UEM/CTC – Departamento de Informática
Curso: Bacharelado em Informática
Estrutura de Dados
Professor: Flávio Rogério Uber

MergeSort

MergeSort

- Utiliza-se da técnica de divisão e conquista
- E um método mais “estável” em relação ao Quicksort
- Consome mais memória (recursividade e vetor auxiliar)
- Tempo O(n logn)

MergeSort

11	8	14	7	12	5	18	3	15	6

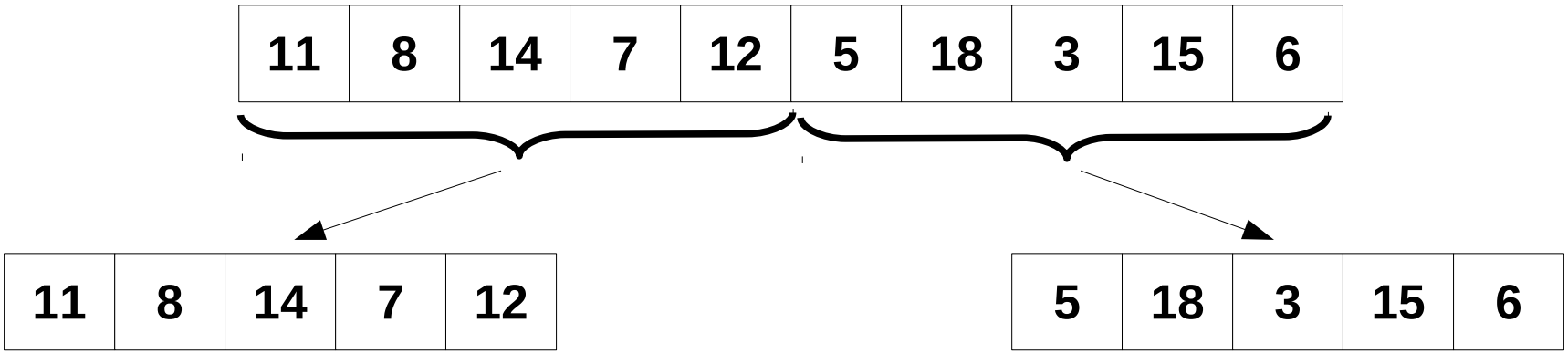
MergeSort

Divisão sucessiva do vetor em duas partes:
de 1 ate n/2 e
de n/2+1 ate n

11	8	14	7	12	5	18	3	15	6

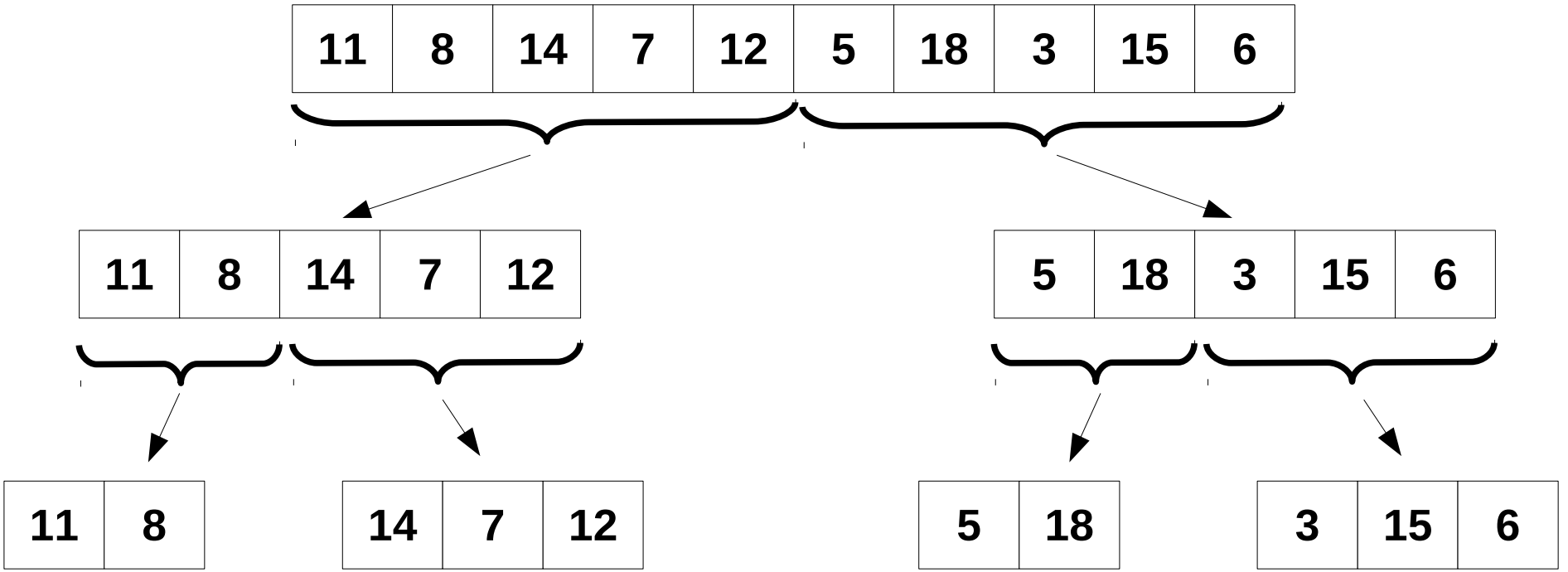
MergeSort

Divisão sucessiva do vetor em duas partes:
de 1 ate n/2 e
de n/2+1 ate n



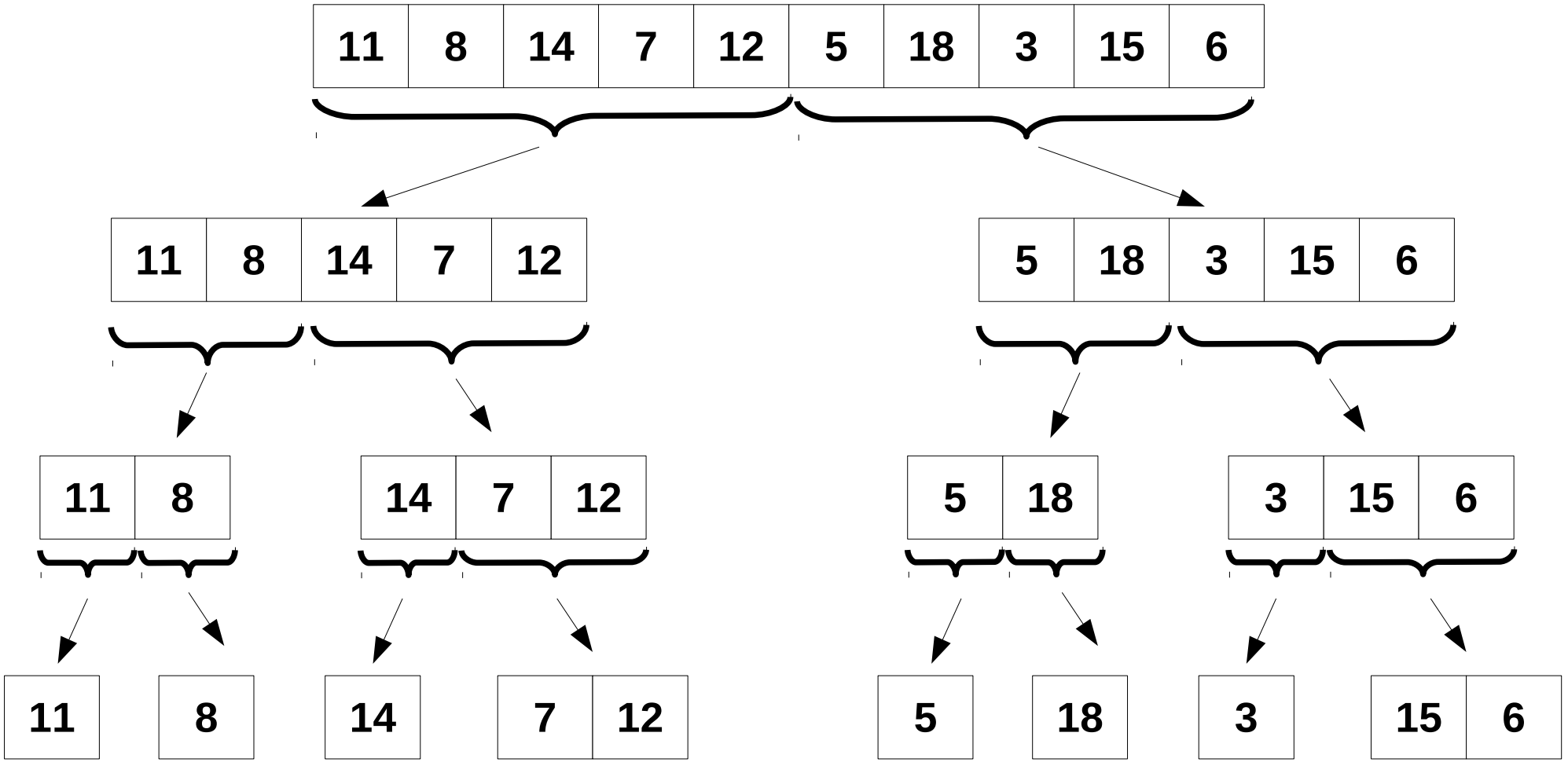
MergeSort

Divisão sucessiva do vetor em duas partes:
de 1 ate n/2 e
de n/2+1 ate n



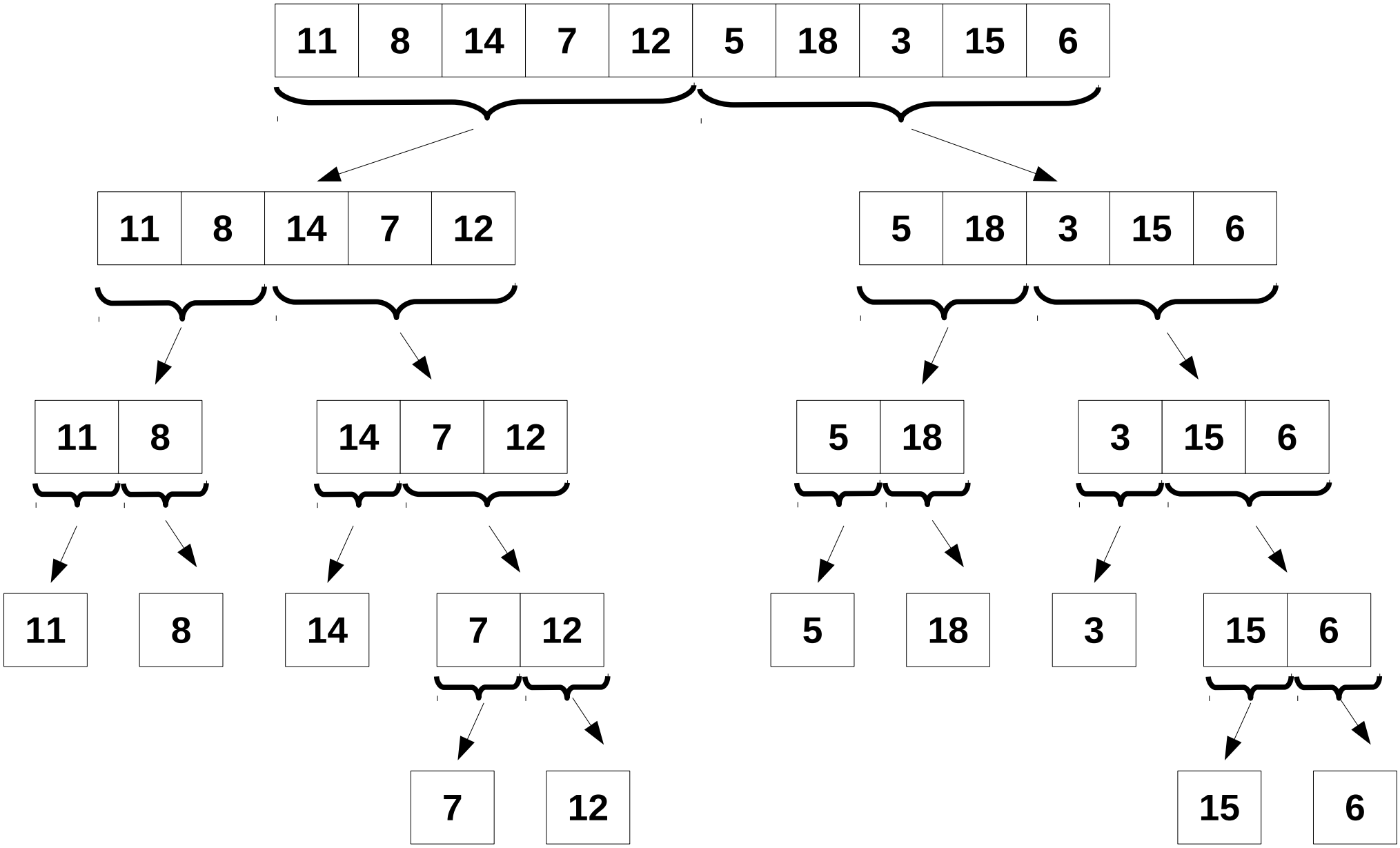
MergeSort

Divisão sucessiva do vetor em duas partes:
de 1 ate n/2 e
de n/2+1 ate n



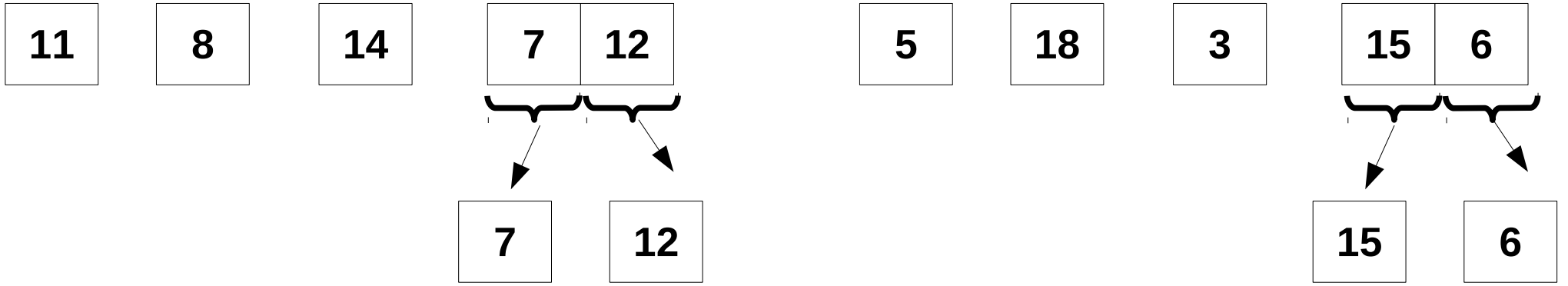
MergeSort

Divisão sucessiva do vetor em duas partes:
de 1 ate n/2 e
de n/2+1 ate n



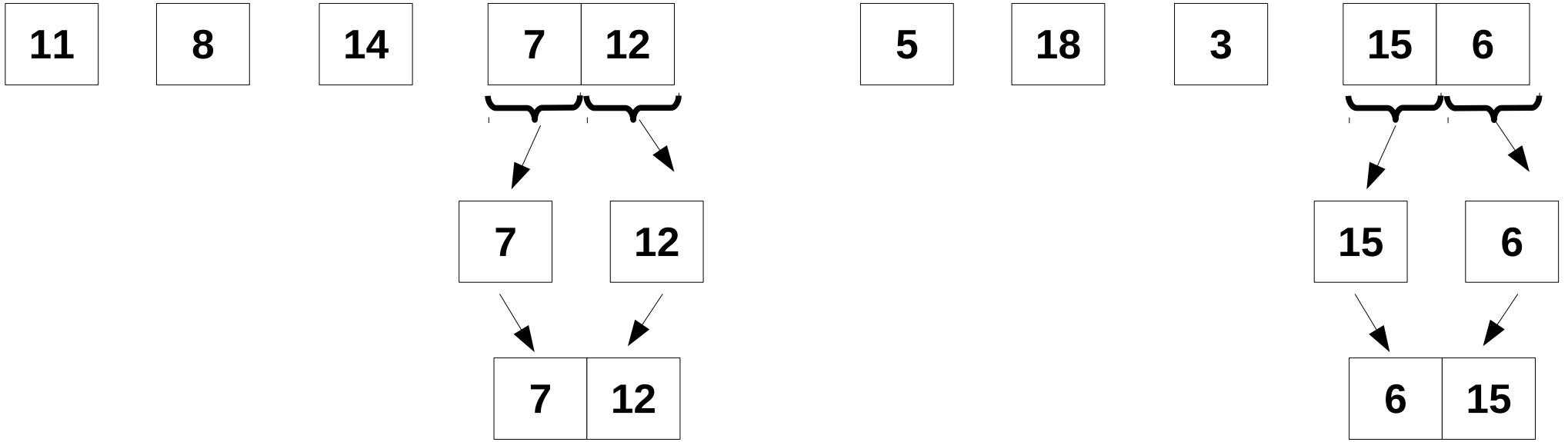
MergeSort

