Trabalho Prático I - Geometria Computacional
DCC 207 - Algoritmos II

Departamento de Ciência da Computação, Instituto de Ciências Exatas, Universidade Federal de Minas Gerais
Belo Horizonte, Minas Gerais
Outono de 2025

2022101086 Augusto Guerra de Lima
augustoguerra@dcc.ufmg.br
2023028234 Cauã Magalhães Pereira
caua.magalhaes@dcc.ufmg.br
2023028579 Heitor Gonçalves Leite
heitorleite@dcc.ufmg.br

1 Introdução

Este trabalho objetiva estudar como estruturas de dados associadas a geometria computacional, a saber, *árvores k-dimensionais*, são empregadas no contexto de georreferenciamento.

Em síntese, foram coletados dados de restaurantes e bares na região de Belo Horizonte. Utilizou-se uma árvore k-dimensional para realizar *buscas ortogonais*, isto é, consultas em subconjuntos retangulares na região da cidade.

A próxima seção tratará de como foi realizada a coleta de dados, como foi implementada a estrutura de dados e como foi criada a interface; A terceira seção discutirá os resultados; As considerações finais e a bibliografia concluirão o texto.

2 Metodologia

2.1 Coleta e processamento de dados

Inicialmente, os dados brutos fornecidos pela PBH foram processados para manter apenas os registros associados a bares e restaurantes. Essa filtragem foi realizada a partir de palavras-chave aplicadas à descrição da CNAE principal. Em seguida, os endereços dos estabelcimentos foram convertidos para coordenadas geográficas (latitude e longitude).

Em um primeiro momento, para a conversão, foi utilizada a API do *OpenStreetMap* com a biblioteca `geopy`, com o auxílio de *RateLimiter* para respeitar os limites de requisição; O processo utilizou cache para evitar reconsultas e paralelização com *ThreadPoolExecutor* para acelerar a geocodificação dos dados. Contudo, muitos dos distintos pontos foram mapeados para as mesmas coordenadas durante a conversão, então foi escolhido utilizar a biblioteca UTM do `Python` que possui função embutida que mapeia as coordenadas UTM para latitude e longitude.

2.2 Integração dos dados do Comida di Buteco

Foram integradas informações adicionais obtidas manualmente a partir dos dados do festival Comida di Buteco. Para os bares participantes, foram incluídas informações como o nome do prato concorrente, imagem e outras descrições, exibidas dinamicamente no mapa interativo. Essa integração exigiu inferir uma correspondência entre os nomes dos estabelecimentos da base da PBH e os registros do festival.

Isso se deve ao fato de que alguns registros do nome fantasia não coincidirem no *join*, sendo necessário um processo manual de ratificação dos dados.

2.3 Estrutura de dados, árvore k-dimensional

Árvores k-dimensionais dividem o conjunto P de pontos em partições. Seja $p \in P$ um ponto tal que $p = (x_1, x_2, ..., x_k)$, isto é, possui k coordenadas, a estrutura de dados particiona recursivamente o espaço alternando entre suas dimensões, de forma a comparar apenas uma dimensão específica por nível, resultado em uma *árvore binária*.

Em particular, nos dados geográficos, as coordenadas *latitude* e *longitude* implicam em uma árvore bidimensional (*2d-tree*); De forma que, se no nível l é utilizada a coordenada x_1 para o particionamento, em $l+1$ tão somente, será utilizada a coordenada x_2, tal que $(x_1, x_2) \in \texttt{float}^2$.

A propósito, seja $a = (a_1, a_2),\ b = (b_1, b_2) \in \texttt{float}^2$, duas coordenadas tal que $a_1 \leq b_1$ e $a_2 \leq b_2$, o conceito de *busca intervalar ortogonal* surge naturalmente: Constitui-se em determinar um subconjunto de pontos $P' = \{p_1, p_2, ..., p_n\} \subset \texttt{float}^2$, onde para todo $p \in P'$, $a \leq p \leq b$; Com a adição de que busca é definida em retângulos paralelos aos eixos.

