

Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Matemática para Computación 2

Sección A

Inga. José Alfredo Gonzales

Segundo Semestre 2023



MANUAL DE USUARIO

APLICACIÓN DE ALGORITMO DE BUSQUEDA POR ANCHURA

Nombre: José Daniel Guzmán
Hernández

Registro académico: 202111612

Fecha: 02/11/2023

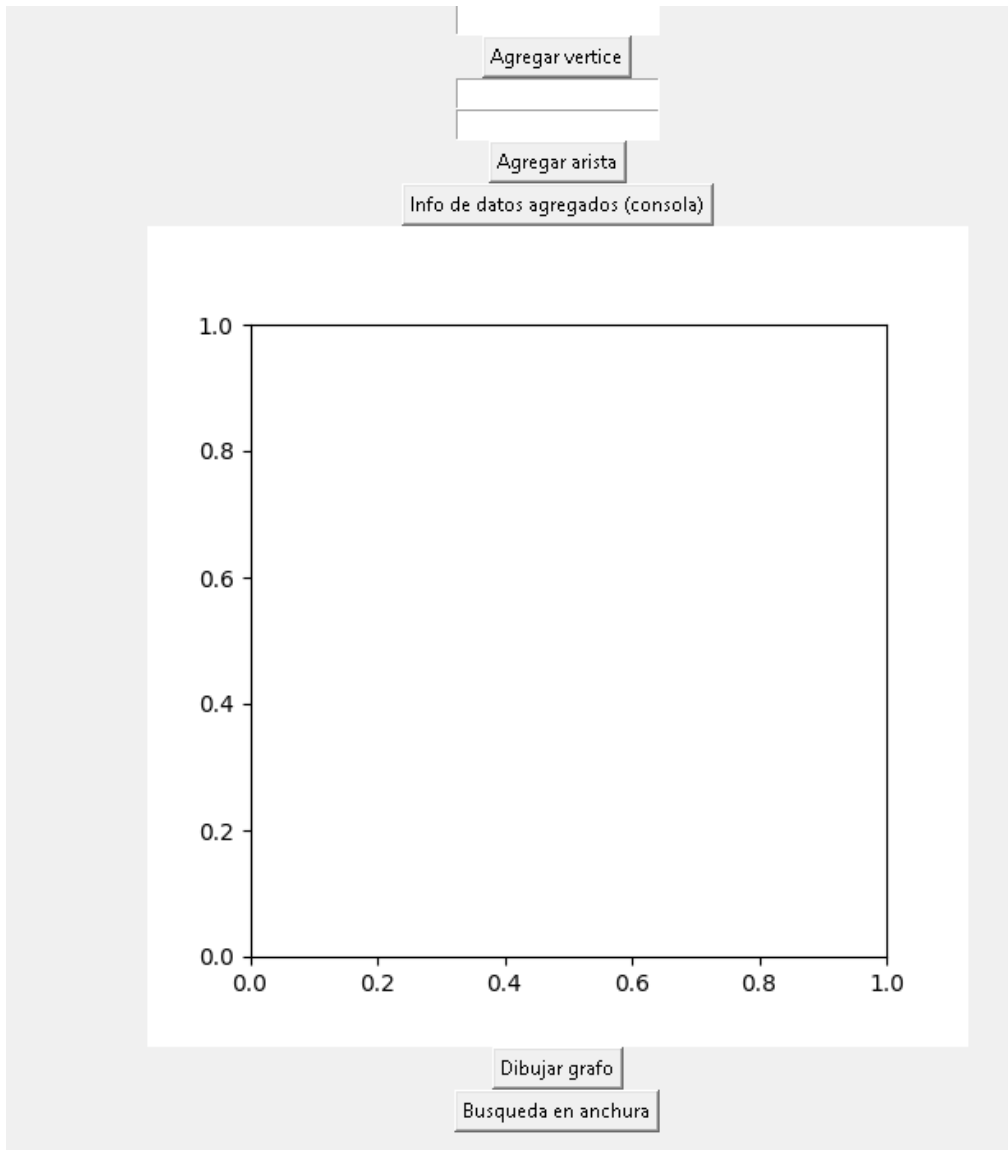
Sección: A

Funcionamiento de la aplicación

Nota

Al abrir el ejecutable esperar unos segundos a que aparezca la ventana del programa ya que tarda un poco en aparecer.

