Relatório – Trabalho Prático 1

Algoritmos 2

Marcelo Ganem, Rafael Paniago, Pedro Loures
{marceloganem,rafaelpaniago,pedroloures}@dcc.ufmg.br
Universidade Federal de Minas Gerais

8 de junho de 2025

1 Especificação do Problema

O código que acompanha este relatório implementa o **armazenamento e consulta de pontos geográficos**, bem como suas informações correspondentes – pertinentes aos bares cadastrados na Prefeitura de Belo Horizonte e ao festival *Comida di Buteco* – em uma **árvore K-dimensional**.

1.1 Dados de entrada

A base de dados primária para a implementação é o relatório em formato `.csv` de localização das atividades econômicas cadastradas no município de Belo Horizonte disponibilizada mensalmente pela Secretaria Municipal da Fazenda da PBH[1].

Adicionalmente, dados **não-estruturados** do festival *Comida di Buteco*, disponíveis na lista de participantes do evento[2], são necessários para a execução do exercício extra.

Por fim, definimos a entrada do usuário como um **retângulo** sob o espaço bidimensional representado pelo mapa de Belo Horizonte.

1.2 Árvores K-Dimensionais

O enunciado propõe a organização dos dados obtidos conforme latitude e longitude em uma árvore K-dimensional. Essa estrutura permite realizar buscas em intervalos ortogonais[3] em tempo $O(\sqrt{n}+k)$ no caso bidimensional (para uma árvore com n nós e retornando k resultados) dado um processo de construção de custo $O(n \log n)$.[4]

[1] `https://dados.pbh.gov.br/dataset/atividades-economicas1`

[2] `https://comidadibuteco.com.br/butecos/belo-horizonte`

[3] Do inglês *orthogonal range search*.

[4] Na implementação ótima. A implementação utilizada aqui constrói a árvore em $O(n \log^2 n)$.

1.3 Ferramentas

A implementação do trabalho na linguagem Python requer a OpenStreetMaps API[5] para obtenção de coordenadas geográficas correspondentes aos endereços dos dados primários e a biblioteca *dash-leaflet*[6] para a construção da interface gráfica. Todas as ferramentas adicionais utilizadas são explicitamente justificadas na Seção 2.

1.4 Requisitos

A tarefa consiste em implementar uma interface web que permita ao usuário selecionar uma área retangular no mapa de Belo Horizonte e, por meio da busca intervalar na árvore K-dimensional, filtrar entre os restaurantes em uma determinada tabela.

1.4.1 Requisitos adicionais (*Comida di Buteco*)

Como uma tarefa opcional, dados do festival Comida di Buteco podem ser cruzados com os dados da tarefa principal, aumentando a interface web com um *pop-up* dinâmico que exibe informações[7] sobre o prato concorrente de restaurantes participantes.

2 Implementação

2.1 Processamento de dados

2.1.1 Filtragem

Um primeiro filtro seleciona somente as entradas de CNAE 5611201.0, 5611204.0 ou 5611205.0, correspondentes a bares e restaurantes. Então,

[5] `https://www.openstreetmap.org/`

[6] `https://www.dash-leaflet.com/`

[7] Vale observar que as informações se limitam às imagens disponíveis na galeria de restaurantes – isso porque as páginas dos estabelecimentos específicos não estão disponíveis desde (pelo menos) 03/06/2025.

descartamos as colunas excedentes, mantendo: nome, endereço – formatado como uma única *string*, data de início das atividades e um campo *booleano* condicional à existência de um alvará emitido pela prefeitura.

2.1.2 Geolocalização

Aumentamos os dados filtrados com coordenadas geográficas correspondentes ao endereço de cada bar utilizando a biblioteca `geopy`[8]. Assim, definimos os valores a serem utilizados para ordenação na árvore K-dimensional.

2.1.3 Deduplicação

Deduplicamos os dados, tendo em vista a duplicação extensiva presente no conjunto original, em função do campo `NOME` da entrada (não processado).

2.1.4 Coleta e casamento (*Comida di Buteco*)

Os dados foram extraídos por meio de *web scraping* utilizando a biblioteca `bs4`[9]. O casamento entre os dados do *Comida di Buteco* e da PBH é feito com base em uma representação normalizada da forma `rua#número`, e então desambiguado com base na contagem de palavras em comum no campo `NOME` – também normalizado. Por *normalização*, nos referimos à remoção de acentos, caracteres não alfanúmericos e palavras e sequências irrelevantes.

2.2 Árvores K-Dimensionais

A implementação da árvore para busca intervalar é feita em C++ e Python. Utilizamos a biblioteca `pybind11`[10] para compilar o código da classe que implementa a árvore K-dimensional em C++, incluindo um método construtor $O(n \log^2 n)$ e um método de consulta $O(\sqrt{n}+k)$. A implementação em Pyhton apresenta métodos equivalentes que invocam o método na classe original, e é incluída por conveniência.

2.3 Interface Web

O aplicativo *dash-leaflet* implementado apresenta um mapa e uma tabela que filtra constantemente os restaurantes sob a seleção retangular atual – que pode ser iniciada pelo usuário clicando no botão correspondente no canto superior esquerdo. Adicionalmente, restaurantes participantes do *Comida di Buteco* aparecem em destaque nas primeiras posições da tabela.

Selecionar um restaurante na tabela ativa um marcador no mapa e centraliza a visualização atual no restaurante selecionado. Passar o mouse (*hover*) no marcador de um restaurante participante do *Comida di Buteco* ativa um *pop-up* mostrando informações do restaurante e uma imagem do prato concorrente.

3 Considerações

1. A geolocalização de dados e a posterior filtragem por dados geolocalizados para *display* no mapa causa uma perda de aproximadamente 3.000 entradas.
2. Os critérios de casamento descritos na Subseção 2.1.4 casam 78 dos 124 restaurantes participantes com os dados deduplicados não geolocalizados, e 51 com os dados filtrados por geolocalização.
3. A árvore é construída sempre que o servidor é incializado. Para otimização, uma ideia é armazenar a estrutura ordenada em disco – essa solução é mais escalável para um número de entradas fora dos limites específicos deste trabalho.

[8] `https://geopy.readthedocs.io/en/stable/`
[9] `https://pypi.org/project/beautifulsoup4/`
[10] `https://github.com/pybind/pybind11`