

Análise de Complexidade - USEI08: Search by Geographical Area

Logistics on Rails - Grupo 023

Dezembro 2025

Resumo

Este documento apresenta a análise detalhada da complexidade algorítmica da implementação da user story USEI08 - Search by Geographical Area. A solução implementa uma KD-Tree balanceada para buscas espaciais eficientes em estações ferroviárias europeias, demonstrando complexidades temporais e espaciais ótimas para operações de range search em grandes datasets.

Conteúdo

1	Introdução	2
1.1	Características do Dataset	2
2	Arquitetura da Solução	2
2.1	KD-Tree Balanceada	2
2.2	Mecanismo de Busca Espacial	2
3	Análise de Complexidade Temporal	2
3.1	Construção da KD-Tree	2
3.2	Busca por Área Geográfica (Range Search)	3
3.2.1	Análise Detalhada do Algoritmo de Busca	3
3.3	Complexidade dos Filtros	4
4	Análise de Complexidade Espacial	4
5	Fatores que Influenciam a Performance	4
5.1	Balanceamento da Árvore	4
5.2	Eficiência da Poda	4
5.3	Otimização de Buckets	5
6	Validação Experimental	5
6.1	Métricas da Implementação	5
6.2	Resultados Esperados para 64k Estações	5
7	Conclusão	5

1 Introdução

A user story USEI08 requer a implementação de um sistema de busca espacial por área geográfica para estações ferroviárias europeias. A solução desenvolvida utiliza uma KD-Tree balanceada que permite buscas eficientes com filtros opcionais por país, tipo de estação (cidade/principal).

1.1 Características do Dataset

- Aproximadamente 64.000 estações europeias
- Coordenadas geográficas (latitude, longitude)
- Metadados: país, timezone, flags (isCity, isMainStation)

2 Arquitetura da Solução

2.1 KD-Tree Balanceada

A estrutura central é uma KD-Tree 2D construída de forma balanceada utilizando as coordenadas (latitude, longitude) das estações.

```
1 public class KDTree {  
2     public static class Node {  
3         private final List<EuropeanStation> stations;  
4         private Node left, right;  
5         private final double latitude, longitude;  
6         private final int depth;  
7     }  
8 }
```

Listing 1: Estrutura da KD-Tree

2.2 Mecanismo de Busca Espacial

```
1 public record SpatialSearch(KDTree kdTree) {  
2     public List<EuropeanStation> searchByGeographicalArea(  
3         double latMin, double latMax,  
4         double lonMin, double lonMax,  
5         String countryFilter,  
6         Boolean isCityFilter,  
7         Boolean isMainStationFilter  
8     );  
9 }
```

Listing 2: Interface do SpatialSearch

3 Análise de Complexidade Temporal

3.1 Construção da KD-Tree

Complexidade: $O(N \log N)$

Justificação:

- Ordenação inicial por latitude e longitude: $O(N \log N)$
- Construção recursiva com partição por medianas: $O(N \log N)$
- Processamento de $O(N)$ elementos em cada nível de recursão

```

1 public void buildBalanced(List<EuropeanStation> stationsByLat,
2                             List<EuropeanStation> stationsByLon) {
3     // Ordenacao inicial: O(N log N)
4     // Construcao recursiva: O(N log N)
5 }
```

Listing 3: Algoritmo de Construção Balanceada

3.2 Busca por Área Geográfica (Range Search)

Cenário	Complexidade	Justificativa
Melhor Caso	$O(\log N)$	Poda eficiente, região seletiva
Caso Médio	$O(\sqrt{N})$	KD-Tree balanceada em 2D
Pior Caso	$O(N)$	Área de busca muito ampla

Tabela 1: Complexidade da Busca Espacial

3.2.1 Análise Detalhada do Algoritmo de Busca

```

1 private void searchInRangeRecursive(KDTree.Node node,
2                                     double latMin, double latMax,
3                                     double lonMin, double lonMax,
4                                     String countryFilter,
5                                     Boolean isCityFilter,
6                                     Boolean isMainStationFilter,
7                                     int depth,
8                                     List<EuropeanStation> results) {
9
10    if (node == null) return; // O(1)
11
12    // Verificação do nó atual: O(1)
13    boolean inLatRange = (currentLat >= latMin && currentLat <= latMax);
14    boolean inLonRange = (currentLon >= lonMin && currentLon <= lonMax);
15
16    if (inLatRange && inLonRange) {
17        // Filtragem das estações no nó: O(K)
18        for (EuropeanStation station : node.getStations()) {
19            if (matchesFilters(station, countryFilter,
20                               isCityFilter, isMainStationFilter)) {
21                results.add(station); // O(1)
22            }
23        }
24    }
25
26    // Poda baseada na dimensão atual
27    int currentDimension = depth % 2;
28    if (currentDimension == 0) {
29        if (latMin <= currentLat) {
```

```

30         searchInRangeRecursive(node.getLeft(), ...); // Recursao
31     }
32     if (latMax >= currentLat) {
33         searchInRangeRecursive(node.getRight(), ...); // Recursao
34     }
35 }
36 // Logica similar para longitude...
37 }
```

