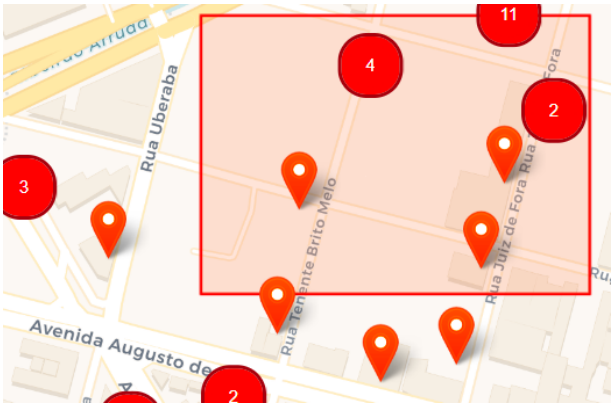
Trabalho Prático 1: Documentação
- Geometria Computacional -



Algoritmos II - 2025/1 - Ciência da Computação, UFMG

Integrantes do grupo:

- **Giovanni Russo Paschoal - giovanni.russo@dcc.ufmg.br** **Matrícula: 2023028390**
- **João Pedro Wadge - wadgejoaopedro@gmail.com** **Matrícula: 2023028340**
- **Wilian Ventura dos Reis - wilianv@ufmg.br** **Matrícula: 2023028129**

Professor: Renato Vimeiro

1- Introdução:

Este trabalho prático se propõe a exercitar alguns aspectos da geometria computacional trabalhados nas aulas expositivas de Algoritmos 2. Seu objetivo consiste em implementar um sistema interativo no qual é possível visualizar todos os bares/restaurantes registrados na base de dados da Prefeitura de Belo Horizonte. A aplicação deve ser desenvolvida em Python, com auxílio da biblioteca *dash-leaflet* para a exibição do mapa interativo, e implementar uma funcionalidade de filtro através de uma K-D Tree de duas dimensões (Árvore 2D). Esse filtro reorganiza restaurantes e bares de uma área retangular desenhada pelo usuário em uma tabela, que exibe, para cada estabelecimento: nome (ou nome fantasia), endereço, data de início e se o local possui ou não alvará.

Para interação com o sistema, é necessário:

- Clicar no botão de desenho, que fica no canto superior esquerdo, abaixo do controle de zoom.
- Desenhar um retângulo cobrindo a área desejada, com dois cliques nos pontos que representam seus limites.

Então a tabela irá retornar os dados dos pontos que estão dentro da área desenhada. Além disso, é possível visualizar os pratos competidores daqueles estabelecimentos participantes do festival Comida Di Buteco de 2025, passando o mouse por cima dos marcadores no mapa (hover).

Nota: Para exemplos práticos, veja o README.md do repositório.

2- Modelagem:

O problema da filtragem na aplicação interativa foi resolvido através da modelagem de três fases principais, que atuam em conjunto: A entrada, o processamento dos dados na 2-D Tree e a saída. É importante ressaltar que durante a execução de todo o programa, ficam carregados os dados de todos os estabelecimentos em uma base na memória e plotados no mapa seus respectivos marcadores geográficos. Esses elementos não são alterados em tempo de execução.

Entrada: A entrada consiste na ferramenta de desenho de retângulos, disponível na interface gráfica. Ao desenhar um retângulo, o usuário recebe um feedback em tempo real que mostra visualmente a área selecionada. Quando a ação de desenho é finalizada, são retornadas somente as coordenadas geográficas (em latitude e longitude) dos limites da área criada.

2-D Tree: A 2-D Tree é uma estrutura de dados responsável por armazenar de forma abstrata os bares e restaurantes, guardando somente as informações de localidade geográfica e seu identificador (índice numérico) na base de dados em memória. Ela divide o espaço recursivamente nas duas dimensões (latitude e longitude) e possui uma função de busca por área, que recebe os limites gerados pela entrada e retorna um vetor com os IDs dos pontos dentro do retângulo.

Saída: Com os identificadores dos estabelecimentos filtrados é possível extrair as informações da memória e exibir na tabela, que é desenhada na interface de usuário.

É importante mencionar que a parte da "saída" referente ao mapa e aos marcadores não depende de uma entrada do usuário, pois está está sempre em exibição na interface. De mesmo modo, caso nenhum retângulo seja desenhado, a tabela exibirá as informações relativas a **todos** os estabelecimentos.

Foram configuradas também funcionalidades de entrada e saída não diretamente relacionadas com o algoritmo principal da 2-D Tree, que já estavam implementadas pelas ferramentas utilizadas. Por exemplo, o mapa pode ser ampliado e arrastado dentro dos limites de Belo Horizonte, e os marcadores de bares/restaurantes são automaticamente agrupados quando o mapa exibe uma área maior.

