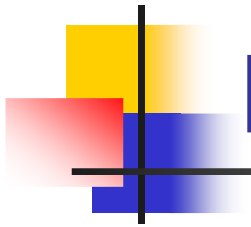
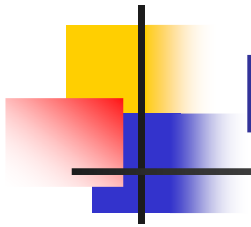


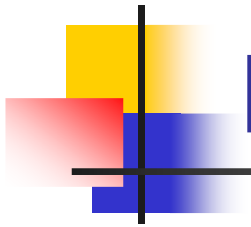
# Divisão e Conquista

- Método usado para projeto de algoritmos:
  - **Divisão**: Se o tamanho da entrada é muito grande para aplicação de uma solução simples, dividir o problema em dois ou mais subproblemas disjuntos
  - **Conquista**: Usar o método recursivamente para resolver os subproblemas
  - **Combinação**: Obter as soluções dos subproblemas e combiná-las para compor uma solução para o problema original



# MergeSort

- **Divisão**: Se S tem $n \geq 2$ elementos, dividir seus elementos em duas subsequências $S_1$ e $S_2$ com respectivamente $\lceil n/2 \rceil$ e $\lfloor n/2 \rfloor$ *elementos*
- **Conquista**: Ordernar as subsequências $S_1$ e $S_2$ usando *MergeSort*
- **Combinação**: Intercale os elementos de $S_1$ e $S_2$ de forma a obter uma seqüência ordenada



# Merge Sort: Algoritmo

- Analogia com o jogo de cartas
  - Temos duas pilhas ordenadas com as cartas de menor valor em cima
  - Desejamos juntar as duas pilhas (fazendo a intercalação) em uma única pilha de saída ordenada
    1) Escolher a menor das duas cartas nas pilhas
    2) Removê-la de sua pilha
    3) Colocá-la sobre a pilha de saída
    4) Repetir os passos 1,2,3 até uma das duas pilhas de entrada esvaziarem

# Merge Sort: Algoritmo

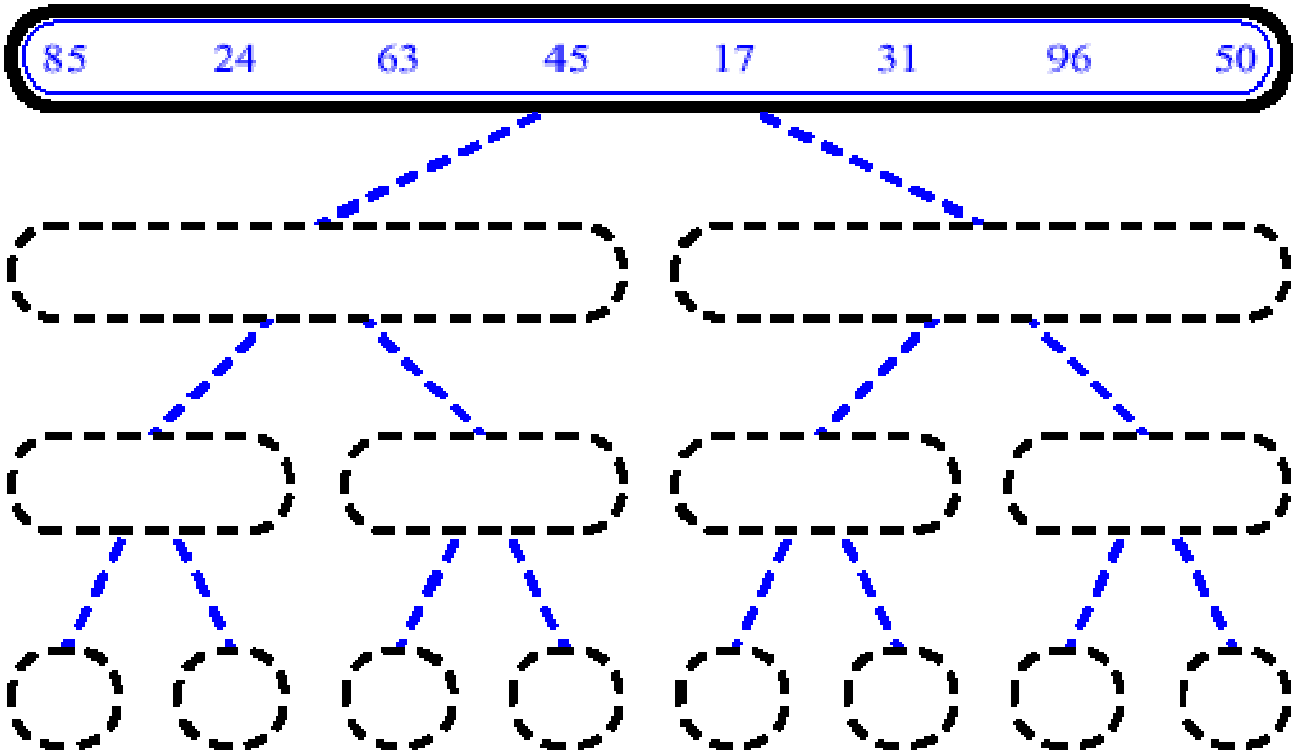
```
Merge-Sort(A, p, r)
    if p < r then
        q ←⌊(p+r)/2⌋
        Merge-Sort(A, p, q)
        Merge-Sort(A, q+1, r)
        Merge(A, p, q, r)
```

A é o arranjo
p, q e r são índices de numeração dos elementos do arranjo $p \leq q < r$

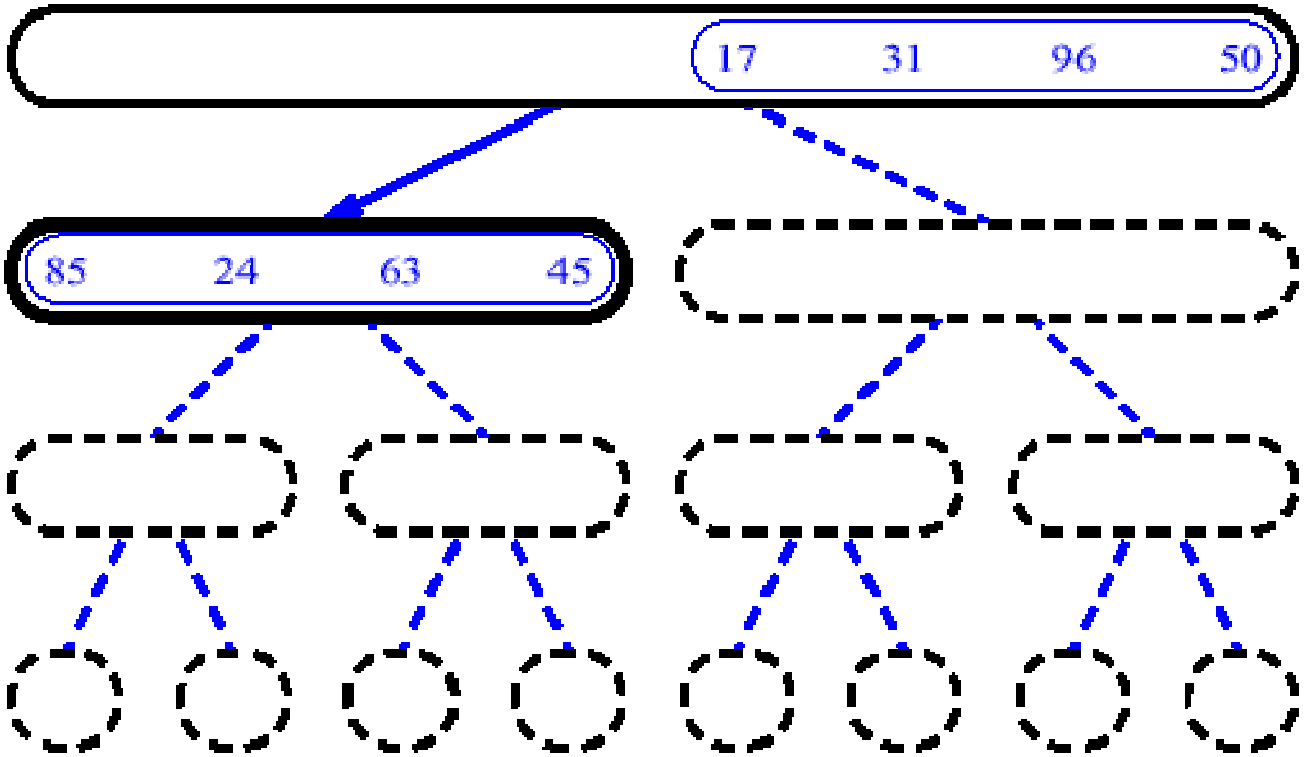
Se $p \geq r$, o subarranjo tem no máximo um elemento e, portanto, já está ordenado

```
Merge(A, p, q, r)
   Retire o menor entre o menor dos elementos de
A[p…q] e A[q+1…r] e acrescente ao resultado. Repita
até que as duas sub-sequências estejam vazias
```

