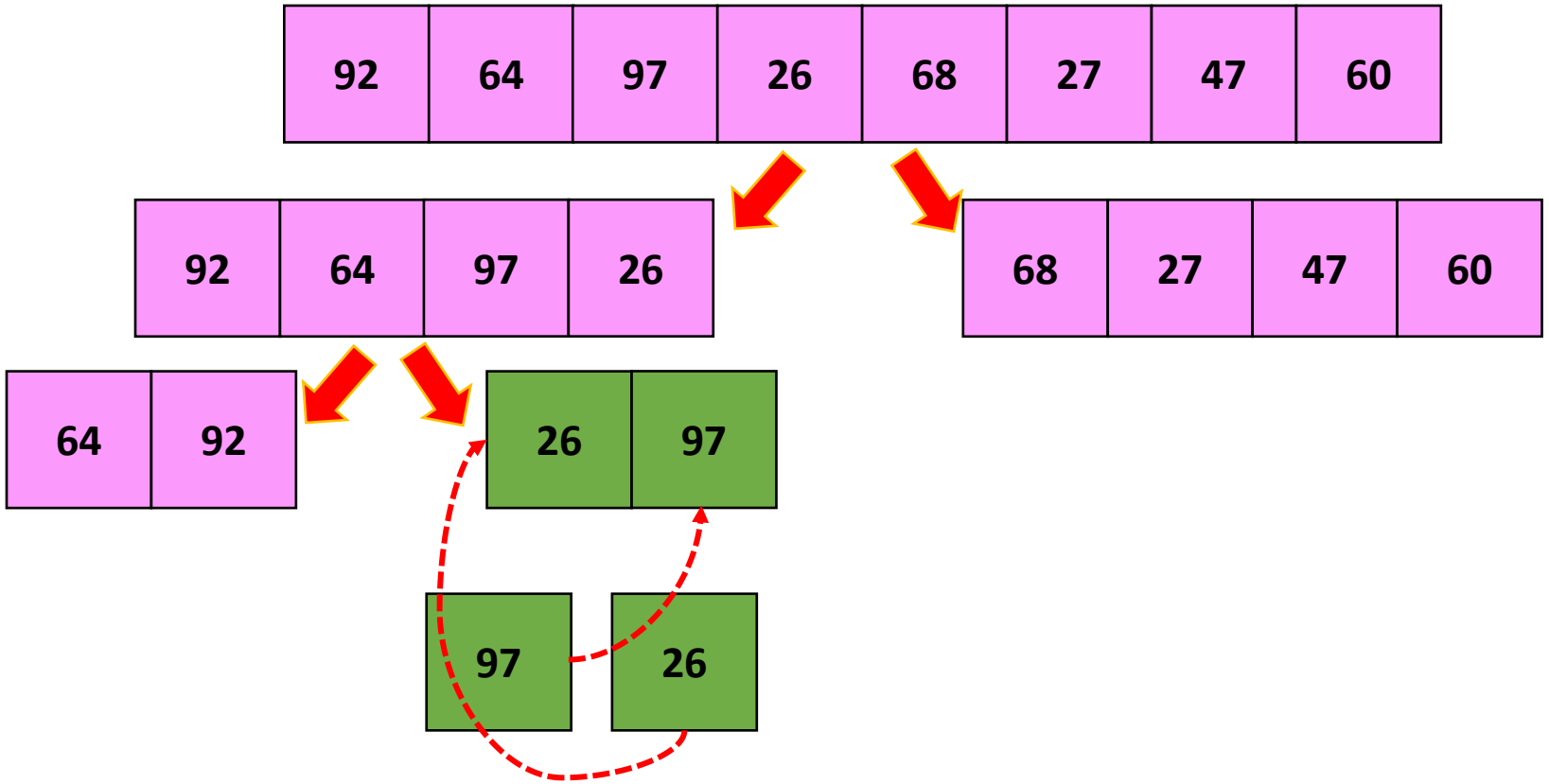
Exemplificando...



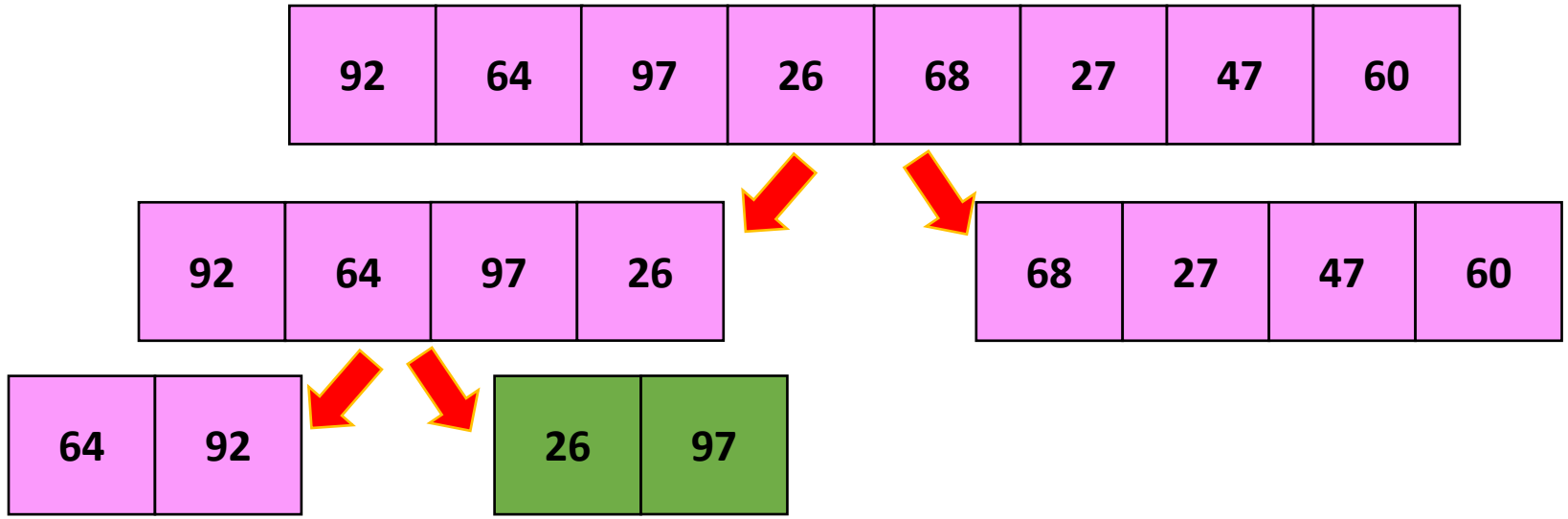
Exemplificando...



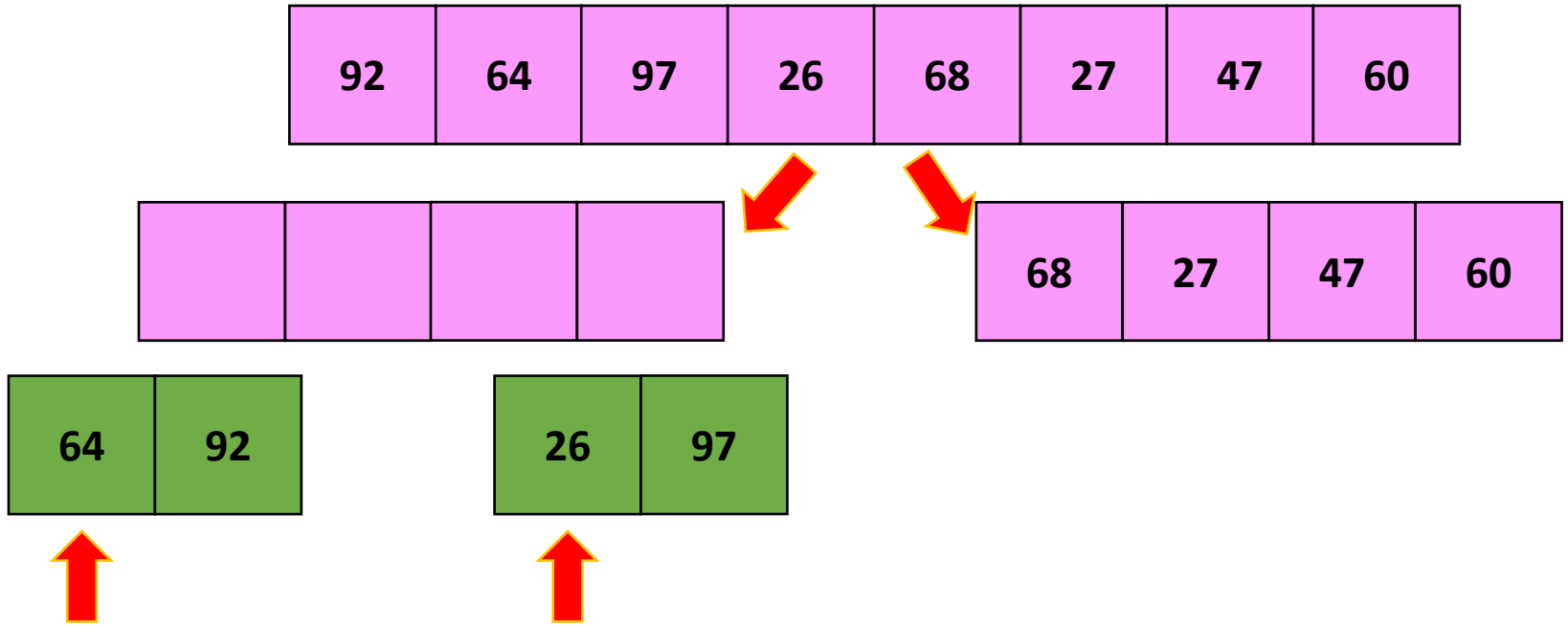
Exemplificando...



