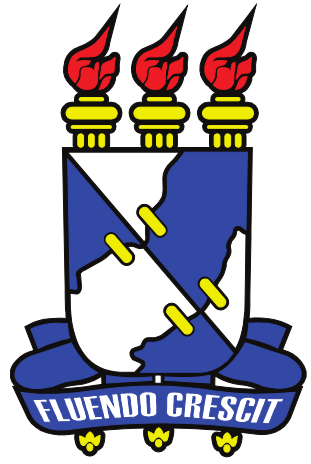


Projeto e Análise de Algoritmos

Mergesort

Bruno Prado

Departamento de Computação / UFS

Introdução

- O que é Mergesort?
  - Criado por John von Neumann em 1945
  - Estratégia de Divisão e Conquista
  - Funciona dividindo o conjunto de dados

Introdução

- Três passos para divisão e conquista em algoritmos
  - Dividir o problema em subproblemas
    - Instâncias menores e mais simples
  - Resolver os subproblemas
    - Mais simples de serem resolvidos
  - Combinar as soluções parciais obtidas para gerar a solução completa
    - Etapa de conquista

Introdução

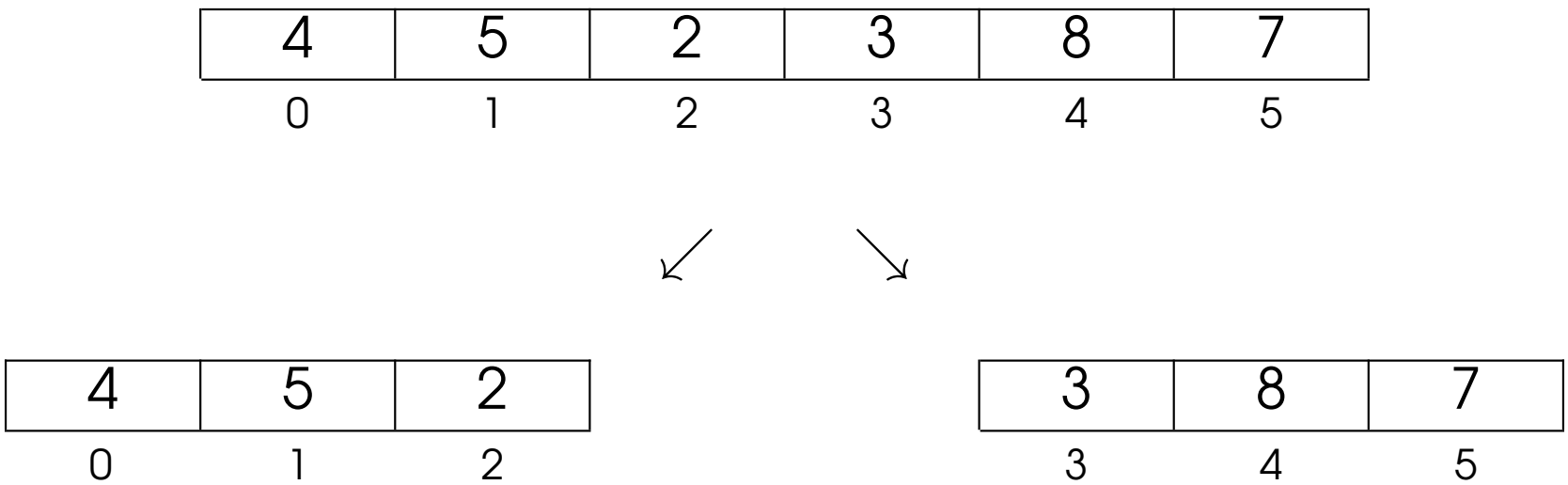
- Vantagens
  - Paralelismo
    - Problema é dividido em partes que podem ser resolvidas separadamente
  - Eficiência algorítmica
    - Complexidade $O(n \log n)$
  - Acesso a memória mais eficiente
    - Dados cabem na memória cache
  - Controle de arredondamento mais preciso
    - Os resultados são combinados ao invés de iterados

Introdução

- Desvantagens
  - Recursão
    - Utilização de pilha que é limitada
    - Menor desempenho por conta do acesso constante a memória
  - Escolha dos casos base
    - Boas escolhas evitam processamento desnecessário para entradas pequenas
  - Subproblemas repetidos
    - É possível obter subproblemas idênticos que vão se calculados repetidamente

