Análise de Complexidade - USEI10: Pesquisa por Raio e Sumário de Densidade

Logistics on Rails - Grupo 023

Resumo

Este documento apresenta a análise detalhada da complexidade algorítmica da implementação da user story **USEI10 - Pesquisa por Raio e Sumário de Densidade**. A solução utiliza uma KD-Tree balanceada para pesquisas espaciais eficientes em estações ferroviárias europeias, combinada com estruturas de dados auxiliares para gerar estatísticas de densidade e ordenação dos resultados.

Arquitetura da Solução

Classe RadiusSearch

java

```
public class RadiusSearch {
    private final KDTree spatialIndex;

    public Object[] radiusSearchWithSummary(double targetLat, double targetLon, double ra
diusKm)
```

Mecanismo de Pesquisa por Raio

- **Entrada**: Coordenadas alvo (lat, lon) + raio em km
- **Saída**:
  - BST ordenada por distância (ASC) e nome (DESC)
  - Sumário de densidade (contagens por país, cidade, estação principal)

Análise de Complexidade Temporal

3.1 Pesquisa por Raio (Radius Search)

Cenário	Complexidade	Justificativa
Melhor Caso	$O(\log N)$	Raio muito seletivo, poda eficiente
Caso Médio	$O(\sqrt{N} + K \log K)$	KD-Tree balanceada + ordenação dos K resultados
Pior Caso	$O(N + K \log K)$	Raio muito amplo, visita todos os nós

Onde:

- **N**: Número total de estações (≈64.000)
- **K**: Número de estações dentro do raio de pesquisa

3.2 Análise Detalhada do Algoritmo

java

```
public Object[] radiusSearchWithSummary(double targetLat, double targetLon, double radi
usKm) {
List<EuropeanStation> stationsInRadius = spatialIndex.radiusSearch(targetLat, targetLon, r
adiusKm);

for (EuropeanStation station : stationsInRadius) {
        double distance = GeoDistance.haversine(...);

Construção da BST balanceada: O(K log K)
    resultBST.buildBalancedTree(distanceList, sd -> sd);

    return new Object[]{resultBST, summary};
}
```

Breakdown da Complexidade:

- **Busca KD-Tree**: `O(√N)` - típico de KD-Trees 2D balanceadas
- **Cálculo Haversine**: `O(1)` por estação → `O(K)` total
- **Agregações (HashMap)**: `O(1)` por operação → `O(K)` total
- **Construção BST balanceada**: `O(K log K)`

3.3 Complexidade dos Filtros e Agregações

java

```
countryCount.merge(station.getCountry(), 1, Integer::sum);
cityCount.merge(station.isCity(), 1, Integer::sum);
mainStationCount.merge(station.isMainStation(), 1, Integer::sum);
```

Análise de Complexidade Espacial

Componente	Complexidade	Justificativa
KD-Tree	`O(N)`	Estrutura principal de indexação
Lista de Resultados	`O(K)`	Estações dentro do raio
BST de Ordenação	`O(K)`	Árvore balanceada com K elementos
HashMaps do Sumário	`O(M)`	M = número de países/categorias distintas
Stack de Recursão	`O(log N)`	Busca recursiva na KD-Tree

Fatores que Influenciam o Desempenho

5.1 Eficiência da Poda Espacial

- A KD-Tree utiliza condições geométricas para podar subárvores
- Redução significativa quando o raio é seletivo
- Haversine calculado apenas para candidatos promissores

5.2 Balanceamento da BST de Resultados

- Construção balanceada garante `O(log K)` para operações futuras
- Ordenação por múltiplos critérios (distância ASC, nome DESC)

5.3 Otimização das Agregações

- Uso de HashMaps para contagens em tempo constante
- Processamento em stream durante a iteração principal

Validação Experimental Esperada

6.1 Métricas para 64k Estações

Operação	Complexidade	Tempo Esperado	K = 10	K = 100	K = 1000
Busca KD-Tree	`O(√N)`	~1-5ms	2ms	3ms	4ms
Processamento K	`O(K log K)`	~0.1-10ms	0.1ms	0.5ms	5ms
Total	`O(√N + K log K)`	~1-15ms	2.1ms	3.5ms	9ms

6.2 Exemplo de Sumário de Densidade

java

```
DensitySummary {
    totalStations: 47,
    countryCount: { "PT": 15, "ES": 32 },
    cityCount: { true: 8, false: 39 },
    mainStationCount: { true: 3, false: 44 }
}
```

Conclusão

A implementação da USEI10 demonstra uma abordagem eficiente e escalável para pesquisas por raio em grandes conjuntos de dados geoespaciais:

- **Desempenho Ótimo**: Complexidade sub-linear no caso médio (`O(√N + K log K)`)
- **Escalabilidade**: Adequada para conjuntos de dados de grande dimensão

- **Funcionalidade Completa**: Combina pesquisa espacial com análise estatística
- **Robustez**: Estruturas balanceadas garantem desempenho consistente

A solução atende plenamente aos requisitos da USEI10, oferecendo tempos de resposta rápidos mesmo para o conjunto completo de 64.000 estações europeias, com capacidade de gerar sumários de densidade detalhados em tempo real.