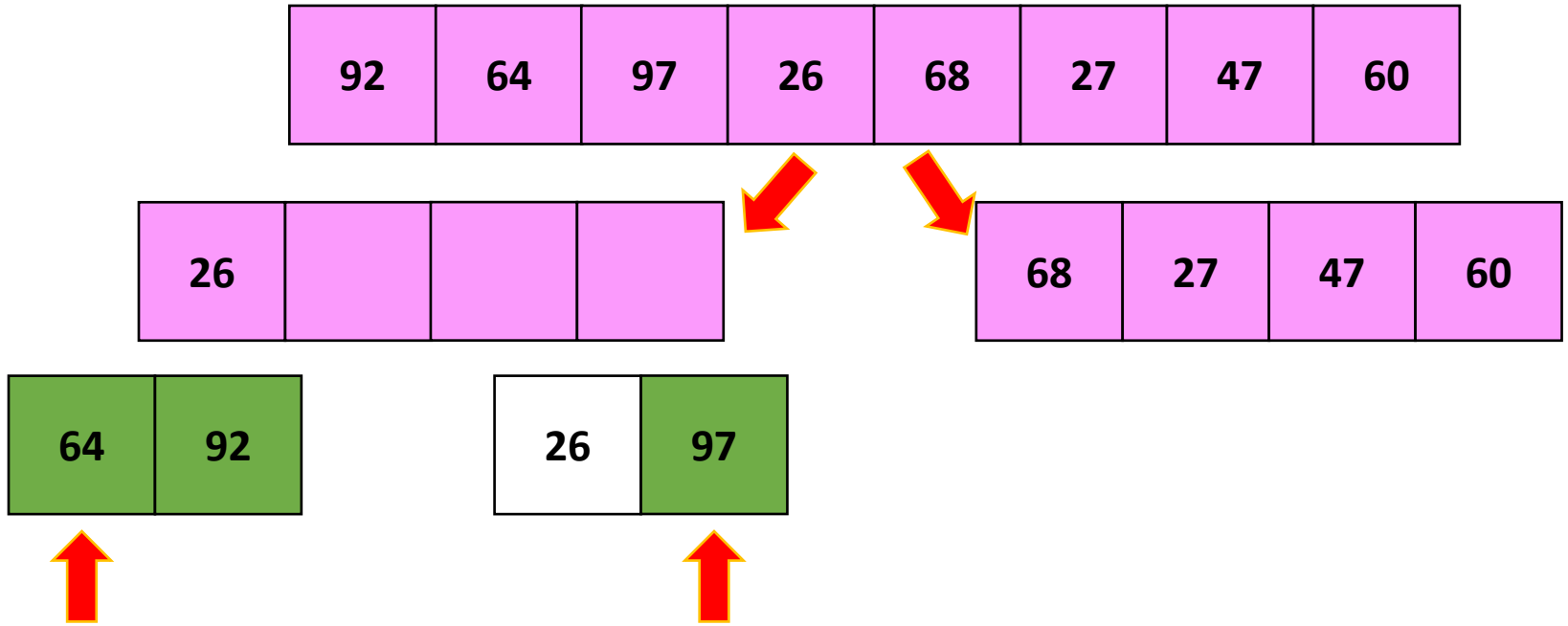
Exemplificando...



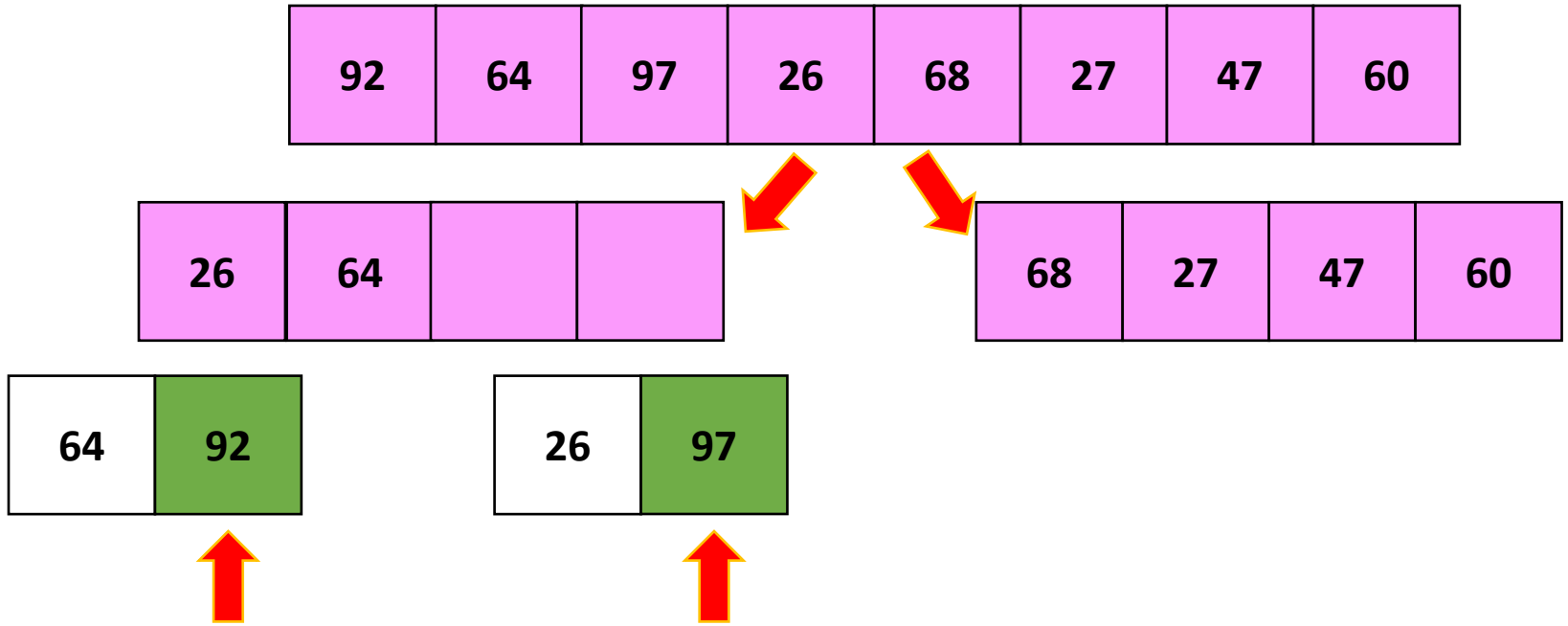
Exemplificando...



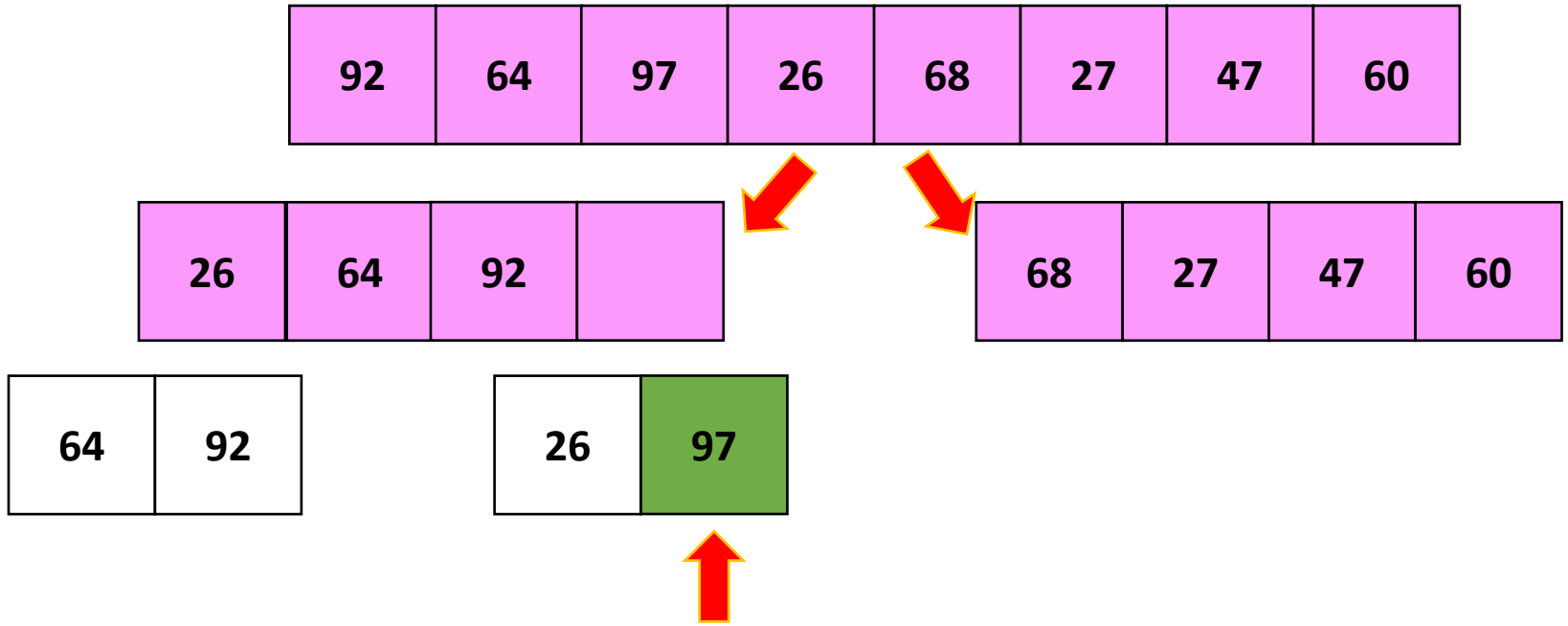
Exemplificando...



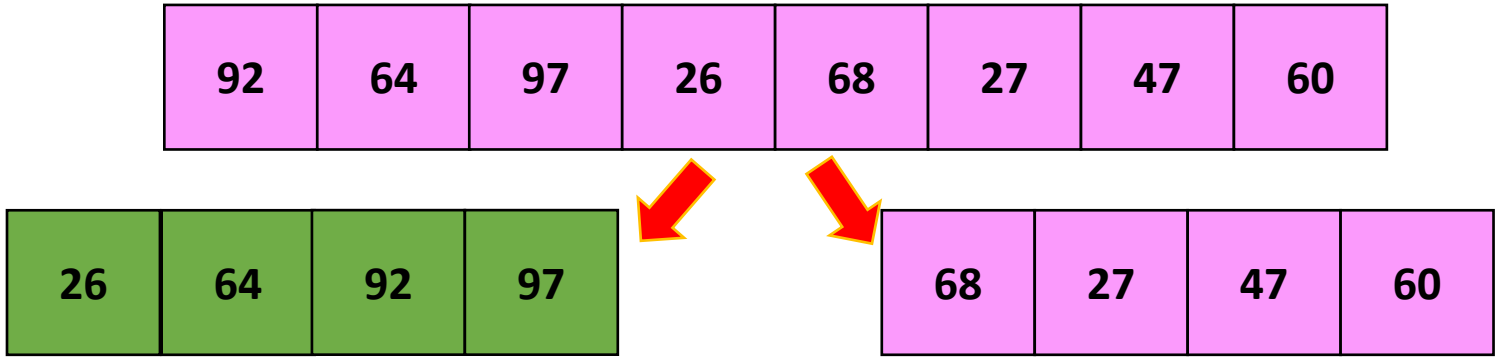
Exemplificando...