Como mostrado em [MD], a *busca intervalar ortogonal* tem uma propriedade útil, designadamente pode ser decomposta em produto de *buscas intervalares unidimensionais*. Justificando a rotatividade de coordenadas.

$$\{p \in \texttt{float}^k : a_i \leq p_i \leq b_i\} = [a_1, b_1] \times [a_2, b_2] \times ... \times [a_k, b_k].$$

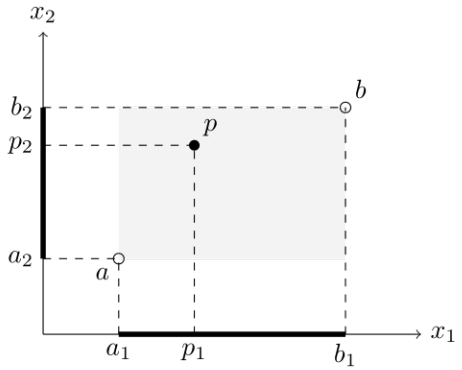

Figure 1: Representação de uma busca intervalar ortogonal em um espaço bidimensional arbitrário.

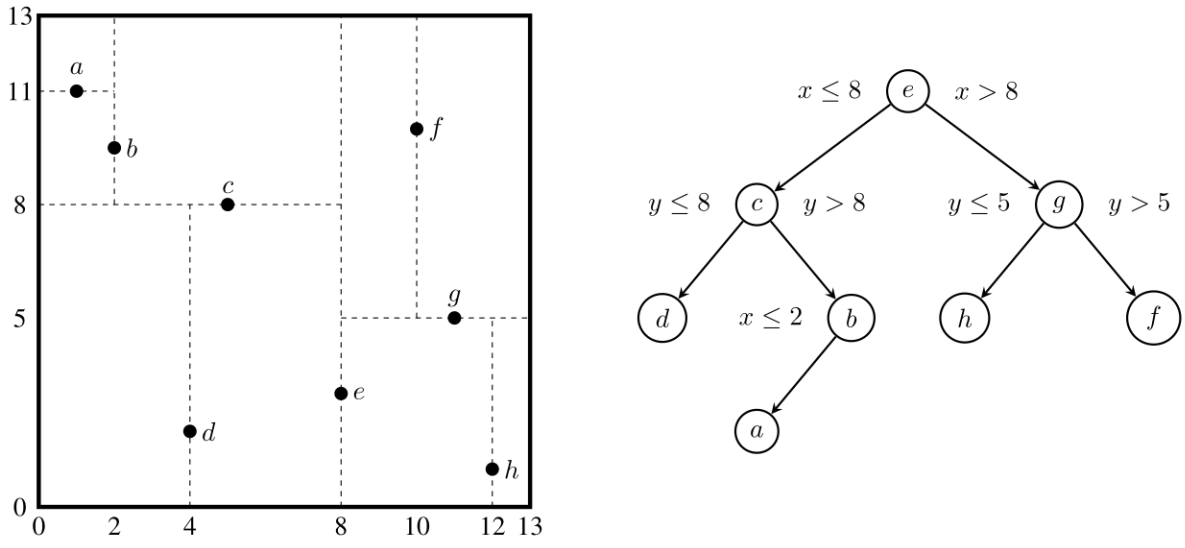

Figure 2: Visualização de pontos no mapa e seu particionamento, e a representação da instancia na estrutura de dados utilizada.

2.3.1 Implementação

A estrutura de dados foi implementada em `C++`. Para integrar com toda a aplicação em `Python`, foi necessário utilizar o *nanobind*, uma pequena biblioteca de integração dessas duas linguagens de programação.

A árvore k-dimensional foi escrita utilizando índices, ou seja, ela apenas armazena um apontador para os *id's* do *dataframe* que é equivalente aos do arquivo CSV, evitando movimentação em massa dos dados.

A prática de implementar estruturas de dados em `C++` para realizar a computação de forma mais eficiente e retornar os resultados para a aplicação `Python` é bastante usual.

Boas práticas do uso de `<template>` e dimensão arbitrária também foram utilizadas. Por fim, para a construção da estrutura em questão com a mediana dos pontos, fora utilizado o algoritmo *quickselect*, assegurando que a árvore é balanceada.

