Engenharia de Computação

Pesquisa e Classificação de Dados

Aula 5 – MergeSort

Prof. Muriel de Souza Godoi
muriel@utfpr.edu.br

MergeSort

- Também conhecido como **ordenação por intercalação**
  - Algoritmo recursivo que usa a ideia de dividir para conquistar para ordenar os dados
    - Parte do princípio de que é mais fácil ordenar um conjunto com poucos dados do que um com muitos
  - O algoritmo **divide os dados** em conjuntos cada vez menores para depois ordená-los e combiná-los por meio de intercalação (*merge*)

MergeSort - Funcionamento

- Divide, recursivamente, o vetor em duas partes
  - Continua até cada parte ter apenas um elemento
- Em seguida, combina dois vetores de forma a obter um vetor maior e ordenado
  - A combinação é feita intercalando os elementos de acordo com o sentido da ordenação (crescente ou decrescente)
- Este processo se repete até que exista apenas um vetor

Merge Sort - Pseudocódigo

- O algoritmo usa **2 funções**
  - **mergeSort**: divide os dados em vetores cada vez menores
  - **merge**: intercala os elementos de forma ordenada em um vetor maior

Merge Sort - Pseudocódigo

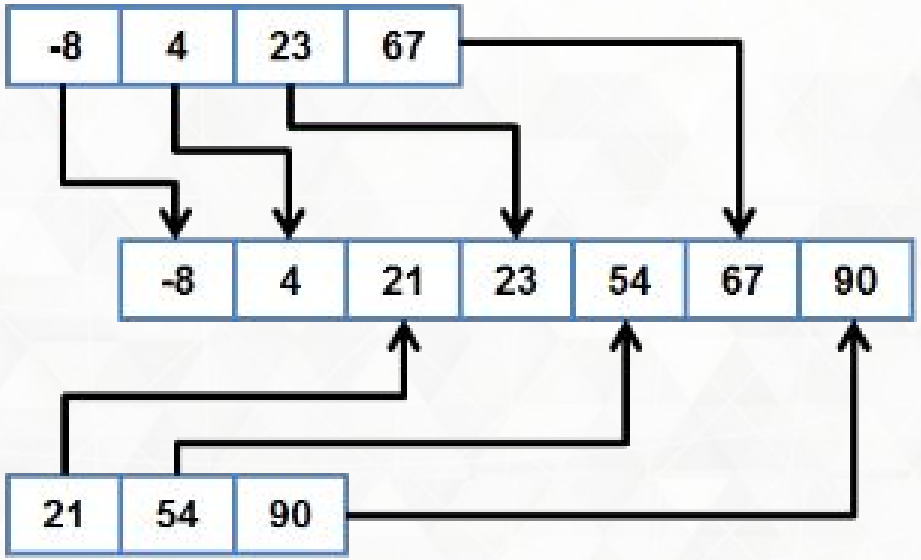
- **mergeSort(v, inicio, fim)**
  - Se inicio < fim então:
    - meio = ((inicio + fim) / 2)
    - mergeSort(v, inicio, meio)
    - mergeSort(v, meio + 1, fim)
    - merge(V, inicio, meio, fim)

Merge Sort - Pseudocódigo

- **merge(v, inicio, meio, fim)**
  - Aloca um vetor temporário
  - p1 ← inicio
  - p2 ← meio + 1
  - **Enquanto** p1 < meio **E** p2 < fim **faça**:
    - Copia para o vetor temp. o menor valor entre p1 e p2
    - Incrementa o contador correspondente
  - **Se** p1 == meio **então:**
    - Copia o que sobrou a partir de p2
  - **Senão**
    - Copia o que sobrou em p1
  - Copia tudo do vetor temporário para o vetor original

