MANUAL TECNICO

VISUALIZACIÓN DE GRAFOS EN PYTHON

Introducción

Propósito del Manual:
Este manual técnico tiene como objetivo detallar el funcionamiento del proyecto Visualización de Grafos en Python, describiendo la arquitectura del código, la lógica detrás de sus funciones principales y las configuraciones necesarias para su ejecución. Este documento está dirigido a programadores y estudiantes interesados en la teoría de grafos y la visualización de grafos dirigidos.

Objetivo del Proyecto:
El proyecto busca crear una aplicación interactiva que permita construir grafos dirigidos, mediante la adición de vértices y aristas, y visualizarlos en una interfaz gráfica. La aplicación es ideal para visualizar la teoría de grafos en contextos educativos y experimentales.

Alcance:
El programa está diseñado para usuarios de nivel intermedio en programación, familiarizados con Python y conceptos básicos de teoría de grafos. Se recomienda su uso en entornos de aprendizaje, tanto para estudiantes como para educadores que desean visualizar grafos.

Funcionalidades Principales del programa:

Añadir Vértices: Permite ingresar y almacenar vértices en una lista visible en la interfaz.

Añadir Aristas: Permite conectar los vértices a través de aristas dirigidas, especificando la dirección en el formato "A -> B".

Generar y Visualizar Grafo: Construye el grafo dirigido en base a los vértices y aristas ingresados y lo visualiza en una ventana gráfica.

Descripción del Entorno de Ejecución

- **Lenguaje de Programación:** Python 3.7 o superior.
- **Bibliotecas Utilizadas:**
  - **tkinter:** Para construir la interfaz gráfica.
  - **networkx:** Para la creación y manipulación de grafos.
  - **matplotlib.pyplot:** Para la visualización gráfica del grafo.
- **Sistema Operativo Recomendado:** Windows, macOS o Linux, siempre que sea compatible con Python 3 y las bibliotecas necesarias.

Requisitos de Hardware y Software

- **Requisitos de Hardware:** No hay requisitos específicos de hardware, pero una resolución de pantalla adecuada facilitará la visualización del grafo.
- **Requisitos de Software:**
  - Python 3.7 o superior: Es necesario tener instalado Python en el sistema.
  - Bibliotecas Externas: networkx y matplotlib. Estas se instalan fácilmente con pip, el gestor de paquetes de Python.

Configuración del programa:

Instalación de Python

Si Python no está instalado, descárgalo desde python.org y sigue las instrucciones de instalación para tu sistema operativo.

Instalación de Dependencias

Instala las bibliotecas requeridas ejecutando el siguiente comando en la terminal: **pip install networkx matplotlib**

El programa comienza con la importación de las bibliotecas tkinter, networkx y matplotlib.pyplot:

```
import tkinter as tk
from tkinter import messagebox
import networkx as nx
import matplotlib.pyplot as plt
```

- **tkinter:** Es la biblioteca estándar de Python para interfaces gráficas. En este proyecto se usa para construir la ventana principal, los botones, y los campos de entrada de datos.
- **networkx**: Permite crear y manipular grafos. En este caso, facilita la creación de grafos dirigidos mediante el objeto DiGraph.
- **matplotlib.pyplot:** Se utiliza para visualizar el grafo final en una ventana gráfica.

Funciones del programa:

Función agregar_vertice:

```
def agregar_vertice():
    vertice = entry_vertice.get()
    if vertice:
        lista_vertices.insert(tk.END, vertice)
        entry_vertice.delete(0, tk.END)
    else:
        messagebox.showwarning("Advertencia", "Ingrese un vértice válido")
```

- **Objetivo**: Permite al usuario ingresar un vértice, verificar si es válido y agregarlo a una lista visible en la interfaz.
- **Lógica:**
- Obtiene el valor del campo entry_vertice.
- Si el valor es válido (no está vacío), lo agrega a lista_vertices, que es una Listbox en la interfaz.
- Si no es válido, muestra un mensaje de advertencia usando messagebox.

Funciones del programa:

Función agregar_arista:

```
def agregar_arista():
    arista = entry_arista.get()
    if arista:
        lista_aristas.insert(tk.END, arista)
        entry_arista.delete(0, tk.END)
    else:
        messagebox.showwarning("Advertencia", "Ingrese una arista válida")
```

- **Objetivo:** Similar a agregar_vertice, esta función permite al usuario ingresar aristas y las almacena en lista_aristas.
- **Lógica:**
- Verifica que el campo entry_arista no esté vacío.
- Agrega el valor en el formato A -> B en lista_aristas.

Funciones del programa:

Función generar_grafo:

```
def generar_grafo():
    G = nx.DiGraph()  # Grafo dirigido
    # Agregar vértices
    for v in lista_vertices.get(0, tk.END):
        G.add_node(v)
    # Agregar aristas
    for a in lista_aristas.get(0, tk.END):
        u, v = a.split('->')
        G.add_edge(u.strip(), v.strip())

    # Dibujar el grafo
    nx.draw(G, with_labels=True, node_color='skyblue',
node_size=700, font_size=10, font_weight='bold')
    plt.show()
```

- **Objetivo:** Construye el grafo dirigido con los vértices y aristas almacenados y lo visualiza.
- **Lógica:**
- **nx.DiGraph():** Inicializa un grafo dirigido.
- Añadir vértices: Itera sobre lista_vertices, agregando cada elemento como un nodo en el grafo.
- Añadir aristas: Para cada arista en lista_aristas, separa los vértices u y v usando split('->'), y luego agrega la arista entre u y v.
- Visualización: Utiliza nx.draw() para representar el grafo, personalizando el color de los nodos y el tamaño de la fuente. Finalmente, muestra el grafo en una ventana gráfica con plt.show().

Funciones del programa:

Interfaz Gráfica:

```
root = tk.Tk()
root.title("Algoritmo de Grafos")
```

La interfaz gráfica es la ventana principal donde se ubican las etiquetas, botones y listas, todos organizados en un diseño de filas y columnas.

```
tk.Label(root, text="Vértice").grid(row=0, column=0)
entry_vertice = tk.Entry(root)
entry_vertice.grid(row=0, column=1)
tk.Button(root, text="Agregar Vértice",
command=agregar_vertice).grid(row=0, column=2)
```

- Se crean etiquetas (Label), campos de entrada (Entry), y botones (Button) para interactuar con la interfaz. Cada elemento es añadido a la ventana root y se ubica en una posición específica mediante el método grid.