

Atividade Avaliativa Síncrona Grafos
Aluno: Pedro Muhamad Suleiman Craveiro
RA: 193934
Curso: Engenharia de Computação
Semestre: 4º Semestre

Processo de Desenvolvimento:

O projeto consiste na criação de um programa em que o usuário pode importar um grafo e/ou criá-lo a partir do menu interativo que o algoritmo oferece. Dessa forma, a ideia foi inspirada por meio de um vídeo do canal do professor Hemerson Pistori “Propriedades e Tipos de Grafos”, no Youtube. Juntamente com os conhecimentos da biblioteca Networkx, somado a aula ministrada no dia 05/09/2023 a respeito da biblioteca pyvis, onde foi amplamente utilizada no código. Ademais, mesclado com outra aula sobre a criação de um Bot no Telegram ministrada em 17/10/2023, desenvolvi um Bot em que o usuário pode enviar arquivos no formato .txt ou .csv e verificar as propriedades do seu Grafo utilizando o algoritmo.

Em seguida, detalharei o desenvolvimento do código “grafos.py”:

Função criar_grafo:

Esta função permite ao usuário interativamente criar um grafo inserindo vértices e arestas. O loop while True continua até o usuário decidir encerrar a entrada digitando "fim". Se ocorrer algum erro na entrada, como vértices inexistentes ou formato de aresta inválido, mensagens de erro são exibidas.

Função adicionar_remove_vertices_arestas:

Permite ao usuário adicionar ou remover vértices e arestas de forma interativa. Usa um loop similar ao da função criar_grafo. Os casos de entrada incluem adicionar/remover vértices e adicionar/remover arestas.

Função abrir_arquivo_ou_criar_grafo:

Permite ao usuário escolher entre criar um novo grafo, abrir um arquivo TXT ou abrir um arquivo CSV. Se escolher abrir um arquivo, a função carregar_grafo_de_arquivo é chamada.

Função carregar_grafo_de_arquivo:

Carrega um grafo a partir de um arquivo TXT ou CSV, dependendo da extensão do arquivo. Se o arquivo contiver informações de peso nas arestas, essas informações também são carregadas.

Função visualizar_grafo:

Usa a biblioteca pyvis para criar uma visualização interativa do grafo. Os nós são posicionados usando um layout personalizado e as arestas são coloridas com base no peso. O resultado é salvo em um arquivo HTML e aberto no navegador.

Função `calcular_numero_cromatico`:

Calcula o número cromático do grafo usando a estratégia "largest_first" do NetworkX.

Função `calcular_arvore_minima_geradora`:

Permite ao usuário escolher entre os algoritmos de Kruskal e Prim para calcular a árvore mínima geradora do grafo. O resultado é visualizado usando a função `visualizar_grafo`.

Funções `salvar_grafo_em_arquivo` e `salvar_grafo_em_csv`:

Salvam o grafo em um arquivo TXT ou CSV, respectivamente, usando as informações de vértices e arestas.

Função `calcular_grau_maximo_minimo`:

Calcula os graus máximo e mínimo do grafo.

Funções `calcular_raio`, `calcular_diametro` e `calcular_perimetro`:

Calculam o raio, o diâmetro e o perímetro do grafo, respectivamente.

Função `propriedades_grafo`:

Fornece ao usuário várias opções para analisar e visualizar propriedades do grafo, como ordem, tamanho, grau médio, conectividade, bipartição, entre outras. Além disso, oferece opções para salvar o grafo e visualizá-lo.

Função `main`:

É a função principal que chama todas as outras funções. Permite ao usuário criar um grafo, visualizá-lo, calcular suas propriedades e salvar ou abrir arquivos.

Em seguida, detalharei as seguintes funções desenvolvidas no código a respeito do Bot no Telegram (`@CSVorTXT_bot`) (Pasta: `telegramCSVeTXT/arquivos.py`):

Definição de Constantes

`UPLOADING`: Define um estado para controle da conversa. Neste exemplo, não é utilizado explicitamente, mas serve para controle de fluxo futuro.

`UPLOADS_FOLDER`: Diretório onde os arquivos enviados serão armazenados.

Função `start`

`start(update: Update, context: CallbackContext)`: Envia uma mensagem de início para o usuário no Telegram e solicita o envio de um arquivo CSV ou TXT para ser armazenado.

Retorna o estado `UPLOADING`.

Função `handle_document`

`handle_document(update: Update, context: CallbackContext)`: Lida com o documento enviado pelo usuário.

Verifica se o arquivo enviado tem extensão `.csv` ou `.txt`. Se sim, realiza o download do arquivo para a pasta `UPLOADS_FOLDER`.

Responde ao usuário com uma mensagem indicando se o arquivo foi recebido e armazenado com sucesso ou se a extensão do arquivo não é aceita.

Função `main`

Inicializa o bot do Telegram com um token específico.

Define o `ConversationHandler` para controlar a conversa.

Configura o comando `/start` para iniciar a conversa com o bot.

Configura o estado `UPLOADING` para lidar com o envio de documentos pelo usuário.

Inicia o bot, aguardando por atualizações no Telegram.

Fluxo de Execução

O usuário inicia a conversa com o bot usando o comando `/start`.

O bot responde solicitando o envio de um arquivo CSV ou TXT.

O usuário envia um arquivo.


O bot verifica se o arquivo é do tipo esperado.

Se for um arquivo válido, o bot faz o download e armazena no diretório `uploads`.

O bot responde ao usuário indicando se o arquivo foi armazenado com sucesso ou se a extensão não é aceita.

Em resumo, o código integra funcionalidades de criação, importação, salvar arquivos e visualização das propriedades de grafos.

Materiais Consultados:

- Biblioteca Python: `networkx`
- Biblioteca Python: `pyvis`
- Biblioteca Python: `webbrowser`
- Biblioteca Python: `os`
- Biblioteca Python: `telegram`
- Biblioteca Python: `csv`
- Vídeo do Professor Hemerson Pistori a respeito do conteúdo de Caminho de Custo Mínimo:  [Propriedades e tipos de grafos](#)
- Github Inovisão Bot_Telegram: [Files · master · inovisao / bot telegram · GitLab](#)

Ajudas Recebidas:

Dúvidas específicas foram sanadas por meio de consultas ao professor e interações com colegas, discutindo as melhores formas de implementação, questionamento sobre bibliotecas e a linguagem de programação Python. Além do mais, foram realizados testes do funcionamento do código com os colegas de laboratório.

Estimativa de Tempo de Desenvolvimento:

A estimativa de tempo foi realizada com base nos commits no GitHub e o desenvolvimento em sala de aula, permitindo uma visão cronológica de todo o processo. Conclui-se que o tempo de desenvolvimento foi em torno de 30 horas.

Código:

Foi utilizado o modelo GPT-3.5 da OpenAI (ChatGPT) como ferramenta de auxílio na elaboração do código, correção de funções e lógicas do algoritmo.

É de suma importância que para utilizar o código você deve instalar as seguintes bibliotecas no terminal: `` pip install networkx pyvis matplotlib ``

E o seguinte na pasta /telegramCSVeTXT: `` pip install python-telegram-bot==13.13 pillow ``

Além de colocar meu token ao rodar o código do Bot do Telegram, eu recomendo abrir 2 terminais, um para testar o grafo.py e outro para deixar o bot funcionando enquanto faz o envio do arquivo em .csv ou .txt.

Link para o Repositório:

O código está disponível em link para o repositório no GitHub ([Github - Atividade Avaliativa Grafos](#)).