UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS

NOMBRE: **ANDY ROBERTO JIMENEZ MACNAB**

CARNÉ: **202111490**

NOMBRE: **ISAI DARDON MAYEN**

CARNÉ: **202200033**

MANUAL TECNICO

QUE ES EL ALGORITMO DE BUSQUEDA DE ANCHURA

El algoritmo de búsqueda en anchura (también conocido como BFS por sus siglas en inglés) es un algoritmo utilizado en teoría de grafos para recorrer y buscar todos los vértices de un grafo. Este algoritmo comienza por un vértice inicial y explora todos los vecinos de ese vértice antes de avanzar al siguiente nivel de vecinos. De esta manera, recorre el grafo en forma de capas concéntricas, desde el vértice inicial hasta los vértices más alejados.

El algoritmo de búsqueda en anchura es útil para encontrar el camino más corto entre dos vértices en un grafo no ponderado, es decir, en un grafo en el que todas las aristas tienen el mismo peso. Además, también se utiliza en la resolución de problemas de optimización, en la detección de ciclos en grafos y en la búsqueda de componentes conexas.

El algoritmo de búsqueda en anchura es fácil de implementar y utiliza una estructura de datos llamada cola (o fila), que permite mantener un registro de los vértices visitados y pendientes de explorar en cada nivel del grafo.

DESCRIPCION DEL PROGRAMA

Se solicita el vértice que va ingresar al programa. También solicita la Arista en el formato A->B y un espacio para el peso de la arista. En el lado derecho se tiene un resumen de los vértices y las aristas que se han ingresado; y finalmente un botón que genera el grafo solicitado. En la parte superior se puede apreciar ambos grafos. Del lado izquierdo el grafo ingresado y a su derecha el grafo con el algoritmo de Prim aplicado. No todos los algoritmos son aplicables a los mismos grafos, por lo que no todos los programas requieren de los mismos campos para el ingreso del grafo. Los algoritmos de búsqueda no requieren de peso en sus aristas. Los árboles AVL no requieren peso ni ingresar aristas directamente.

A continuación se dejara procedimiento y explicación de como funciona el programa, donde este fue realizado es en Python.

Este código es un fragmento de código Python que utiliza la biblioteca Tkinter para crear una ventana principal y mostrar una interfaz gráfica de usuario. En particular, este código importa la clase Tk de la biblioteca Tkinter y la clase Principal del módulo app.mainView.

El código se ejecuta cuando se invoca como un script principal (en lugar de ser importado como un módulo) y comienza por crear una instancia de la clase Tk. Luego, crea una instancia de la clase Principal, pasando como argumento la instancia de la clase Tk recién creada. Finalmente, el bucle principal de la ventana (mainloop) se inicia y se mantiene en ejecución hasta que el usuario cierra la ventana.

La clase Principal que se crea en este código probablemente contiene la lógica y la interfaz gráfica necesarias para el programa en cuestión. La instancia de la clase Tk se utiliza como un objeto padre para la ventana principal, y le permite a la instancia de la clase Principal tener un marco en el que dibujar la interfaz gráfica.

En resumen, este código sirve para crear una ventana principal y mostrar una interfaz gráfica de usuario en Python utilizando la biblioteca Tkinter.

```python
from tkinter import Tk
from app.mainView import Principal

if __name__=="__main__":

    wPrincipal=Tk()
    application=Principal(wPrincipal)
    wPrincipal.mainloop()
```

Este código que se realizo es una aplicación básica de interfaz gráfica de usuario (GUI) que permite al usuario generar un grafo y aplicar un algoritmo de búsqueda de anchura (BFS) en ese grafo.

En la primera parte del código, se importan los módulos necesarios de la biblioteca tkinter, incluyendo Tk (para crear una ventana principal), Button (para crear botones), scrolledtext (para crear un área de texto con desplazamiento), Label (para crear etiquetas de texto), Entry (para crear campos de entrada de texto) y messagebox (para mostrar mensajes emergentes). También se importan las funciones generarGrafo y generarAlgoritmo del módulo Grafo y del módulo controlador, respectivamente.

La clase "Principal" es el marco principal de la aplicación. En el método Init, se inicializa la ventana principal y se configura su título y capacidad de redimensionamiento. Luego, se crea un marco para los botones y se configura su tamaño.

Se crearon dos botones: uno para generar el grafo y otro para generar el algoritmo de búsqueda de anchura. Se configura el tamaño de los botones y se les asigna una función de llamada cuando se hace clic en ellos.

También se crean dos etiquetas para solicitar al usuario que ingrese los vértices y aristas del grafo, y dos campos de entrada de texto para que el usuario ingrese estos datos.

En el método pasoDatos, se obtienen los valores ingresados por el usuario para los vértices y las aristas y se llama a la función generarGrafo para crear un grafo a partir de estos datos. Si se genera correctamente, se habilita el botón de generación de algoritmo, de lo contrario, se muestra un mensaje emergente informando que ha ocurrido un error. En el método generoAlgoritmo, se llama a la función generarAlgoritmo con el grafo generado en el paso anterior para aplicar el algoritmo de búsqueda de anchura en el grafo.