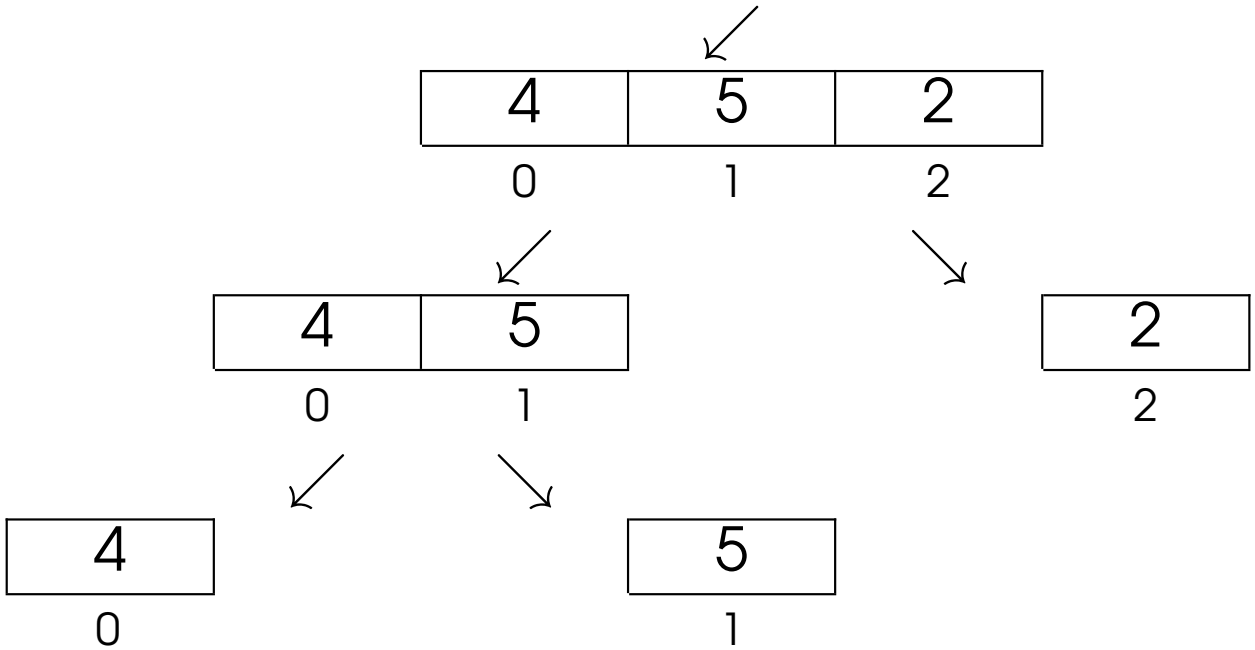
Mergesort

- Princípio de funcionamento
  - Etapa de divisão



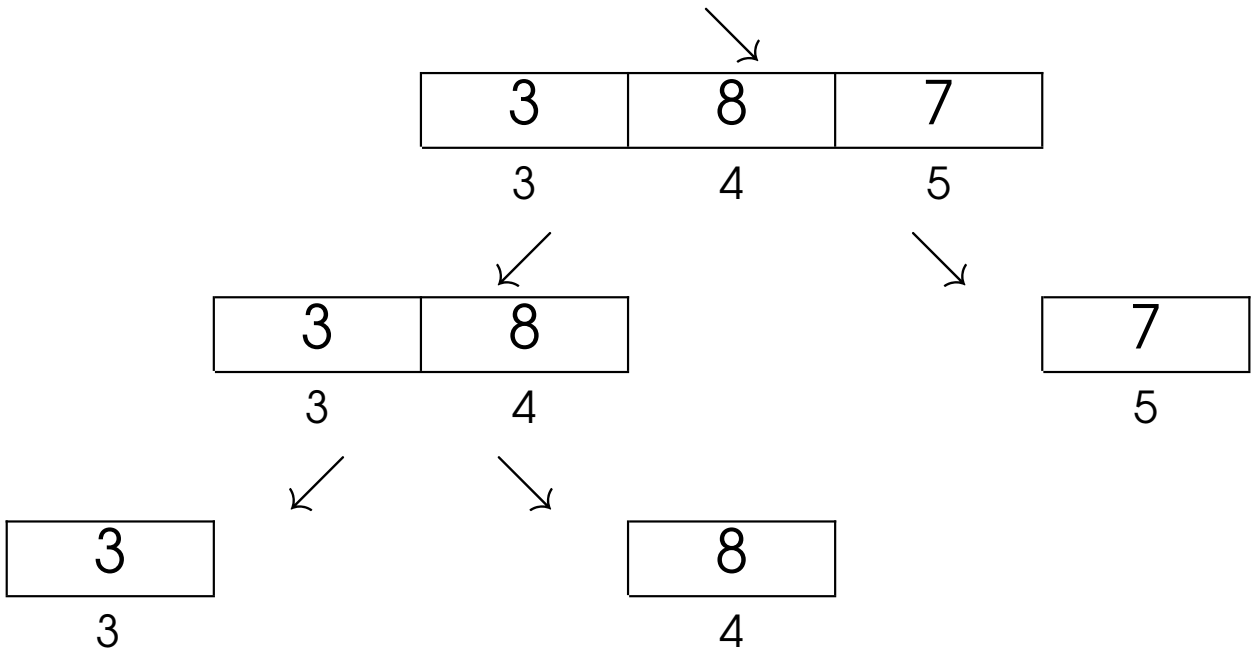
Mergesort

- Princípio de funcionamento
  - Etapa de divisão



Mergesort

- Princípio de funcionamento
  - Etapa de divisão



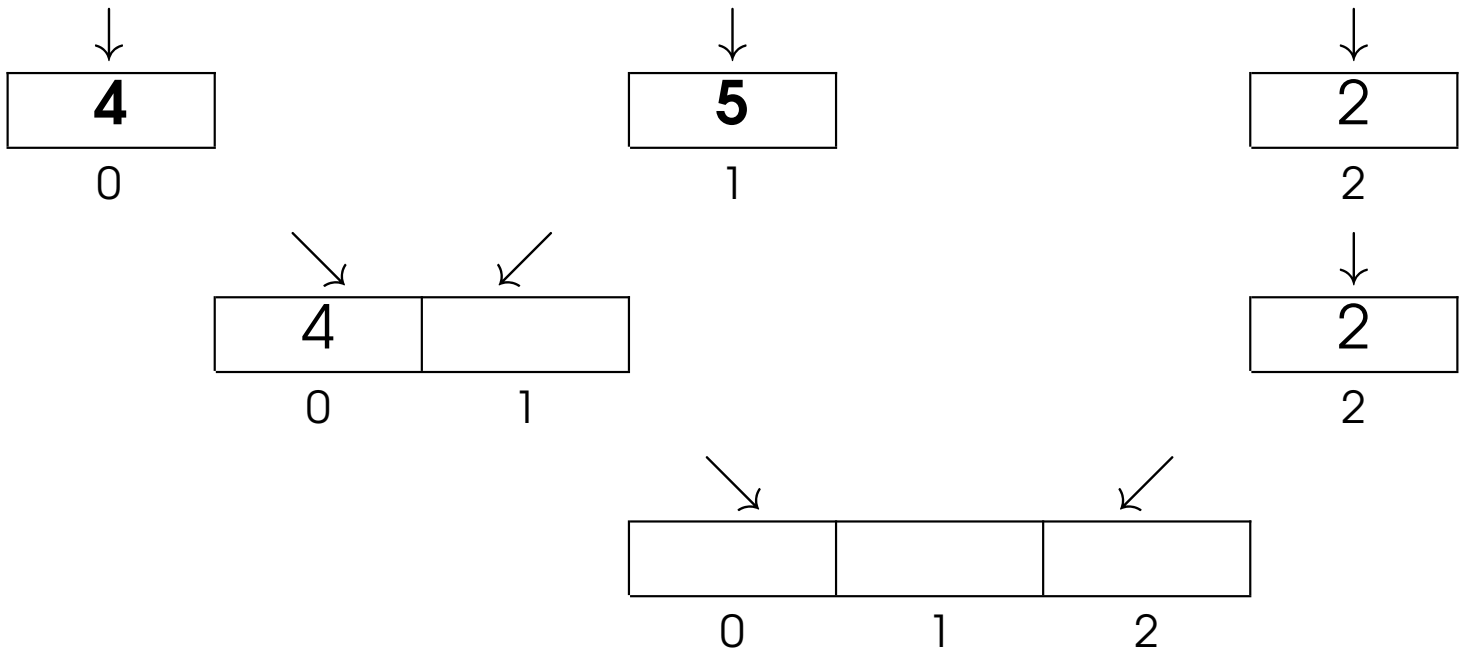
Mergesort

- Etapa de divisão
  - Procedimento mergesort

```
void mergesort(int S(), int E(), int ini, int fim) {
        if(ini < fim) {
                int meio = ini + (fim - ini) / 2;
                mergesort(S, E, ini, meio);
                mergesort(S, E, meio + 1, fim);
                intercalar(S, E, ini, meio, fim);
        }
}
```