3- Tratamento dos dados:

A base de dados utilizada foi o arquivo CSV de atividade econômica de 2025-04-01, disponibilizado publicamente no site da Prefeitura de Belo Horizonte, no qual há informações referentes a todos os estabelecimentos economicamente ativos no município. Por isso, foi feito um pré-processamento inicial nos estabelecimentos, a fim de manter apenas aqueles que continham "bar" ou "restaurante" em sua descrição da CNAE principal. No total foram obtidos 13801 bares e restaurantes.

Para obter as coordenadas dos bares e restaurantes, foi utilizada a API do OpenStreetMap de pesquisa por endereço, uma vez que esse é composto por atributos presentes na base de dados. Os endereços foram propriamente formatados e pesquisados na ferramenta, resultando em cerca de 11 600 pontos geográficos (com latitude e longitude) na região de Belo Horizonte. Porém, algumas centenas dos pontos encontrados estavam com erros (alguns até estavam em outras cidades), por diversos motivos diferentes. Tanto no OpenStreetMap quanto no site da prefeitura haviam ruas e bairros com nomes

desatualizados (Exemplo: alguns endereços no Novo Santa Cecília estavam com outro nome de bairro), duplicados (Exemplo: existem 6 ruas "Dois" distintas) ou com nomes diferentes cadastrados para as mesmas regiões (Exemplo: Petrópolis e Vila Petrópolis são o mesmo lugar). O restante dos pontos não foi encontrado pelo OpenStreetMap.

Assim, foi elaborada uma segunda estratégia, que no lugar de utilizar a API do OpenStreetMap, converteu o atributo GEOMETRIA(x,y) presente no CSV em latitude e longitude. Esse atributo continha a localização dos estabelecimentos em coordenadas UTM Leste/Norte para a zona 23 do hemisfério Sul. Após identificação e conversão, todos os 13801 pontos tiveram suas coordenadas geográficas propriamente obtidas. A precisão da conversão foi visualmente avaliada como suficiente.

Foi então criado um novo arquivo CSV chamado *bares_restaurantes.csv*, com todas as alterações mencionadas.

Ademais, para a obtenção dos dados do site "ComidaDiButeco" foi realizado um processo de web scraping, que consistiu em abrir a "página principal" do domínio com o filtro para a cidade de Belo Horizonte e em selecionar as classes correspondentes aos estabelecimentos — entrando no hiperlink sob o nome de "Detalhes" para cada uma destas. Isso permitiu extrair as informações desejadas sobre os botecos. Nesse sentido, foram exploradas todas as páginas da pesquisa com a filtragem, totalizando 124 estabelecimentos, cujos dados — nome, endereço, prato, descrição do prato e imagem deste — foram armazenados em um único arquivo JSON. Sob tal ótica, integrar tal base com a obtida anteriormente ocorreu por meio de 4 etapas, as quais avançam se há falha em cruzar um dado: encontrar casamentos entre os nomes fantasia, tentar parear os endereços, investigar de forma manual o restante dos dados, e completar a base *bares_restaurantes.csv*. Assim, a terceira etapa permitiu corrigir erros como divergência de nomes, logradouros, números e bairros; a quarta ocorreu em virtude dos itens que chegam nessa fase serem tidos como inexistentes, o que levanta a necessidade de preencher a tabela com os dados, relevantes e possível de serem encontrados, para o trabalho. Além disso, é válido ressaltar que, dentre os 6 elementos que alcançaram a última parte, todos tiveram o atributo "IND_POSSUI_ALVARA" marcado como "SIM", em razão da inviabilidade de descobrir esse dado, e 3 receberam "00-00-0000" como "DATA_INICIO_ATIVIDADE", pelo mesmo motivo.

4- Implementação:

Na implementação, foram utilizadas majoritariamente as bibliotecas Python *dash, dash_leaflet e dash_extensions* para criar o mapa e as funções necessárias para seu comportamento. Além disso, foram utilizadas as bibliotecas *pandas*, para carregamento dos dados do CSV em memória, *plotly*, para criar a tabela dinâmica que exibe as informações dos pontos, *requests*, para acessar as páginas do "Comida Di Buteco" e baixar o HTML delas, *BeautifulSoup*, para filtrar a estrutura do HTML, e *urllib*, para lidar com os hiperlinks. A interface da aplicação está dividida em 2 colunas: na da esquerda há a tabela com os dados filtrados e na da direita o mapa interativo. O código começa lendo o arquivo CSV em um data frame do *pandas*, inserindo os índices e coordenadas de cada estabelecimento na 2D-Tree, e montando a tabela inicial do *plotly*.

Dash Leaflet: O *Dash Leaflet* é uma biblioteca que possui "módulos" que facilitam a implementação da biblioteca leaflet. Ela executa um código em JavaScript, que cria um mapa a partir dos dados do OpenStreetMap, já integrado com HTML e CSS. Esse mapa foi inicializado com os marcadores

dos pontos plotados estaticamente, com auxílio de um *geojson*, além de ter sido limitado para cobrir somente Belo Horizonte, estando centralizado na praça Raul Soares. A ferramenta *Edit Tool* do dash foi também alterada para permitir ao usuário desenhar apenas retângulos, ao clicar em um botão.