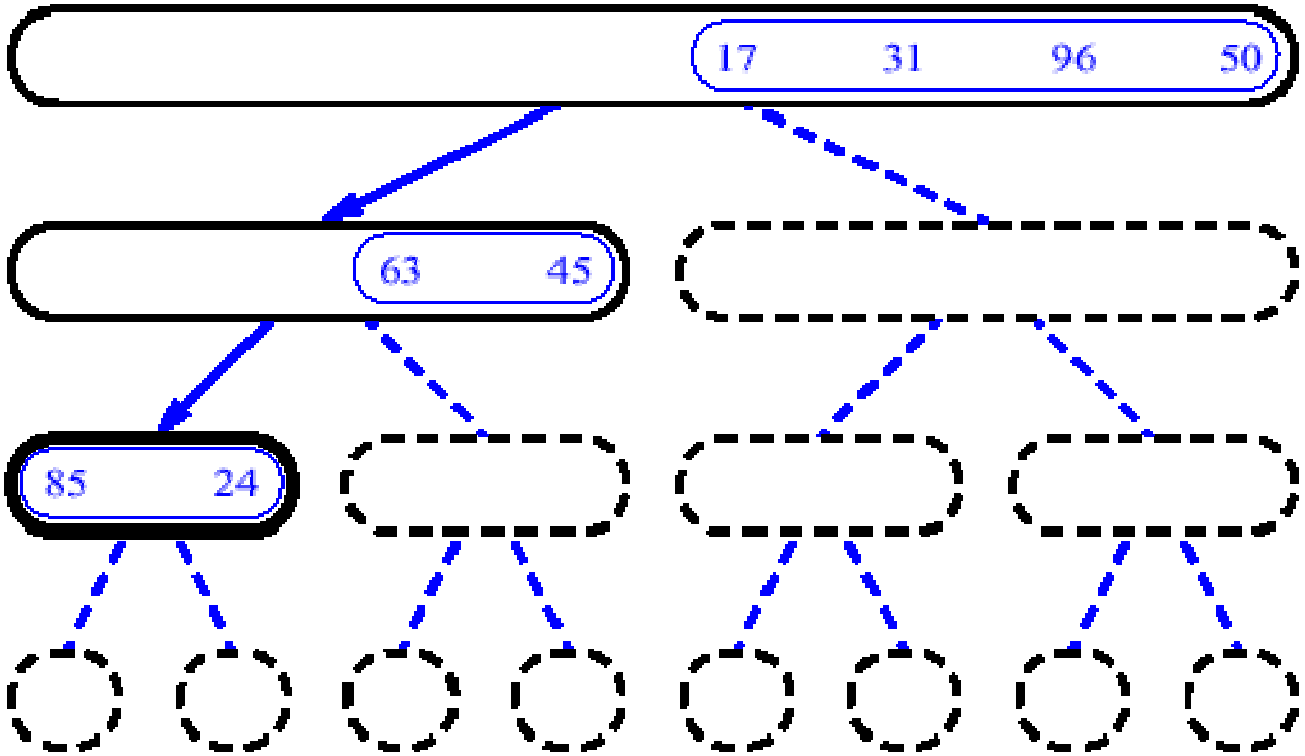
# MergeSort (Exemplo) - 1



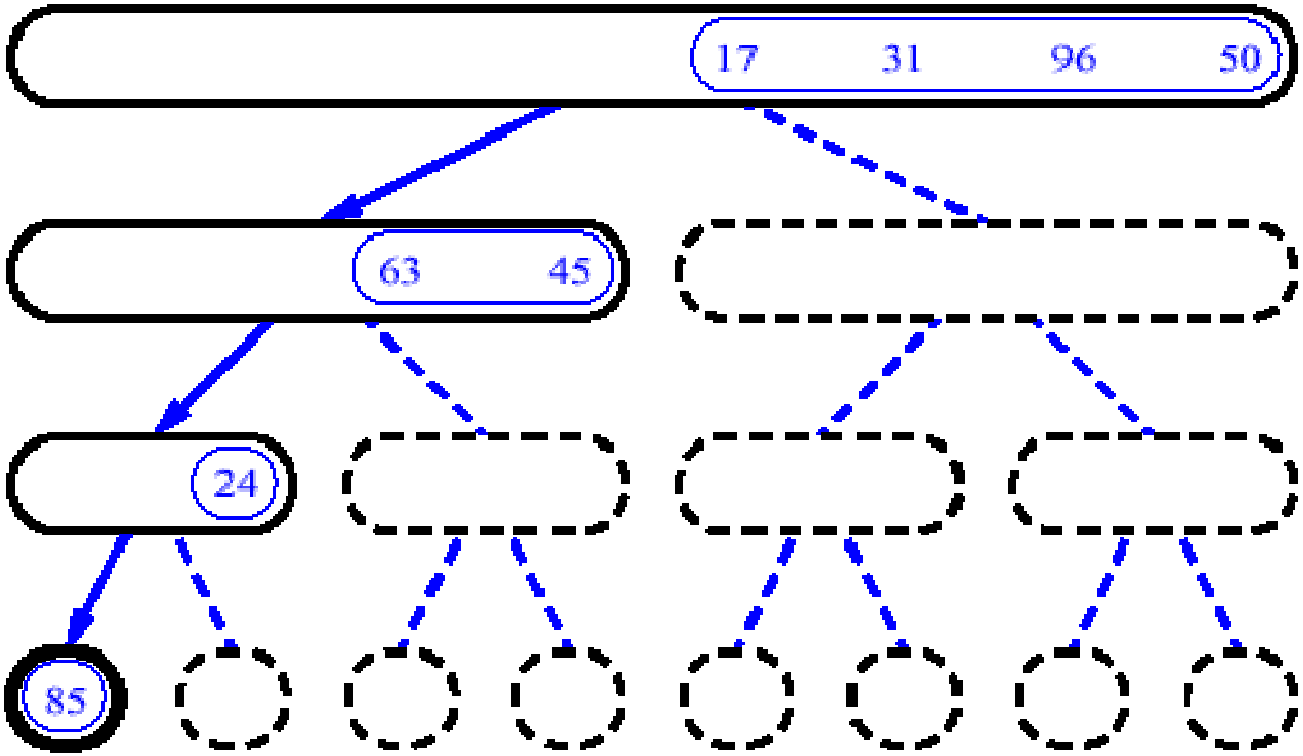
# MergeSort (Exemplo) - 2



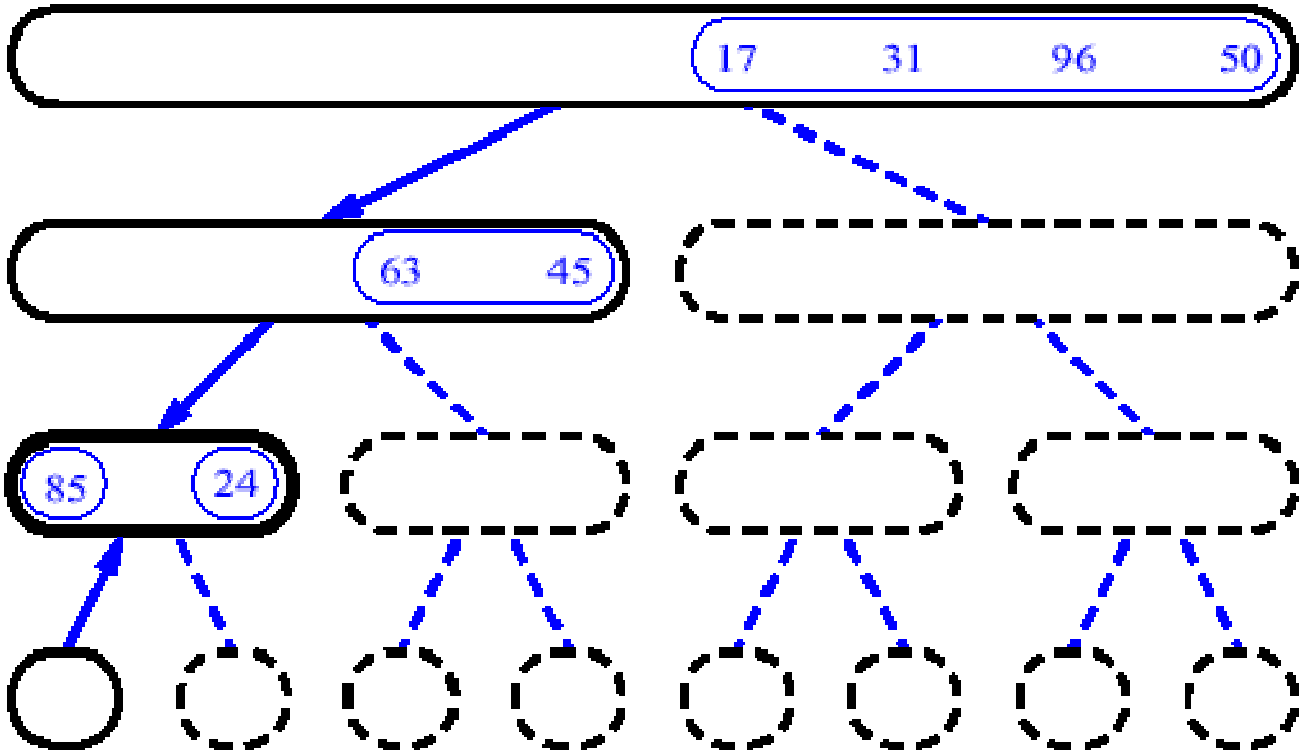
# MergeSort (Exemplo) - 3



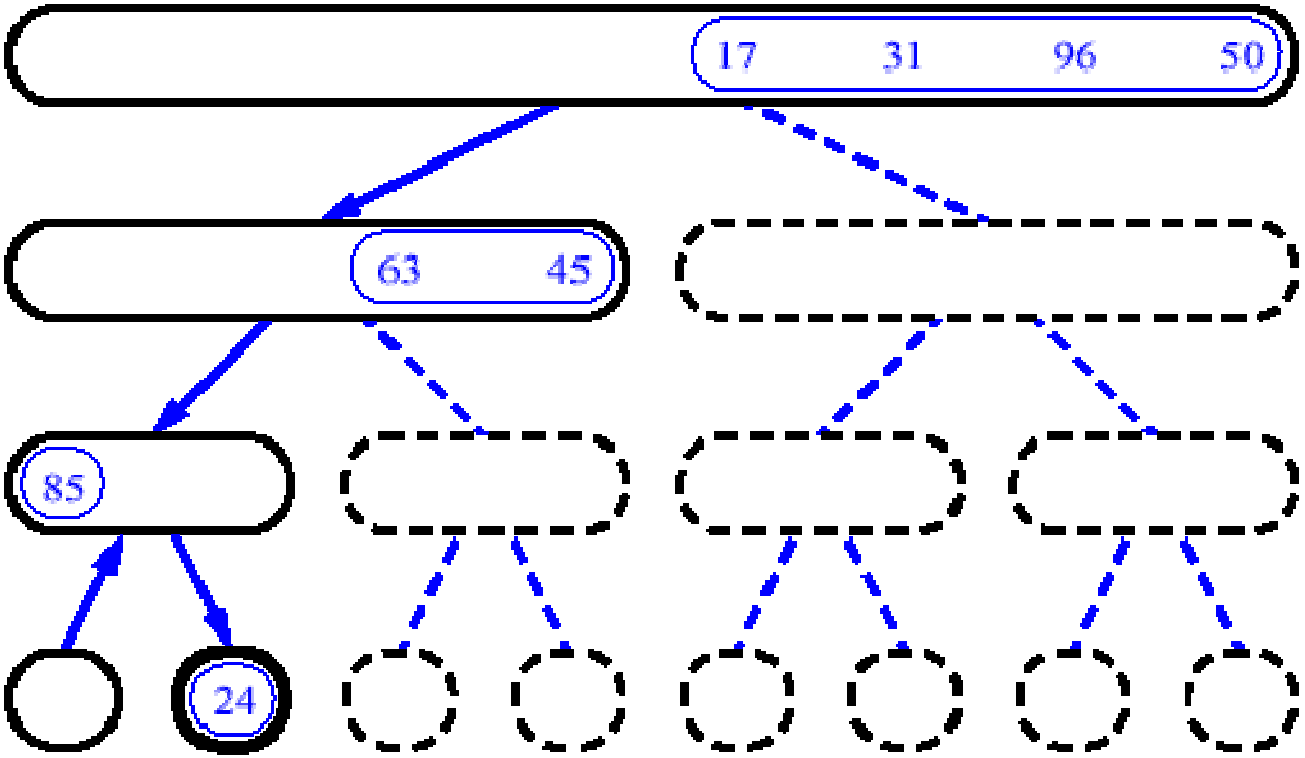
# MergeSort (Exemplo) - 4



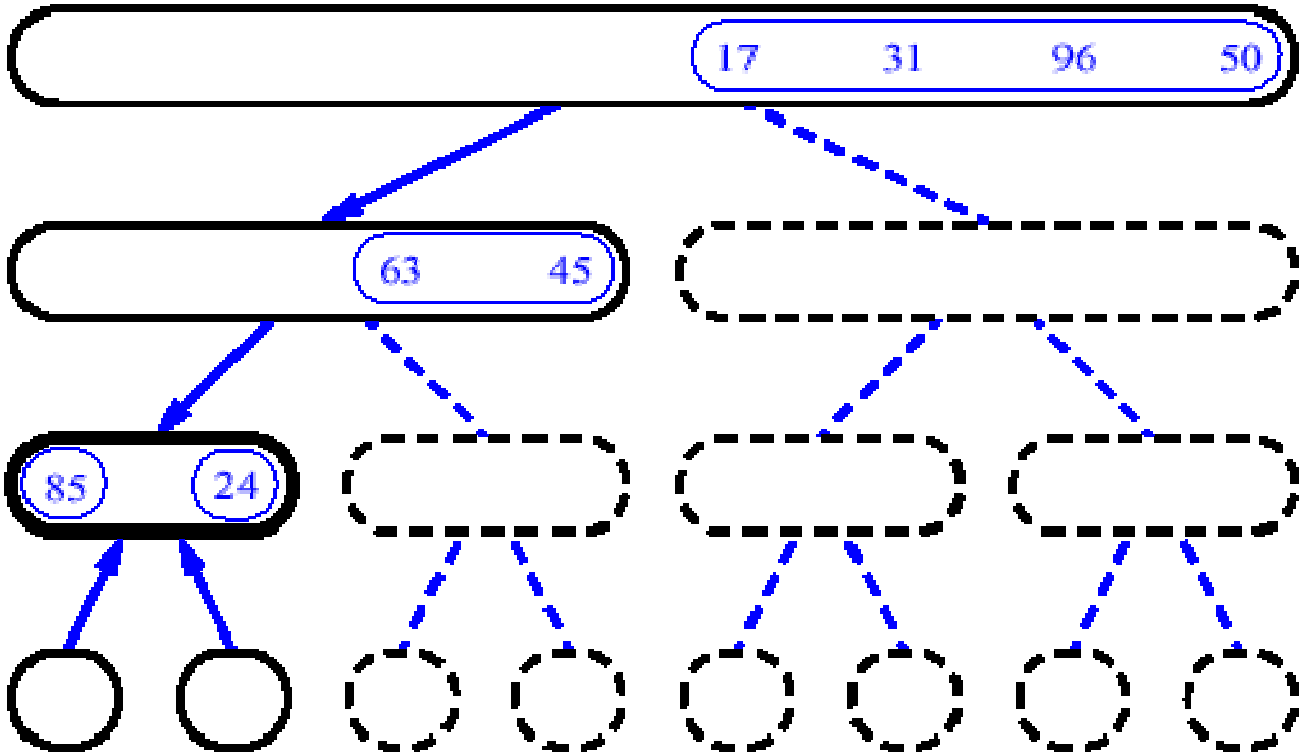
# MergeSort (Exemplo) - 5



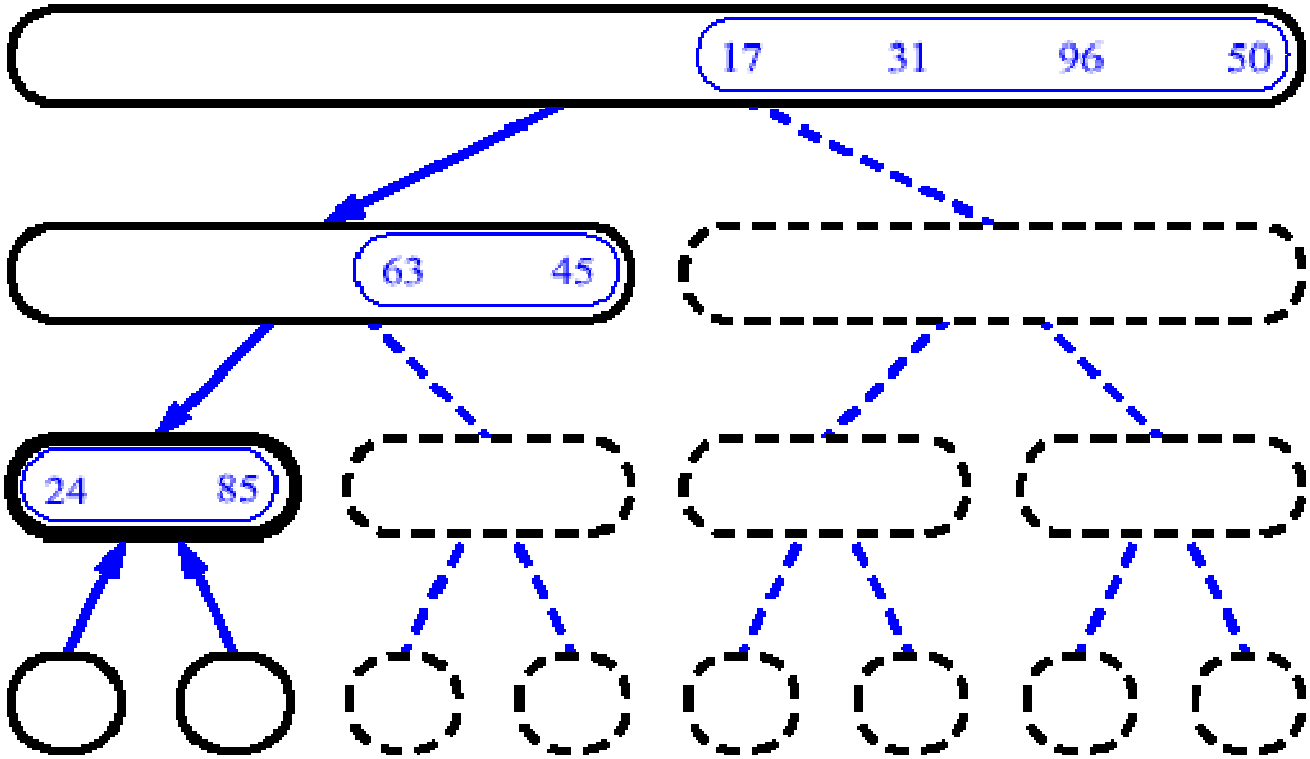
# MergeSort (Exemplo) - 6



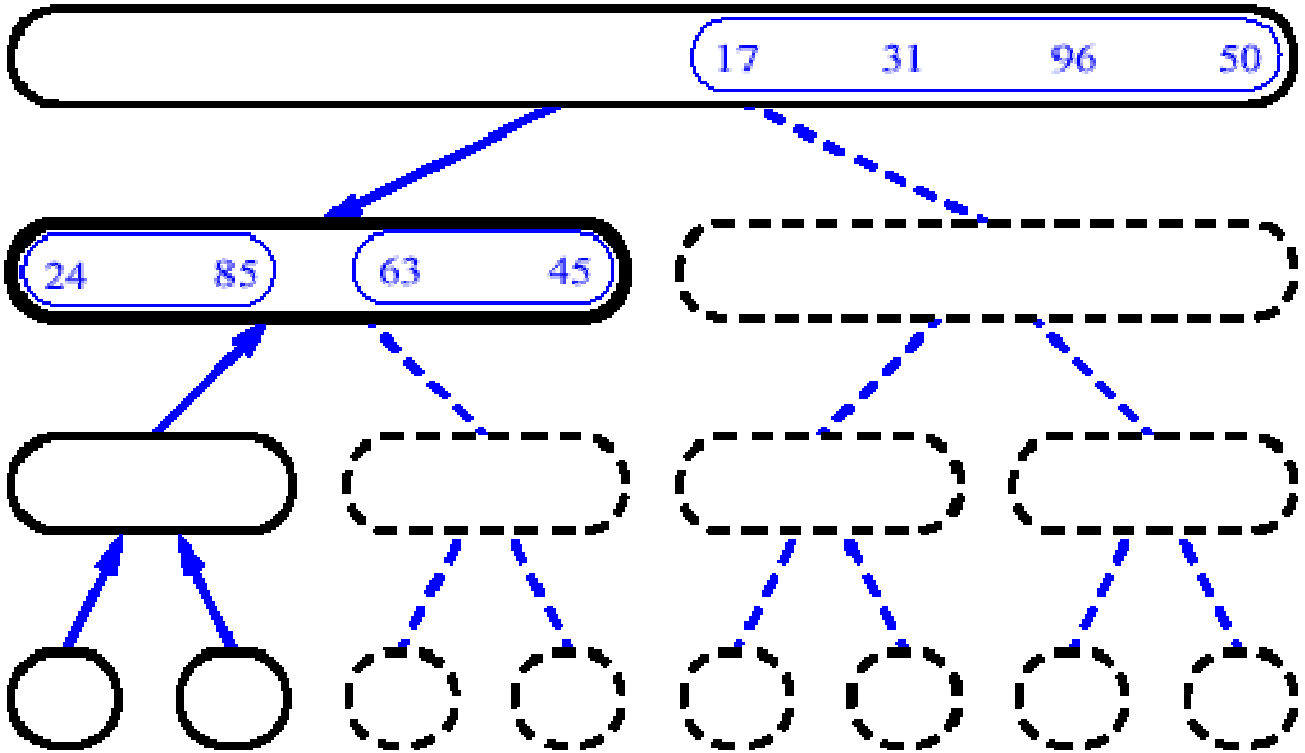