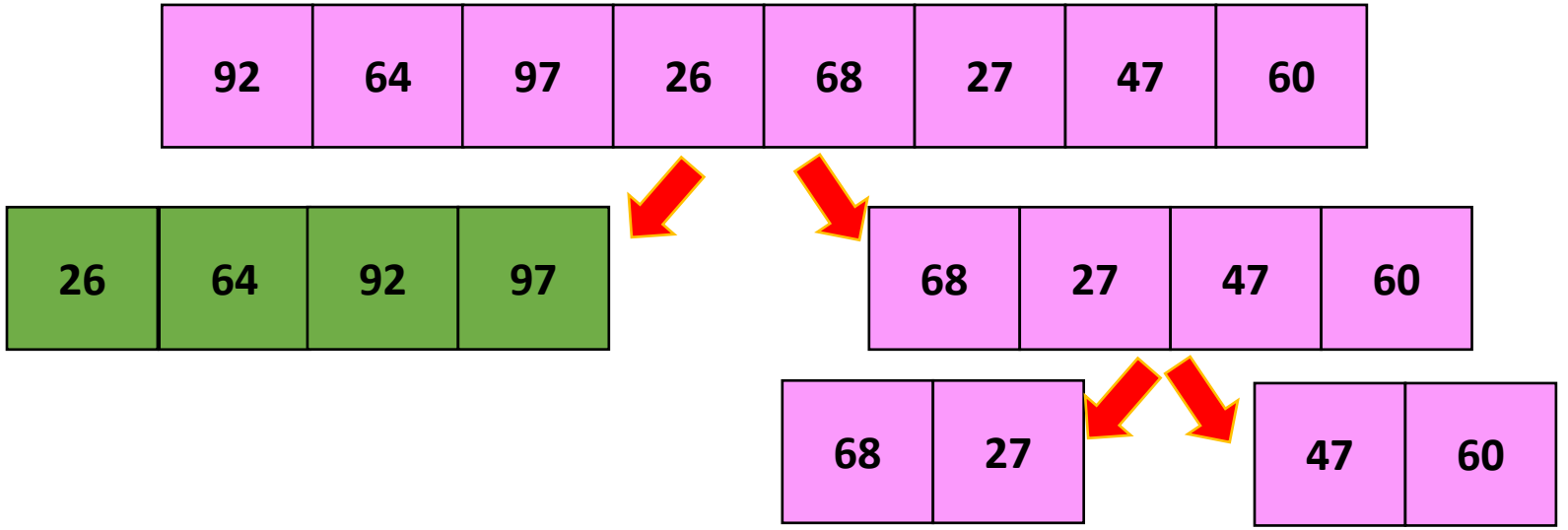
Exemplificando...



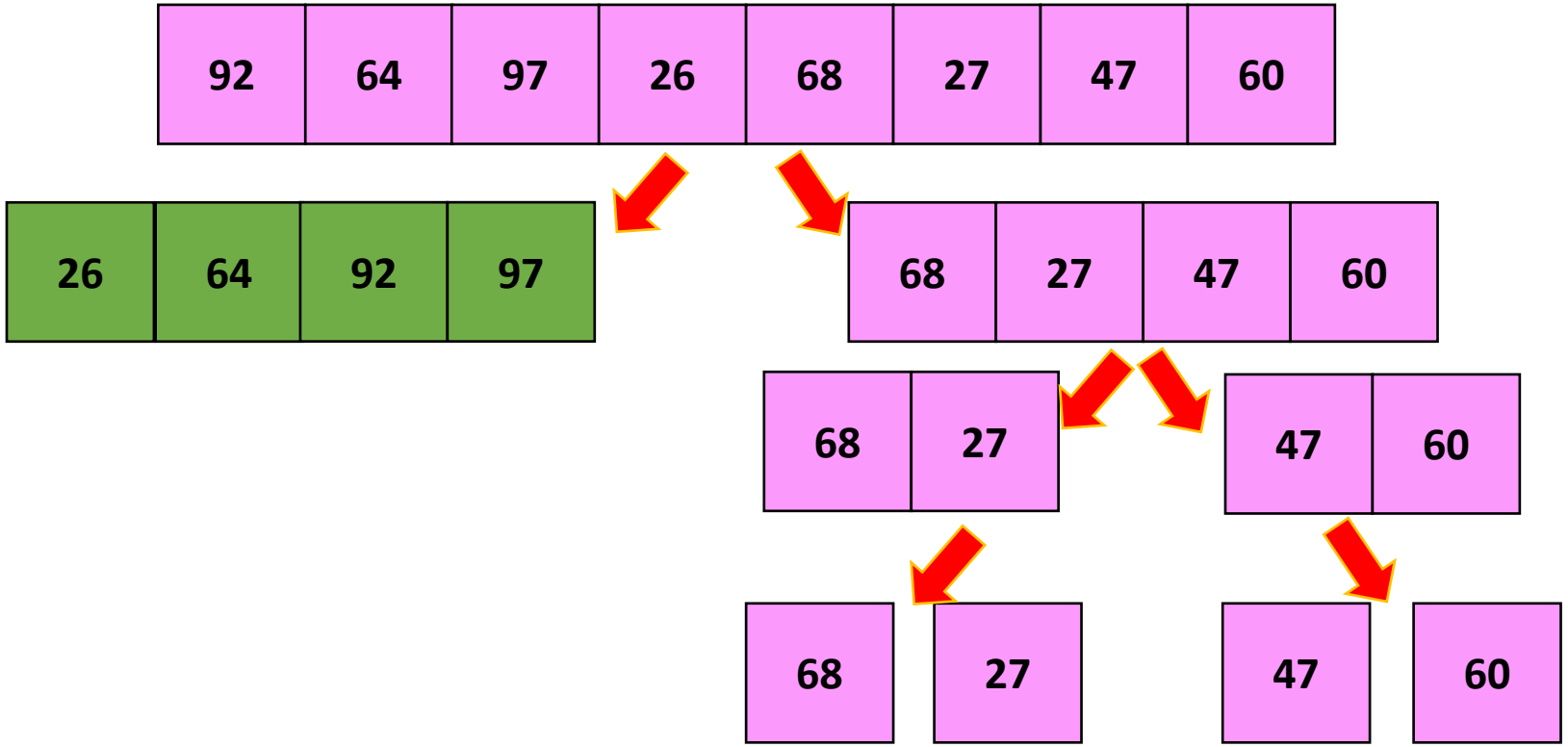
Exemplificando...



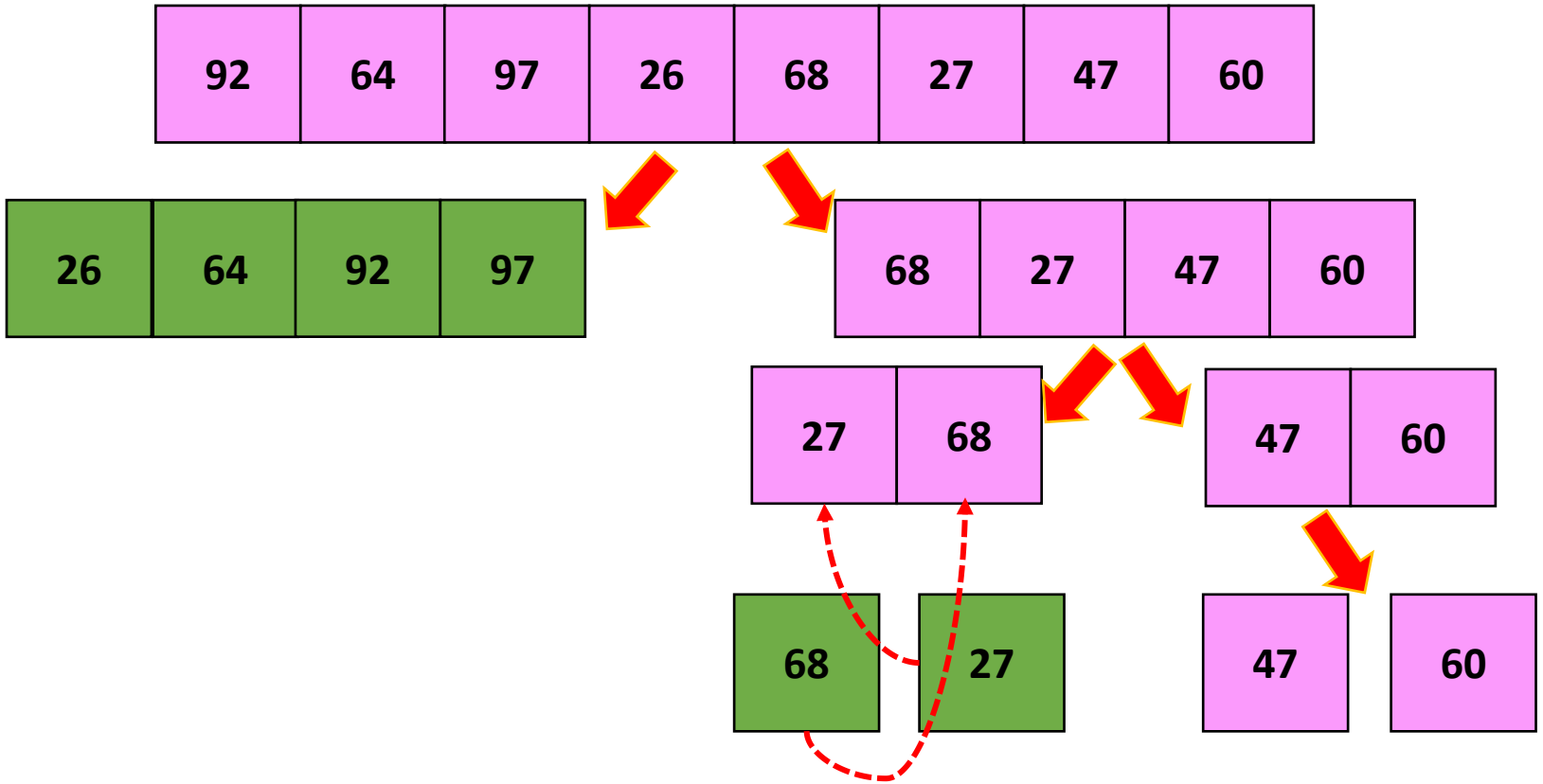
Exemplificando...