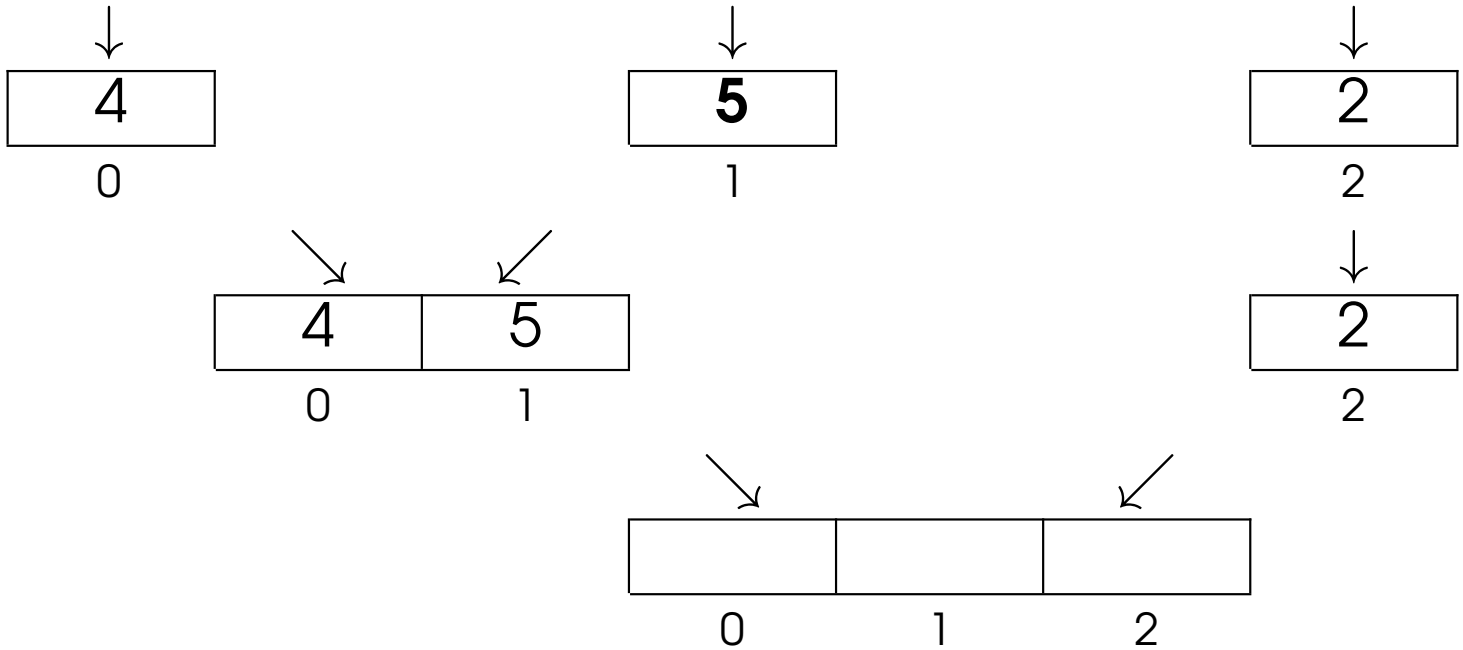
Mergesort

- Princípio de funcionamento
  - Etapa de conquista



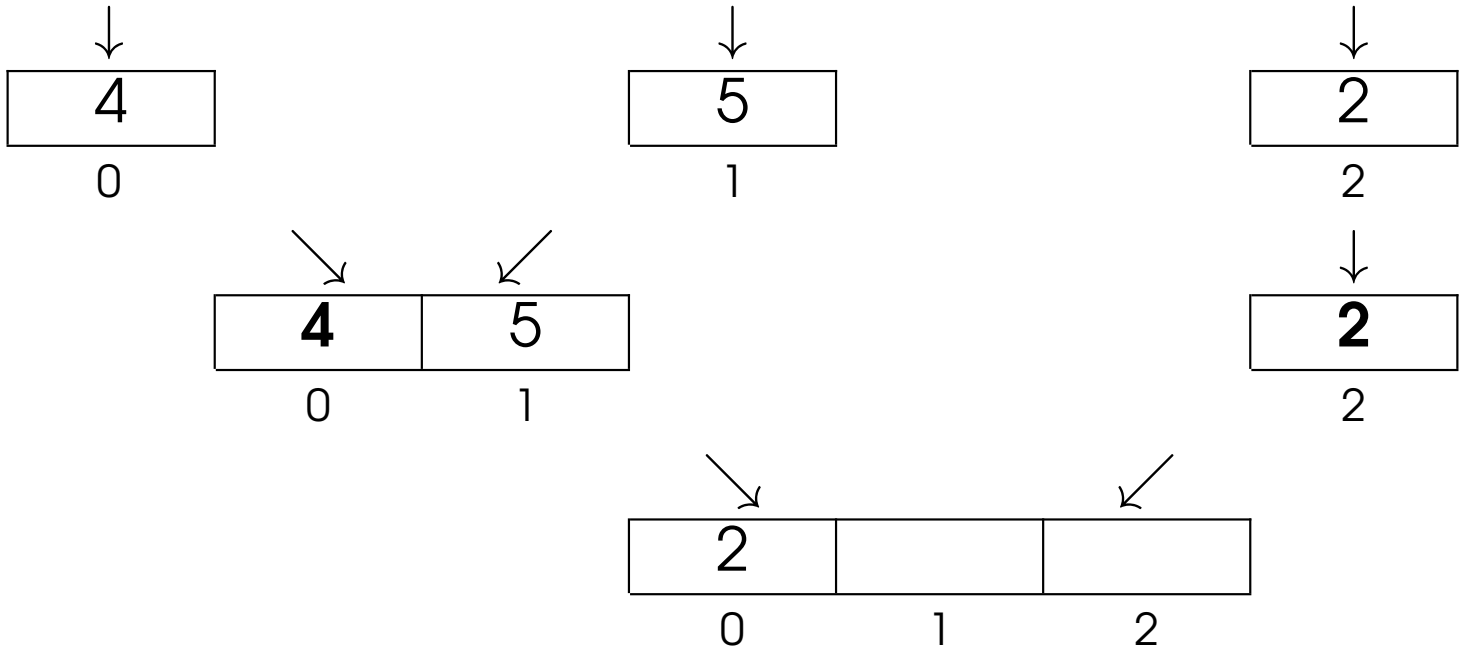
Mergesort

- Princípio de funcionamento
  - Etapa de conquista



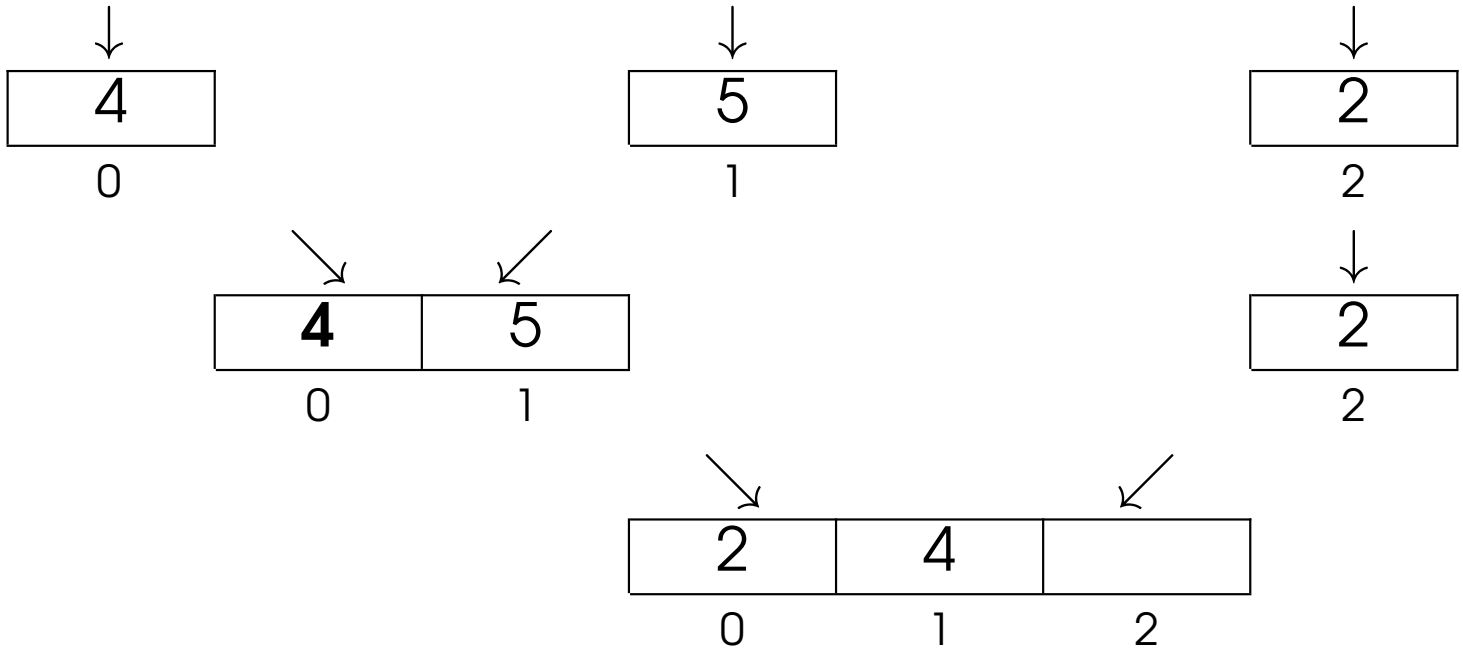
Mergesort

- Princípio de funcionamento
  - Etapa de conquista



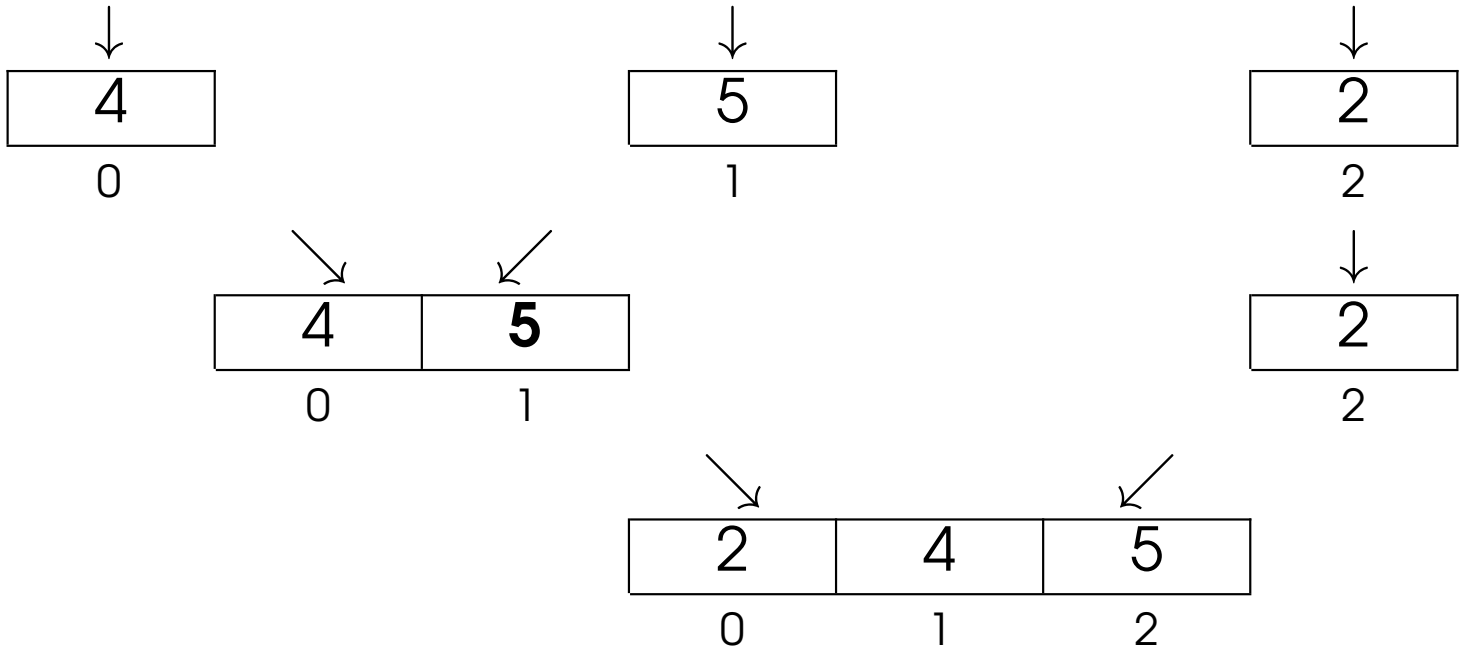
Mergesort

- Princípio de funcionamento
  - Etapa de conquista



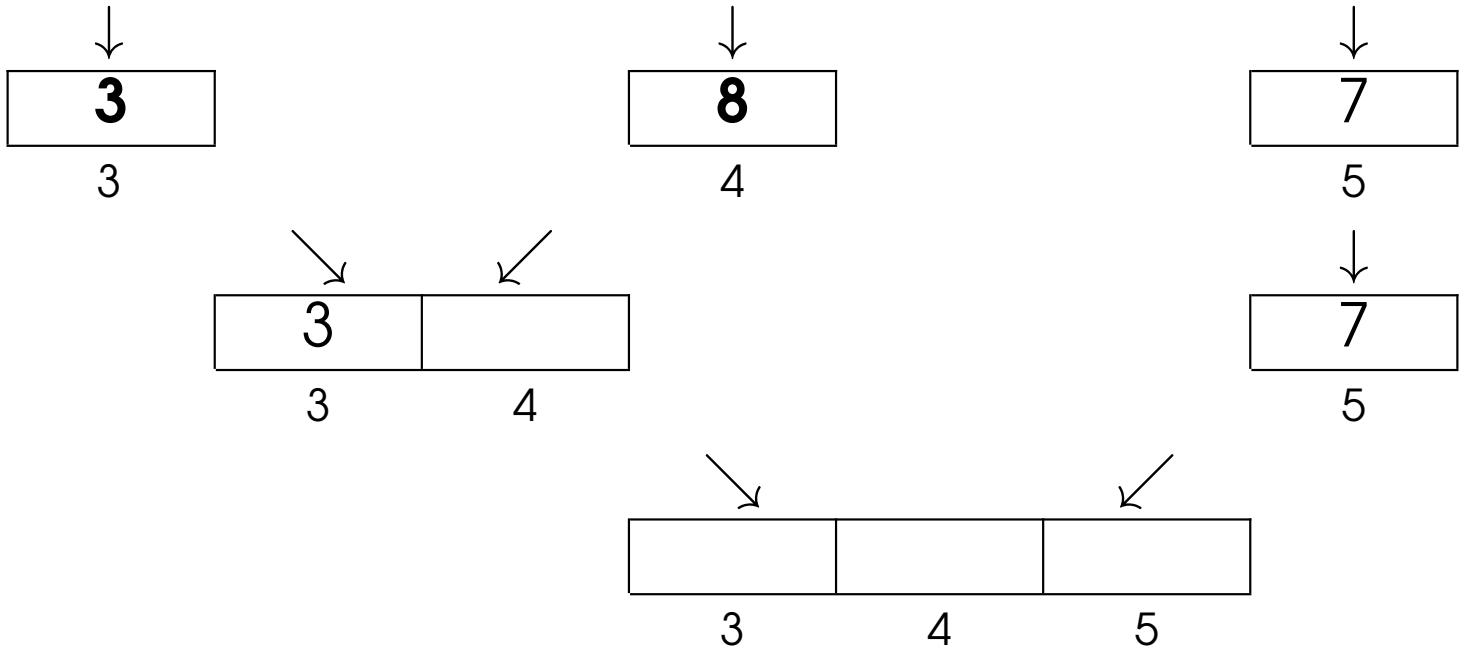
Mergesort

- Princípio de funcionamento
  - Etapa de conquista



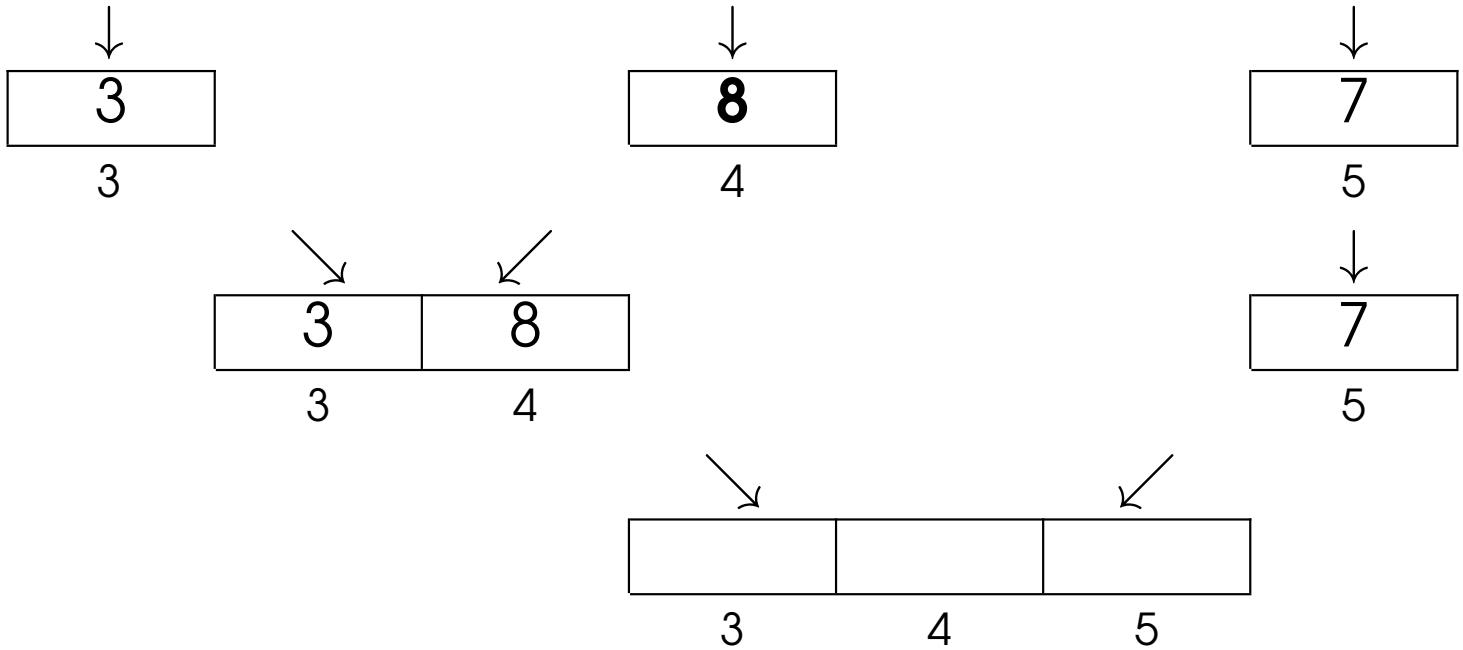
Mergesort

- Princípio de funcionamento
  - Etapa de conquista



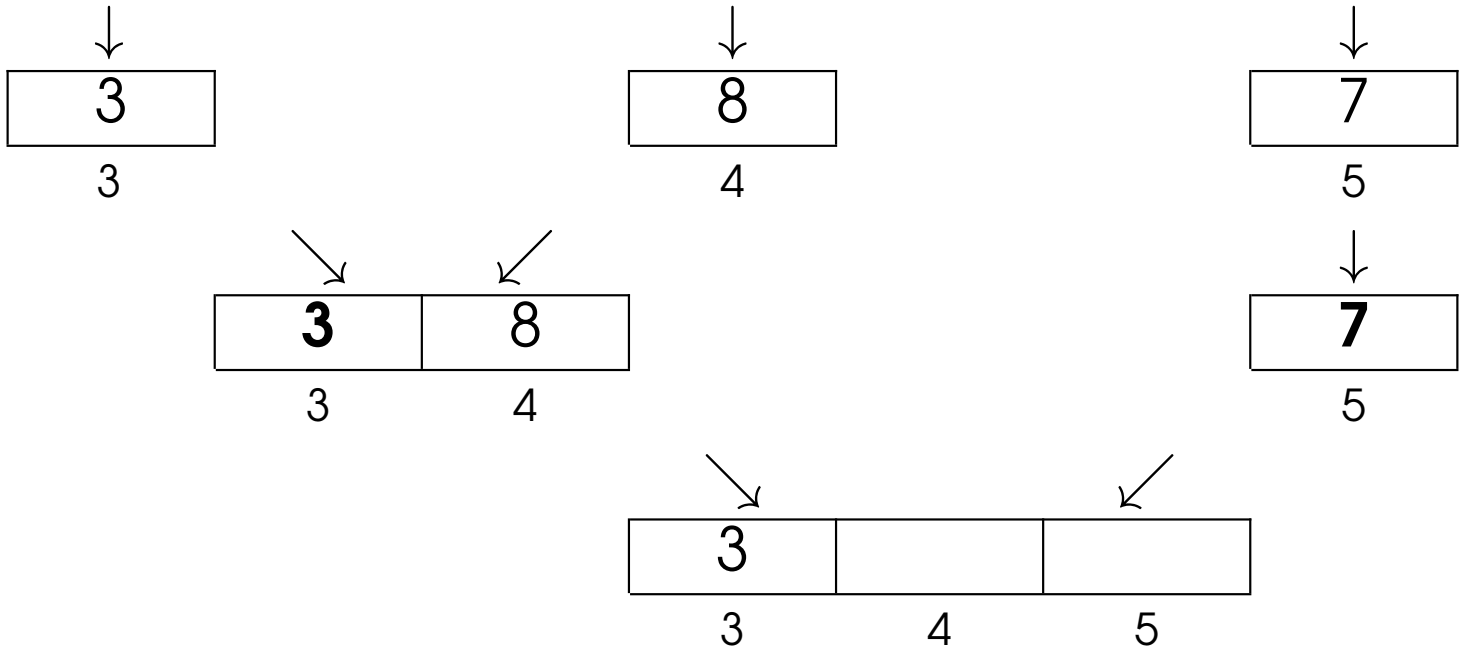
Mergesort

- Princípio de funcionamento
  - Etapa de conquista



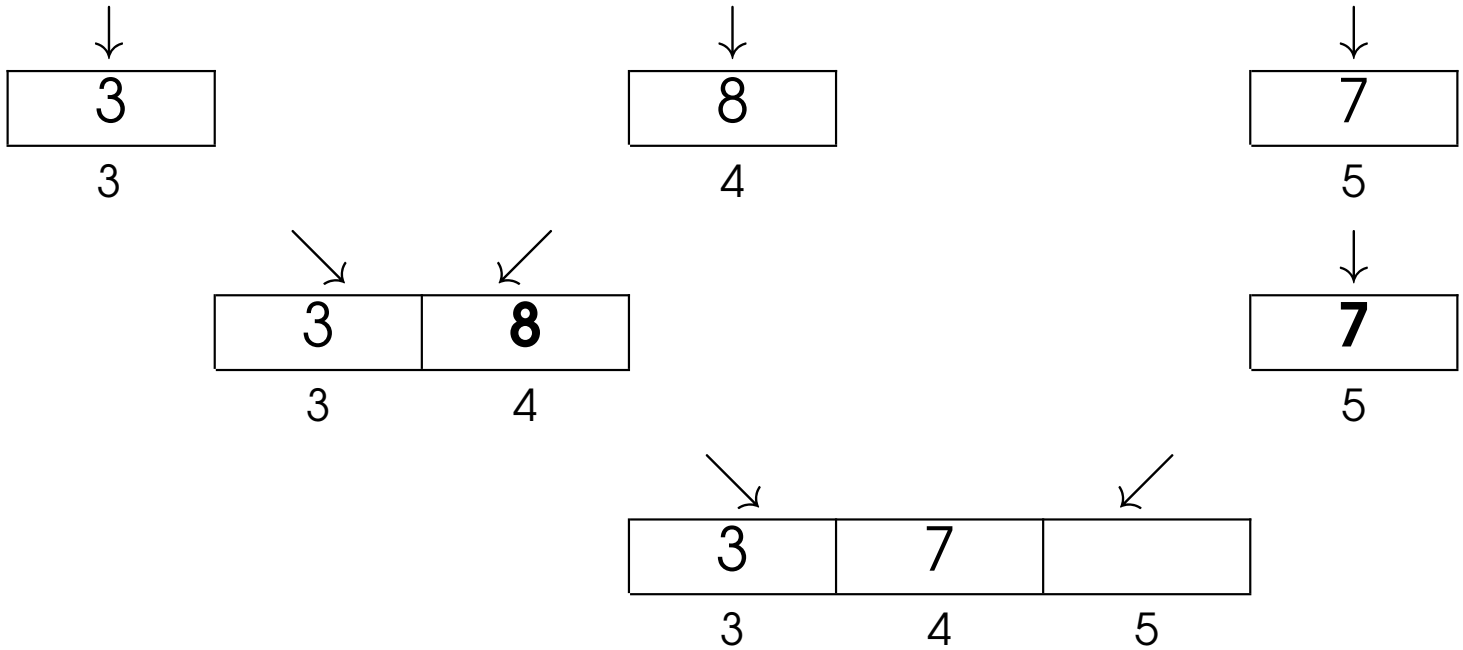
Mergesort

- Princípio de funcionamento
  - Etapa de conquista



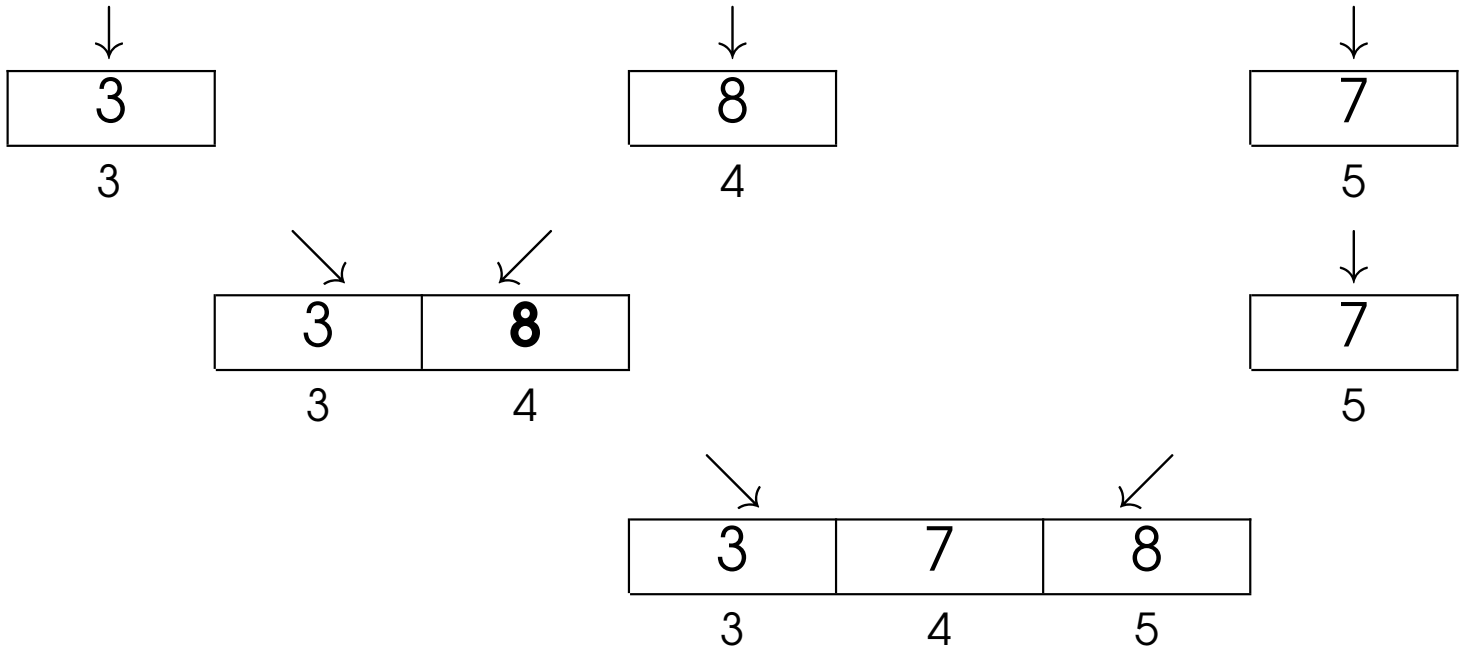