A continuación se dejara el codigo del cual se esta hablando:

```python
from tkinter import Button, scrolledtext as st
from tkinter import Label, Entry,messagebox
from tkinter import messagebox as messagebox
from tkinter import Frame
from controller.Grafo import generarGrafo,generarAlgoritmo

class Principal:
    def __init__(self,window):
        self.principal = window
        self.principal.title("Algoritmo de busqueda de anchura")
        self.principal.resizable(width=False,height=False)

        self.frameFunciones= Frame(self.principal)
        self.frameFunciones.grid(row=0,column=0)
        self.frameFunciones.config(width="250",height="300" )

        #Botones
        self.generarGrafo=Button(self.principal,text="Generar Grafo",command=lambda:self.pasoDatos(),
                                 width=16,height=4)
        self.generarGrafo.grid(row=0,column=2,padx=15)

        self.generarAlgoritmo=Button(self.principal,text="Generar BFS", command=lambda:self.generoAlgoritmo(), width=16,height=4)
        self.generarAlgoritmo.configure(state="disabled")
        self.generarAlgoritmo.grid(row=1,column=2,padx=15,pady=15)

        #Labels
        self.aristas=Label(self.principal,text="Ingrese las aristas: (AB,BC,CB)")
        self.aristas.grid(row=1,column=0)
        self.vertices=Label(self.principal,text="Ingrese los vertices: (A,B,C)")
        self.vertices.grid(row=0,column=0)

        #Entrys
        self.verticesEntry=Entry()
        self.verticesEntry.grid(row=0,column=1)

        self.aristasEntry=Entry()
        self.aristasEntry.grid(row=1,column=1)

    def pasoDatos(self):
        vertices=self.verticesEntry.get()
        aristas=self.aristasEntry.get()
        self.grafo=generarGrafo(vertices,aristas,self.generarAlgoritmo)
        if (self.grafo):
            print("todo bien")
        else:
            messagebox.showinfo(message="Ha ocurrido un error para generar el grafo", title="Error del Sistema")

    def generoAlgoritmo(self):
        generarAlgoritmo(self.grafo)

```

El código es una implementación del algoritmo de búsqueda de anchura (BFS) en un grafo, utilizando la biblioteca NetworkX para manejar el grafo y matplotlib.pyplot para visualizarlo. La función generarGrafo toma dos listas de entrada: listalistaVertices y listaAristaslistaAristas, que representan los vértices y las aristas del grafo, respectivamente. La función crea un objeto nx. Gráficonx.Graph() de NetworkX, agrega los nodos y las aristas a través de los métodos add_nodeadd_node() y add_edgeadd_edge() y, finalmente, utiliza nx.drawnx.draw() para visualizar el grafo. Si todo sale bien, la función habilita el botón habilitarBoton y devuelve el grafo.

La función generarAlgoritmoG como entrada y realiza un BFS en el grafo desde el nodo inicioinicio, que es el primer nodo en la lista de nodos del grafo. La función nx.bfs_treenx.bfs_tree() realiza el BFS y devuelve un subárbol BFS en forma de un objeto de gráfico. La función luego utiliza nx.drawnx.draw() para visualizar el subárbol y plt.showplt.show() para mostrar la visualización. La función no devuelve nada, ya que solo se utiliza para visualizar el resultado del BFS. En general, este código se utiliza para generar y visualizar grafos y para aplicar el algoritmo de búsqueda de anchura en ellos.

```python
import networkx as nx
import matplotlib.pyplot as plt

def generarGrafo(listaVertices,listaAristas,habilitarBoton):
    try:
        G = nx.Graph()

        vertices=listaVertices.split(",")

        for element in vertices:
            G.add_node(element)

        aristas=listaAristas.split(",")

        for elemento in aristas:
            G.add_edge(elemento[0],elemento[1])

        nx.draw(G, with_labels=True)
        habilitarBoton["state"]="normal"
        plt.figure(1)
        plt.show()
        return G

    except:
        return False


    #Realizamos el algoritmo de busqueda de anchura
def generarAlgoritmo(G):
    inicio= list(G.nodes)[0]

    ruta=nx.bfs_tree(G,inicio)

    nx.draw(ruta, with_labels=True)
    plt.figure(1)
    plt.show()
```

Lo que se hablo anteriormente fue por partes del codigo, ahora una breve explicación de los códigos juntos y como se implementan uno para el otro para que funcione adecuadamente.

Los tres códigos proporcionados están relacionados y se complementan entre sí para crear una aplicación completa de búsqueda de anchura.

- El primer código define la interfaz gráfica de usuario (GUI) utilizando la biblioteca Tkinter. Esta GUI tiene dos botones, uno para generar un gráfico y otro para ejecutar el algoritmo de búsqueda de anchura en el gráfico generado. El código también define las funciones para leer las entradas del usuario y generar el grafo y el algoritmo de búsqueda de anchura correspondiente.

- El segundo código genera el grafo utilizando la biblioteca NetworkX y lo visualiza utilizando la biblioteca Matplotlib. La función 'generarGrafo toma dos listas, una para los vértices y otra para las aristas, las procesa para crear un objeto de grafo de NetworkX y lo muestra en una ventana de gráfico utilizando Matplotlib. También habilita el botón para ejecutar el algoritmo de búsqueda de anchura.

- El tercer código realiza el algoritmo de búsqueda de anchura en el grafo utilizando la biblioteca NetworkX. La función 'generarAlgoritmo toma el objeto de grafo generado por la función 'generarGrafogenerarGrafo y lo procesa para ejecutar el algoritmo de búsqueda de anchura. El resultado se muestra en una ventana de gráfico utilizando Matplotlib.

En conclusión se puede decir que el primer código proporciona una interfaz de usuario para que el usuario ingrese los datos de entrada y ejecute el algoritmo de búsqueda de anchura. El segundo código genera el grafo y lo muestra en una ventana de gráfico. El tercer código ejecuta el algoritmo de búsqueda de anchura en el grafo generado y muestra el resultado en una ventana de gráfico. En conjunto, estos tres códigos forman una aplicación completa de búsqueda de anchura.

Finalmente el programa queda ejecutado y se ve de la siguiente manera, de igual manera en el otro documento se adjunto un MANUAL DE USUARIO donde explica la funcionalidad física de este.