2.3.2 Análise de complexidade

Seja k o número de pontos encontrados em uma *busca intervalar ortogonal*, a complexidade do *tempo de execução* da busca pertence ao conjunto $\mathcal{O}(\sqrt{n})$, mas reportar todas as ocorrências está em $\mathcal{O}(\sqrt{n}+k)$.

A cada dois níveis da árvore k-dimensional, o número de nós que podem ser atingidos dobra. A relação de recorrência a seguir modela a discussão; Seja $T(n)$ o número de nós acessados em uma subárvore com n pontos.

$T(n) \leq 2$ se $n \leq 4$; $T(n) \leq 2T(\frac{n}{4}) + 1$ contrariamente.

Expandindo a relação de recorrência,

$$T(n) \leq 2T(\frac{n}{4}) + 1$$

$$\leq 2(2T(\frac{n}{4^2}) + 1) + 1$$

$$... \leq \sum_{i=0}^{k-1} 2^i + 2^k T(\frac{n}{4^k})$$

$$\Rightarrow T(n) \leq (2^{\log_4 n} - 1) + 2^{\log_4 n} T(1) \leq 3\sqrt{n} \in \mathcal{O}(\sqrt{n}).$$

2.4 Interface interativa e publicação

A interface do sistema foi desenvolvida com a biblioteca *dash-leaflet*, que combina os recursos de visualização do *Dash* com a renderização geográfica do *Leaflet*. Foram implementadas funcionalidades de visualização dos estabelecimentos no mapa com pinos georreferenciados, seleção interativa de regiões, atualização da tabela de dados e exibição de detalhes complementares. O sistema final foi publicado em ambiente web utilizando a plataforma *Render*, com o servidor *gunicorn* para servir a aplicação `Python`.

3 Resultados

Ao final do trabalho desenvolvemos um sistema robusto e *user-friendly*, capaz de visualizar os dados de bares e restaurantes de Belo Horizonte, cruzados com os dados do Comida di Buteco. A Figura 3 ilustra a tela inicial da aplicação 3.

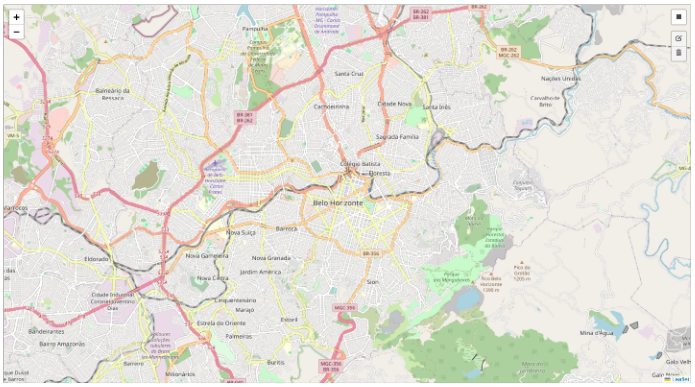

Figure 3: Tela inicial da aplicação

A aplicação permite selecionar um conjunto de áreas retangulares (inclusive com interseção entre elas) e realizar a busca dos pontos nessas áreas. Além disso, também é possível editar e excluir as regiões selecionadas, como mostram as Figuras 4 e 5.