Plotly: O *Plotly* é uma biblioteca gráfica que permite criar sub-plotagens de diversos tipos diferentes utilizando uma base de dados, além de poder ser integrado com o *Dash Leaflet*. A figura da tabela é criada com 4 colunas fundamentais que recebem os dados formatados da memória. Conforme o usuário interage com a ferramenta de desenhar as áreas de interesse, os dados são atualizados a partir dos índices.

Callback do algoritmo: A entrada e a saída do programa são organizadas a partir de *Callbacks*, ou seja, toda interação do usuário com a interface, como cliques no mapa ou em botões, é gerenciada por funções assíncronas, tratadas como eventos. Todas as 3 etapas da busca (entrada, processamento na 2d-tree, saída) foram organizadas em um único callback, que é ativado quando o retângulo termina de ser desenhado. Esse callback recebe os limites do retângulo, no formato *geojson* e realiza a busca na 2D-Tree. Por fim, ele remove o retângulo recém desenhado e retorna uma tabela do *plotly* atualizada a partir dos índices encontrados.

5- Análise de Complexidade:

A 2D-Tree implementada apresenta uma complexidade de construção **O(n log n)**, já que, além das duas ordenações presentes no método *build*, a chamada recursiva neste mesmo método também introduz o mesmo custo, em virtude de a cada nível da recursão a lista de pontos ser dividida duas vezes na mediana. Por consequência, existem **log n** níveis, cada um deles com complexidade linear. Ademais, a complexidade espacial de tal função é **O(n log n)**, pois, para todas as etapas da recursão, há a criação de listas temporárias para a filtragem, que gasta um espaço linear, resultando, no custo mencionado. Apesar disso, vale citar que a complexidade espacial da 2d-tree, após sua construção, é **O(n)**.

Por último, a busca ortogonal existente na estrutura de dados trabalhada possui complexidade **O(k +** $\sqrt{n}$ **)**, na qual **k** é a quantidade de elementos presentes na região procurada. Este custo tem como intuição o fato de a 2D-Tree dividir o plano em regiões aproximadamente quadradas, sendo apenas necessário visitar os nós em regiões que interceptam o retângulo de busca, o qual, em casos razoáveis, cruza $\sqrt{n}$ e ,após caminhar pela árvore, coletar os **k** pontos presentes na área de interesse.

6- Parte Extra - Integração com o Comida de Buteco:

Alguns dos restaurantes e bares de Belo Horizonte estão participando do festival Comida Di Buteco de 2025. Por isso, foi implementada também uma integração com seus dados, extraídos do site oficial do evento. Ao passar o mouse sobre os restaurantes participantes (hover), é possível visualizar as informações básicas do prato concorrente no festival, assim como a sua imagem, publicada no site. Para o restante dos estabelecimentos, é exibido somente o nome. Essa é uma funcionalidade que, assim como os marcadores, foi implementada estaticamente, isto é, não muda com a interação do usuário.

7- Considerações finais:

Este projeto foi de grande relevância para aplicar o conhecimento teórico sobre árvores K-dimensionais (K-D trees) vistas em sala de aula, aplicando seus conceitos e funcionamento em um

problema real. Além disso, foram trabalhadas habilidades adicionais nas bibliotecas Python, que permitiram o desenvolvimento de um projeto completo. Ainda, incentivou o trabalho em equipe para o desenvolvimento conceitual e a implementação de soluções para os problemas enfrentados em situações de maior complexidade, algumas das quais fogem do escopo da matéria — como obter, tratar e cruzar bancos de dados divergentes e integrar diferentes APIs. Ademais, a possibilidade de observar e interagir, com propósito construtivo sobre dados reais, no lugar de bases bem tratadas, se provou engrandecedor, por representar bem esse tipo de problema na prática.

8- Referências:

Fontes:

- Data Base PBH - https://dados.pbh.gov.br/dataset/atividades-economicas1
- Comida de Buteco - https://comidadibuteco.com.br/butecos/belo-horizonte/

Ferramentas utilizadas:

- Visual Studio Code - https://code.visualstudio.com/
- Python3 - https://www.python.org/
- OpenStreetMap - https://www.openstreetmap.org/
- Plotly - https://plotly.com/python/table-subplots/
- Dash Leaflet - https://www.dash-leaflet.com/
- CartoDBBasemapStyles - https://github.com/CartoDB/basemap-styles
- requests - https://pypi.org/project/requests/
- BeautifulSoup - https://pypi.org/project/beautifulsoup4/
- urllib - https://docs.python.org/3/library/urllib.html