# MergeSort (Exemplo) - 7



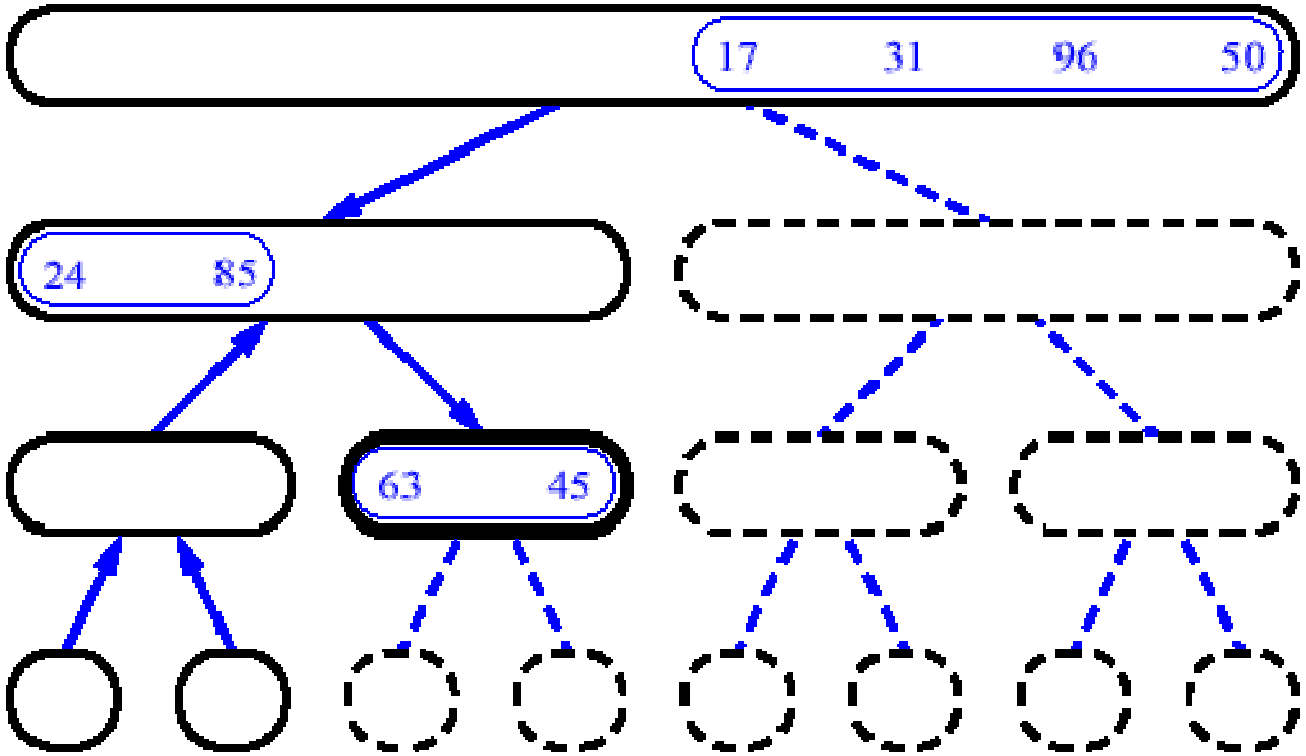
# MergeSort (Exemplo) - 8



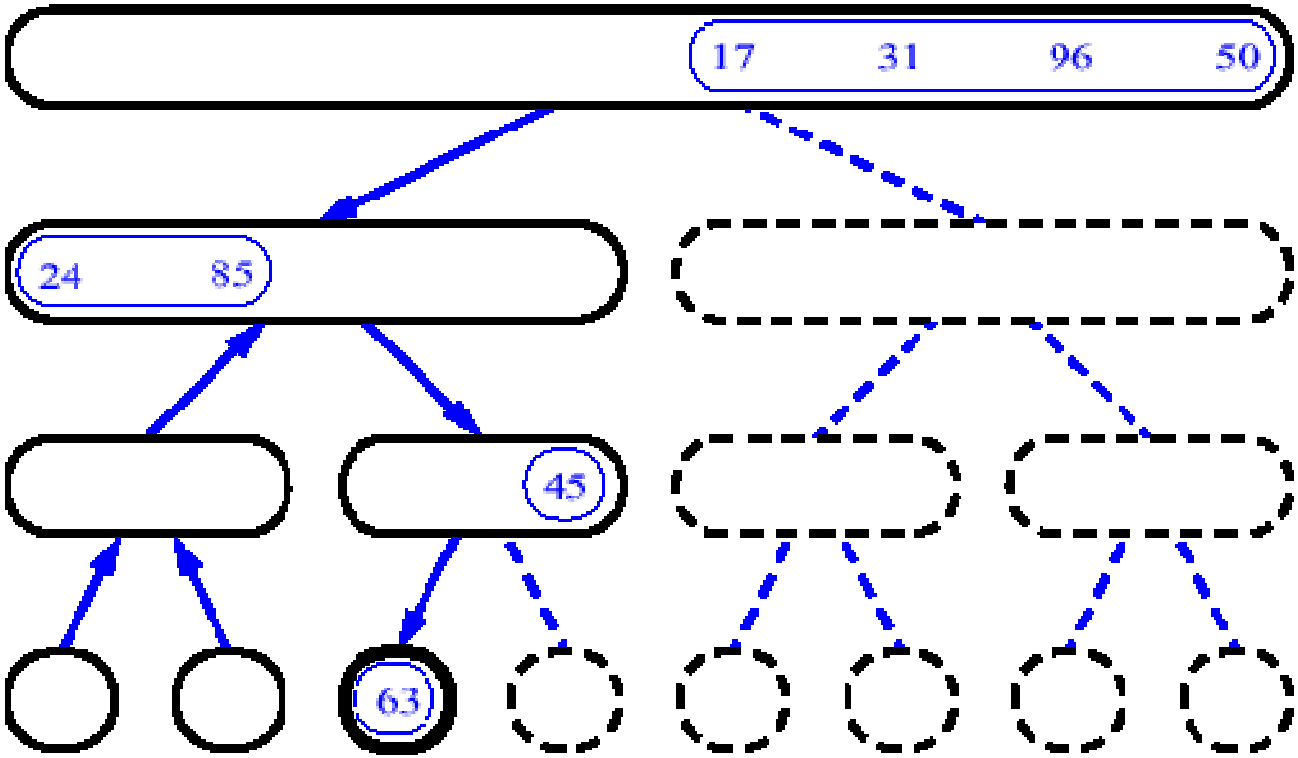
# MergeSort (Exemplo) - 9



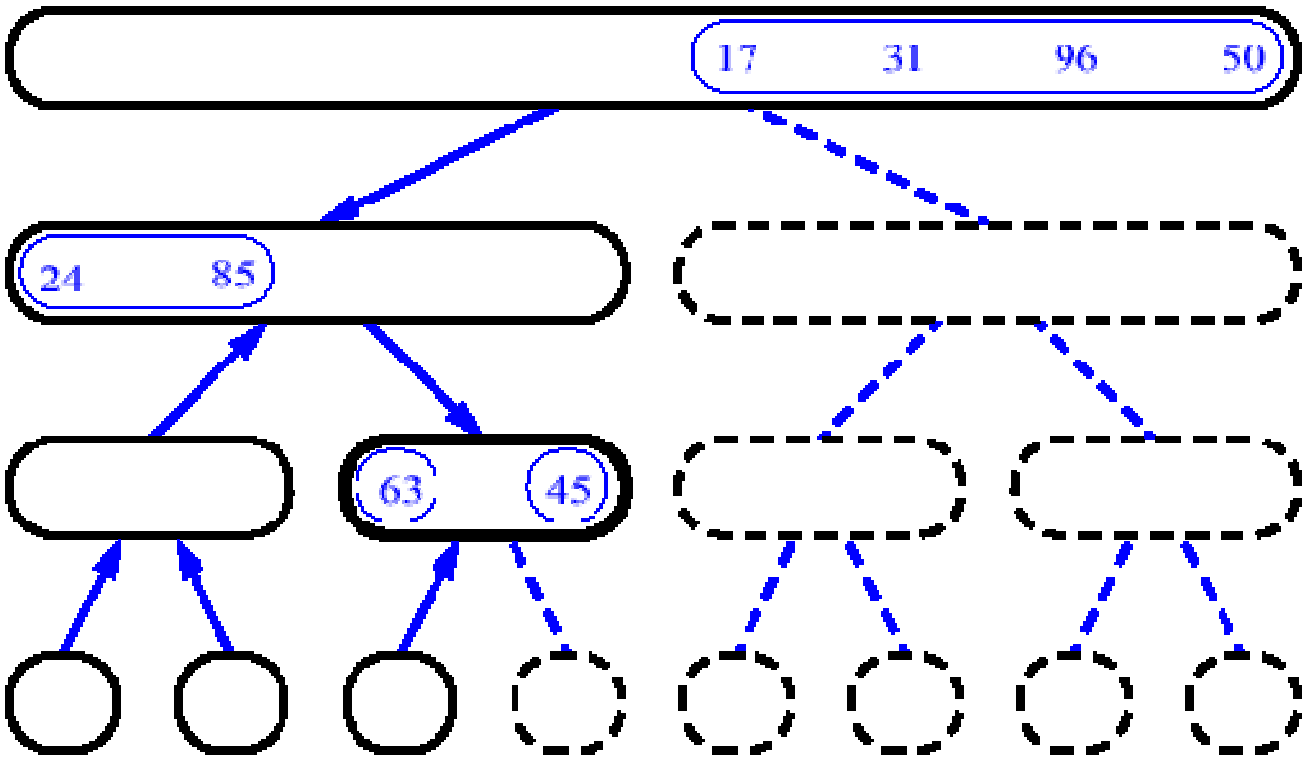
# MergeSort (Exemplo) - 10



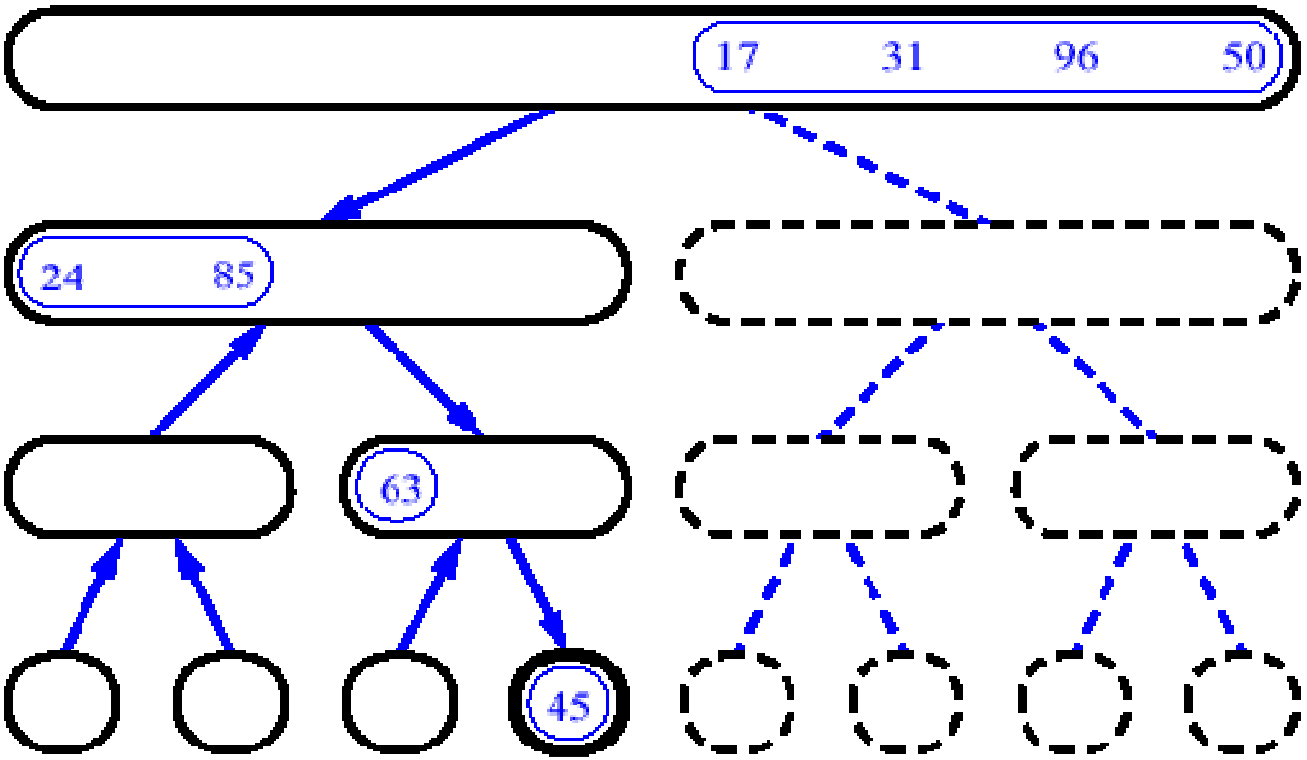
# MergeSort (Exemplo) - 11



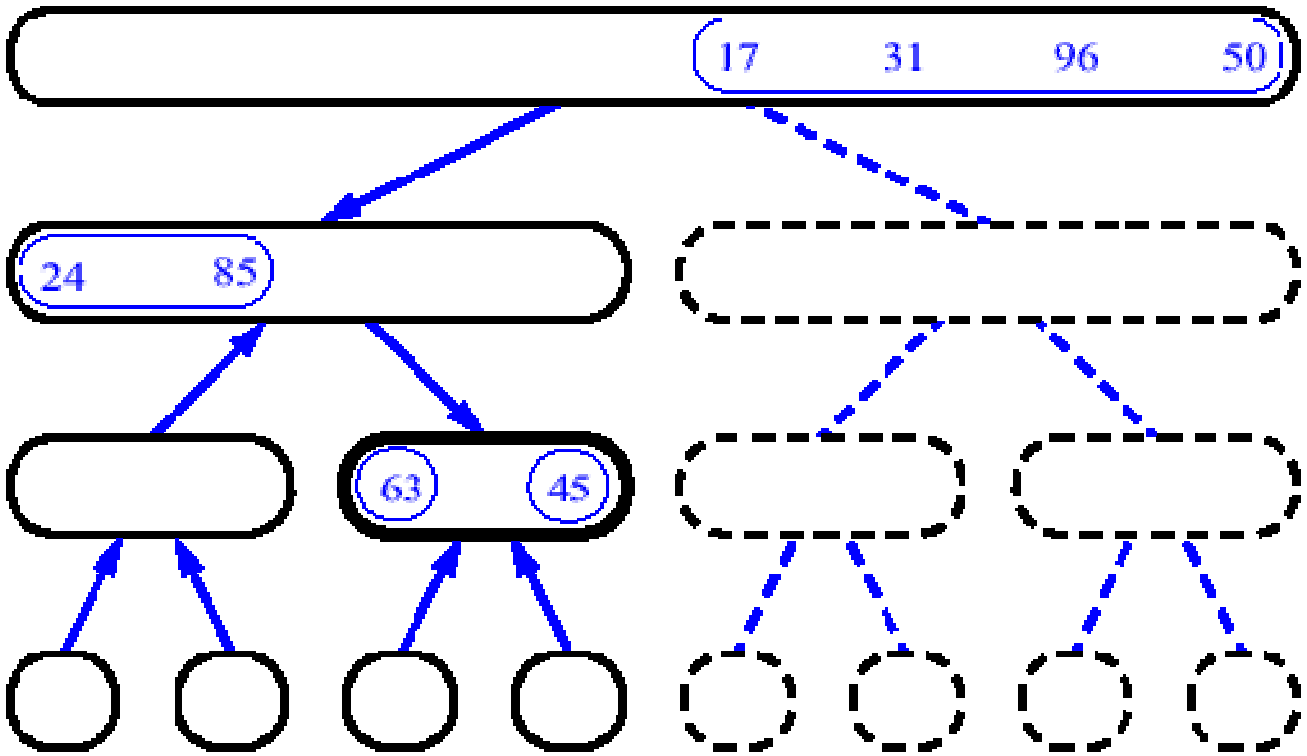
# MergeSort (Exemplo) - 12



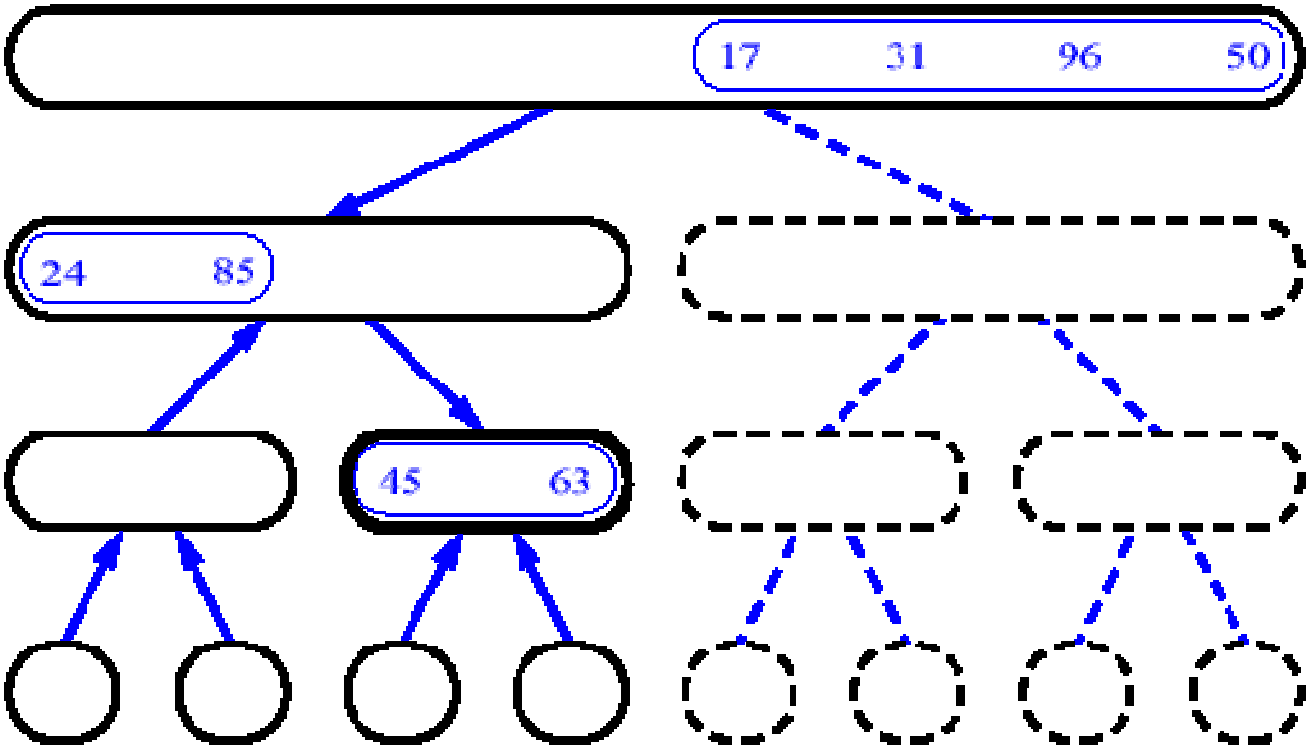
