Ranking de Corridas - Documentação

Algoritmos Escolhidos

Algoritmos de Ordenação

1. Merge Sort (Ordem Crescente)

- **Complexidade**: O(n log n) no melhor, médio e pior caso
- **Estável**: Sim
- **Espaço**: O(n)

2. Quick Sort (Ordem Decrescente)

- **Complexidade**: O(n log n) médio, $O(n^2)$ pior caso
- **Estável**: Não
- **Espaço**: O(log n) médio

Justificativa da Escolha

Por que Merge Sort e Quick Sort?

1. **Eficiência**: Ambos têm complexidade O(n log n) no caso médio, adequada para 100 elementos
2. **Comparação**: Permite avaliar algoritmo estável vs não-estável
3. **Performance**: Quick Sort geralmente mais rápido na prática, Merge Sort mais consistente
4. **Diferentes características**: Merge Sort garante O(n log n), Quick Sort pode ser $O(n^2)$ no pior caso

Algoritmo de Busca

Busca Linear e Binária

- **Linear**: Para lista não ordenada, O(n)
- **Binária**: Para lista ordenada, O(log n)

Vantagens e Desvantagens

Merge Sort

Vantagens:

- Complexidade garantida O(n log n)
- Algoritmo estável
- Performance consistente

Desvantagens:

- Usa O(n) espaço extra
- Overhead de criação de arrays temporários

Quick Sort

Vantagens:

- Rápido na prática
- Ordenação in-place (menos memória)
- Boa performance média

Desvantagens:

- Pior caso $O(n^2)$
- Não é estável
- Performance varia com dados de entrada

Busca

Linear:

- Simples de implementar
- Funciona em qualquer lista
- O(n) - lenta para listas grandes

Binária:

- Muito rápida O(log n)
- Requer lista ordenada
- Mais complexa de implementar

Conclusão para o Problema

Para ranking de corridas com 100 elementos:

- **Merge Sort** é ideal quando precisamos de consistência e estabilidade
- **Quick Sort** é ideal quando precisamos de velocidade máxima
- **Busca Binária** é muito superior após ordenação
- A diferença de performance é mais perceptível com listas maiores