

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

MATEMATICAS PARA COMPUTACION 2

CATEDRÁTICO: ING. JOSÉ ALFREDO GONZÁLES

TUTOR ACADÉMICO: ROBERTO CARLOS GÓMEZ DONIS



MANUAL TECNICO
PROGRAMA DE GRAFOS
Y ALGORITMO DE
PROFUNDIDAD Y
ANCHURA

Eduardo Antonio Reyes Pineda	202200091
Pablo Roberto García Sanun	202300448
Cecilia Judith Diéguez Solís	201801041
Rene Alejandro Yuman Barco	202307687
Samuel Alejandro de León Maldonado	202111587

GUATEMALA, 28 DE ABRIL DEL 2,024

ÍNDICE

ÍNDICE.....	2
OBJETIVOS DEL SISTEMA.....	3
GENERAL.....	3
ESPECÍFICOS.....	3
INTRODUCCIÓN.....	3
INFORMACION DEL SISTEMA.....	4
EXPLORACIÓN DETALLADA DE LOS CÓDIGOS FUENTE.....	4
LINK DEL VIDEO.....	8

OBJETIVOS DEL SISTEMA

GENERAL

Desarrollar un sistema de interfaz gráfica en Python utilizando el módulo tkinter para la gestión y visualización de datos relacionados con grafos, permitiendo al usuario ingresar y almacenar vértices y aristas, así como realizar búsquedas en anchura y profundidad sobre el grafo generado.

ESPECÍFICOS

- **Objetivo 1:** Crear ventanas de entrada de datos para la inserción de vértices y aristas, facilitando al usuario el ingreso de información como el nombre del vértice, su posición en la fila y columna, y los nombres de los vértices que conforman una arista.
- **Objetivo 2:** Implementar funciones que permitan recibir los datos ingresados por el usuario, almacenarlos adecuadamente utilizando clases como CrearVertice y CrearArista, y cerrar las ventanas de entrada una vez que se ha completado la operación de inserción de datos.

INTRODUCCIÓN

Este manual técnico proporciona una guía detallada para el desarrollo y uso de un sistema de interfaz gráfica en Python utilizando el módulo tkinter. El sistema está diseñado para la gestión y visualización de datos relacionados con grafos, permitiendo al usuario ingresar, almacenar y manipular vértices y aristas, así como realizar operaciones de búsqueda en anchura y profundidad sobre el grafo generado. Con una interfaz intuitiva y funcionalidades bien definidas, este manual busca brindar a desarrolladores y usuarios una comprensión completa del sistema y sus capacidades, así como instrucciones detalladas para su implementación y uso eficiente.

INFORMACIÓN DEL SISTEMA

El sistema desarrollado en Python utilizando el módulo tkinter proporciona una interfaz gráfica intuitiva y funcional para la gestión de grafos. Permite al usuario ingresar y almacenar vértices y aristas, así como realizar operaciones de búsqueda en anchura y profundidad. La estructura del sistema se basa en una arquitectura modular que facilita su mantenimiento y extensión, mientras que la implementación de clases y métodos bien definidos garantiza un código organizado y fácilmente comprensible.

EXPLORACIÓN DETALLADA DE LOS CÓDIGOS FUENTE

1. AlmacenarDatos, CrearVertice y CrearArista. La clase AlmacenarDatos gestiona la información de vértices y aristas, con métodos para agregar, obtener y vaciar elementos de listas. Las clases CrearVertice y CrearArista son modelos simples para representar vértices y aristas, con métodos `__init__()` que inicializan atributos específicos para cada uno.

```
1  class AlmacenarDatos:
2      definición __en eso__(ser):
3          ser.Vértices = []
4          ser.Aristas = []
5
6      definición LlenarLista(ser,vértice):
7          ser.Vértices.adjunatar(vértice)
8
9      definición RetornarÚltimoVértice(ser):
10         si ser.Vértices:
11             ultimo_vertice = ser.Vértices[-1]
12             devolver ultimo_vertice.nombre,ultimo_vertice.fila,ultimo_vertice.columna
13         demás:
14             devolver Ninguno
15
16     definición RetornarLista(ser):
17         devolver ser.Vértices
18
19     definición LlenarListaAristas(ser,Arista):
20         ser.Aristas.adjunatar(Arista)
21
22     definición RetornarUltimaArista(ser):
23         si ser.Aristas:
24             ultima_arista = ser.Aristas[-1]
25             devolver ultima_arista.NombreVérticeInicial,ultima_arista.NombreVérticeFinal
26         demás:
27             devolver Ninguno
28
```

