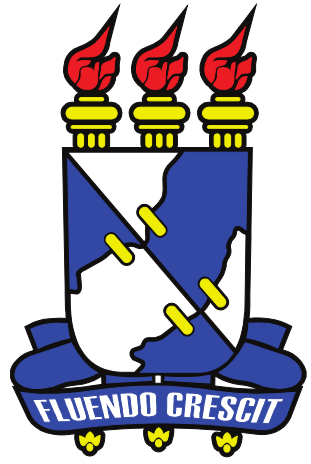


Estrutura de Dados I

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

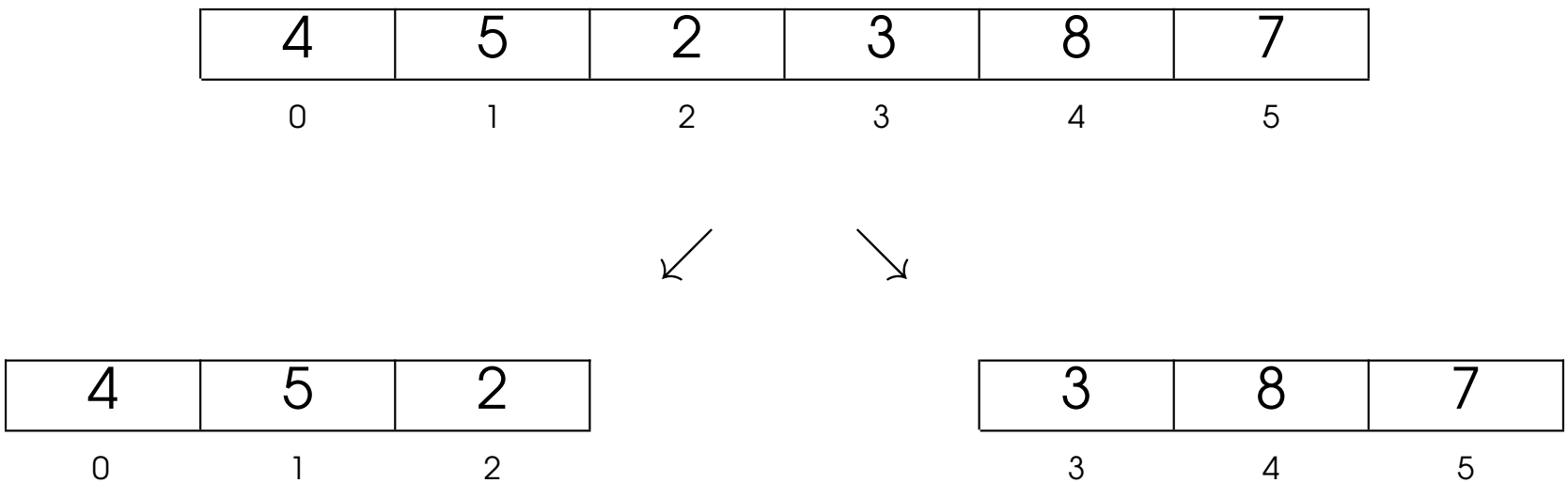
- Vantagens
  - Paralelismo
    - Problema é dividido em partes que podem ser resolvidas separadamente
  - Eficiência algorítmica
    - Complexidade $O(n \log n)$
  - Acesso a memória mais eficiente
    - Dados cabem na memória cache
  - Controle de arredondamento mais preciso
    - Os resultados são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

