Grupo: Felipe Padilha, Guilherme Zamboni, Jade Evelyn, Lara Chiodini, Maykon Douglas e Rafael Nunes

Arquitetura do Projeto:

Arquivo	Função
data_structs.py	classes Team e Match
main.py	leitura do CSV, geração do summary, BSTs, AVL e ordenação
sorting.py	Insertion Sort (O n^2), Merge Sort (O n log n) e cálculo de pontos
avl.py	AVL balanceada por pontos das seleções
bst.py	Duas BSTs: ordenada por nome e ordenada por gols
matches_summary.csv	arquivo gerado ao final

Complexidade das operações:

- **Etapa 1:**

Operação	Complexidade
Criar Match	O(1)
Criar Team	O(1)
total_goals	O(1)
to_list	O(1)

- **Etapa 2:**

Operação	Complexidade
Leitura do arquivo	O(n)
Criação dos objetos Match	O(n)
Escrita do resumo	O(n)

- **Etapa 3:**

Operação	Média	Pior caso
Inserção	O(log n)	O(n)
Inorder	O(n)	O(n)

- **Etapa 4:**

Algoritmo	Melhor	Médio	Pior
Insertion Sort	O(n)	$O(n^2)$	$O(n^2)$
Merge Sort	O(n log n)	O(n log n)	O(n log n)

- **Etapa 5:**

Operação	Complexidade
Inserção	O(log n)
Rotação	O(1)
Altura da árvore	O(1)
Percurso inorder	O(n)

Comparação Teórica BST X AVL:

- **BST:**
  - Pode degradar para lista encadeada
  - Pior caso: altura = n
  - Inserção e busca podem virar O(n)
  - Vantagem: mais simples, maior velocidade média se entradas forem aleatórias

- **AVL:**
  - Garantidamente balanceada
  - Altura ≈ $\log_2(n)$
  - Inserção e busca sempre O(log n)
  - Vantagem: desempenho consistente
  - Custo extra: rotações para balancear

- **Conclusão:** Para ranking de times por pontos, a AVL garante consultas e inserções consistentes e rápidas → melhor escolha.

Descrição das Etapas:

1. **Etapa 1 - Modelagem (Team, Match) - Guilherme Zamboni**

- Implementação das Classes Team e Match, contendo: date, home_team, away_team, home_score, away_score, tournament, city, country, neutral, Método total_goals() e Método to_list().
- Tratamento de dados faltantes
- Verificação de: data válida, nome dos times, gols numéricos, país não vazio, linhas inválidas foram descartadas.

2. **Etapa 2 - Leitura do CSV e preenchimento das estruturas - Felipe Padilha**

- Leitura usando csv.DictReader
- Conversão de data usando strptime com fallback para fromisoformat
- Conversão de gols com int()
- Lista Python usada como estrutura base para armazenar os Match
- Arquivo matches_summary.csv gerado com: year,country, home_team, away_team e score.

3. **Etapa 3 - BSTs (Árvores de Busca Binária) - Rafael Nunes**

- BST ordenada pelo nome das seleções:
  Chave = nome do país
  Valor = {"team": nome, "goals": total_gols}
- BST ordenada pelo total de gols:
  Chave = gols
  Valor = {"team": nome, "goals": total_gols}

- Métodos: insert, inorder, construir_bst_por_nome, construir_bst_por_gols

4. **Etapa 4 - Ordenação - Jade Evelyn**

- Insertion Sort (O n^2): Comparações e deslocamentos, ideal para listas pequenas.
- Merge Sort (O n log n): Ordenação estável baseada em dividir para conquistar.
- Ambos usados para ordenar seleções por pontos, calculados como:

  Vitória = 3 pontos

  Empate = 1 ponto

  Derrota = 0

- Saídas geradas: Top 10 maiores pontuadores, Top 10 menores pontuadores

5. **Etapa 5 - AVL por Pontos - Maykon Douglas**

- Altura por nó
- Rotações LL, RR, LR, RL
- Inserção balanceada
- Ordenação por chave (pontos, nome)
- AVL usada para armazenar ranking completo de seleções.

6. **Etapa 6 - Geração do matches_summary.csv - Lara Chiodini**

- Arquivo contém: year,country,home_team,away_team,score

7. **Etapa 7 - Desenvolvimento do relatório Final - Lara Chiodini**