Esta será la ventana inicial que aparecerá



Ahora pulsaremos el botón dibujar grafo

The image shows a web application interface for graph management. It features a light gray background with a large white rectangular area in the center. At the top, there is a vertical stack of controls: an empty text input field, a button labeled 'Agregar vertice', another empty text input field, a third empty text input field, a button labeled 'Agregar arista', and a button labeled 'Info de datos agregados (consola)'. At the bottom, there are two buttons: 'Dibujar grafo' and 'Busqueda en anchura'. A blue arrow points from the left towards the 'Dibujar grafo' button.

Ahora agregamos el vértice que necesitamos y le damos al botón agregar vértice

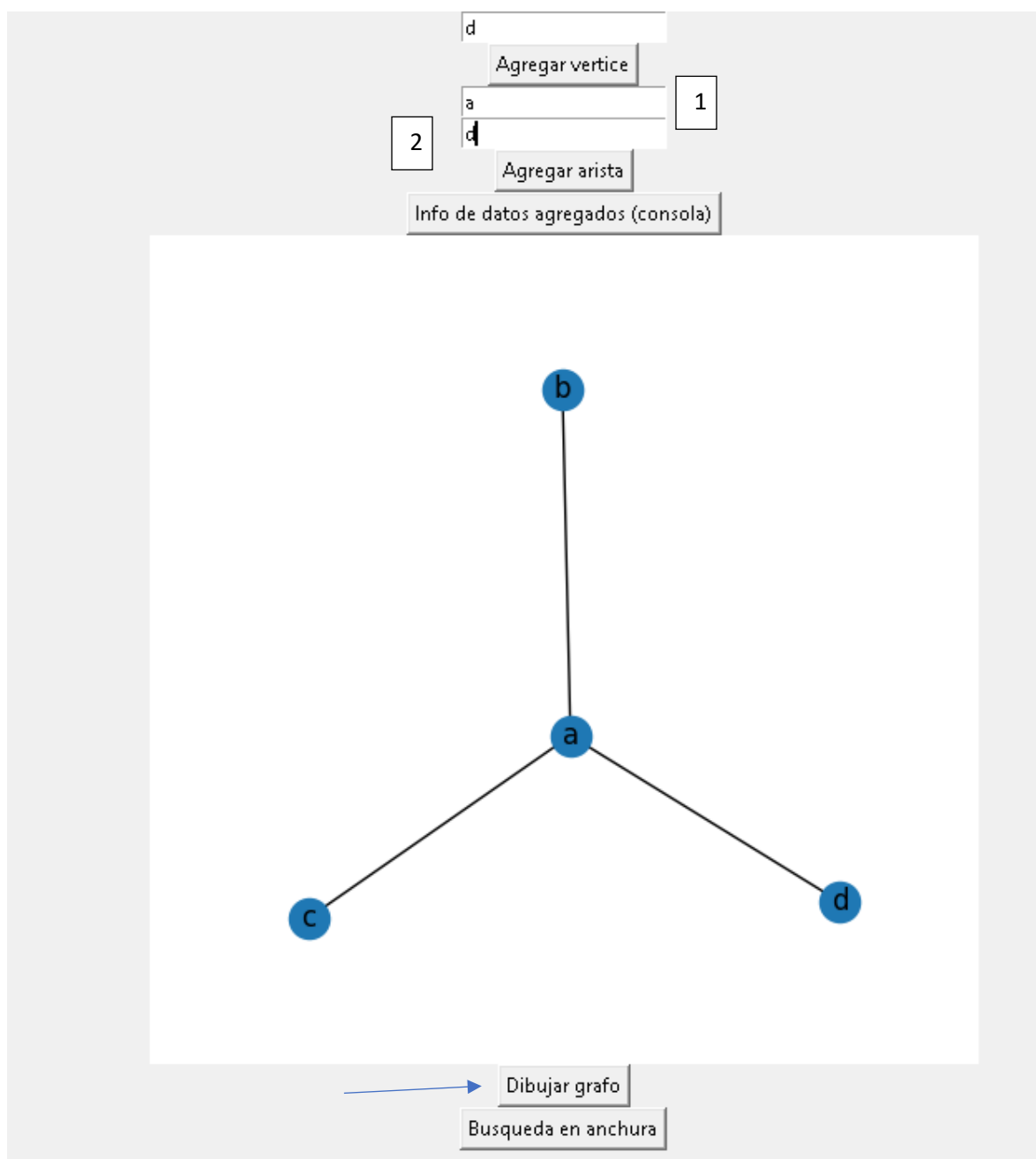
The interface contains the following elements:

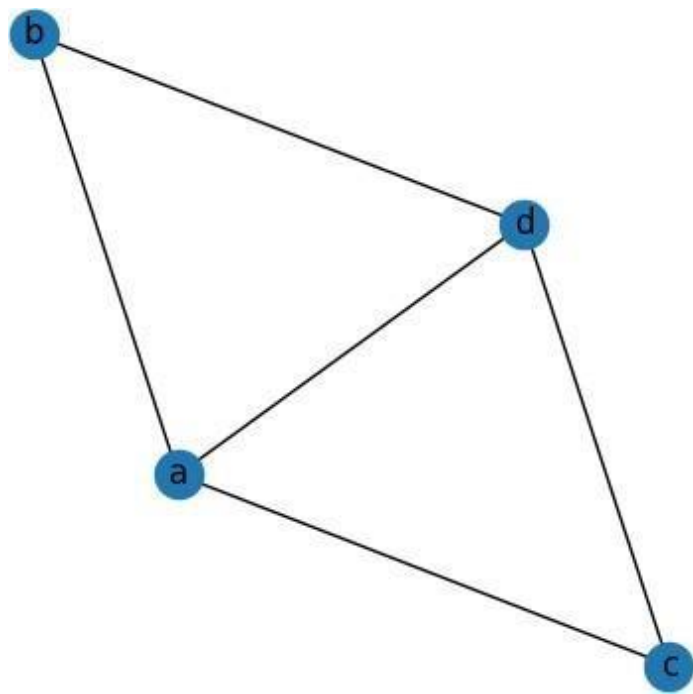
- A text input field at the top left containing the letter 'a', with a blue arrow pointing to it from the right.
- A button labeled 'Agregar vertice' located to the right of the input field.
- A single-line text input field directly below the 'Agregar vertice' button.
- A button labeled 'Agregar arista' located below the single-line input field.
- A button labeled 'Info de datos agregados (consola)' located below the 'Agregar arista' button.
- A large, empty white rectangular area in the center of the interface.
- A button labeled 'Dibujar grafo' located at the bottom center of the interface.
- A button labeled 'Busqueda en anchura' located directly below the 'Dibujar grafo' button.

Ahora le damos dibujar grafo y nos graficara el vértice

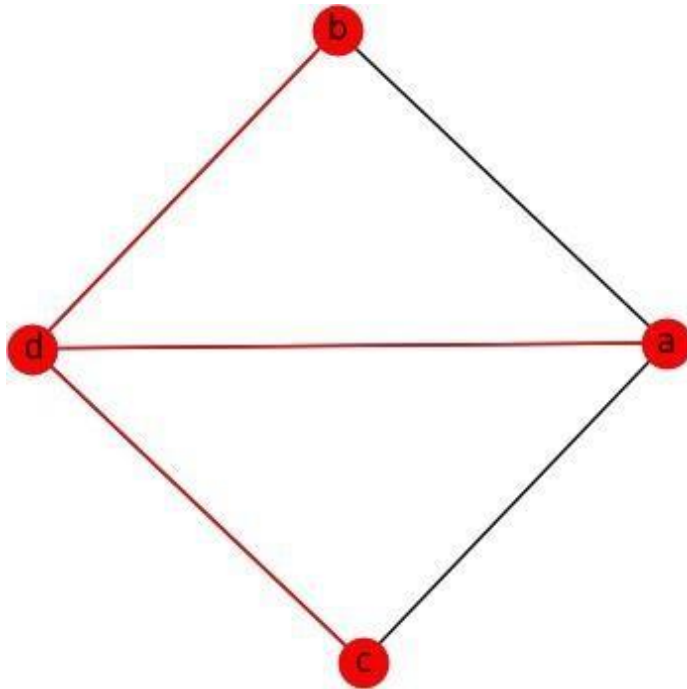
The screenshot shows a web-based application for graph management. At the top, there is a text input field containing the letter 'a', followed by an 'Agregar vertice' button. Below this is another empty text input field and an 'Agregar arista' button. A button labeled 'Info de datos agregados (consola)' is positioned below the 'Agregar arista' button. The central area of the application is a large white square canvas. In the center of this canvas is a single blue circular vertex labeled 'a'. At the bottom of the interface, there are two buttons: 'Dibujar grafo' and 'Busqueda en anchura'. A blue arrow points from the 'Dibujar grafo' button towards the central canvas area.