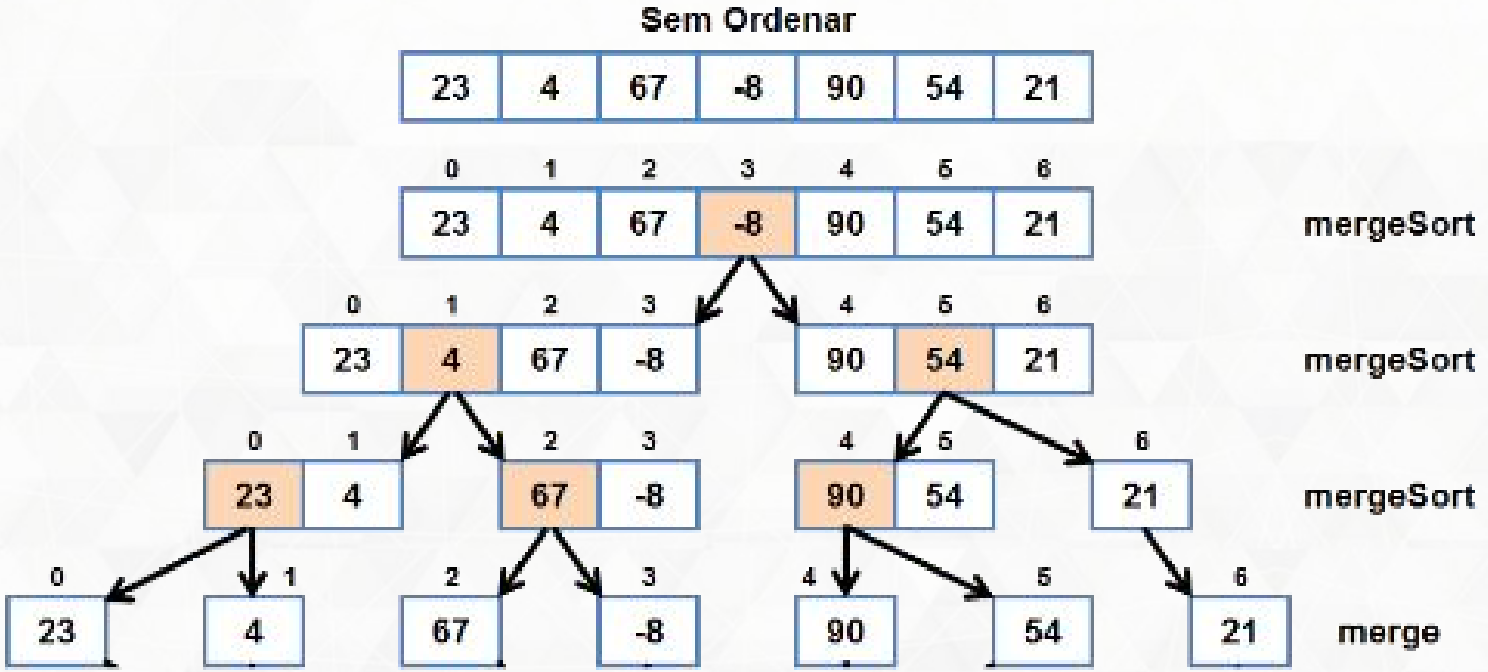
Merge Sort – Passo a passo

- Função **merge**
  - **Intercala** os dados de forma ordenada em um vetor maior
  - Utiliza um vetor auxiliar