Mergesort

- Princípio de funcionamento
  - Etapa de conquista



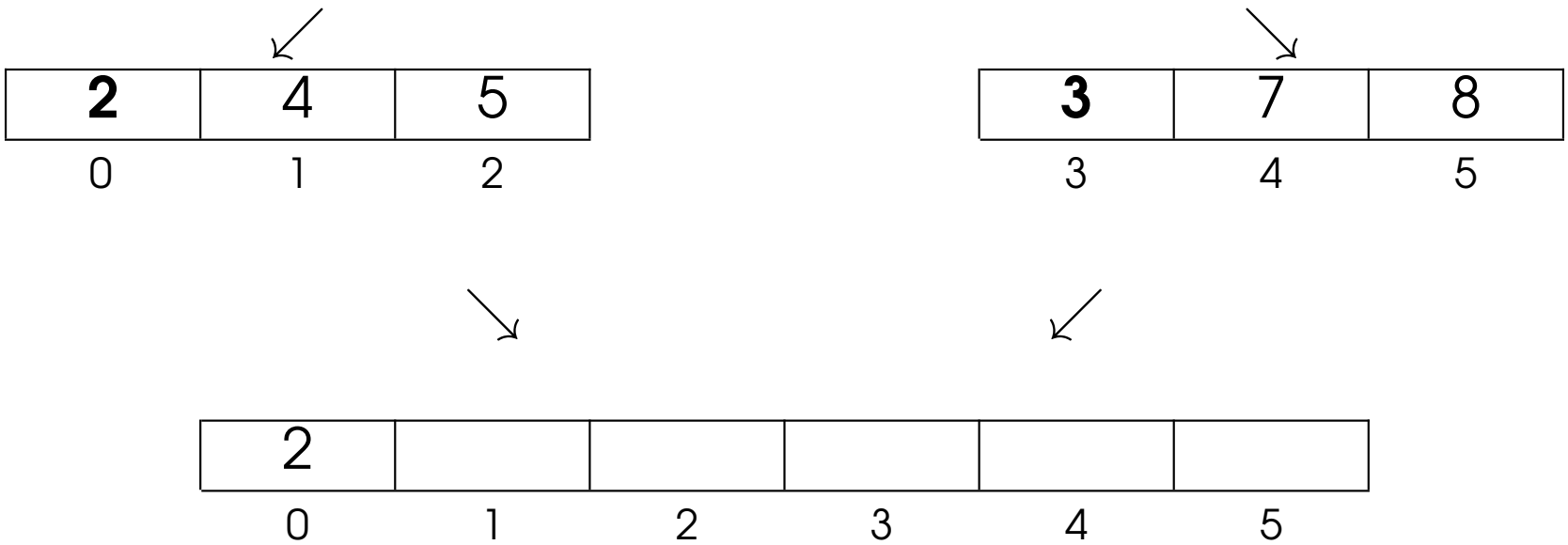
Mergesort

- Princípio de funcionamento
  - Etapa de conquista



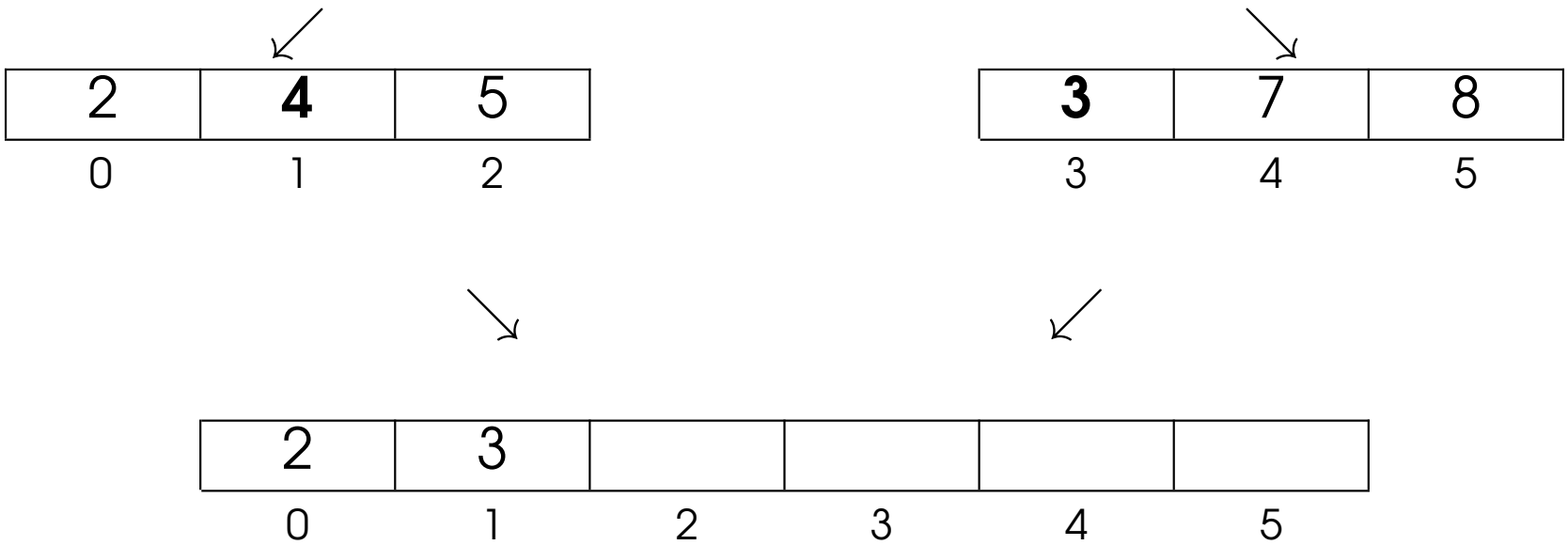
Mergesort

- Princípio de funcionamento
  - Etapa de conquista



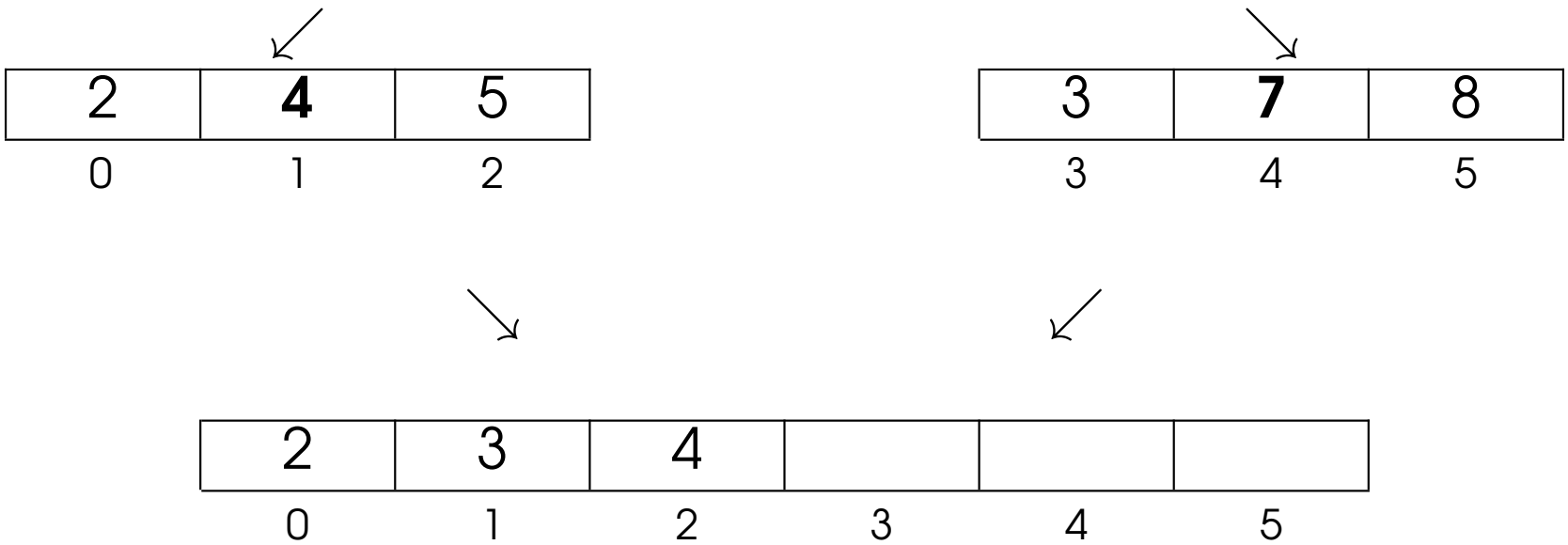
Mergesort

- Princípio de funcionamento
  - Etapa de conquista



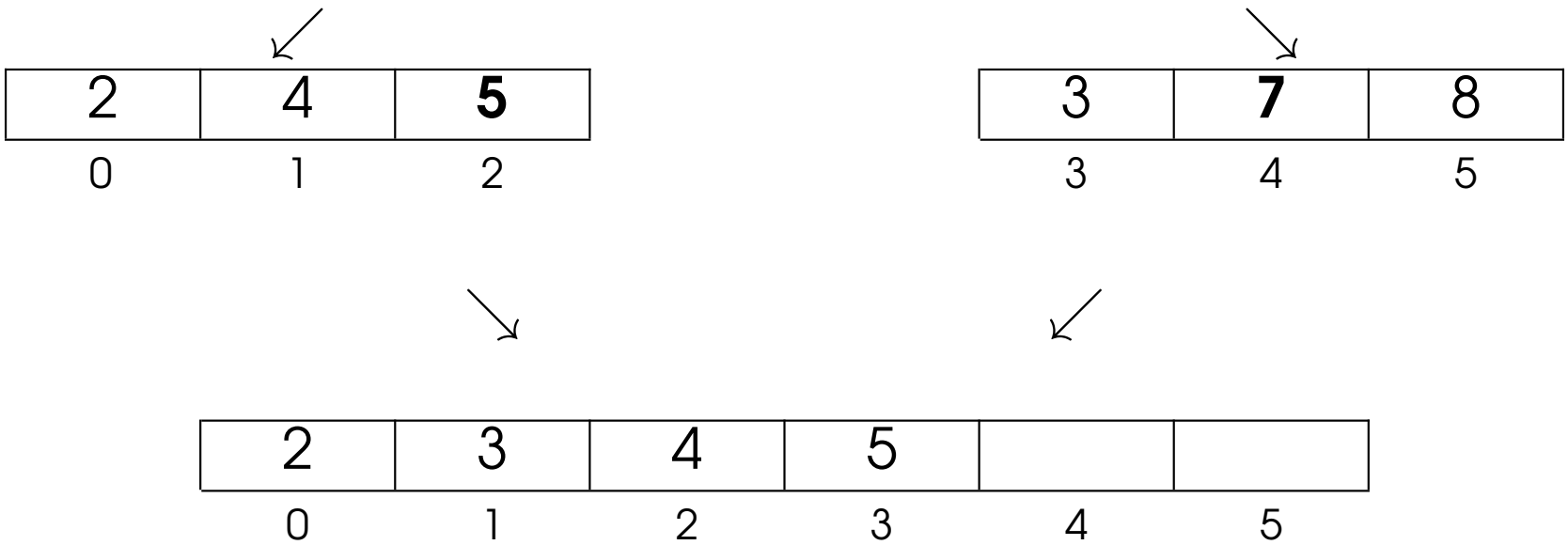
Mergesort

- Princípio de funcionamento
  - Etapa de conquista



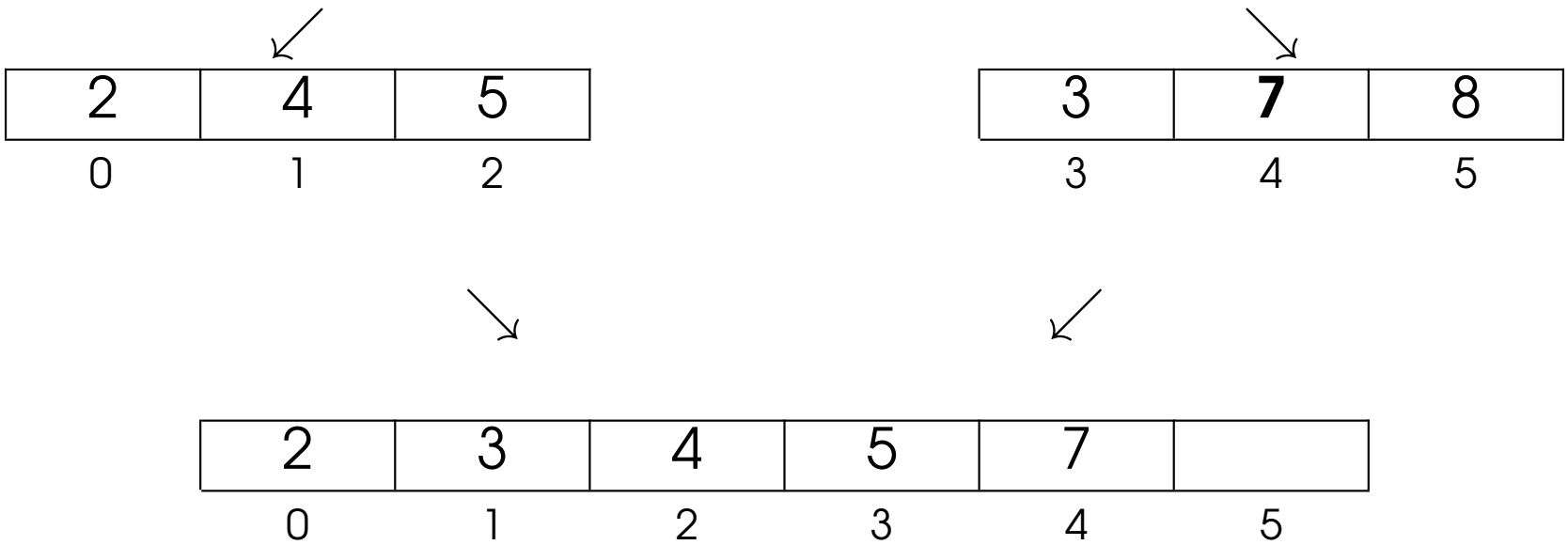
Mergesort

- Princípio de funcionamento
  - Etapa de conquista



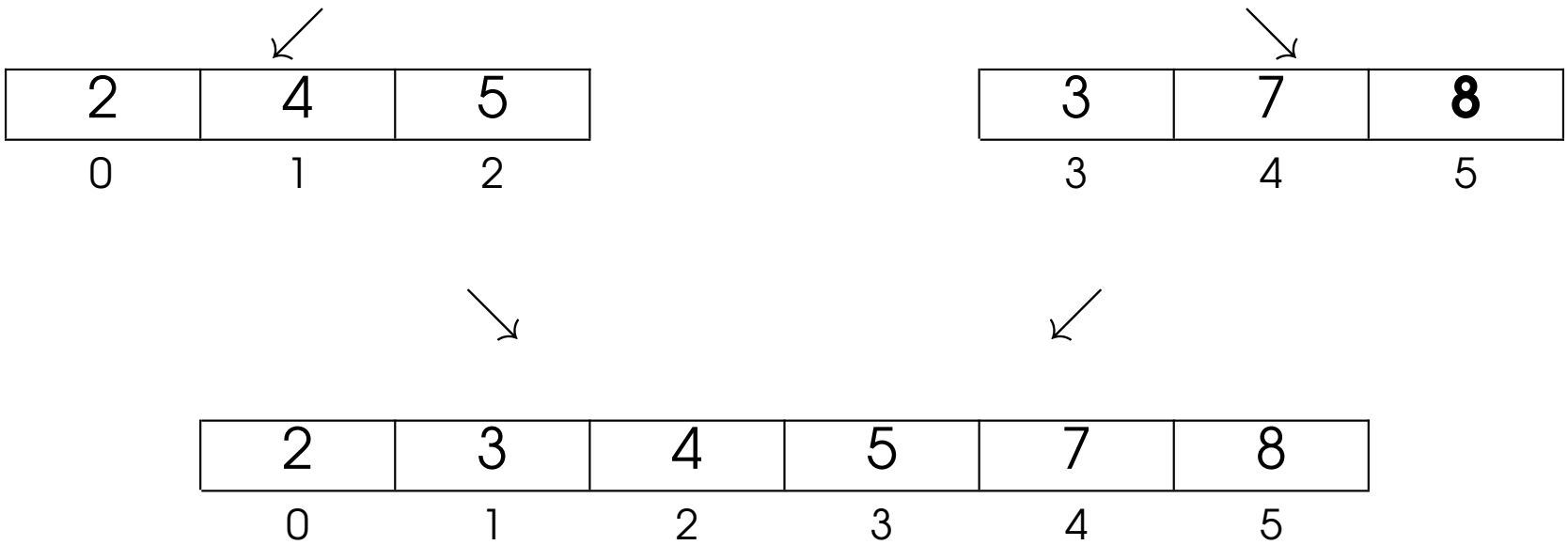
Mergesort

- Princípio de funcionamento
  - Etapa de conquista



Mergesort

- Princípio de funcionamento
  - Etapa de conquista



Mergesort

- Etapa de conquista
  - Procedimento intercalar

```
void intercalar(int S(), int E(), int ini, int meio, int fim) {