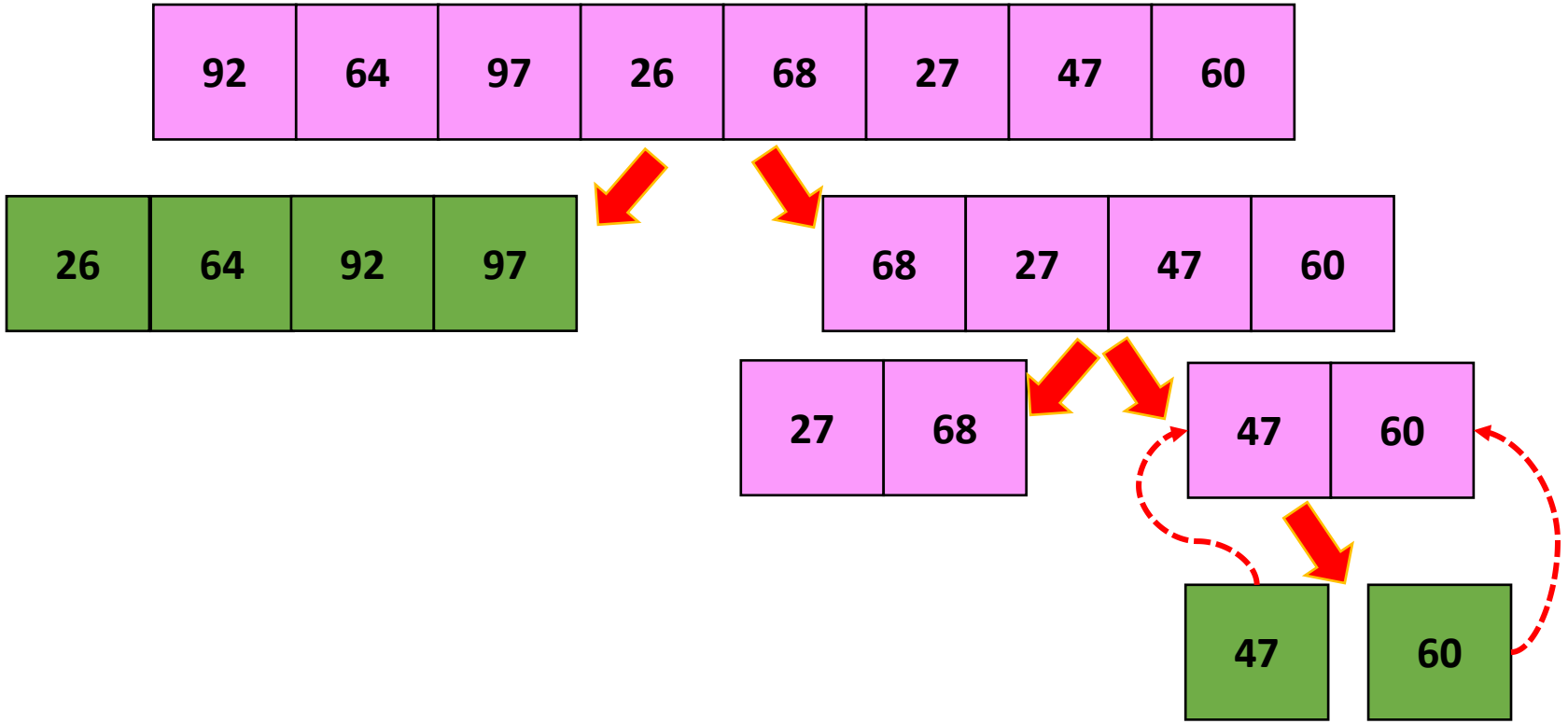
Exemplificando...



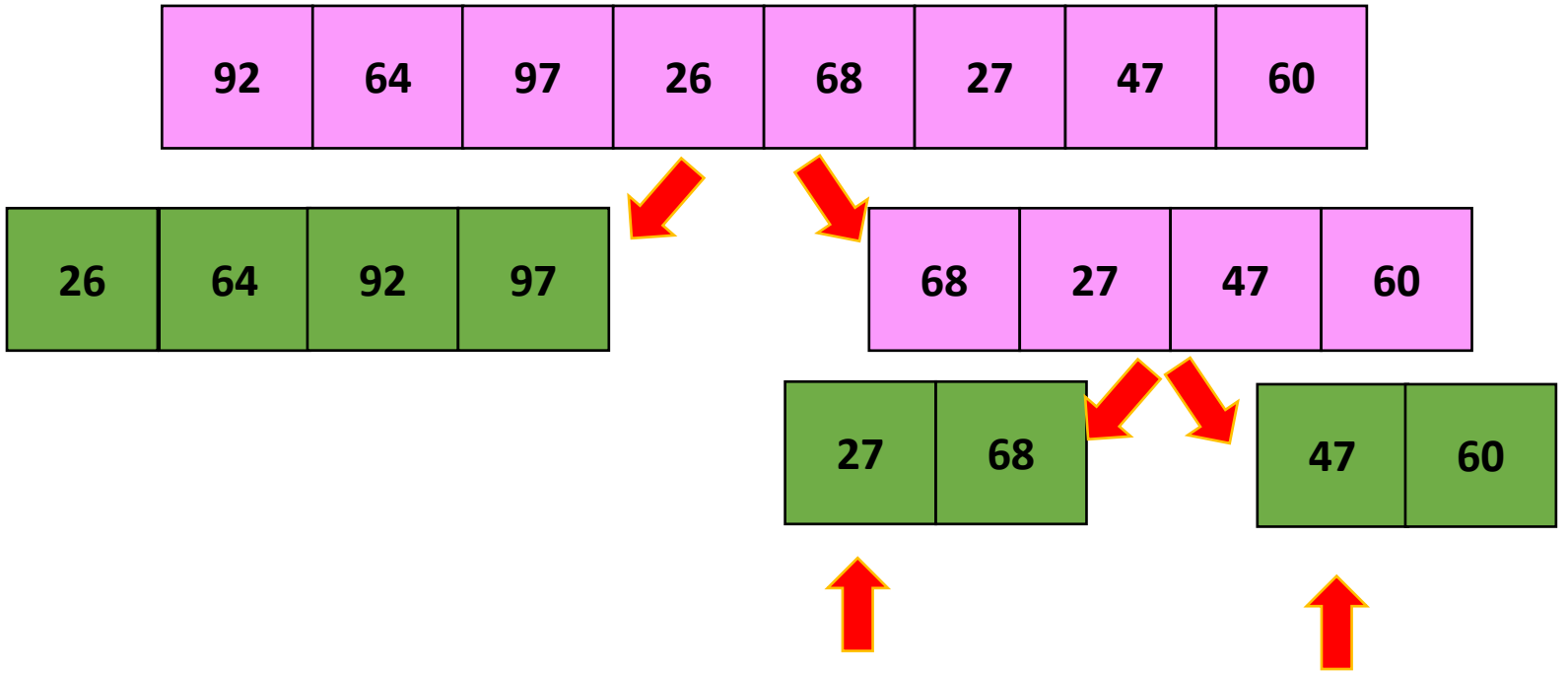
Exemplificando...



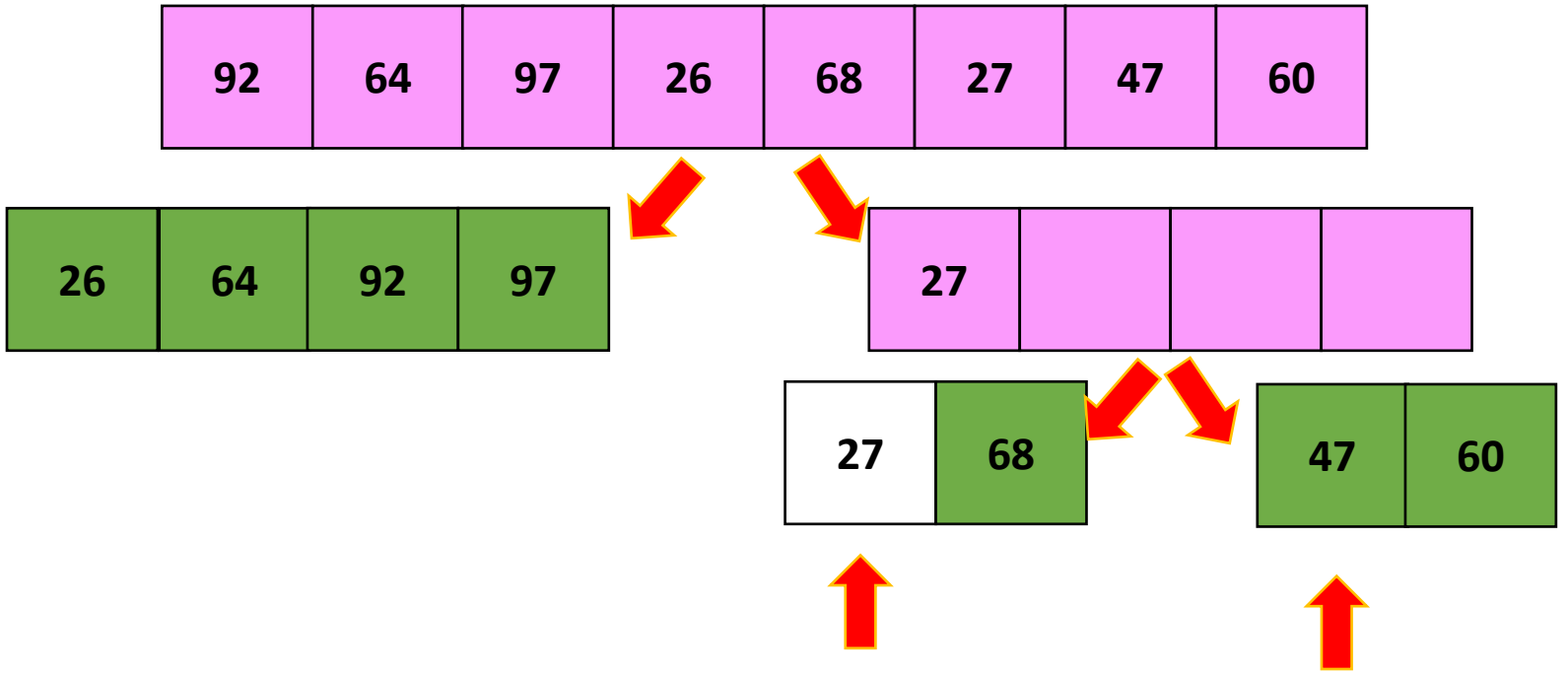
Exemplificando...