Listing 4: Algoritmo de Busca Recursiva

3.3 Complexidade dos Filtros

```

1 private boolean matchesFilters(EuropeanStation station,
2                                 String countryFilter,
3                                 Boolean isCityFilter,
4                                 Boolean isMainStationFilter) {
5     // Todas as operacoes: O(1)
6     // Filtros nulos sao ignorados
7     // Comparacao case-insensitive para pais
8 }
```

Listing 5: Aplicação de Filtros

Complexidade: $O(1)$ por estação

4 Análise de Complexidade Espacial

Componente	Complexidade
Armazenamento KD-Tree	$O(N)$
Stack de Recursão (caso médio)	$O(\log N)$
Resultados da Busca	$O(K)$
Estruturas Auxiliares	$O(1)$

Tabela 2: Complexidade Espacial

5 Fatores que Influenciam a Performance

5.1 Balanceamento da Árvore

- **Altura ótima:** $O(\log N)$
- **Construção balanceada** garante performance consistente
- **Métrica de balanceamento:** $h \leq 2 \times \log_2(N)$

5.2 Eficiência da Poda

- **Redução do espaço de busca** através de condições geométricas
- **Evita subárvores** quando a região de busca não as intersecta
- **Decisões baseadas** na dimensão atual (latitude/longitude)

5.3 Otimização de Buckets

- **Agrupamento** de estações com coordenadas idênticas
- **Redução** do número total de nós
- **Eficiência** em operações de filtragem

6 Validação Experimental

6.1 Métricas da Implementação

```
1 public String getComplexityAnalysis() {
2     return String.format("""
3         KD-Tree Properties:
4         - Height: %d
5         - Nodes: %d
6         - Balance: %s
7
8         Time Complexity:
9         - Best case: O(log n)
10        - Average case: O( n )
11        - Worst case: O(n)
12
13        Space Complexity:
14        - Auxiliary: O(1)
15        - Recursion stack: O(log n)
16        """
17        ,
18        kdTree.height(),
19        kdTree.size(),
20        kdTree.height() <= 2 * Math.log(kdTree.size()) / Math.log(2)
21        ? "Good" : "Could be improved");
22 }
```

Listing 6: Métricas de Performance

6.2 Resultados Esperados para 64k Estações

Operação	Complexidade	Tempo Esperado
Construção KD-Tree	$O(N \log N)$	$\sim 100\text{-}200\text{ms}$
Busca Seletiva	$O(\log N)$	$\sim 0.1\text{-}1\text{ms}$
Busca Média	$O(\sqrt{N})$	$\sim 1\text{-}10\text{ms}$
Busca Completa	$O(N)$	$\sim 10\text{-}100\text{ms}$

Tabela 3: Performance Esperada ($N = 64.000$)

7 Conclusão

A implementação da USEI08 demonstra uma abordagem eficiente e escalável para buscas espaciais em grandes datasets:

- **Performance Ótima:** Complexidade sub-linear no caso médio ($O(\sqrt{N})$)

- **Escalabilidade:** Adequada para datasets de grande dimensão
- **Flexibilidade:** Suporte a múltiplos filtros sem impacto significativo
- **Robustez:** Balanceamento automático garante performance consistente

A solução atende plenamente aos requisitos da USEI08, oferecendo tempos de resposta rápidos mesmo para o dataset completo de 64.000 estações europeias.

Anexos

A. Notação Assintótica Utilizada

- $O(N)$: Complexidade linear
- $O(\log N)$: Complexidade logarítmica
- $O(\sqrt{N})$: Complexidade sub-linear típica de KD-Trees 2D
- $O(N \log N)$: Complexidade log-linear

B. Referências

- Cormen, T. H. et al. "Introduction to Algorithms"
- Bentley, J. L. "Multidimensional Binary Search Trees Used for Associative Searching"
- Documentação do Projeto: Enunciado_Projeto_Sprint2.pdf