2. La clase `BusquedaEnAnchura` se encarga de realizar búsquedas en anchura o profundidad en un grafo representado por una lista de aristas. En su método `__init__()` se inicializan variables como la instancia de `AlmacenarDatos`, la lista de aristas y el tipo de búsqueda. `BuscarTodosLosVertices` identifica todos los vértices presentes en el grafo y los almacena. `DefinirConexiones` crea una lista de conexiones entre los vértices. `CrearListaParaGrafoAnchura()` y `CrearListaParaGrafoProfundidad()` realizan la búsqueda en anchura o profundidad respectivamente. Además, hay métodos para buscar aristas y aristas iniciales durante la búsqueda. Las clases `ConexionVertices` y `CrearNuevasAristas` ayudan a representar conexiones entre vértices y a crear nuevas aristas durante la búsqueda, respectivamente.

```
1  class BusquedaEnAnchura:
2      definición __en eso__(ser,instancia,lista,tipo):
3          ser.AlmacenarDatos = instancia
4          ser.ListaAristas = lista
5          ser.TipoBusqueda = tipo
6          ser.Vértices = []
7          ser.ListaConexiones = []
8          ser.NuevasAristas = []
9          ser.VérticesTotales = []
10         ser.VérticeAnterior = ""
11         ser.BuscarTodosLosVertices()
12
13     definición BuscarTodosLosVertices(ser):
14         para i en ser.ListaAristas:
15             si len(ser.Vértices)> 0:
16                 si i.NombreVérticeInicial no en ser.Vértices:
17                     ser.Vértices.adjuntar(i.NombreVérticeInicial)
18                 si i.NombreVérticeFinal no en ser.Vértices:
19                     ser.Vértices.adjuntar(i.NombreVérticeFinal)
20             demás:
21                 ser.Vértices.adjuntar(i.NombreVérticeInicial)
22                 ser.Vértices.adjuntar(i.NombreVérticeFinal)
23
24         ser.Vértices.clasificar()
25         para k en ser.Vértices:
26             ser.VérticesTotales.adjuntar(cadena(k))
27     --
```

3. La clase `InterfazGrafica` define una interfaz gráfica utilizando el módulo `customtkinter`, donde se crea una ventana con diversos elementos como cuadros de texto, botones y lienzo para dibujar. Se proporcionan métodos para agregar texto a los cuadros de texto, dibujar vértices y aristas en el lienzo, limpiar la interfaz, obtener datos de vértices y aristas, y ejecutar algoritmos de búsqueda en anchura o profundidad. La ventana se

inicializa con una configuración específica y se ejecuta el bucle principal de la interfaz. Cada vez que se agrega un vértice o una arista, se actualiza la visualización en el lienzo correspondiente, y al ejecutar los algoritmos de búsqueda, se muestran los resultados en el lienzo de salida.

```

1  import personalizado como ctk
2  import tkinter
3  import tkinter as tk
4  de SolicitarDatosVertice importar *
5  de SolicitarDatosAristas importar *
6  de AlmacenarDatos importar *
7  de Búsqueda importar *
8
9  class InterfazGráfica:
10     def __init__(self):
11         self.ConfiguraciónInicial()
12         self.CrearCuadros()
13         self.CrearBotones()
14         self.dibujar_cuadricula()
15         self.AlmacenamientoDeDatos = AlmacenarDatos()
16
17     def ConfiguraciónInicial(self):
18         self.VentanaInterfaz = ctk.CTk()
19         ctk.establish_mode_appearance("luz")
20         ctk.set_default_color_theme("verde")
21         self.VentanaInterfaz.resizable(0,0)
22         self.VentanaInterfaz.title("Grafos")
23         self.anchura_ventana = 1200
24         self.alto_ventana = 675
25         self.anchura_pantalla = self.VentanaInterfaz.winfo_screenwidth()
26         self.alto_pantalla = self.VentanaInterfaz.winfo_screenheight()
27         self.posición_x = (self.anchura_pantalla - self.anchura_ventana) // 2

```

- Este bloque de código utiliza PyInstaller para empaquetar un programa Python en un archivo ejecutable independiente. Se inicia con la configuración de una instancia de la clase Analysis para analizar los módulos y sus dependencias. Luego, se definen las opciones de empaquetado, como la inclusión de archivos binarios y datos, la exclusión de ciertos módulos, la compresión con UPX, y la configuración del archivo ejecutable final, como el nombre, la depuración y la habilitación de la consola. Al final, PyInstaller crea el archivo ejecutable con todas las especificaciones dadas.

```

1      # -*- modo: python; codificación: utf-8 -*-
2
3
4  ✓  a = Análisis(
5      ['InterfazGráfica.py'],
6      patex=[],
7      binarios=[],
8      datos=[],
9      importaciones ocultas=[],
10     camino de gancho=[],
11     configuración de ganchos={},
12     runtime_hooks=[],
13     excluye=[],
14     sin archivo=FALSO,
15     optimizar=0,
16     )
17     pyz = PIZ(a.puro)
18
19  ✓  exe = EXE(
20     pyz,
21     a.guiones,
22     a.binarios,
23     a.datos,
24     [],
25     nombre='InterfazGráfica',
26     depurar=FALSO,
27     gestor de arranque_ignorar_señales=FALSO,