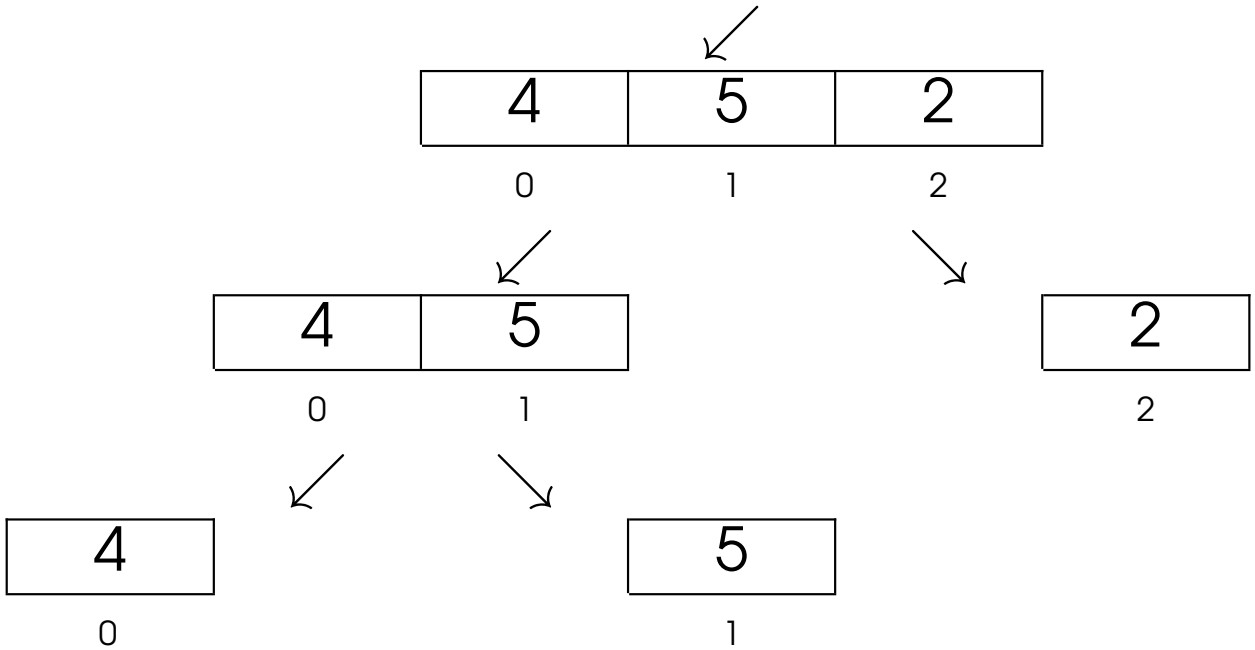
Mergesort

- Princípio de funcionamento
  - Etapa de divisão



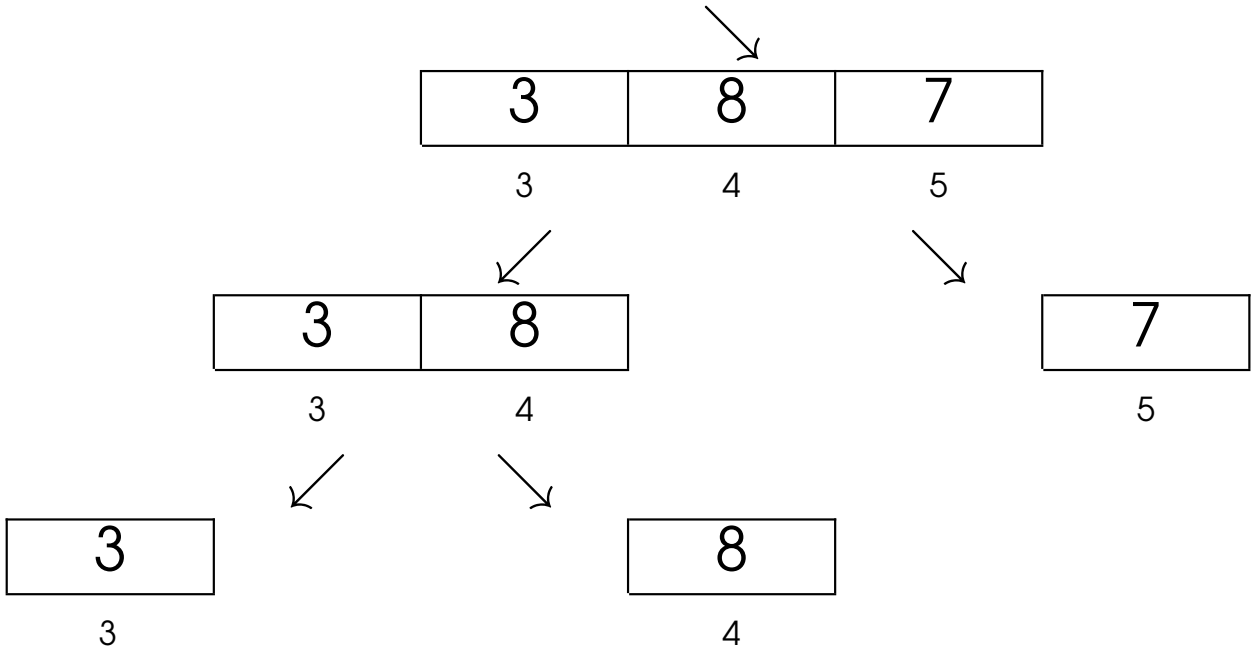
Mergesort

- Princípio de funcionamento
  - Etapa de divisão



Mergesort

- Princípio de funcionamento
  - Etapa de divisão



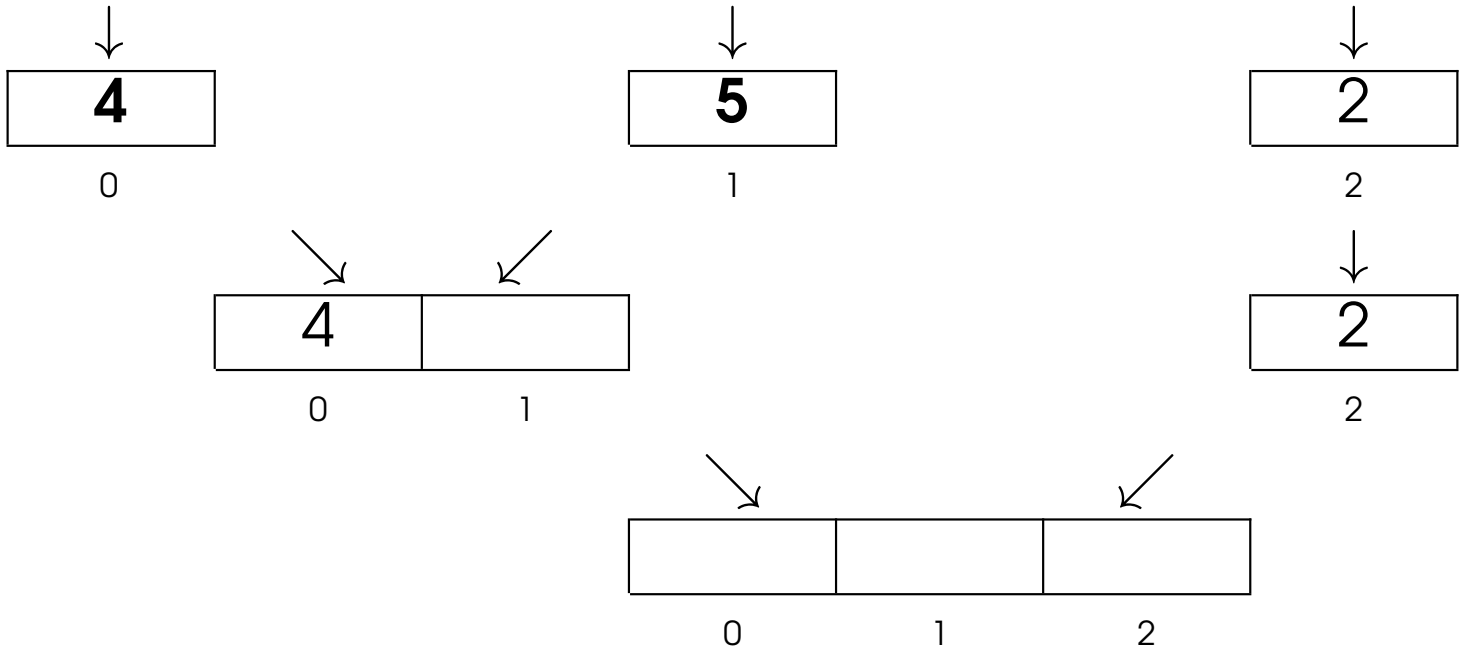
Mergesort

- Etapa de divisão
  - Procedimento mergesort

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

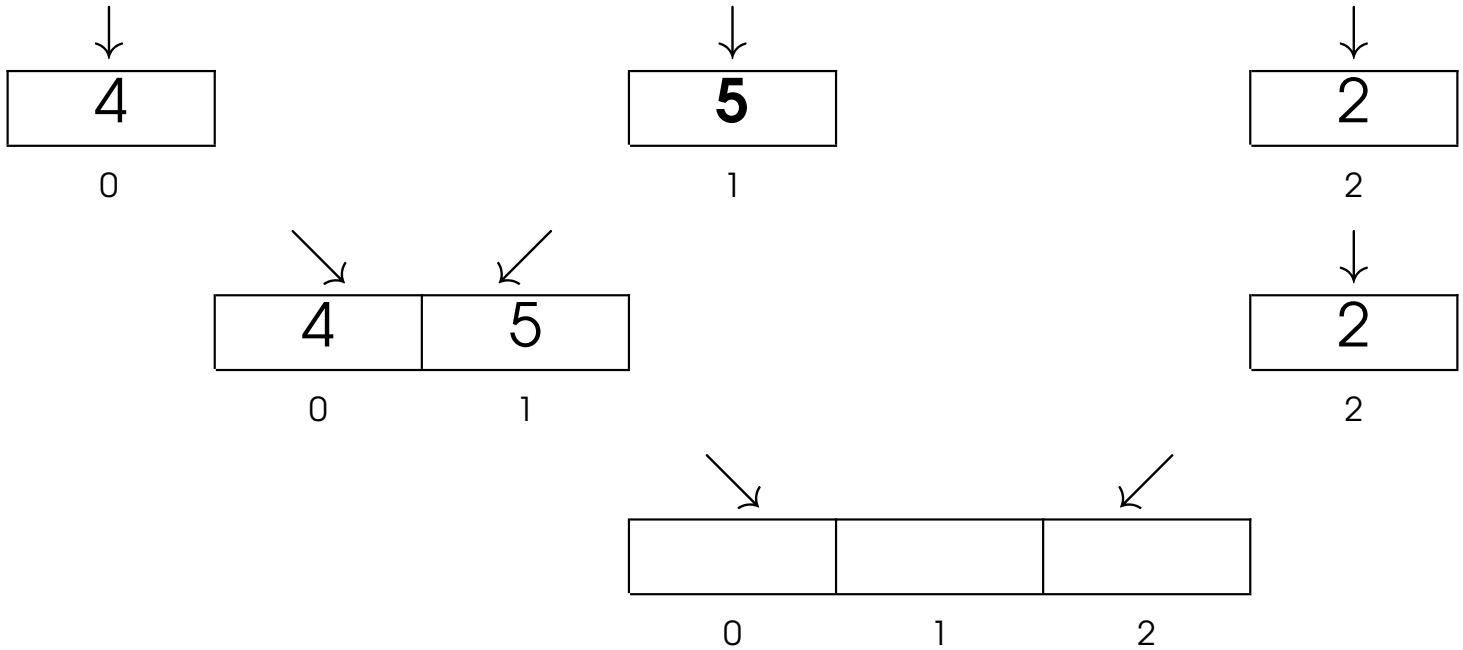
Mergesort

- Princípio de funcionamento