        int i = ini, j = meio + 1, k = ini;
        while(i <= meio && j <= fim) {
                if(E(i) <= E(j)) S(k++) = E(i++);
                else S(k++) = E(j++);
        }
        if(i > meio) copiar(&S(k), &E(j), fim - j + 1);
        else copiar(&S(k), &E(i), meio - i + 1);
        copiar(&E(ini), &S(ini), fim - ini + 1);
}
```

Mergesort

- Análise de complexidade
  - Caso base $T(1) = 1$
    - O vetor só possui um único elemento
  - Recorrência $T(n) = 2T(n/2) + n$
    - Dividindo vetor em dois
    - Realizando conquista de custo n

Mergesort

- Análise de complexidade
  - Resolvendo recorrência

$$T(n) = 2\left[2T\left(\frac{n}{4}\right) + \frac{n}{2}\right] + n = 4T\left(\frac{n}{4}\right) + 2n$$

$$T(n) = 4\left[2T\left(\frac{n}{8}\right) + \frac{n}{4}\right] + 2n = 8T\left(\frac{n}{8}\right) + 3n$$

$$T(n) = 8\left[2T\left(\frac{n}{16}\right) + \frac{n}{8}\right] + 3n = 16T\left(\frac{n}{16}\right) + 4n$$

$$T(n) = 2^k T\left(\frac{n}{2^k}\right) + kn$$

Mergesort

- Análise de complexidade
  - Resolvendo recorrência

$$T(n) = 2^k T\left(\frac{n}{2^k}\right) + kn$$

$$T(1) = 1 \longrightarrow \frac{n}{2^k} = 1 \longrightarrow k = \log_2 n$$

$$T(n) = 2^{\log_2 n} T(1) + n \log_2 n \longrightarrow T(n) = n + n \log_2 n$$

$$T(n) = O(n \log_2 n)$$

Mergesort

- Análise de complexidade
  - Espaço $O(\log n + n) = O(n)$
  - Tempo $O(n \log_2 n)$
  - É estável

Mergesort

- Aplicações
  - Paralelismo
  - Dispositivos de acesso sequencial
  - Grande volume de dados
  - ...

Exemplo

- Considerando o algoritmo de ordenação Mergesort, ordene a sequência abaixo
  - Sequência 23, 32, 54, 92, 74, 23, 1, 43, 63,12
  - Critério crescente de ordenação
  - Execute passo a passo

Exercício

