ENGENHARIA INFORMÁTICA E DE COMPUTADORES

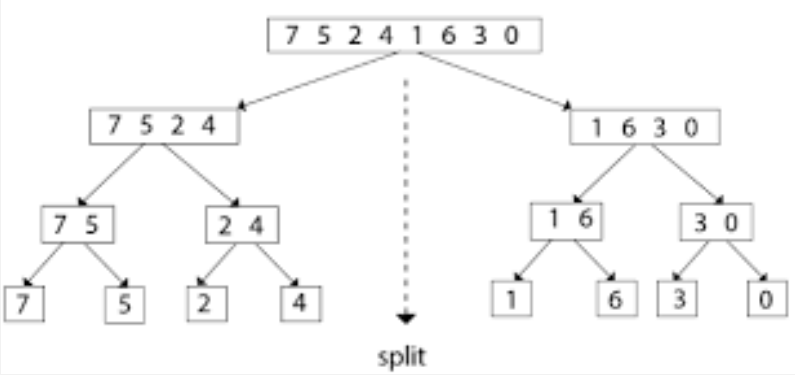
Algoritmos e Estruturas de Dados

(parte 3 – Algoritmo de Ordenação Merge Sort)

2° Semestre 2022/2023
Instituto Superior de Engenharia de Lisboa
Paula Graça

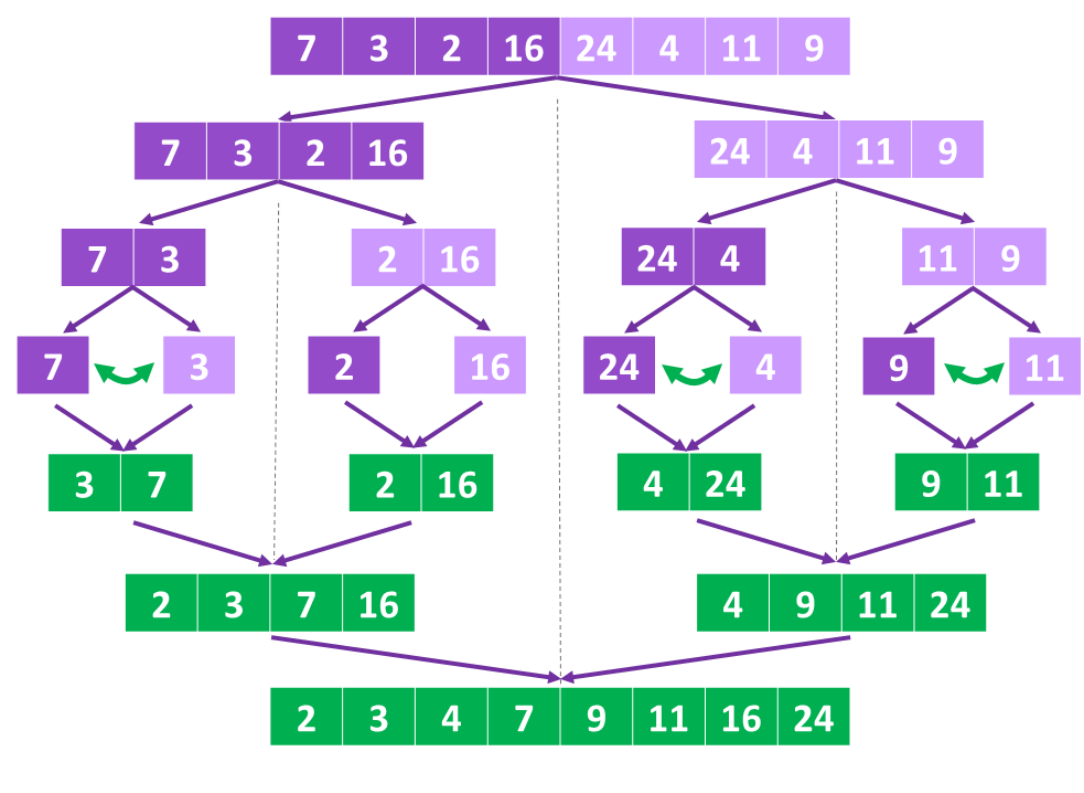
MERGE SORT

- Solução: O Merge Sort é uma aplicação perfeita da técnica dividir para conquistar
  - **Divide** o problema particionando o *array* ao meio sucessivamente, criando dois *subarrays,* um com a metade esquerda e outro com a metade direita, até terem dimensão 1, não podendo ser mais particionado



  - **Conquista** quando os sub-problemas (*subarrays*) ficam com dimensão 1, já estão ordenados
  - **Combina** os resultados progredindo inversamente, juntando ambas as metades (*subarrays*) ordenando os elementos entre si, até obter o *array* original ordenado