  - Etapa de conquista



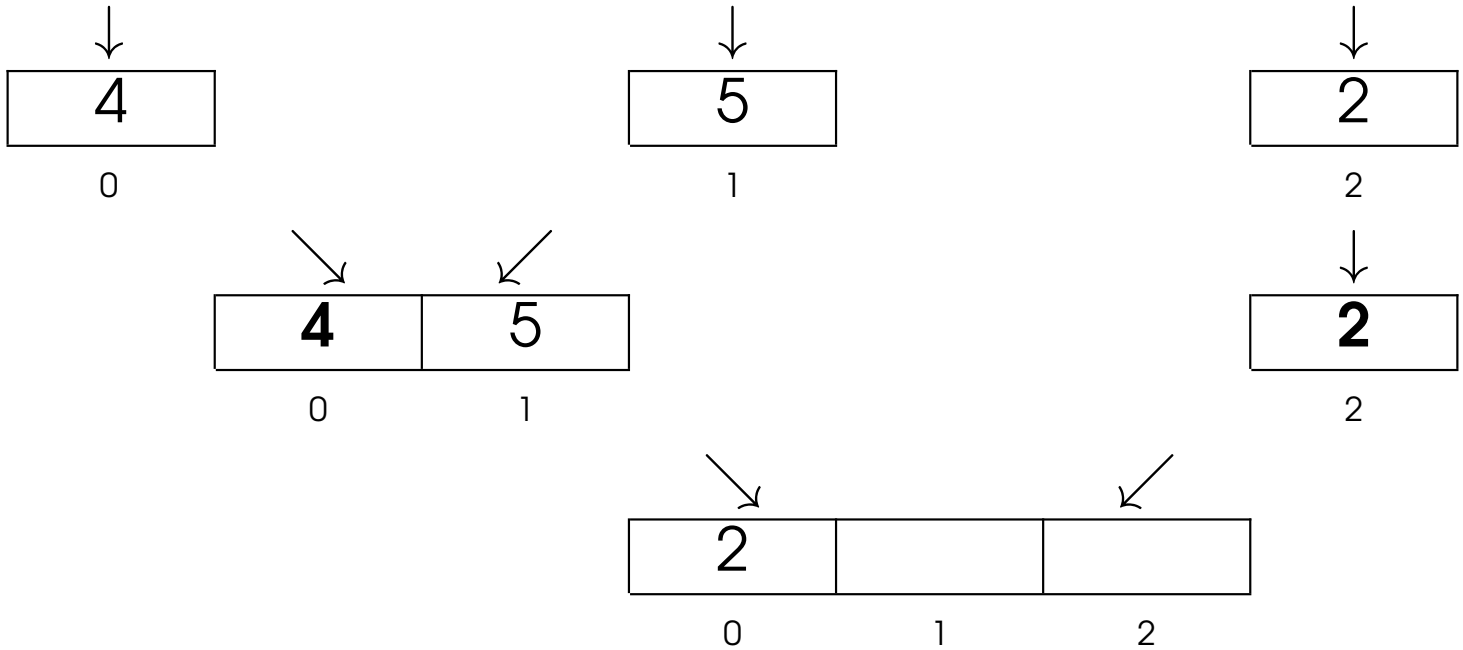
Mergesort

- Princípio de funcionamento
  - Etapa de conquista



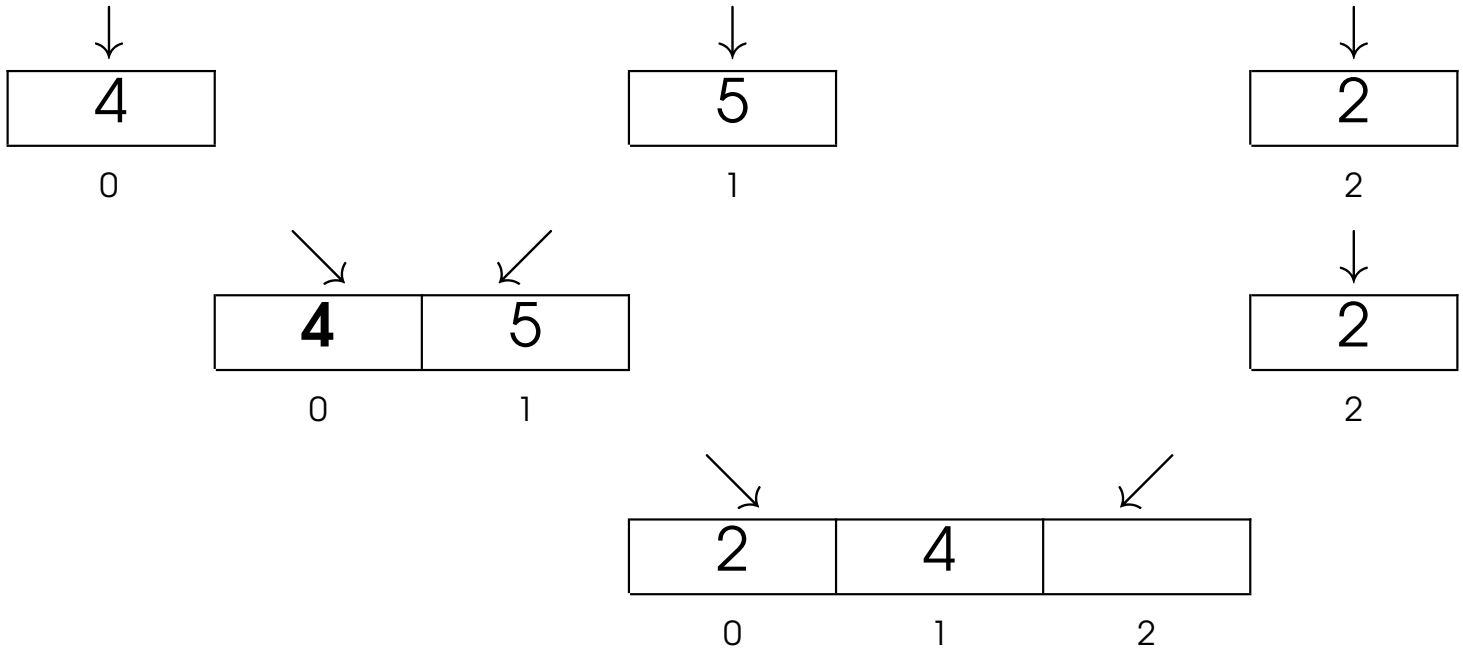
Mergesort

- Princípio de funcionamento
  - Etapa de conquista



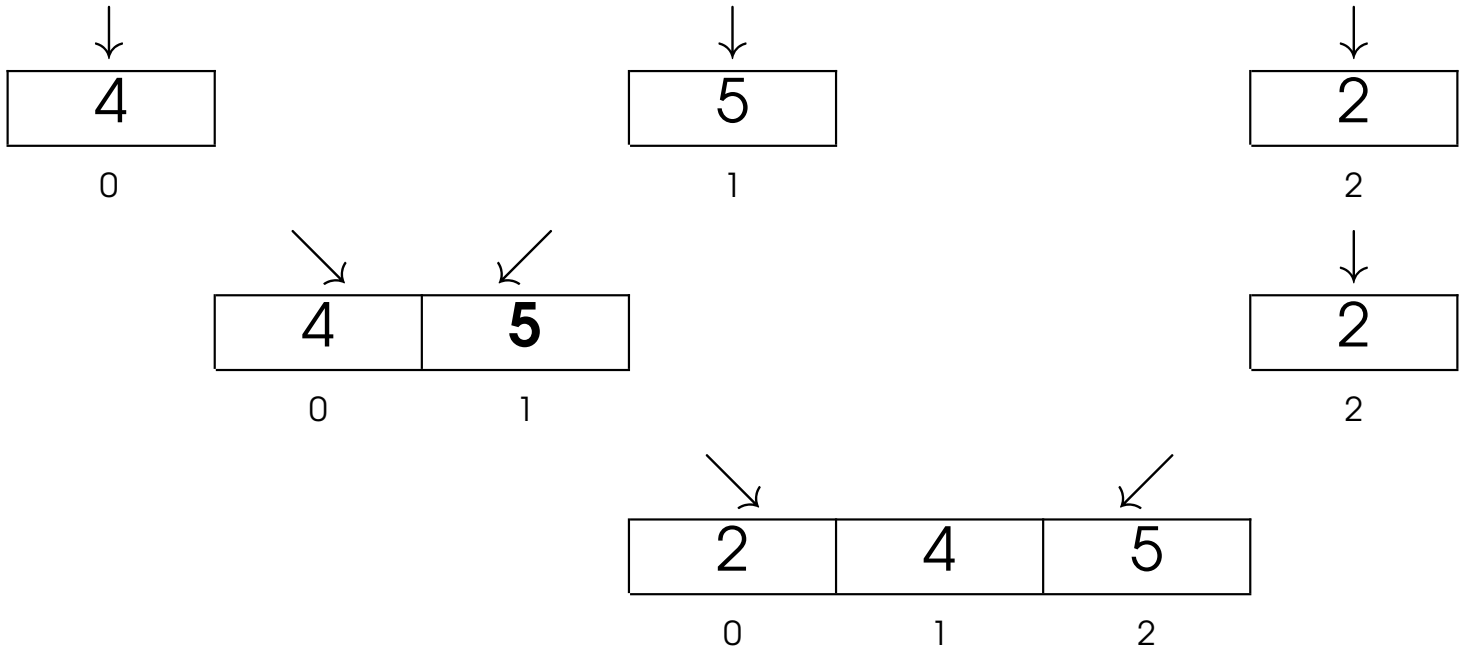
Mergesort

- Princípio de funcionamento
  - Etapa de conquista



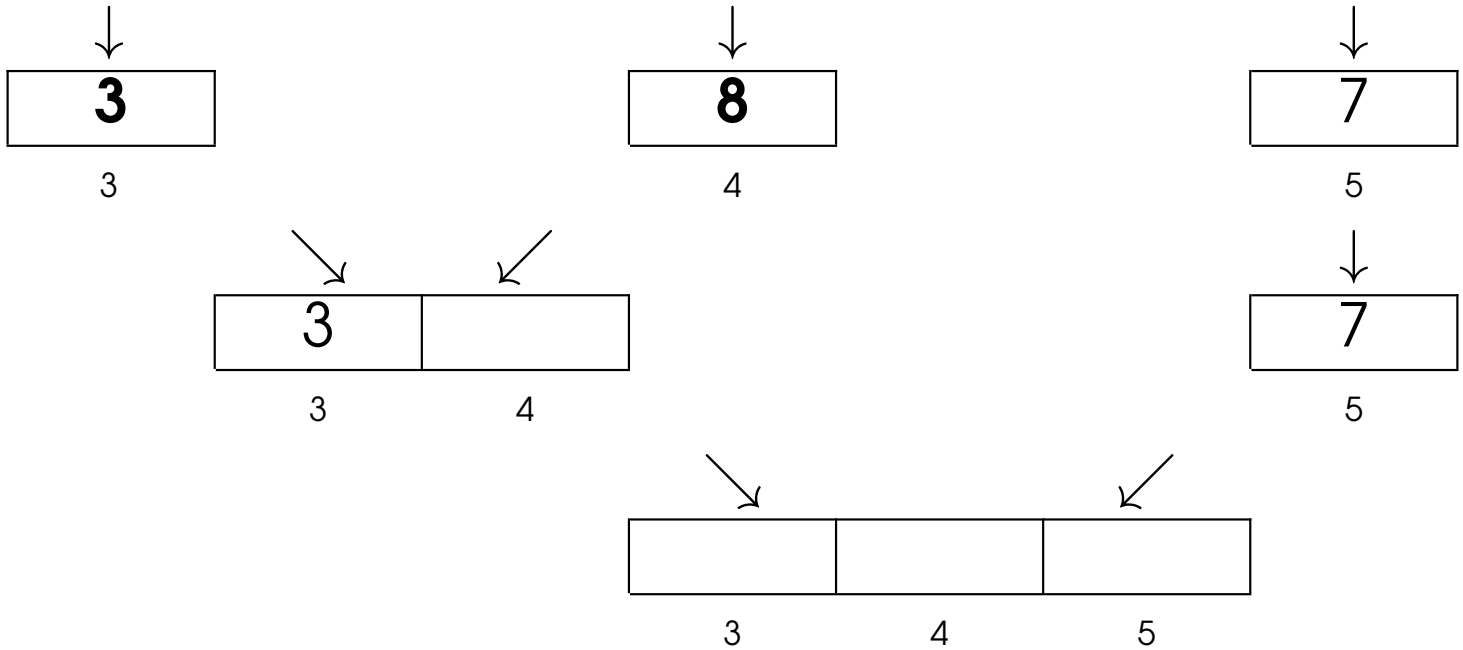
Mergesort

- Princípio de funcionamento
  - Etapa de conquista



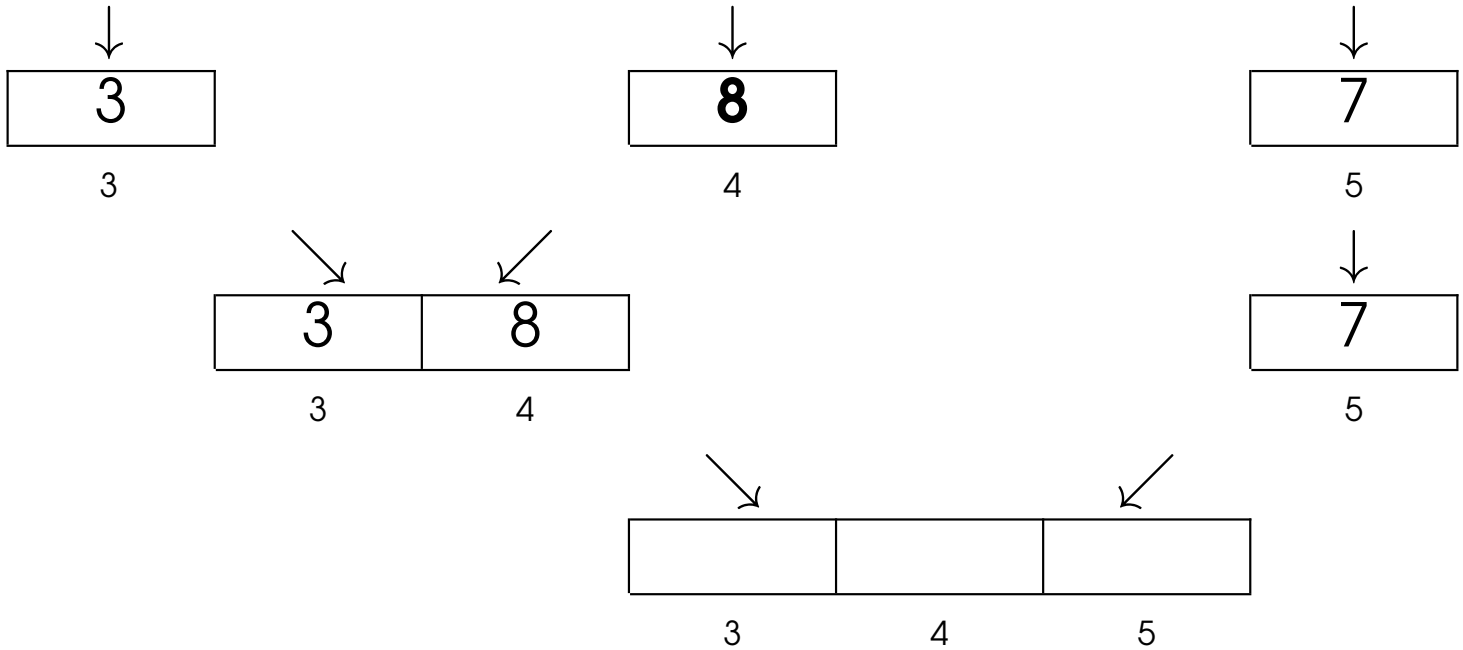
Mergesort