No retorno da recursão ordena-se os vetores parciais



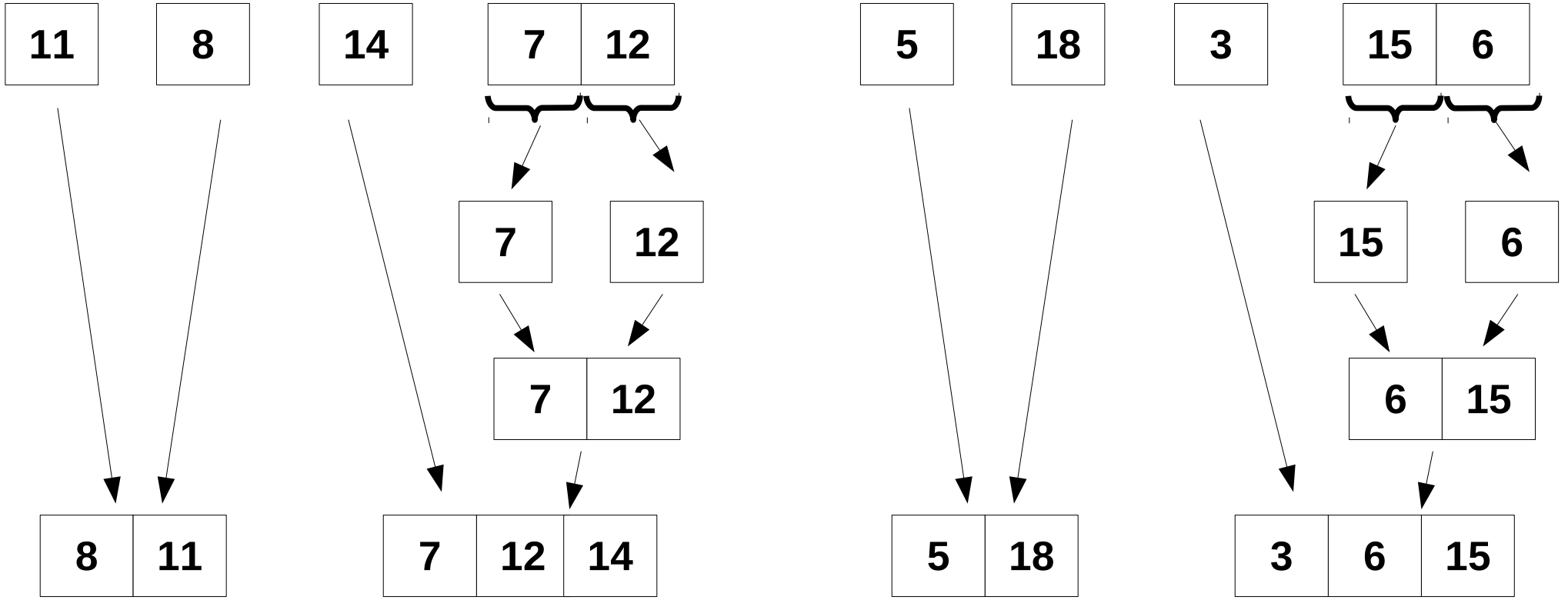
MergeSort

No retorno da recursão ordena-se os vetores parciais



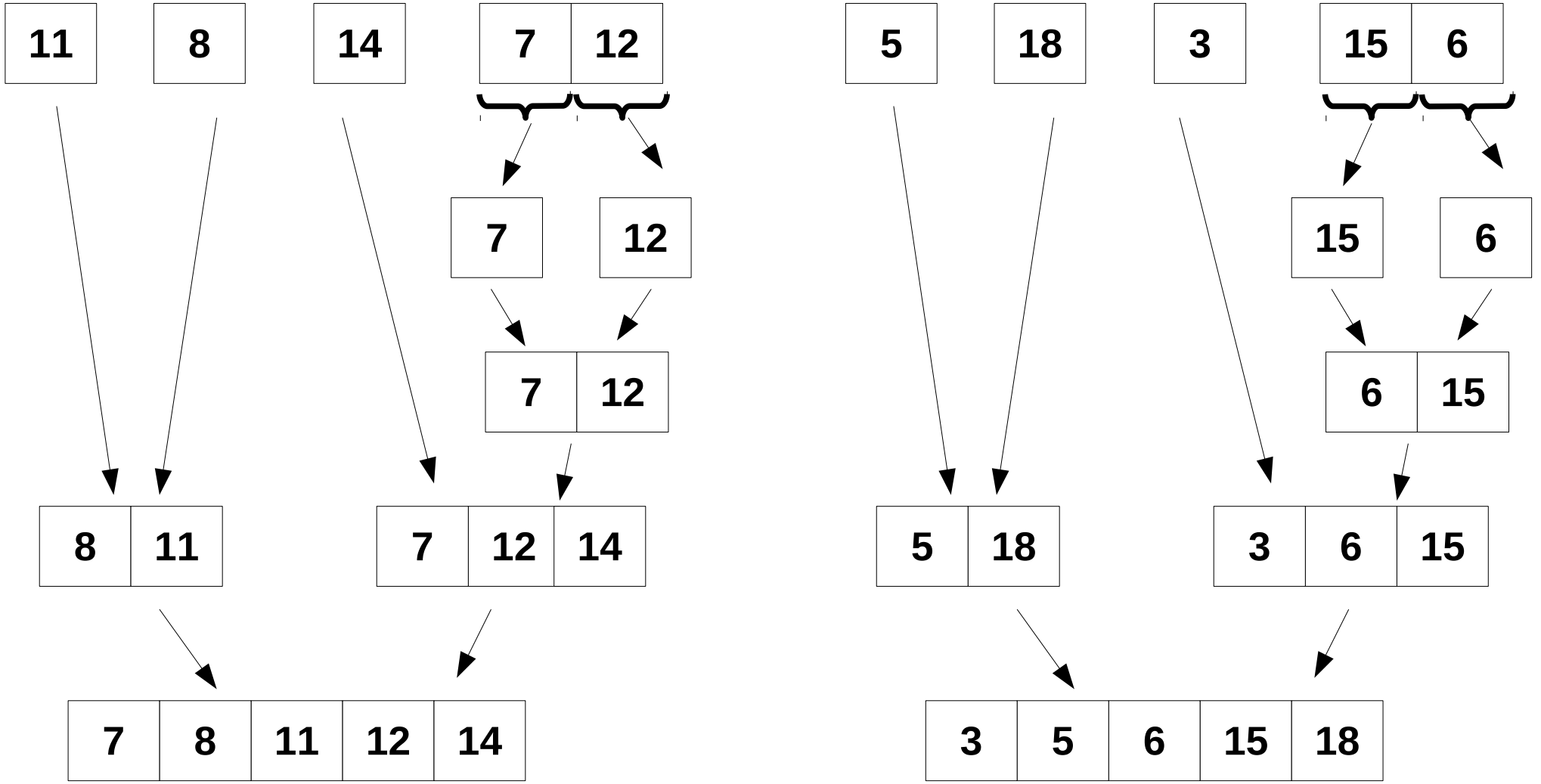
MergeSort

No retorno da recursão ordena-se os vetores parciais



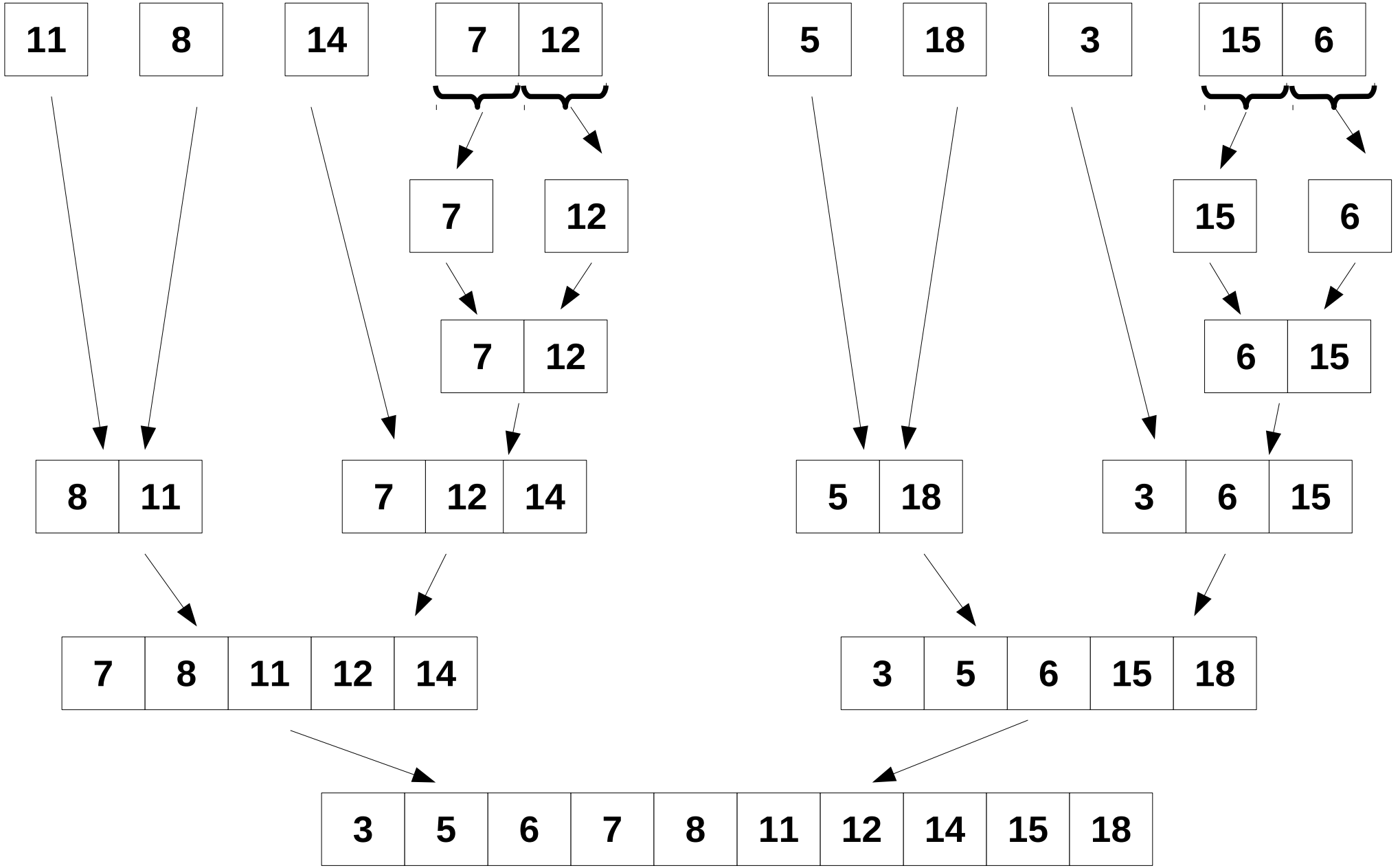
MergeSort

No retorno da recursão ordena-se os vetores parciais