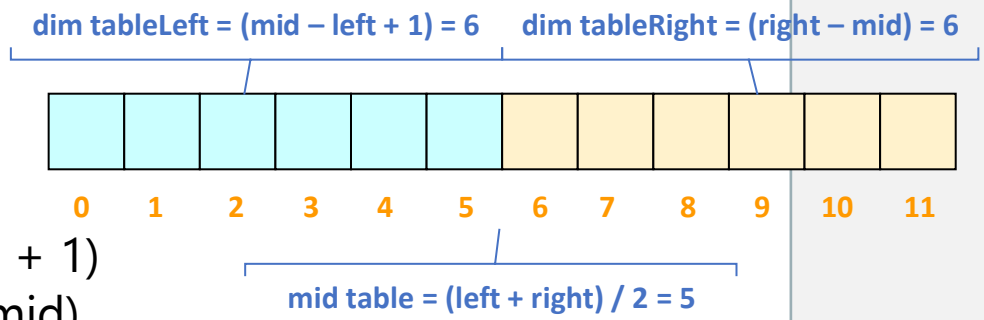
MERGE SORT



- É um algoritmo estável (*stable*), pois os valores repetidos mantêm as mesmas posições relativas
- Utiliza espaço de armazenamento adicional para a ordenação (*not in-place*)

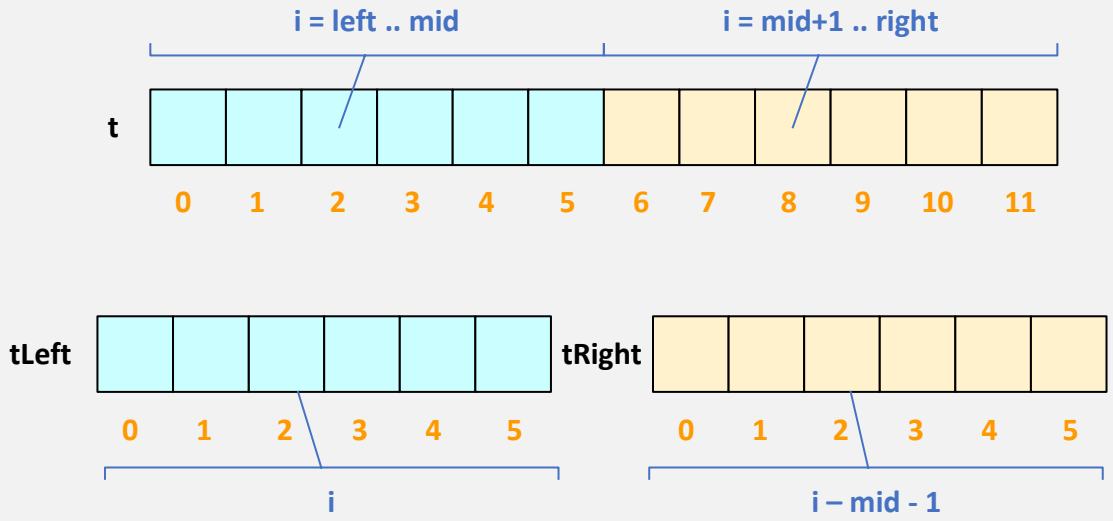
MERGE SORT

```
fun mergeSort(table: IntArray, left: Int, right: Int) {
    if (left < right) {
        val mid = (right + left)/ 2


        val tableLeft = IntArray(mid - left + 1)
        val tableRight = IntArray(right - mid)


        //divide os elementos de table por tableLeft e tableRight
        divide(table, tableLeft, tableRight, left, mid, right)

        mergeSort(tableLeft, 0, mid)                  //repete para tableLeft
        mergeSort(tableRight, 0, right - mid - 1) //repete para tableRight

        // junta ambas as metades em table, ordenando-as
        merge(table, tableLeft, tableRight, left, mid, right)
    }
}
```



MERGE SORT

```
fun divide(t: IntArray, tLeft: IntArray, tRight: IntArray, left: Int,
        mid: Int, right: Int) {
    for (i in left..mid)              // copia para tLeft
        tLeft[i] = t[i]
    for (i in mid + 1..right)
        tRight[i - mid - 1] = t[i] // copia para tRight
}
```



MERGE SORT

```
fun merge(t: IntArray, tLeft: IntArray, tRight: IntArray, left: Int, mid: Int, right: Int) {
    var i = 0
    var j = 0
    var k = left
    // faz o merge ordenado de tLeft com tRight
    while (i < tLeft.size && j < tRight.size)
        if (tLeft[i] <= tRight[j])
            t[k++] = tLeft[i++]
        else t[k++] = tRight[j++]
    while (i < tLeft.size)        // copia os restantes elementos de tLeft
        t[k++] = tLeft[i++]
    while (j < tRight.size)       // copia os restantes elementos de tRight
        t[k++] = tRight[j++]
}
```

PROPRIEDADES DA ORDENAÇÃO

- Um algoritmo de ordenação é dito

  - Adaptativo (*adaptive*) - se tira partido da ordem existente dos dados para otimizar a ordenação
  - Estável (*stable*) - se preserva a ordem relativa dos elementos com chaves repetidas de acordo com o critério de ordenação
  - Interno (*in-place*) - se o conjunto de todos os dados a ordenar cabe na memória utilizada para os armazenar