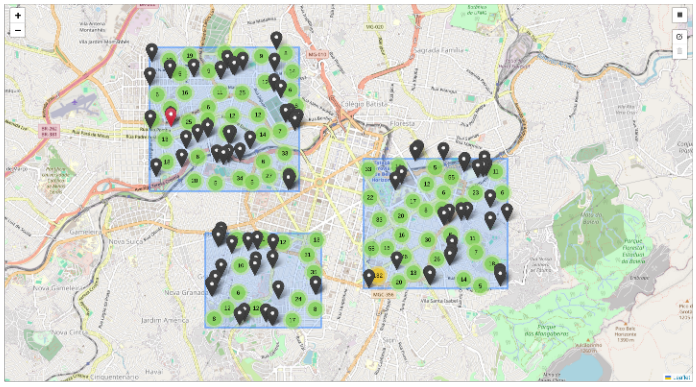

Figure 4: Seleção de áreas para a busca

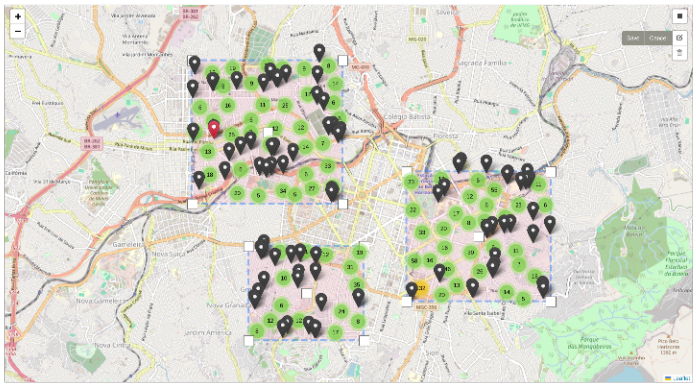

Figure 5: Edição das áreas selecionadas

A partir de uma seleção, é possível visualizar os dados que satisfazem aquela busca, como mostra a Figura 6.

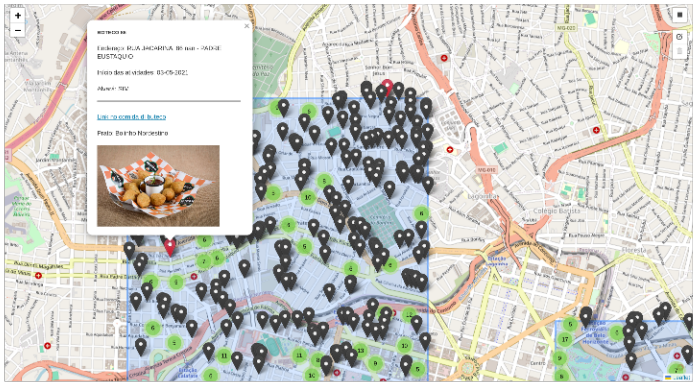

Figure 6: Visualização dos dados

4 Considerações finais

Neste trabalho, implementamos uma árvore k-dimensional para realizar buscas ortogonais em dados georreferenciados de bares e restaurantes de Belo Horizonte, integrando informações adicionais do festival Comida di Buteco.

Essas estruturas fundamentais para buscas eficientes em espaços multidimensionais, sendo amplamente utilizadas em aplicações de geolocalização e visão computacional.

A experiência foi interessante para explorar ferramentas novas para lidar com dados e desenvolvimento de aplicações, além de proporcionar maior contato com programação em `Python`, especialmente na integração com código em `C++`.

5 Bibliografia

References

[CDB] Comida di Buteco. Disponível em: `https://comidadibuteco.com.br/`. Acesso em: 2 jun. 2025.

[DL] *Dash-Leaflet Documentation*. Disponível em: `https://dash-leaflet.herokuapp.com/`. Acesso em: 1 jun. 2025.

[GB] Leonidas J. Guibas. *Geometric Range Searching Kinetic Data Structures Clustering Mobile Nodes*; Stanford University. Disponível em: `https://graphics.stanford.edu/courses/cs428-03-spring/03Talks/Range.pdf`. Acesso em: 1 jun. 2025.

[GPY] *Geopy Documentation*. Disponível em: `https://geopy.readthedocs.io/`. Acesso em: 16 mai. 2025.

[MD] MOUNT, Dave. CMSC 754: *Lecture* 14 - *Orthogonal Range Searching and kd-Trees*; University of Maryland. Disponível em: `https://www.cs.umd.edu/class/fall2021/cmsc754/Lects/lect14-kd-tree.pdf`. Acesso em: 1 jun. 2025.

[NB] *Nanobind Repository*. Disponível em: `https://github.com/wjakob/nanobind`. Acesso em: 6 de jun. 2025.

[RD] *Render Documentation*. Disponível em: `https://render.com/docs`. Acesso em: 5 jun. 2025.