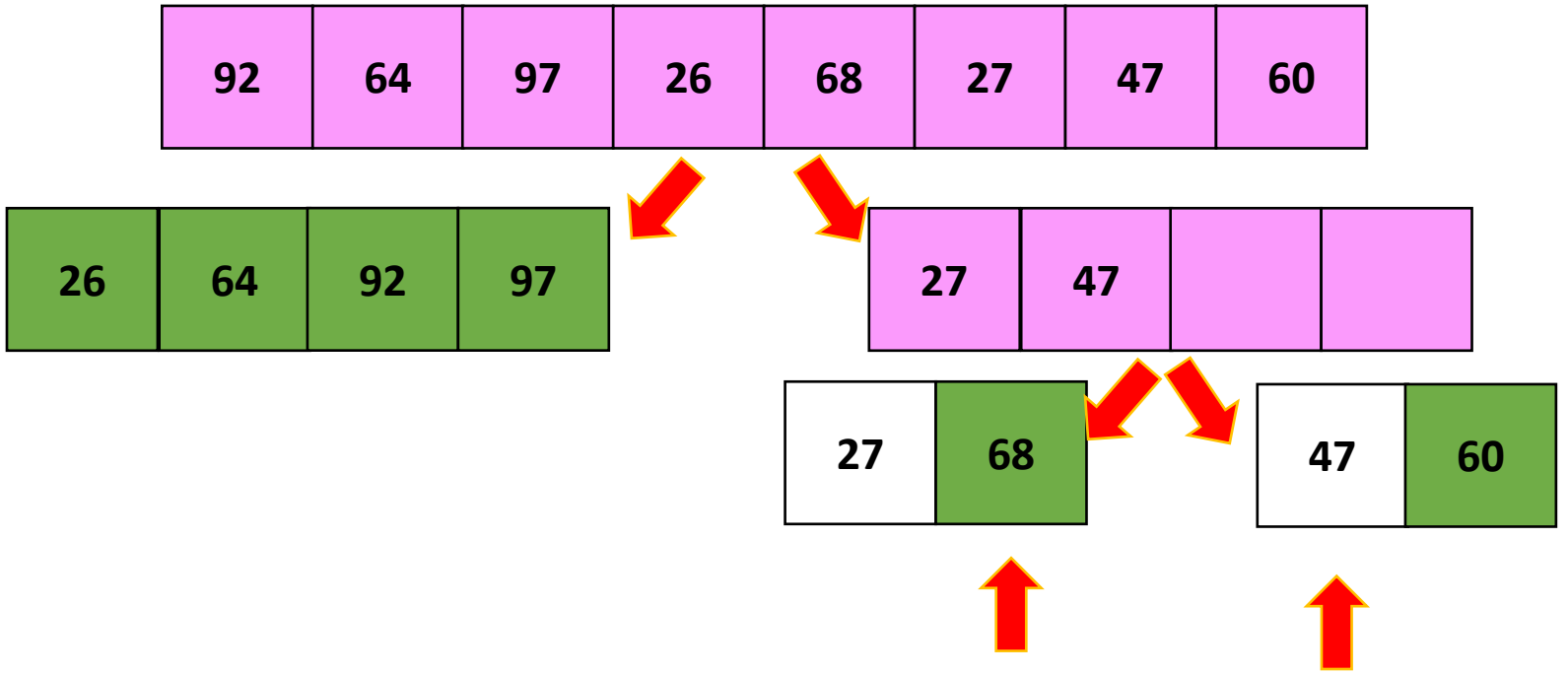
Exemplificando...



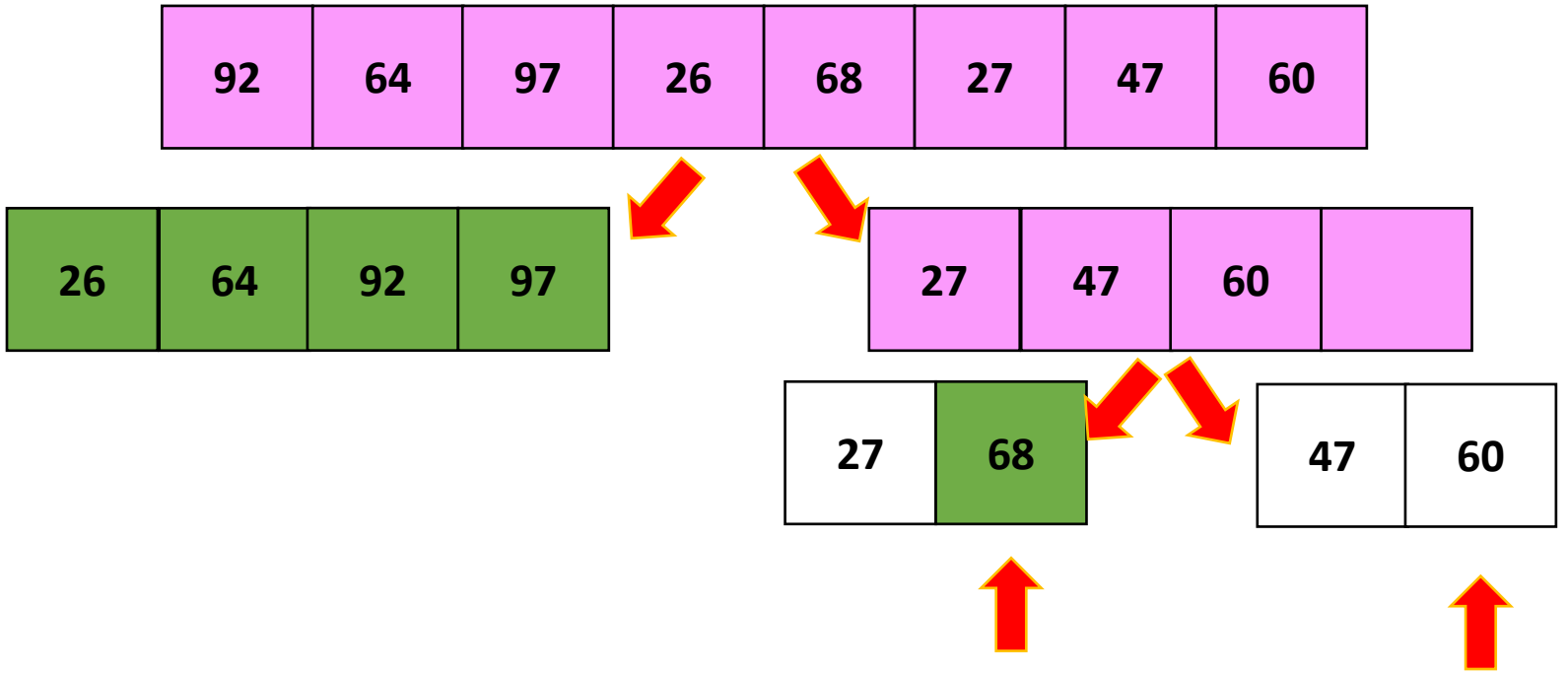
Exemplificando...



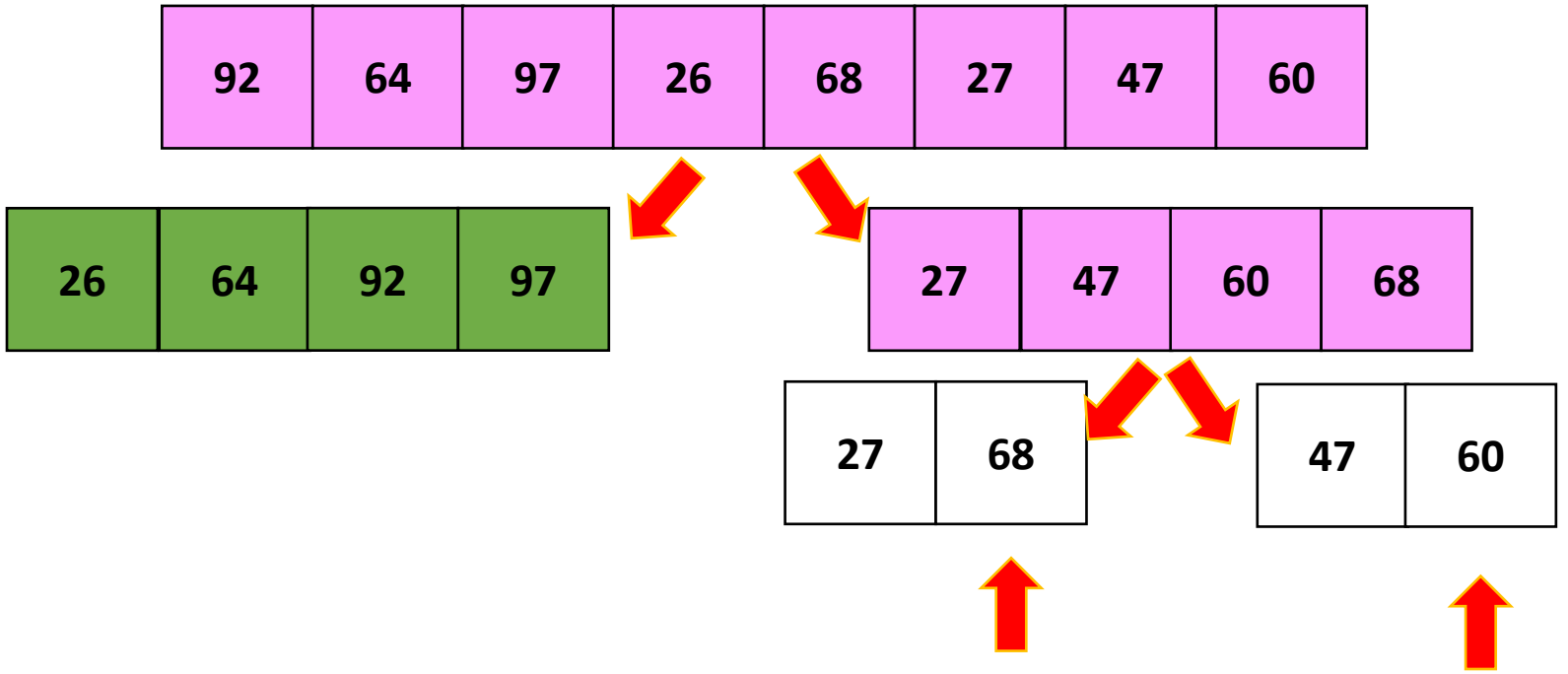
Exemplificando...