# MergeSort (Exemplo) - 13



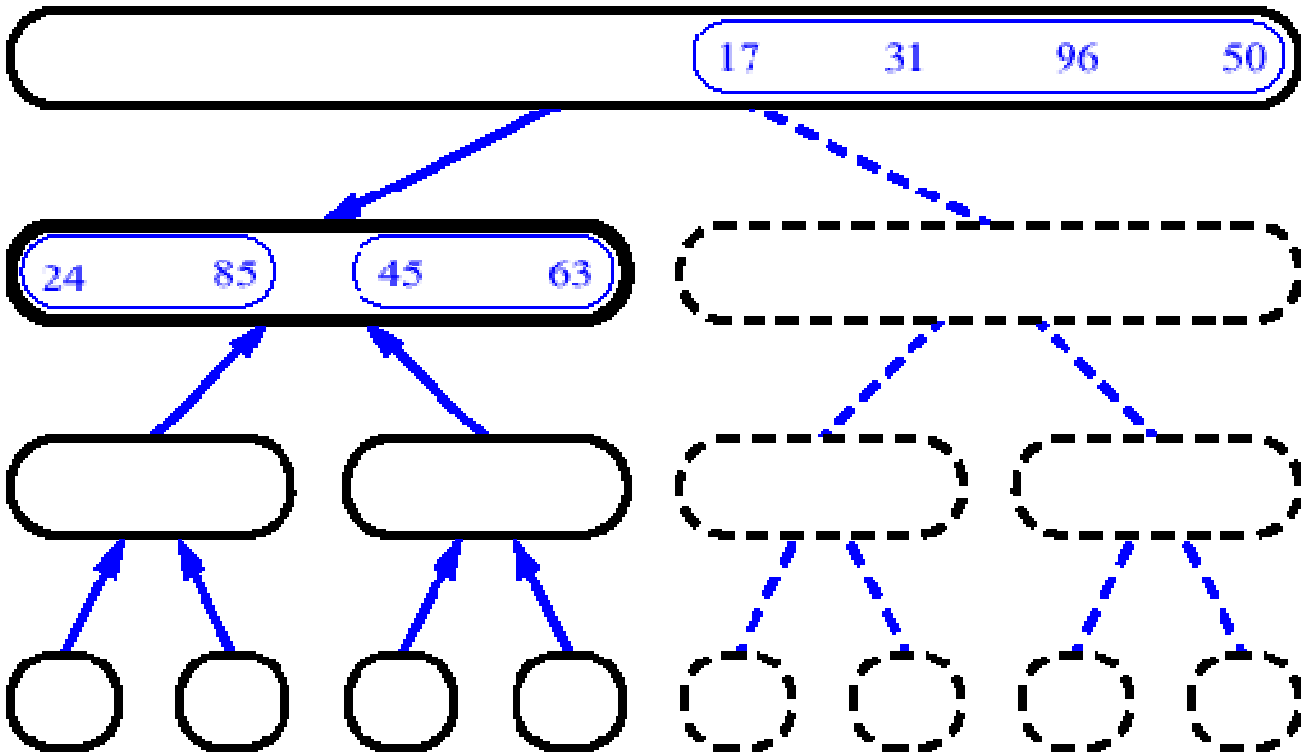
# MergeSort (Exemplo) - 14



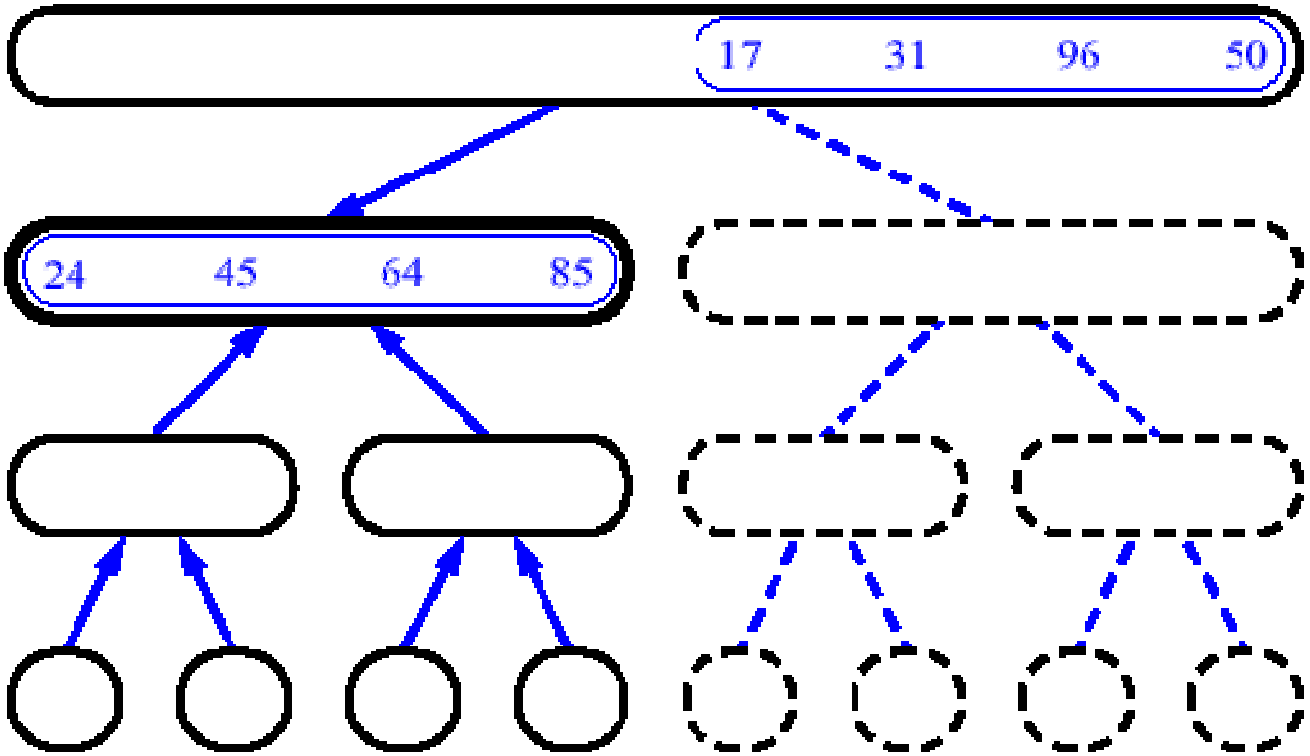
# MergeSort (Exemplo) - 15



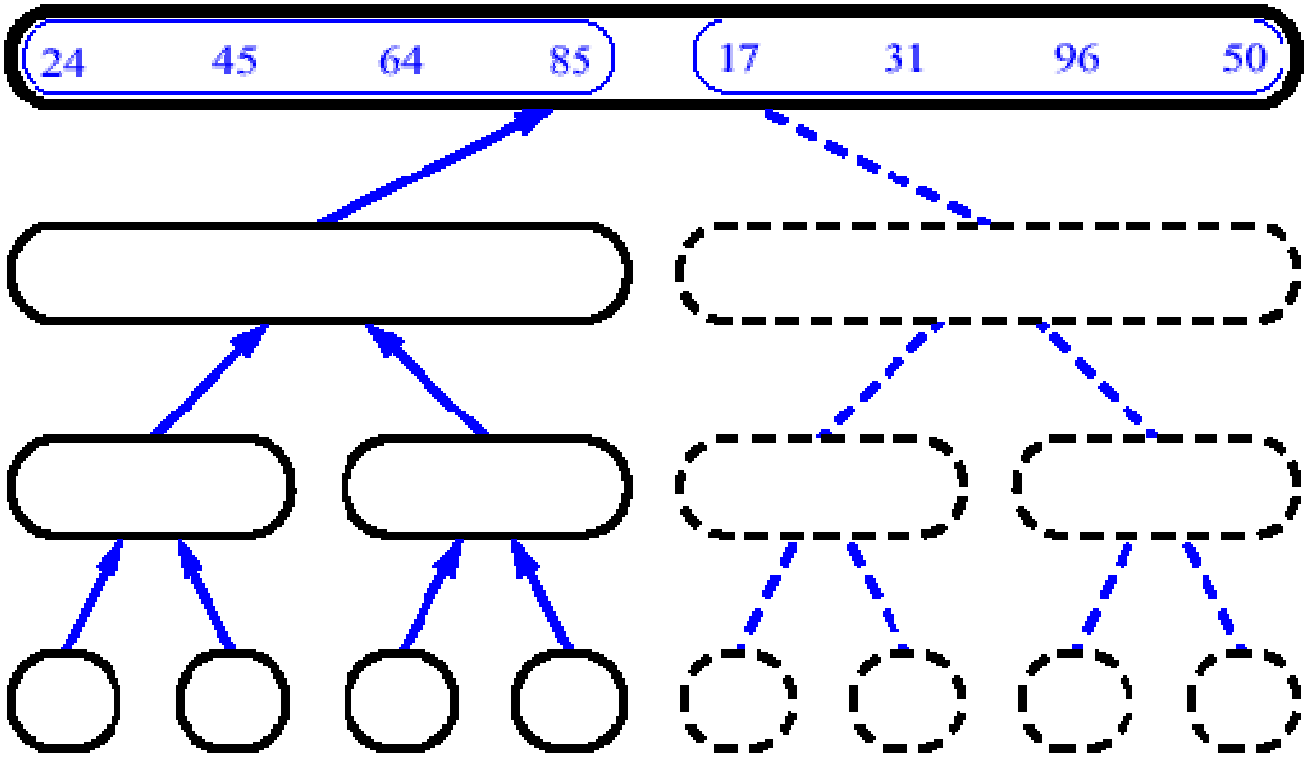
# MergeSort (Exemplo) - 16



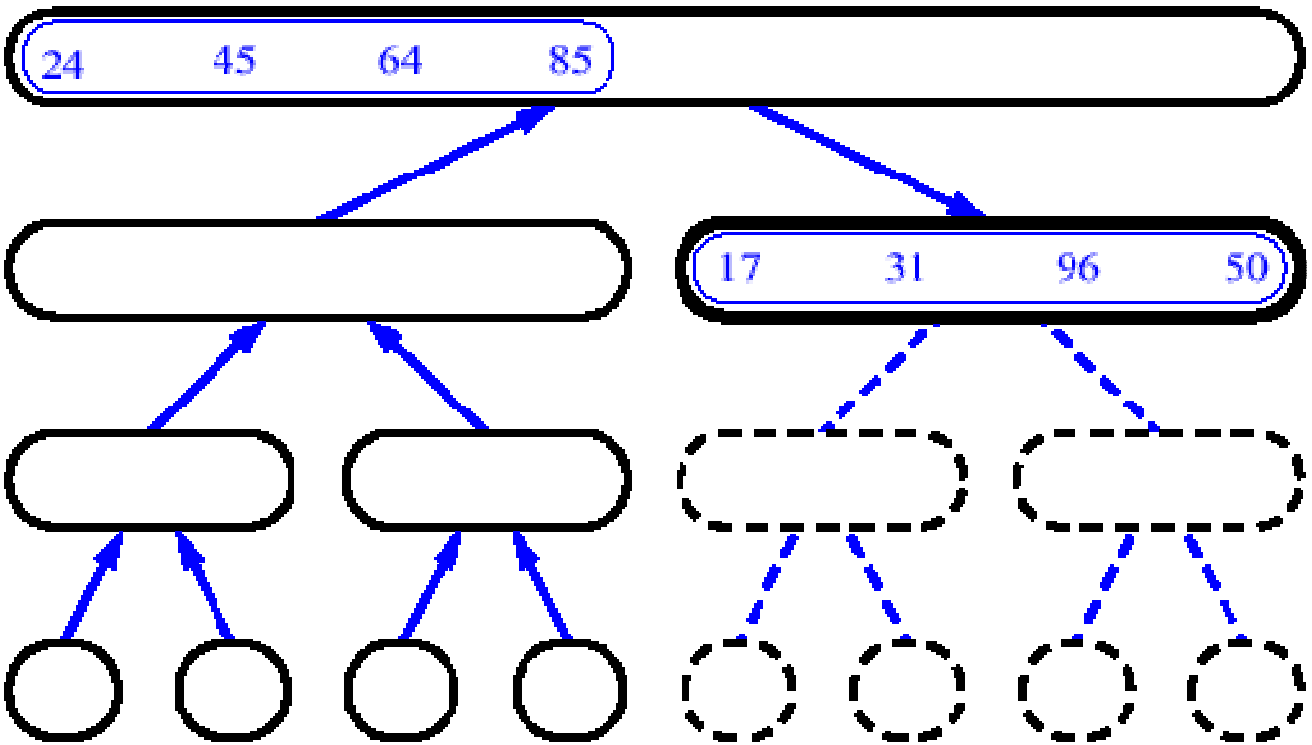
# MergeSort (Exemplo) - 17



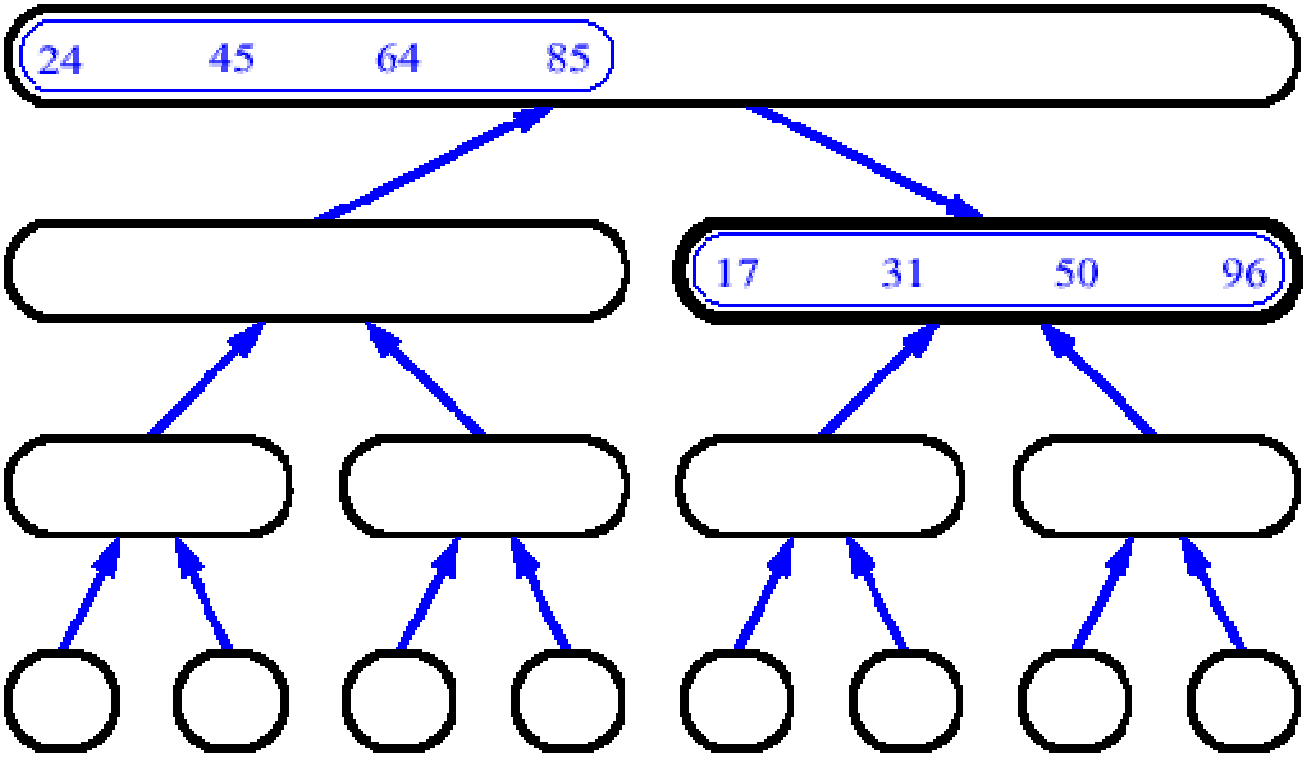
# MergeSort (Exemplo) - 18



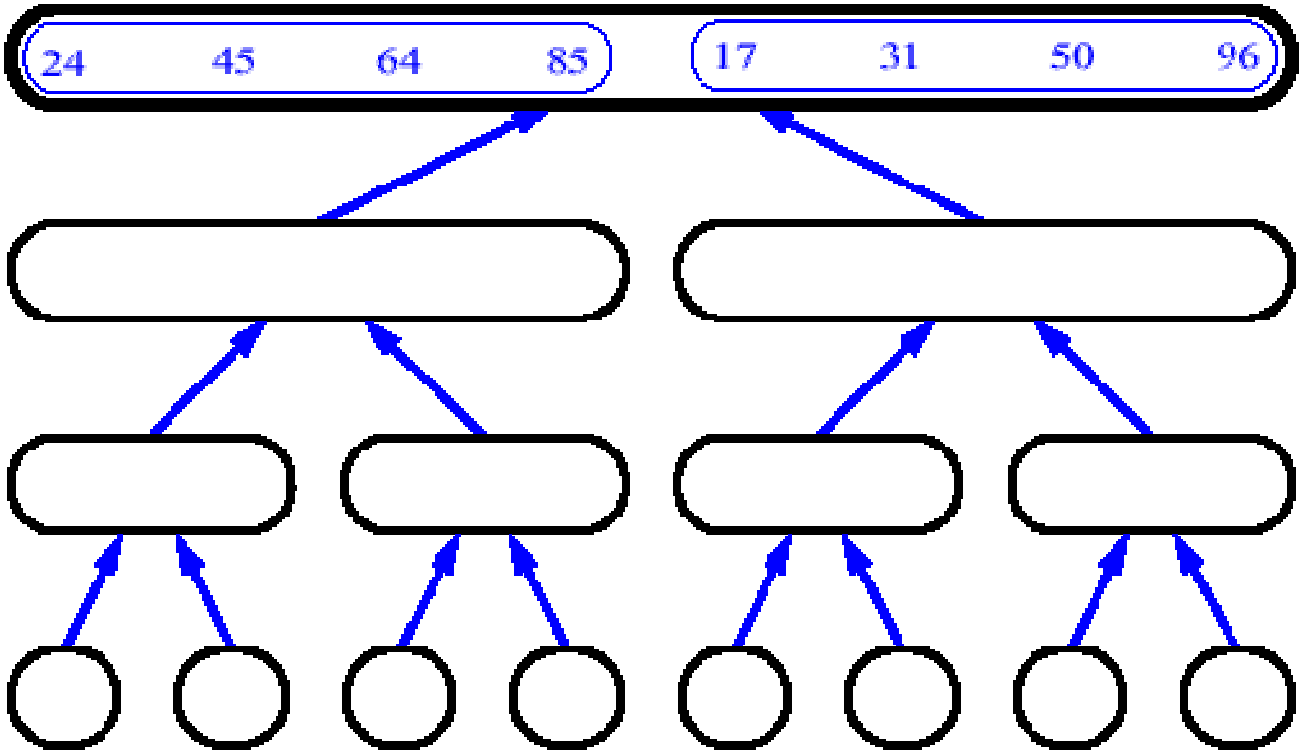