Para agregar aristas en la casilla uno va el vértice inicial y en la casilla dos el vértice final y le damos agregar arista y después al botón dibujar grafo





Ya teniendo el grafo le damos al botón búsqueda en anchura y el resultado será el grafo de contorno rojo



Funcionamiento técnico de la aplicación

Se utilizan las siguientes librerías para el correcto funcionamiento de la aplicación:

```
import networkx as nx
import tkinter as tk
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.figure import Figure
```

1. Tkinter: Es una biblioteca de GUI (Interfaz Gráfica de Usuario) estándar de Python que se utiliza para crear aplicaciones de escritorio con ventanas, botones, cajas de texto, etc. Esta biblioteca se basa en la biblioteca gráfica Tcl/Tk y se utiliza para crear interfaces de usuario.
2. NetworkX: Es una biblioteca de Python utilizada para la creación, manipulación y estudio de estructuras de red. Se utiliza para modelar relaciones entre objetos, como la interacción entre objetos y líneas, para este proyecto son vértices y aristas.
3. Matplotlib: Es una biblioteca de Python utilizada para crear visualizaciones en 2D y 3D de datos. Se utiliza comúnmente para crear gráficos, histogramas, diagramas de dispersión y otros tipos de visualizaciones para ayudar a visualizar datos.

Explicación del código:

```
5  G = nx.Graph()
6  root = tk.Tk()
7  root.title("Algoritmo de búsqueda en Anchura")
8  vertex_entry = tk.Entry(root)
9  vertex_entry.pack()
10 add_vertex_button=tk.Button(root, text="Agregar vertice", command=lambda:G.add_node(vertex_entry.get()))
11 add_vertex_button.pack()
12
13 edge_entry_1=tk.Entry(root)
14 edge_entry_1.pack()
15 edge_entry_2=tk.Entry(root)
16 edge_entry_2.pack()
17 add_edge_button = tk.Button(root, text="Agregar arista", command=lambda:G.add_edge(edge_entry_1.get(),edge_entry_2.get()))
18 add_edge_button.pack()
19
20 print_info_button=tk.Button(root, text="Info de datos agregados (consola)", command=lambda:print("Numero de nodos:",G.number_of_nodes()))
21 print_info_button.pack()
22
23 figure = Figure(figsize=(5,5))
24 ax = figure.add_subplot(111)
25 canvas=FigureCanvasTkAgg(figure, root)
26 canvas.get_tk_widget().pack()
27
```

Se inicializa la variable G que identifica un grafo de la librería networkx (nx.Graph)

Se inicializa la ventana de Tkinter root, aquí se insertan todos los objetos que dan forma a la interfaz grafica del usuario (tk.Tk)

Vertex_entry: Variable que contiene una casilla de texto de tipo entry, se utiliza escribir cualquier texto, se agrega a la ventana root

Add_vertex_button: Variable que contiene un botón de tipo Button (Tkinter), se utiliza para realizar alguna acción dentro de la ejecución del programa, en este caso, indica que se agregue la información contenida en vertex_entry al grafo de networkX G

Edge_entry_1: Variable que contiene una casilla de texto de tipo entry, se utiliza para escribir cualquier texto, se agrega a la ventana root

Edge_entry_2: Variable que contiene una casilla de texto de tipo entry, se utiliza para escribir cualquier texto, se agrega a la ventana root

Add_edge_button: Variable que contiene un botón de tipo Button (Tkinter), se utiliza para realizar alguna acción dentro de la ejecución del programa, en este caso, indica que se agregue la información contenida en Edge_entry_1 y Edge_entry_2 al grafo de networkX G

Print_info_button: Variable que contiene un botón de tipo Button (Tkinter), se utiliza para realizar alguna acción dentro de la ejecución del programa, en este caso se indica que, muestre la información agregada a G en la consola, ya sean vértices o aristas, únicamente de uso informativo.