```

5. Este código define una clase VentanaCrearArista que crea una ventana de interfaz gráfica usando el módulo tkinter para solicitar datos de una arista. La ventana solicita al usuario que ingrese el nombre del vértice inicial y final, luego, al presionar un botón, los datos ingresados se envían a una instancia de AlmacenarDatos para ser almacenados como una nueva arista. Una vez enviado, la ventana se cierra.

```

1      importar tkinter como tk
2      de AlmacenarDatos importar CrearArista
3
4  ✓  clase VentanaCrearArista:
5      definición __en eso__(ser):
6          ser.ConfiguraciónInicial()
7          ser.CreandoEspacios()
8
9      definición Recibir Instancia(ser,instancia):
10         ser.AlmacenarDatos = instancia
11
12  ✓  definición ConfiguraciónInicial(ser):
13         ser.VentanaInterfaz = tk.tk()
14         ser.VentanaInterfaz.titulo("Datos para vértice")
15         ser.ancha_ventana = 300
16         ser.alto_ventana = 230
17         ser.ancha_pantalla = ser.VentanaInterfaz.winfo_screenwidth()
18         ser.alto_pantalla = ser.VentanaInterfaz.winfo_screenheight()
19         ser.posición_x =(ser.ancha_pantalla - ser.ancha_ventana)// 2
20         ser.posición_y =(ser.alto_pantalla - ser.alto_ventana)// 2
21         ser.VentanaInterfaz.geometría(F"{ser.ancha_ventana}X{ser.alto_ventana}+{ser.posición_x}+{ser.posición_y}")
22
23  ✓  definición CreandoEspacios(ser):
24         ser.etiqueta_vertice_inicial = tk.Etiqueta(maestro=ser.VentanaInterfaz,texto="Ingrese el nombre del vértice IN
25         ser.etiqueta_vertice_inicial.embalar(padx=20,arroz=(20,0))
26         ser.campo_vertice_inicial = tk.Entrada(maestro=ser.VentanaInterfaz, ancho=30)
27         ser.campo_vertice_inicial.embalar(padx=20,arroz=(0,10))

```

6. Este código define una clase `VentanaCrearVertice` que crea una ventana de interfaz gráfica utilizando `tkinter` para solicitar datos de un vértice, incluyendo su nombre, fila y columna. Una vez que el usuario completa los campos y presiona el botón "Agregar vértice", los datos ingresados se envían a una instancia de `AlmacenarDatos` para ser almacenados como un nuevo vértice. Luego de enviar los datos, la ventana se cierra.

```
1  importar tkinter como tk
2  de AlmacenarDatos importar CrearVértice
3
4  class VentanaCrearVértice:
5      definición __en eso__(ser):
6          ser.ConfiguraciónInicial()
7          ser.CreandoEspacios()
8
9      definición Recibir Instancia(ser,instancia):
10         ser.AlmacenarDatos = instancia
11
12     definición ConfiguraciónInicial(ser):
13         ser.VentanaInterfaz = tk.tk()
14         ser.VentanaInterfaz.titulo("Datos para vértice")
15         ser.anchos_ventana = 300
16         ser.alto_ventana = 350
17         ser.anchos_pantalla = ser.VentanaInterfaz.winfo_screenwidth()
18         ser.alto_pantalla = ser.VentanaInterfaz.winfo_screenheight()
19         ser.posición_x =(ser.anchos_pantalla - ser.anchos_ventana)// 2
20         ser.posición_y =(ser.alto_pantalla - ser.alto_ventana)// 2
21         ser.VentanaInterfaz.geometría(F"{ser.anchos_ventana}X{ser.alto_ventana}+{ser.posición_x}+{ser.posición_y}")
22
23     definición CreandoEspacios(ser):
24         ser.etiqueta_nombre = tk.Etiqueta(maestro=ser.VentanaInterfaz,texto="Ingrese el nombre del vértice:")
25         ser.etiqueta_nombre.embalar(padx=20,arroz=(20,0))
26         ser.campo_nombre = tk.Entrada(maestro=ser.VentanaInterfaz, ancho=30)
27         ser.campo_nombre.embalar(padx=20,arroz=(0,10))
```

LINK DEL VIDEO

<https://drive.google.com/file/d/1P-9A9KdJ9QWXvvJi-pJ--jkoPjJpFDPb/view?usp=drivesdk>