# MergeSort (Exemplo) - 19



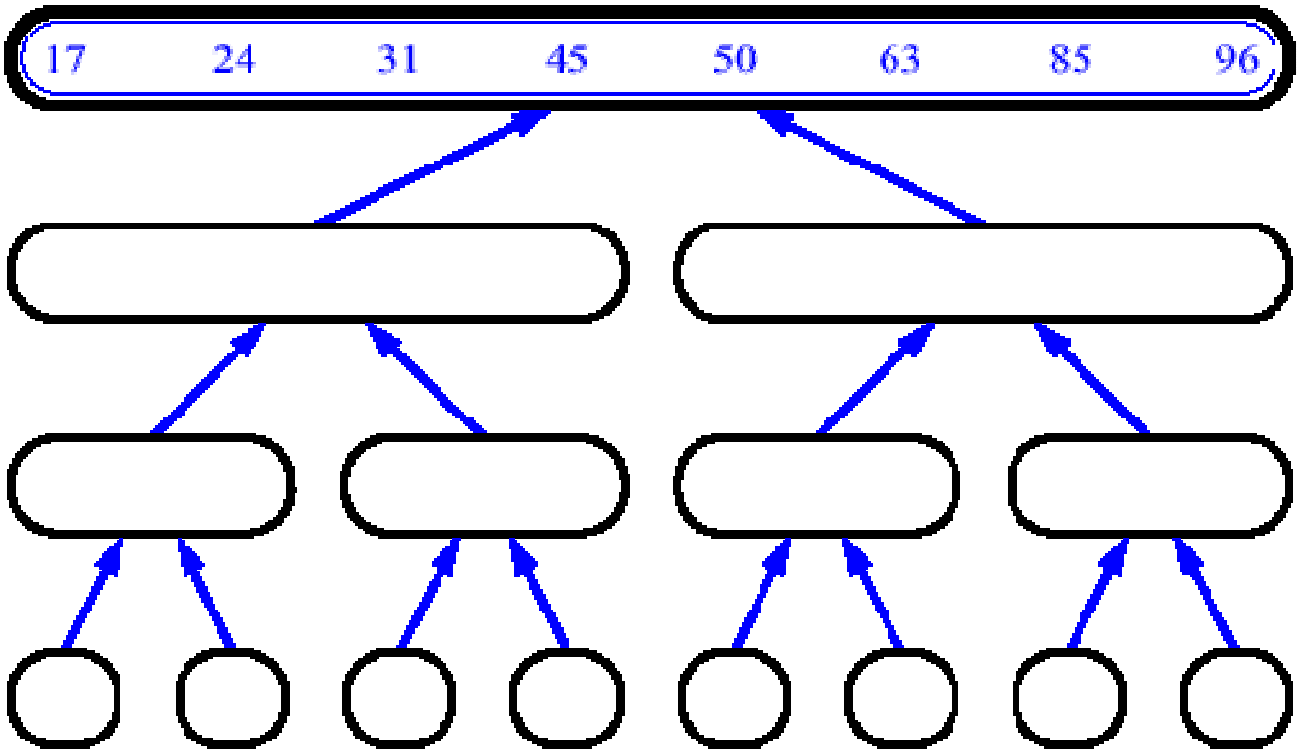
# MergeSort (Exemplo) - 20

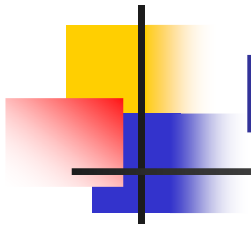


# MergeSort (Exemplo) - 21



# MergeSort (Exemplo) - 22





# Recorrências

- As **chamadas recursivas** nos algoritmos podem ser descritas usando-se equações(ou inequações) de **recorrência**

- Recorrência é uma equação ou desigualdade que descreve uma função em termos dos seus valores para entradas menores