Figure y ax: estas líneas de código indican que tipo de grafico generar y en que rango x,y

Canvas: se inicializa este objeto de matplotlib, que agrega los datos de figure y root al grafico y agregar la información cuando se llame

```
28 def draw_graph(bfs_edges=None):
29     ax.clear()
30     if bfs_edges:
31         pos = nx.spring_layout(G)
32         nx.draw(G, pos=pos, ax=ax, with_labels=True)
33         nx.draw_networkx_edges(G, pos=pos, edgelist=bfs_edges, edge_color='r', ax=ax)
34         nx.draw_networkx_nodes(G, pos=pos, nodelist=[vertex_entry.get()+[v for u, v in bfs_edges], node_color='r', ax=ax)
35     else:
36         nx.draw(G, ax=ax, with_labels=True)
37     canvas.draw()
```

Función draw_graph

```
28 def draw_graph(bfs_edges=None):
29     ax.clear()
30     if bfs_edges:
31         pos = nx.spring_layout(G)
32         nx.draw(G, pos=pos, ax=ax, with_labels=True)
33         nx.draw_networkx_edges(G, pos=pos, edgelist=bfs_edges, edge_color='r', ax=ax)
34         nx.draw_networkx_nodes(G, pos=pos, nodelist=[vertex_entry.get()+[v for u, v in bfs_edges], node_color='r', ax=ax)
35     else:
36         nx.draw(G, ax=ax, with_labels=True)
37     canvas.draw()
```

Esta función llamada draw_graph se utiliza para dibujar un grafo utilizando la librería networkx y la interfaz gráfica matplotlib.

El parámetro bfs_edges es una lista de aristas que representan el recorrido en anchura del grafo, si este parámetro tiene un valor, se dibujará el grafo con los nodos y aristas coloreados en rojo. Si el parámetro es None, se dibujará el grafo sin coloración.

Primero, se limpia el eje actual con el método `ax.clear()`.

Luego, si se proporciona una lista de aristas para un recorrido en anchura, se obtiene la posición de los nodos utilizando el algoritmo de `spring_layout` de `networkx`. A continuación, se dibuja el grafo utilizando `nx.draw`, pasando la posición de los nodos y el objeto `ax`.

Después, se dibujan las aristas del recorrido en anchura utilizando `nx.draw_networkx_edges`, pasando la lista de aristas `bfs_edges`, la posición de los nodos y el objeto `ax`.

Finalmente, se dibujan los nodos del recorrido en anchura utilizando `nx.draw_networkx_nodes`, pasando una lista que contiene el nodo de entrada y los nodos visitados en el recorrido.

Si no se proporciona la lista de aristas del recorrido en anchura, se dibuja simplemente el grafo utilizando `nx.draw`, y se actualiza el objeto `canvas` con el método `canvas.draw()`.

`Draw_button`: Variable que contiene un botón de tipo `Button` (Tkinter), se utiliza para realizar alguna acción dentro de la ejecución del programa, en este caso, indica que inicialice la función `draw_graph`

Función `bfs_show`

```
42 def show_bfs():
43     bfs_edges=list(nx.bfs_edges(G,source=vertex_entry.get()))
44     draw_graph(bfs_edges)
45     canvas.draw()
```

Se define una función llamada `show_bfs()` que utiliza la biblioteca `NetworkX` para realizar un recorrido BFS (Breadth-First Search o Búsqueda en Anchura) en un grafo llamado `G`. El vértice de partida del recorrido BFS es obtenido a través de la función `vertex_entry.get()`, que probablemente obtiene el valor de una entrada de texto en una interfaz gráfica de usuario.

Luego, se llama a la función `nx.bfs_edges(G,source=vertex_entry.get())` para realizar el recorrido BFS en el grafo `G` y obtener una lista de aristas que conforman el árbol BFS. Esta lista de aristas se almacena en la variable `bfs_edges`.

Después, se llama a la función `draw_graph(bfs_edges)` para dibujar el grafo en una interfaz gráfica de usuario. Si la variable `bfs_edges` tiene algún valor, se dibuja el árbol BFS en rojo, y los nodos correspondientes al recorrido BFS también se resaltan en rojo. Si `bfs_edges` está vacío, se dibuja simplemente el grafo original.

Finalmente, se llama a la función `canvas.draw()` para actualizar la interfaz gráfica y mostrar el grafo dibujado.

`root.mainloop()`: únicamente mantiene la ejecución de la ventana en uso hasta que se decida finalizar el programa.