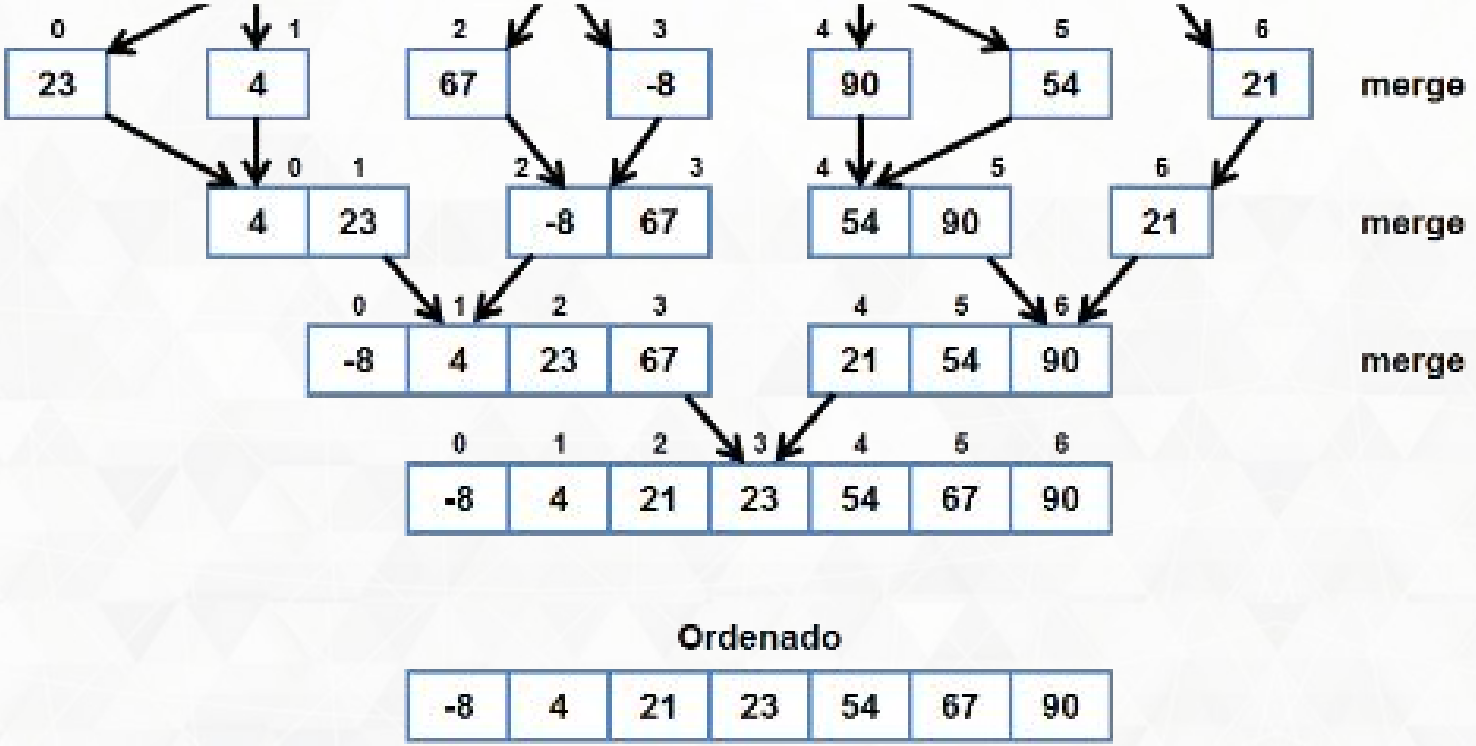
Merge Sort – Passo a passo

- Função **mergeSort**
  - **Divide** recursivamente o vetor até ter **n** vetores de 1 elemento cada



Merge Sort – Passo a passo

- **Intercala** os vetores até obter um único vetor de **n** elementos



MergeSort

- Vantagens
  - Estável: não altera a ordem dos dados iguais
- Desvantagens:
  - Possui um gasto extra de espaço de memória em relação aos demais métodos de ordenação
    - Ele cria uma cópia do vetor para cada chamada recursiva
    - Em outra abordagem, é possível utilizar um único vetor auxiliar ao longo de toda a sua execução

MergeSort - Complexidade

- Considerando um vetor com **n** elementos, o tempo de execução é de ordem **O(n log n)** em todos os casos
- Sua eficiência não depende da ordem inicial dos elementos
  - No pior caso, realiza cerca de 39% menos comparações do que o QuickSort no seu caso médio
  - Já no seu melhor caso, o MergeSort realiza cerca de metade do número de iterações do seu pior caso

Exercício

- Implemente o **MergeSort** em C considerando as seguintes assinaturas de função
  - **Função 1: mergeSort**

```
/**
 * \brief Ordena o vetor usando MergeSort
 *
 * \param v vetor a ser ordenado
 * \param inicio índice do início do vetor
 * \param fim índice do final do vetor
 *
 * Ordena o vetor usando o método MergeSort
 * Esse algoritmo tem um comportamento assintótico O(n log n)
 */
void mergeSort(int *v, int inicio, int fim);
```

Exercício

- **Função 2: merge**

```
/**
 * \brief Intercala dois vetores
 *
 * \param v vetor a ser intercalado
 * \param inicio índice do início do vetor
 * \param meio índice do último elemento do primeiro vetor
 * \param fim índice do final do segundo vetor
 *
 * Intercala um vetor dividido em duas partes de maneira
 * ordenada
 */
void merge(int *v, int inicio, int meio, int fim);
```