- Princípio de funcionamento
  - Etapa de conquista



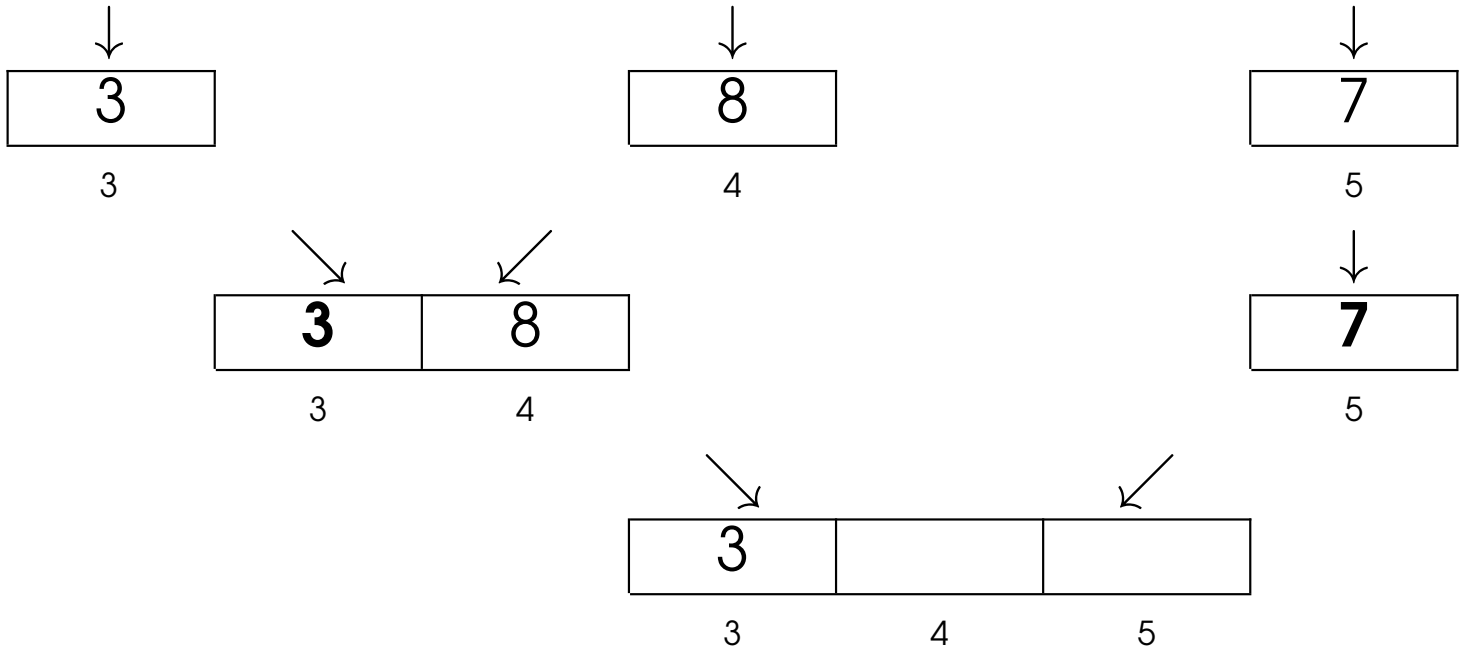
Mergesort

- Princípio de funcionamento
  - Etapa de conquista



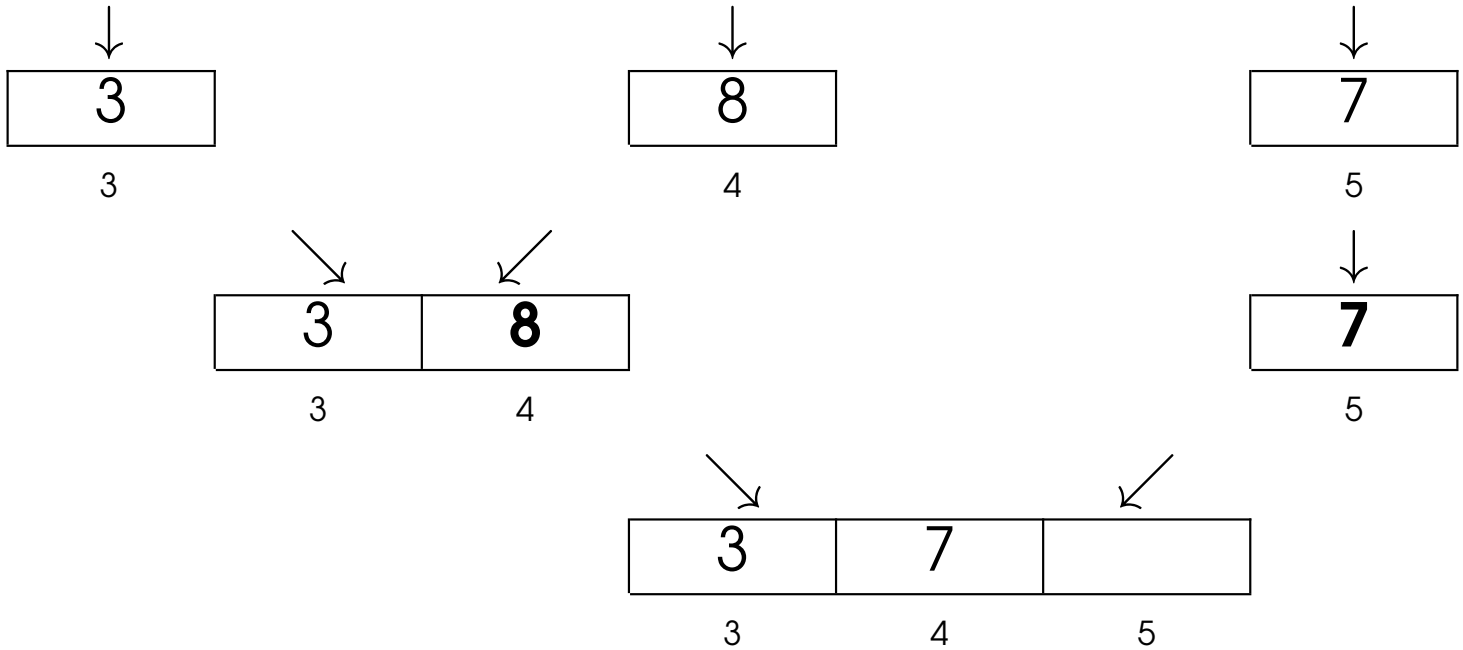
Mergesort

- Princípio de funcionamento
  - Etapa de conquista



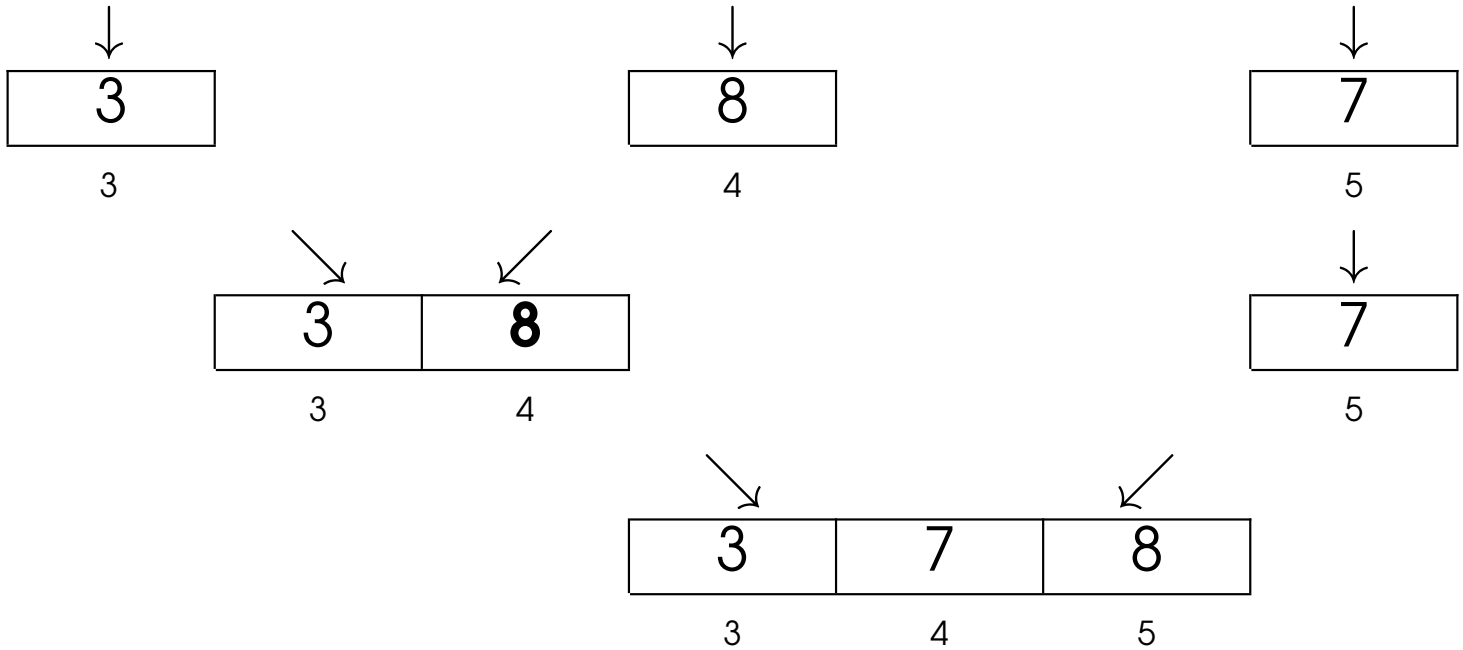
Mergesort

- Princípio de funcionamento
  - Etapa de conquista



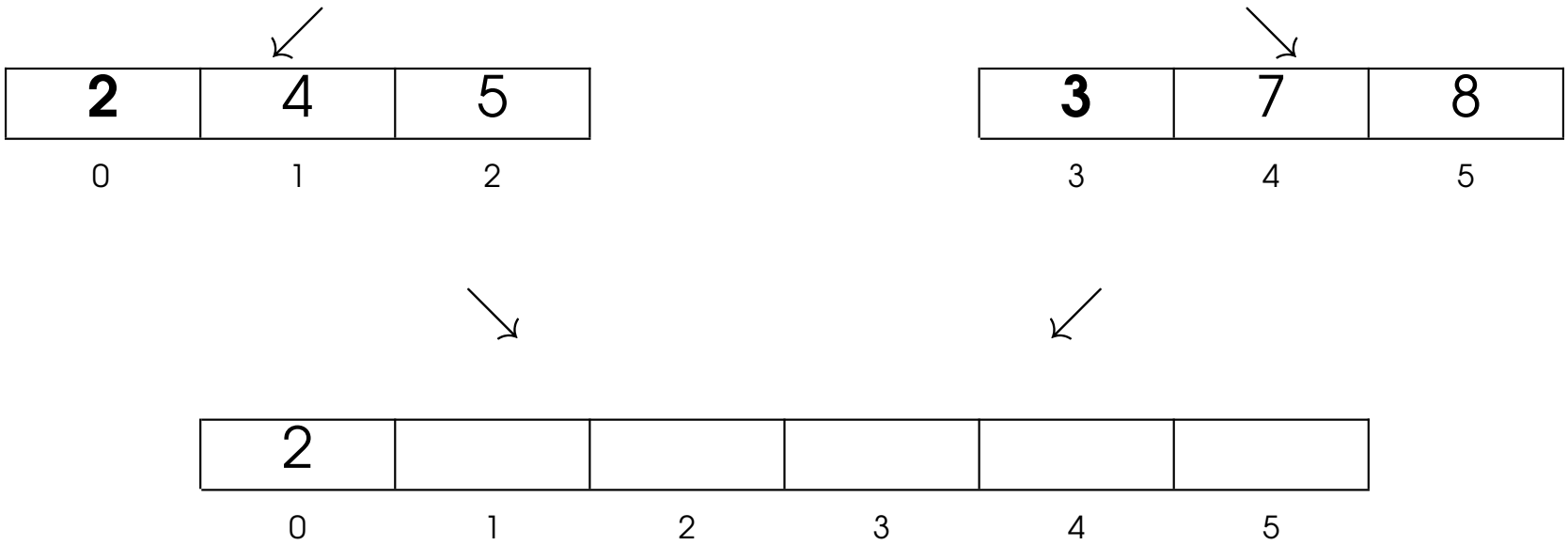
Mergesort

- Princípio de funcionamento
  - Etapa de conquista



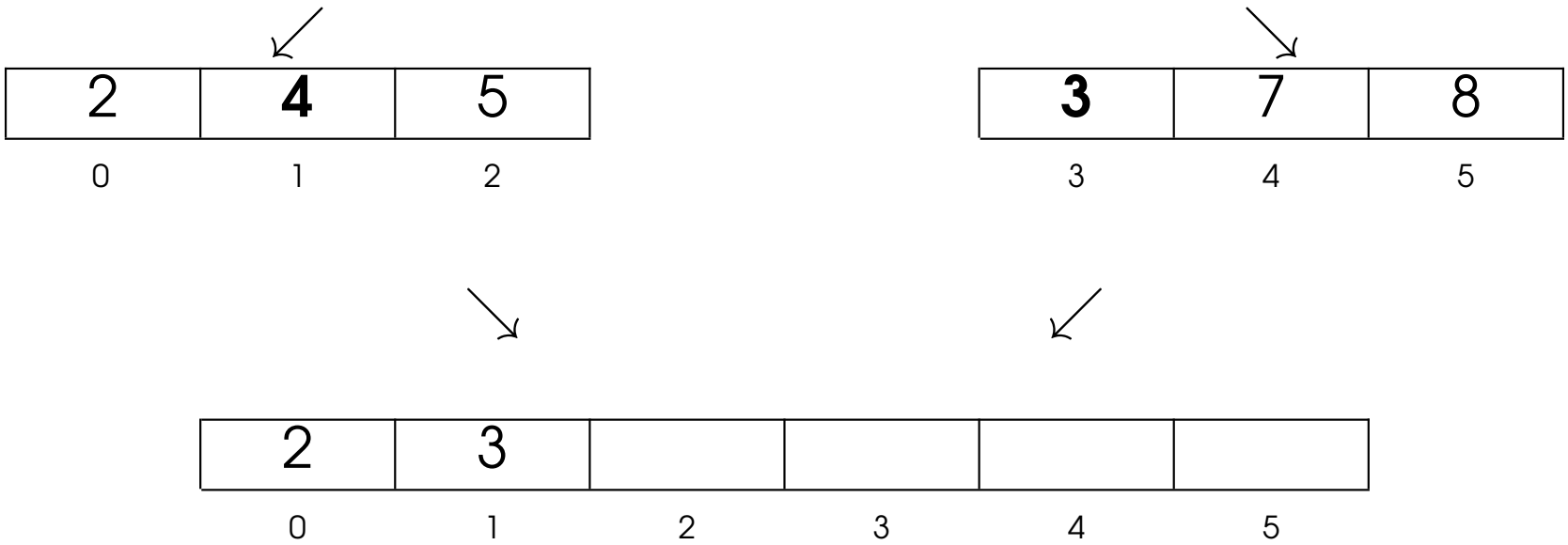
