Manual Técnico: Visualización de Algoritmos de Teoría de Grafos

Grupo de Trabajo - Matemática de Computación 2 A

Segundo Semestre 2024

Contents

Chapter 1

Introducción

El presente manual técnico describe la arquitectura, módulos, y algoritmos del programa desarrollado para la visualización interactiva de grafos. El enfoque principal es explicar la estructura del código y su implementación para facilitar futuras modificaciones o mejoras.

Chapter 2

Arquitectura del Sistema

El programa sigue una arquitectura modular, permitiendo separar la lógica de los algoritmos, la representación de los grafos y la interfaz gráfica. Cada módulo tiene responsabilidades claras que facilitan el mantenimiento del código.

2.1 Componentes Principales

- **main.py**: Archivo principal que inicia la aplicación y carga la interfaz gráfica.
- **graph.py**: Define la estructura del grafo y proporciona métodos para añadir vértices y aristas.
- **algorithms.py**: Implementa los algoritmos de búsqueda en anchura (BFS) y búsqueda en profundidad (DFS).
- **ui.py**: Gestiona la interfaz gráfica utilizando PyQt5 para la interacción con el usuario.

Chapter 3

Descripción de Módulos

A continuación se detallan los módulos clave del programa, su propósito y su contenido.

3.1 Módulo: `main.py`

Este archivo inicializa la aplicación y coordina las acciones entre la interfaz gráfica y la lógica interna del programa.

```
from ui import GraphicalInterface
from graph import Graph
from algorithms import bfs, dfs

if __name__ == "__main__":
    app = GraphicalInterface()
    app.run()
```

Listing 3.1: main.py

3.1.1 Funcionalidad

- Importa los módulos necesarios para manejar el grafo y los algoritmos.
- Crea una instancia de la interfaz gráfica y la ejecuta.

3.2 Módulo: `graph.py`

Este módulo contiene la clase `Graph`, que representa un grafo como una estructura de datos. Permite la manipulación de vértices y aristas, y mantiene una representación del grafo en memoria.

```python
class Graph:
    def __init__(self):
        self.vertices = {}

    def add_vertex(self, vertex):
        if vertex not in self.vertices:
            self.vertices[vertex] = []

    def add_edge(self, u, v):
        if u in self.vertices and v in self.vertices:
            self.vertices[u].append(v)
```

Listing 3.2: graph.py

3.2.1 Funcionalidad

- `add_vertex(vertex)`: Agrega un vértice al grafo.
- `add_edge(u, v)`: Agrega una arista entre dos vértices.
- Estructura interna del grafo: Diccionario donde las claves son vértices y los valores son listas de vértices adyacentes.

3.3 Módulo: `algorithms.py`

Este módulo contiene las implementaciones de los algoritmos de búsqueda en anchura y búsqueda en profundidad.

```python
def bfs(graph, start):
    visited = []
    queue = [start]

    while queue:
        vertex = queue.pop(0)
        if vertex not in visited:
            visited.append(vertex)
            queue.extend(set(graph[vertex]) - set(visited))

    return visited

def dfs(graph, start, visited=None):
    if visited is None:
        visited = []
    visited.append(start)

    for neighbor in graph[start]:
        if neighbor not in visited:
```

```
20              dfs(graph, neighbor, visited)
21
22      return visited
```

Listing 3.3: algorithms.py

3.3.1 Funcionalidad

- `bfs(graph, start)`: Implementa el algoritmo de búsqueda en anchura, recorriendo el grafo por niveles.
- `dfs(graph, start)`: Implementa el algoritmo de búsqueda en profundidad, recorriendo el grafo por ramas.
- Ambos algoritmos devuelven una lista de los vértices visitados en el orden correspondiente.

3.4 Módulo: `ui.py`

Este módulo gestiona la interfaz gráfica utilizando PyQt5. Proporciona una manera interactiva de ingresar vértices y aristas, así como ejecutar los algoritmos sobre el grafo.

```
1  from PyQt5 import QtWidgets
2  from graph import Graph
3  from algorithms import bfs, dfs
4
5  class GraphicalInterface:
6      def __init__(self):
7          self.graph = Graph()
8
9      def run(self):
10         # Configuraci n de la interfaz y eventos.
11         pass
```

Listing 3.4: ui.py

3.4.1 Funcionalidad

- Configura la ventana principal de la aplicación.
- Administra la interacción del usuario con la inserción de datos y visualización del grafo.

Chapter 4

Implementación de Algoritmos

Los algoritmos de búsqueda en anchura (BFS) y búsqueda en profundidad (DFS) se implementan utilizando estructuras de datos básicas (listas y conjuntos) para almacenar los vértices visitados y el orden de recorrido.

4.1 Búsqueda en Anchura (BFS)

El algoritmo BFS recorre el grafo nivel por nivel, comenzando desde el vértice inicial. Se utiliza una cola para gestionar los vértices que deben ser explorados.

4.2 Búsqueda en Profundidad (DFS)

El algoritmo DFS sigue una rama del grafo tan profundamente como sea posible antes de retroceder. Utiliza la recursión para explorar cada vértice.

Chapter 5

Recomendaciones para Extensiones

5.1 Soporte para Grafos Dirigidos

Actualmente, el programa trabaja con grafos no dirigidos. Para implementar grafos dirigidos, se deben modificar los métodos de `add_edge` para permitir aristas con dirección.

5.2 Optimización de la Interfaz

Para mejorar la experiencia de usuario, se recomienda incluir una visualización dinámica de los algoritmos, mostrando el avance paso a paso de cada algoritmo sobre el grafo.

Chapter 6

Conclusión

Este manual técnico detalla la arquitectura y funcionamiento interno del programa de visualización de grafos, proporcionando información clave para futuros desarrollos o mejoras. La estructura modular facilita la extensión del código, y los algoritmos implementados cumplen con el objetivo de demostrar los recorridos BFS y DFS de manera clara y eficiente.