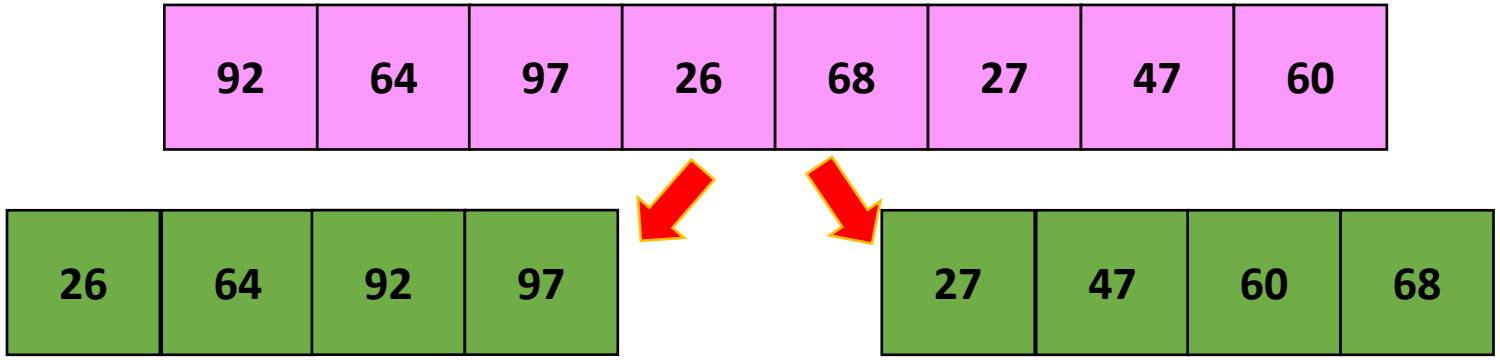
Exemplificando...



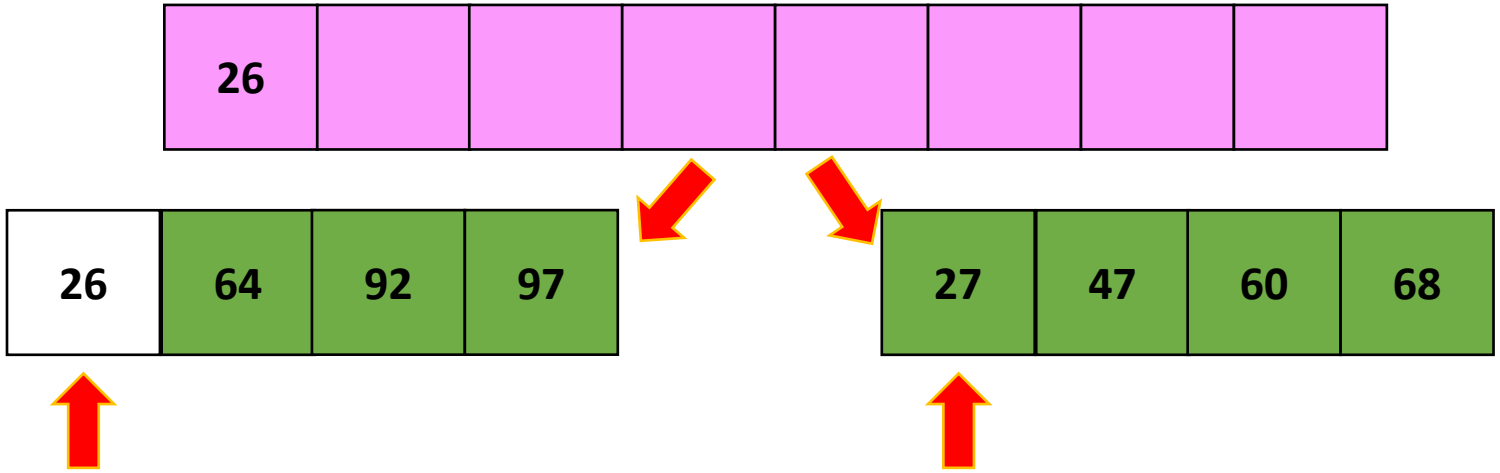
Exemplificando...



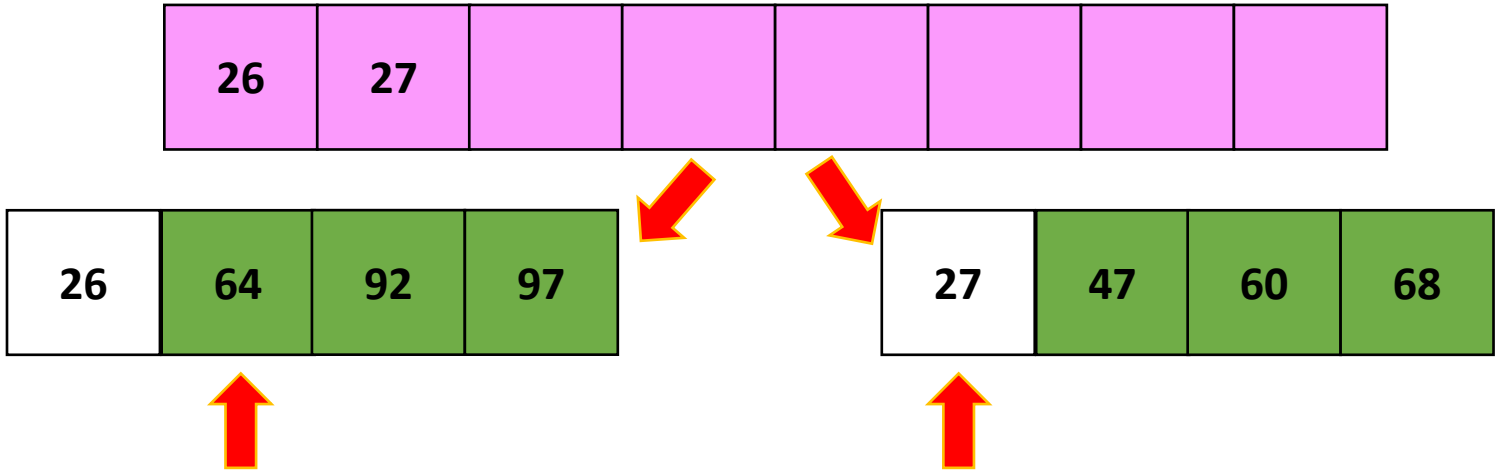
Exemplificando...



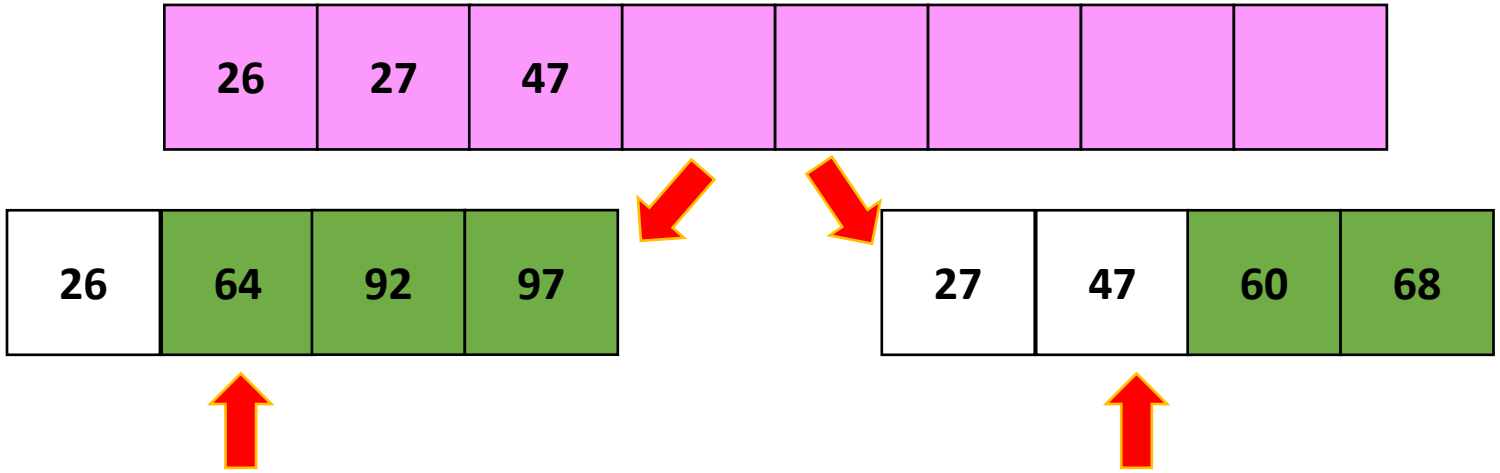
Exemplificando...