Mergesort

- Princípio de funcionamento
  - Etapa de conquista



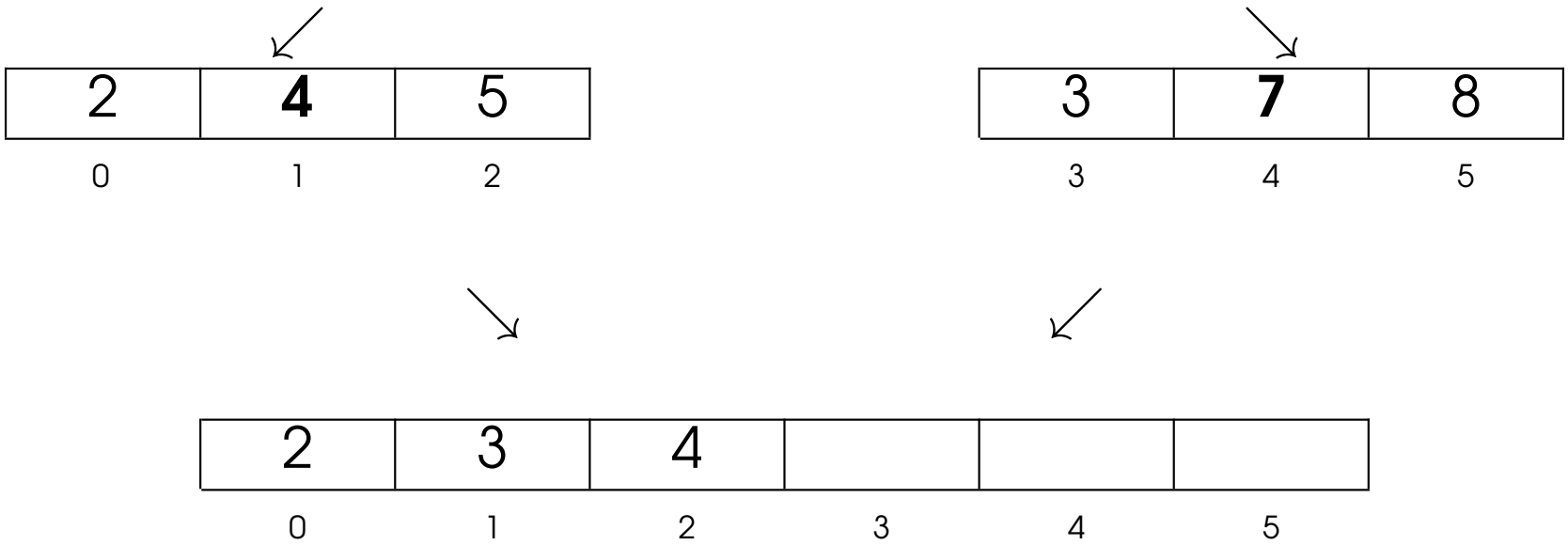
Mergesort

- Princípio de funcionamento
  - Etapa de conquista



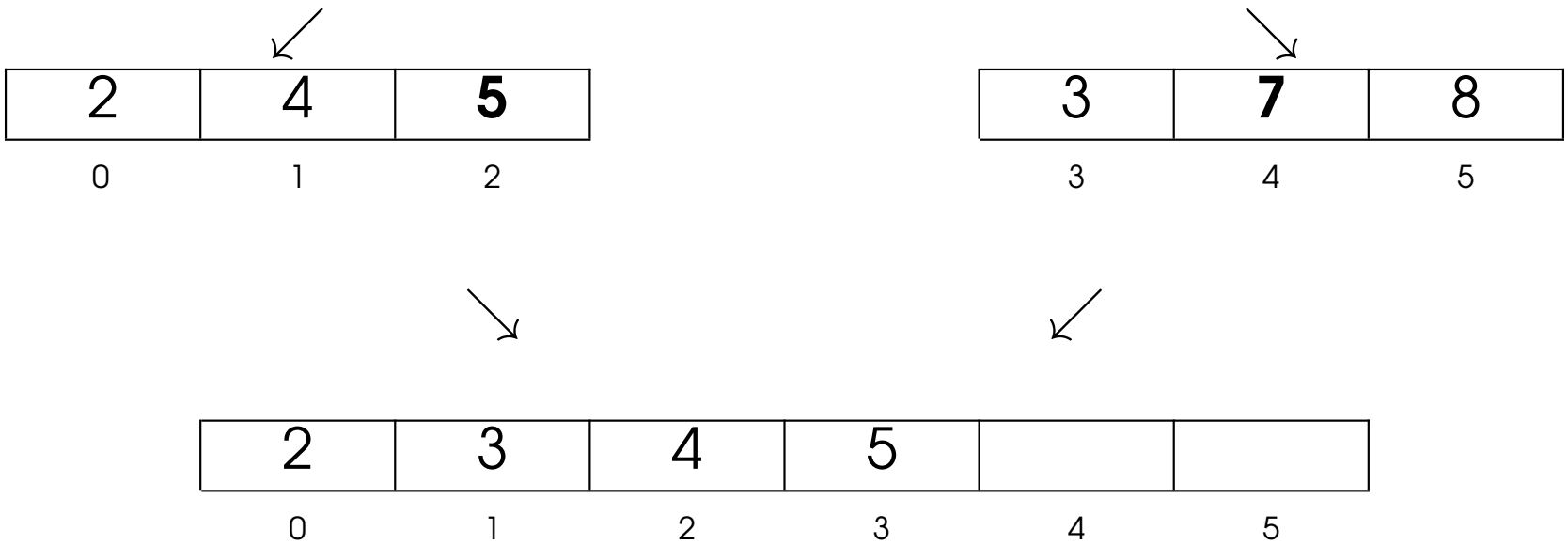
Mergesort

- Princípio de funcionamento
  - Etapa de conquista



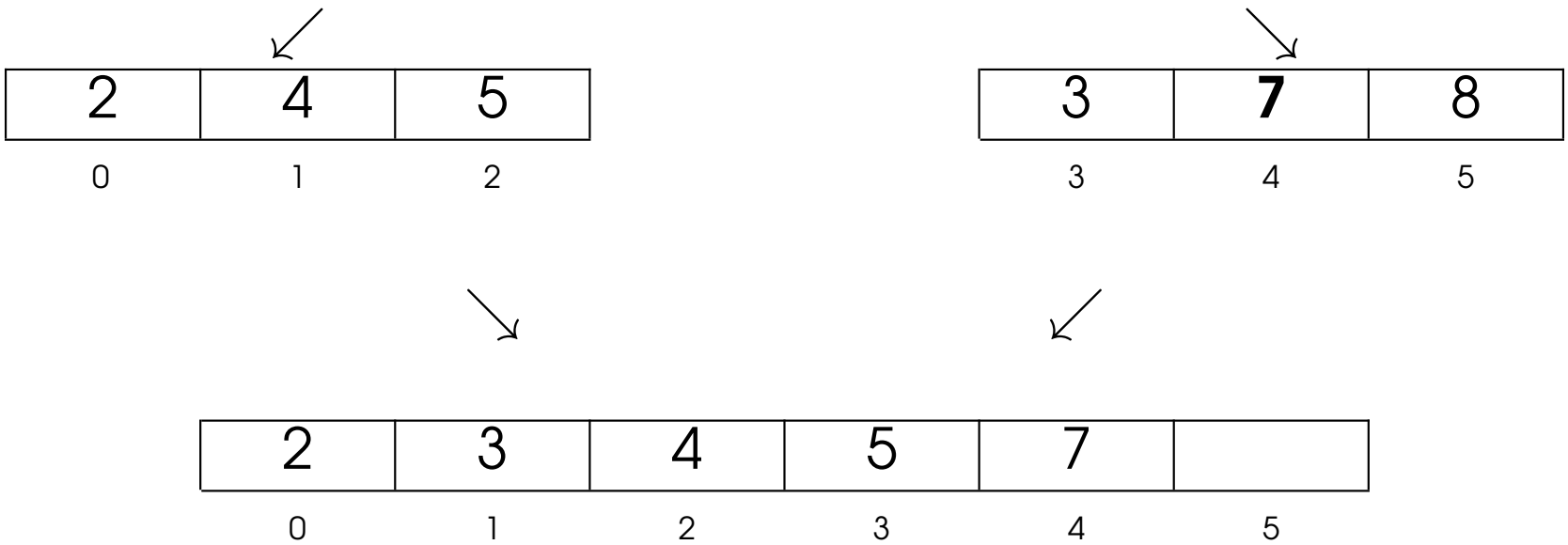
Mergesort

- Princípio de funcionamento
  - Etapa de conquista



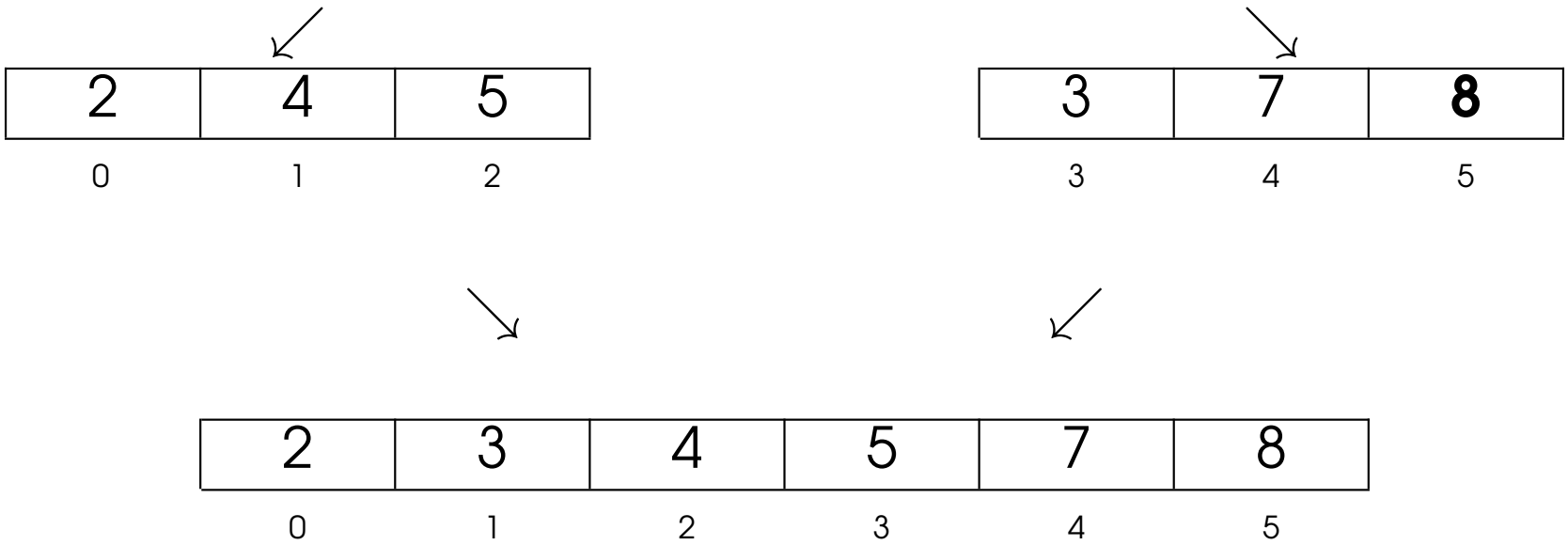
Mergesort

- 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 $\Theta(n \log_2 n)$