# Recorrências (2)

- Exemplo: Busca Binária

```
BuscaBin(A[1…n],q)
    if n=1
       then if A[n]=q then return n
                      else return 0

     k←(n+1)/2
     if q < A[k] then BuscaBin(A[1…k-1],q)
                 else BuscaBin(A[k…n],q)
```

$$T(n) = \begin{cases} 1 & \text{if } n = 1; \\ 1 + T(n/2) & \text{if } n > 1; \end{cases}$$

# Exemplo: *MergeSort*

- Suposição: O tamanho do problema original é uma potência de dois (simplificação)

Duas subsequências de tamanho n/2

```
Merge-Sort(A, p, r)
    if p < r then
        q←⌊(p+r)/2⌋
        Merge-Sort(A, p, q)
        Merge-Sort(A, q+1, r)
        Merge(A, p, q, r)
```

$$T(n) = \begin{cases} 0 & \text{if } n = 1; \\ T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + T_{\text{merge}}(n) & \text{if } n > 1; \end{cases}$$

# Procedimento do *Merge*

```
Merge(A, p, q, r)
  n1 = q-p+1                                   Θ(1)
  n2 = r-q
  criar arranjos L[0..n1] e R[0..n2]
  for i=0 to n1-1                              Θ(n1)
    do L[i]=A[p+i]
  for j=0 to n2-1                              Θ(n2)
    do R[j]=A[q+j+1]
  L[n1]=∞                                      Θ(1)
  R[n2]=∞
  i=0
  j=0
  for k=p to r                                 Θ(n)
    do if L[i] ≤ R[j]
      then A[k]=L[i]
        i=i+1
      else A[K]=R[j]
        j=j+1
```