MergeSort

No retorno da recursão ordena-se os vetores parciais



```
void merge(int *saida, int *auxiliar, int inicio, int meio, int fim){
    int i= inicio, j= meio + 1, k= inicio;
    while(i <= meio && j <= fim){
        if(auxiliar[i] < auxiliar[j]){
            saida[k] = auxiliar[i];
            i++;}
        else{
            saida[k] = auxiliar[j];
            j++;}
        k++; }
    while(i <= meio){
        saida[k] = auxiliar[i];
        i++;
        k++;}
    while(j <= fim){
        saida[k] = auxiliar[j];
        j++;
        k++; }
    //Copia os elementos que foram ordenados para o auxiliar
    for(int p = inicio; p <= fim; p++)
        auxiliar[p] = saida [p];
}
void mergeSort(int *saida, int *auxiliar, int inicio, int fim){
    if(inicio < fim){
        int meio = (inicio + fim) / 2;
        mergeSort(saida, auxiliar, inicio, meio);
        mergeSort(saida, auxiliar, meio + 1, fim);
        merge(saida, auxiliar, inicio, meio, fim);}
}
```

MergeSort

- Exercício
  - Ordene a seguinte sequência utilizando MergeSort e mostre passo a passo:

12	3	15	13	17	2	9	14	5