Mergesort

- Aplicações
  - Ordenação estável
  - Execução paralela
  - Dispositivos de acesso sequencial
  - ...

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 tecnologia Poxim Tech está desenvolvendo um sistema para controle de carregamento de veículos de transporte da Merge Express com o objetivo de otimizar o acesso aos pacotes durante as entregas
  - Todos os pacotes são classificados levando em conta 3 níveis de prioridade: ECxxxxxxxxxBR (mínima), PTxxxxxxxxxBR (média) e XPxxxxxxxxxBR (máxima), onde os caracteres 'x' são substituídos por números
  - Por conta da classificação de prioridade, nenhum pacote com menor nível deve ser entregue antes de um que possui maior prioridade de entrega e o peso dos pacotes nunca excederá a capacidade total de carga da empresa
  - Deve ser gerada uma sequência de carregamento para cada veículo que permita atender à prioridade dos pacotes e garantir que pacotes com mesma prioridade sejam tratados de acordo com sua ordem de postagem

Exercício

- Formato de arquivo de entrada
  - [*Quantidade de veiculos*] [*Capacidade de carga*]
  - [*Quantidade de pacotes*]
  - [C_0] [P_0]
  - [C_1] [P_1]
  - $\vdots$
  - [C_{n-1}] [P_{n-1}]

```
2 10
6
EC123456789BR 3
XP000111222BR 1
PT789012567BR 4
EC765839726BR 10
PT929374817BR 2
PT830173947BR 7
```

Exercício

- Formato de arquivo de saída
  - Sequências de carregamento para cada veículo

```
[V0] EC123456789BR PT789012567BR XP000111222BR
[V1] EC765839726BR
[V0] PT830173947BR PT929374817BR
```