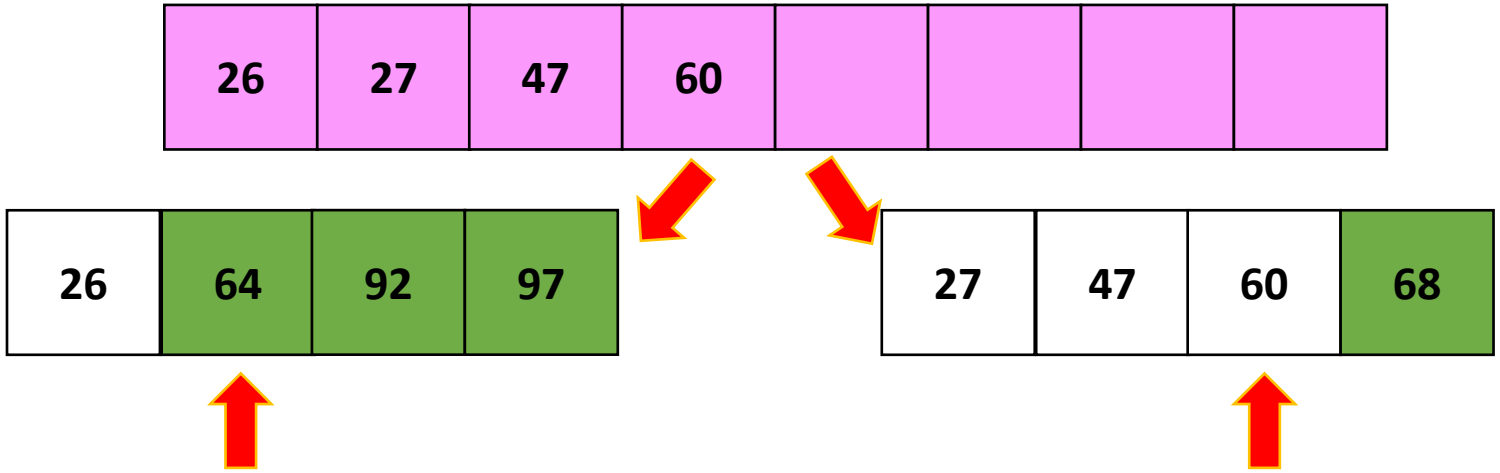
Exemplificando...



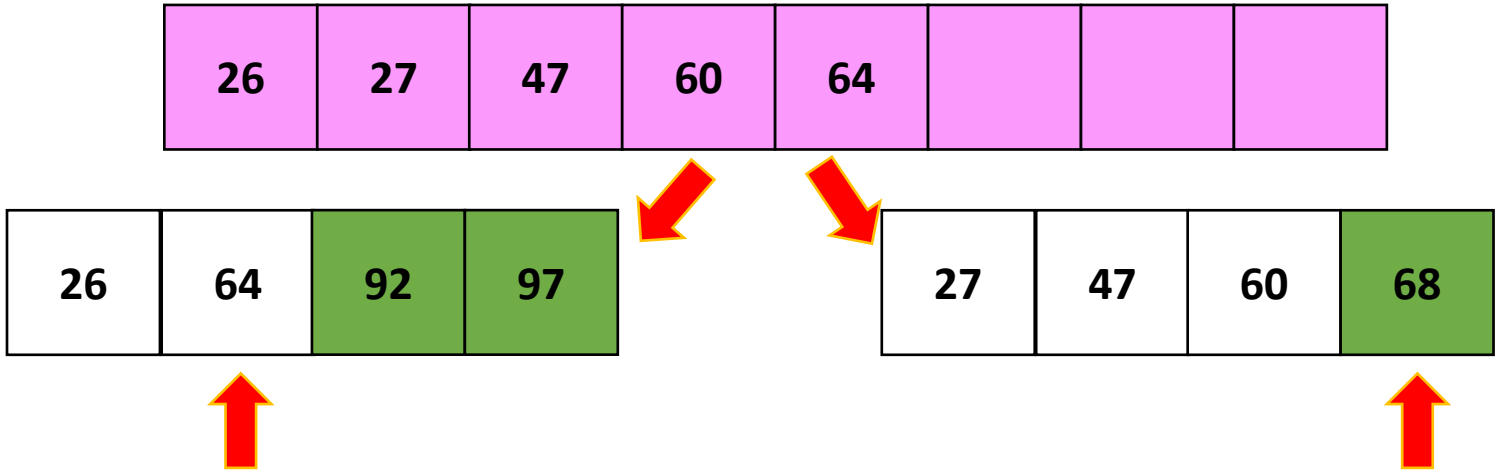
Exemplificando...



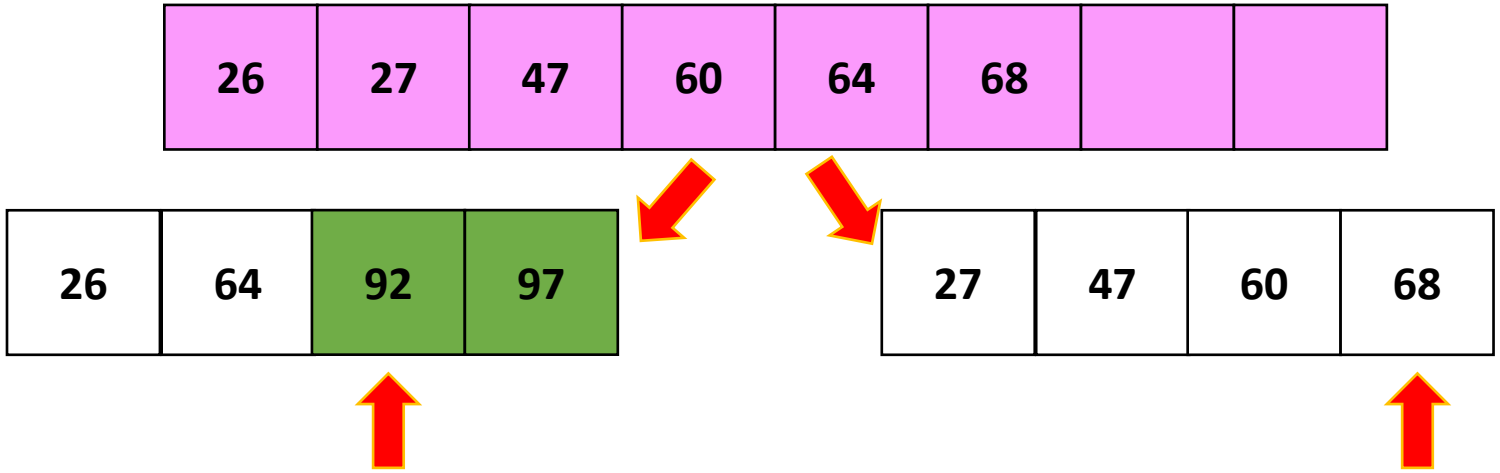
Exemplificando...



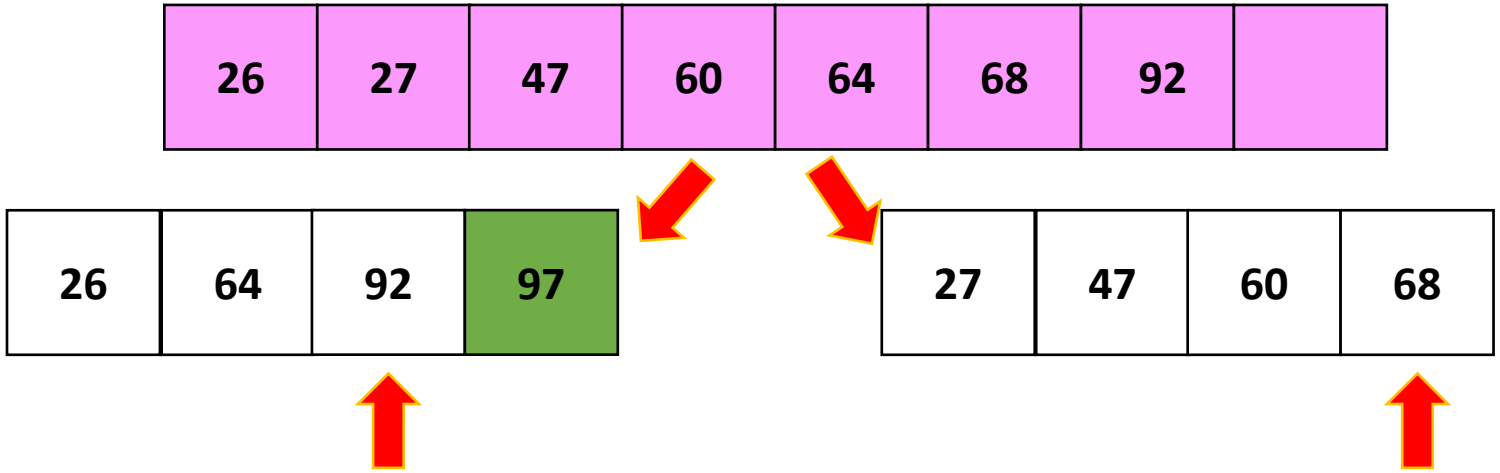
Exemplificando...



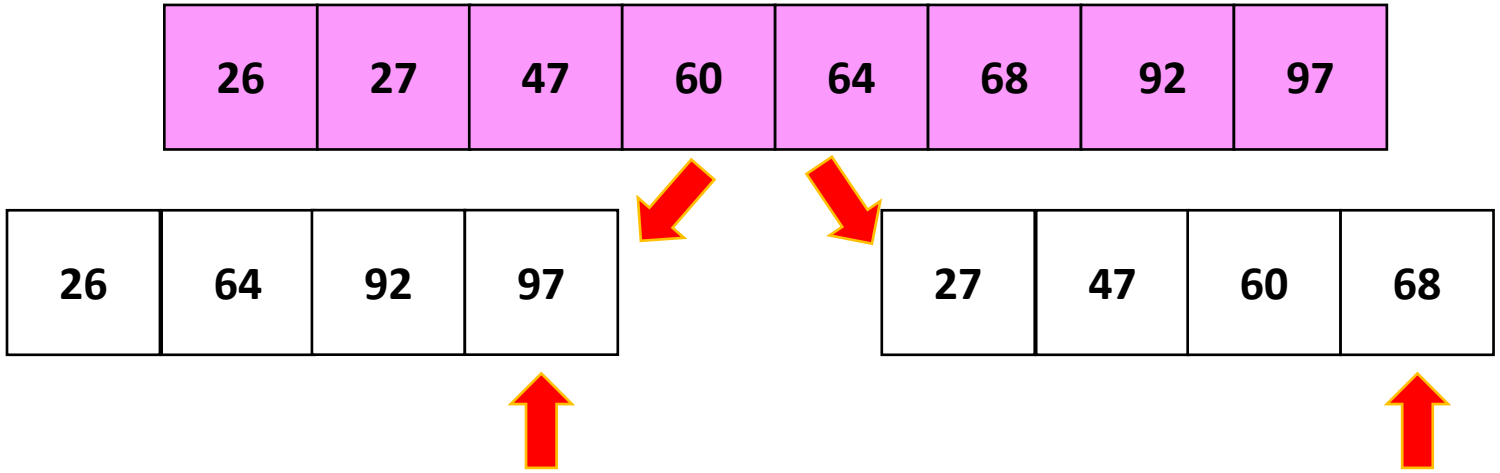
Exemplificando...