- A empresa de automação portuária Poxim Tech está desenvolvendo um sistema para movimentação automatizada dos contêineres de carga de origem internacional no Porto de Sergipe para maximizar a eficiência da fiscalização aduaneira
  - Todos os contêineres possuem um cadastro eletrônico contendo informações sobre o código do contêiner, o CNPJ da empresa importadora e o peso líquido em quilos da carga
  - A inspeção dos contêineres é realizada sempre que existe alguma divergência entre as informações cadastradas, como o CNPJ informado ou a diferença percentual maior do que 10% no peso líquido
  - Na triagem dos contêineres são fiscalizados os contêineres com a seguinte ordem de prioridade:
    1. Divergência de CNPJ
    2. Maior diferença percentual de peso líquido

Exercício

- Formato de arquivo de entrada
  - [#*n contêineres cadastrados*]
  - [*Código* 1] [*CNPJ* 1] [*Peso* 1]
  - . . .
  - [*Código n*] [*CNPJ n*] [*Peso n*]
  - [#*m contêineres selecionados*]
  - [*Código* 1] [*CNPJ* 1] [*Peso* 1]
  - . . .
  - [*Código m*] [*CNPJ m*] [*Peso m*]

```
6
QOZJ7913219 34.699.211/9365-11 13822
FCCU4584578 50.503.434/5731-28 16022
KTAJ0603546 20.500.522/6013-58 25279
ZYHU3978783 43.172.263/4442-14 24543
IKQZ7582839 51.743.446/1183-18 12160
HAAZ0273059 25.699.428/4746-79 16644
5
ZYHU3978783 43.172.263/4442-14 29765
IKQZ7582839 51.743.446/1113-18 18501
KTAJ0603546 20.500.522/6113-58 17842
QOZJ7913219 34.699.211/9365-11 16722
FCCU4584578 50.503.434/5731-28 16398
```

Exercício

- Formato de arquivo de saída
  - Sequência de fiscalização dos contêineres do navio, com a causa da triagem e seguindo a ordem de cadastramento dos contêineres

```
KTAJ0603546: 20.500.522/6013-58<->20.500.522/6113-58
IKQZ7582839: 51.743.446/1183-18<->51.743.446/1113-18
QOZJ7913219: 2900kg (21%)
ZYHU3978783: 5222kg (21%)
```