Algoritmos e Estruturas de Dados

FECAP

São Paulo

2025

Vitor dos Santos 23025502

Caio Gomes 24026876

Estrutura de Dados em Projetos Educacionais

Ordenação de Tarefas em Projetos Educacionais Usando MergeSort

No contexto de projetos educacionais, é essencial organizar as tarefas de maneira eficiente para assegurar que o cronograma seja cumprido. Um exemplo prático seria classificar as tarefas de aulas, reuniões e eventos por data ou prioridade. Para isso, utilizaremos o algoritmo MergeSort para ordenar as tarefas de forma eficaz.

Aqui está uma nova classe chamada Tarefa, que armazena as informações de uma tarefa dentro do projeto educacional.

Algoritmo de Ordenação: MergeSort

Código:

```
using System;


public class Tarefa
{
   private int _id;
   private string _titulo;
   private string _descricao;
   private DateTime _dataEntrega;
   private string _prioridade;
   private string _status;

   // Construtor da classe Tarefa
   public Tarefa(int id, string titulo, string descricao, string dataEntrega, string prioridade, string status)
   {
```

```
    this._id = id;
    this._titulo = titulo;
    this._descricao = descricao;
    this._dataEntrega = DateTime.Parse(dataEntrega); // Converte string para DateTime
    this._prioridade = prioridade;
    this._status = status;
}

// Getters e Setters Manuais
public int Id
{
    get { return _id; }
    set { _id = value; }
}

public string Titulo
{
    get { return _titulo; }
    set { _titulo = value; }
}

public string Descricao
{
    get { return _descricao; }
    set { _descricao = value; }
}

public DateTime DataEntrega
{
```

```
        get { return _dataEntrega; }
        set { _dataEntrega = value; }
    }

    public string Prioridade
    {
        get { return _prioridade; }
        set { _prioridade = value; }
    }

    public string Status
    {
        get { return _status; }
        set { _status = value; }
    }
}

class Program
{
    // Função para comparar duas tarefas pela data de entrega
    static int CompareTarefas(Tarefa tarefa1, Tarefa tarefa2)
    {
        return tarefa1.DataEntrega.CompareTo(tarefa2.DataEntrega); // Compara as datas
    }

    // Função MergeSort para ordenar tarefas
    static Tarefa[] MergeSort(Tarefa[] tarefas)
    {
        if (tarefas.Length <= 1)
```

```
        return tarefas; // Caso base: lista com 1 ou nenhum item

    int middle = tarefas.Length / 2;
    Tarefa[] left = new Tarefa[middle];
    Tarefa[] right = new Tarefa[tarefas.Length - middle];

    Array.Copy(tarefas, 0, left, 0, middle); // Copia a primeira metade
    Array.Copy(tarefas, middle, right, 0, tarefas.Length - middle); // Copia a segunda metade

    left = MergeSort(left); // Ordena a primeira metade
    right = MergeSort(right); // Ordena a segunda metade

    return Merge(left, right); // Combina as duas metades ordenadas
}

// Função para combinar duas metades ordenadas
static Tarefa[] Merge(Tarefa[] left, Tarefa[] right)
{
    Tarefa[] sorted = new Tarefa[left.Length + right.Length];
    int i = 0, j = 0, k = 0;

    while (i < left.Length && j < right.Length)
    {
        if (CompareTarefas(left[i], right[j]) < 0)
        {
            sorted[k++] = left[i++];
        }
        else
        {
```

```
            sorted[k++] = right[j++];
        }
    }

    // Adiciona os elementos restantes das listas left e right
    while (i < left.Length)
        sorted[k++] = left[i++];

    while (j < right.Length)
        sorted[k++] = right[j++];

    return sorted;
}

static void Main(string[] args)
{
    // Criação de algumas tarefas de exemplo
    Tarefa[] tarefas = new Tarefa[]
    {
        new Tarefa(1, "Aula de Biologia", "Ecossistemas", "2025-03-15T09:00:00", "Alta", "Em andamento"),
        new Tarefa(2, "Reunião de Coordenação", "Reunião com professores", "2025-03-12T14:00:00", "Média", "Pendente"),
        new Tarefa(3, "Aula de Química", "Reações Químicas", "2025-03-14T11:00:00", "Baixa", "Concluída")
    };

    // Ordena as tarefas pela data de entrega
    Tarefa[] tarefasOrdenadas = MergeSort(tarefas);
```

```
        // Exibe as tarefas ordenadas
        foreach (Tarefa tarefa in tarefasOrdenadas)
        {
            Console.WriteLine($"Tarefa: {tarefa.Titulo}, Data de Entrega: {tarefa.DataEntrega}");
        }
    }
}
```

Exemplo de Saída:

Tarefa: Reunião de Coordenação, Data de Entrega: 12/03/2025 14:00:00

Tarefa: Aula de Química, Data de Entrega: 14/03/2025 11:00:00

Tarefa: Aula de Biologia, Data de Entrega: 15/03/2025 09:00:00

Como funciona

Classe Tarefa:

A classe Tarefa representa uma tarefa dentro de um projeto educacional. Ela tem atributos como:

Id: Identificador único.

Titulo: Título da tarefa.

Descricao: Descrição da tarefa.

DataEntrega: A data limite para a tarefa.

Prioridade: A urgência da tarefa.

Status: O status da tarefa (por exemplo, "Concluída", "Pendente").

Função MergeSort:

A função MergeSort recebe um array de tarefas e ordena esse array com base na data de entrega.

Se o array tiver apenas 1 ou nenhum item, ele retorna o próprio array.

Caso contrário, divide o array em duas metades e ordena cada metade recursivamente.

As duas metades ordenadas são então combinadas usando a função Merge.

Função Merge:

A função Merge recebe duas sublistas (metades ordenadas) e as combina em uma única lista ordenada com base na comparação das datas de entrega das tarefas.

Exibição dos Resultados:

Após a ordenação, o código exibe as tarefas no console com seus títulos e datas de entrega ordenadas.