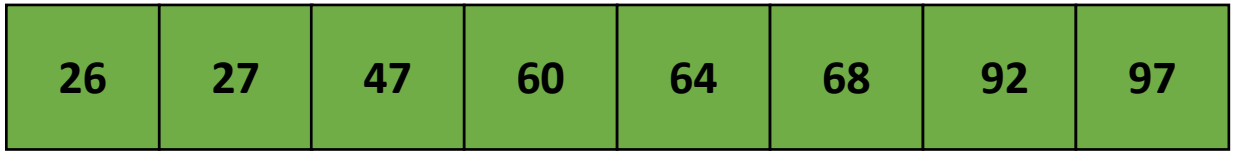
Exemplificando...



Exemplificando...



Exemplificando...



Algoritmo Mergesort (procedimento Mergesort)

```
mergesort(v, esq, dir)
1.   se esq < dir então
2.        centro ← ⌊(esq + dir)/2⌋;
3.        mergesort(v, esq, centro);
4.        mergesort(v, centro+1, dir);
5.        intercalar(esq, centro+1, dir);
```

Algoritmo Mergesort (procedimento Mergesort)

```
mergesort(v, esq, dir)
1.   se esq < dir então
2.        centro ← ⌊(esq + dir)/2⌋;
3.        mergesort(v, esq, centro);
4.        mergesort(v, centro+1, dir);
5.        intercalar(esq, centro+1, dir);
```

Chama a função para cada uma das metades

Algoritmo Mergesort (procedimento Mergesort)

```
mergesort(v, esq, dir)
1.   se esq < dir então
2.        centro ← ⌊(esq + dir)/2⌋;
3.        mergesort(v, esq, centro);
4.        mergesort(v, centro+1, dir);
5.        intercalar(v, esq, centro+1, dir);
```

combina as metades de forma ordenada

Procedimento Intercalar

```
intercalar(L, Tmp, inicio, centro, fim);
1. vAux[], p1, p2, tamanho, i, j , k
2. fim1 ← 0; fim2 ← 0
3. tamanho ← fim - inicio + 1
4. p1 ← inicio
5. p2 ←meio +1;
6.            para i ← 0 até tamanho faça
7.                                  se (!fim1 e !fim2)
8.                                      se(L[p1] < L[p2])
9.                                          vAux[i] ←  L[p1++]
10.                                       senão
11.                                         vAux[i] ← vAux[p2++]
12.                                     se(p1 > centro) então fim1 ← 1
13.                                     se(p2 > centro) então fim2 ← 1
14.                                 senão
15.                                      se (!fim1)
16.                                          vAux[i] = L[p1++]
17.                                      else
18.                                          vAux[i] = L[p2++]
19.           para j ← 0 e k ← inicio até tamanho faça
20.                 L[k] ← vAux[j]
```

combina ordenando

Procedimento Intercalar

```
intercalar(L, Tmp, inicio, centro, fim);
1. vAux[], p1, p2, tamanho, i, j , k
2. fim1 ← 0; fim2 ← 0
3. tamanho ← fim - inicio + 1
4. p1 ← inicio
5. p2 ←meio +1;
6.            para i ← 0 até tamanho faça
7.                                  se (!fim1 e !fim2)
8.                                        se(L[p1] < L[p2])
9.                                            vAux[i] ←  L[p1++]
10.                                        senão
11.                                            vAux[i] ← vAux[p2++]
12.                                       se(p1 > centro) então fim1 ← 1
13.                                       se(p2 > centro) então fim2 ← 1
14.                                 senão
15.                                      se (!fim1)
16.                                            vAux[i] = L[p1++]
17.                                       else
18.                                            vAux[i] = L[p2++]
19.            para j ← 0 e k ← inicio até tamanho faça
20.                 L[k] ← vAux[j]
```

verifica se o vetor acabou

Procedimento Intercalar

```
intercalar(L, Tmp, inicio, centro, fim);
1. vAux[], p1, p2, tamanho, i, j , k
2. fim1 ← 0; fim2 ← 0
3. tamanho ← fim - inicio + 1
4. p1 ← inicio
5. p2 ←meio +1;
6.            para i ← 0 até tamanho faça
7.                                  se (!fim1 e !fim2)
8.                                        se(L[p1] < L[p2])
9.                                            vAux[i] ←  L[p1++]
10.                                        senão
11.                                            vAux[i] ← vAux[p2++]
12.                                       se(p1 > centro) então fim1 ← 1
13.                                       se(p2 > centro) então fim2 ← 1
14.                                 senão
15.                                    se (!fim1)
16.                                           vAux[i] = L[p1++]
17.                                    else
18.                                           vAux[i] = L[p2++]
19.            para j ← 0 e k ← inicio até tamanho faça
20.                 L[k] ← vAux[j]
```

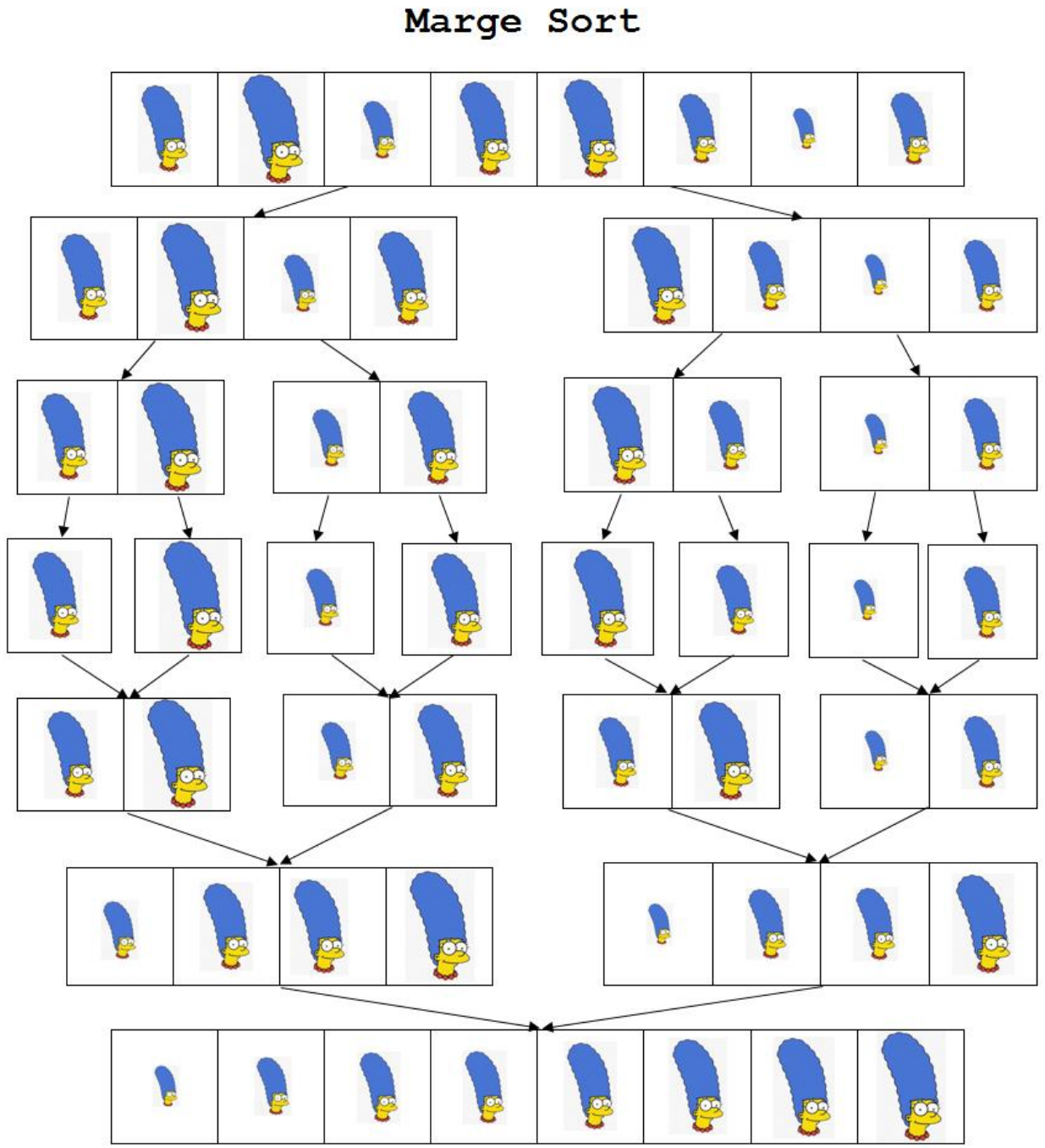
Copia o que sobrar (ordenando)

Procedimento Intercalar

```
intercalar(L, Tmp, inicio, centro, fim);
1. vAux[], p1, p2, tamanho, i, j , k
2. fim1 ← 0; fim2 ← 0
3. tamanho ← fim - inicio + 1
4. p1 ← inicio
5. p2 ←meio +1;
6.            para i ← 0 até tamanho faça
7.                                  se (!fim1 e !fim2)
8.                                        se(L[p1] < L[p2])
9.                                            vAux[i] ←  L[p1++]
10.                                        senão
11.                                            vAux[i] ← vAux[p2++]
12.                                        se(p1 > centro) então fim1 ← 1
13.                                        se(p2 > centro) então fim2 ← 1
14.                                 senão
15.                                       se (!fim1)
16.                                            vAux[i] = L[p1++]
17.                                        else
18.                                            vAux[i] = L[p2++]
19.           para j ← 0 e k ← inicio até tamanho faça
20.                 L[k] ← vAux[j]
```

copiar do vetor auxiliar para o vetor original

Exemplo



Material complementar

Utilize o link abaixo para visualizar o algoritmo Mergesort:

https://www.hackerearth.com/pt-br/practice/algorithms/sorting/merge-sort/visualize/

Livros:

Estrutuda de Dados e seus Algoritmos - Jayme L. Szwarcfiter e Lilian Markenzon

Linguagem C: Completa e Descomplicada - André Backes