Búsqueda en Grafos: BFS y DFS aplicados al Metro CDMX

Introducción

En este proyecto se implementaron los algoritmos de búsqueda en grafos Breadth-First Search (BFS) y Depth-First Search (DFS), aplicándolos al sistema del Sistema de Transporte Colectivo Metro en la Ciudad de México.

El objetivo fue modelar el Metro como un grafo y analizar formalmente:

- Completitud
- Optimalidad
- Complejidad en tiempo
- Complejidad en espacio

Además, se ejecutaron tres casos obligatorios para comparar experimentalmente ambos algoritmos.

Modelado del Grafo

El sistema fue modelado como un grafo:

$G=(V,E)$

Donde:

- V (nodos) = estaciones del Metro.
- E (aristas) = conexión directa entre estaciones contiguas de una misma línea.

Supuestos del modelo

- Grafo no dirigido (las conexiones son bidireccionales).
- Grafo no ponderado (cada tramo tiene costo 1).
- No se consideraron tiempos reales ni penalización por transbordo.

Este modelo representa correctamente la estructura física del sistema, donde las estaciones están conectadas linealmente dentro de cada línea y comparten nodos en estaciones de correspondencia.

Representación de Adyacencia

Se utilizó lista de adyacencia, implementada mediante un diccionario:

graph = {

"Observatorio": ["Tacubaya"],

"Tacubaya": ["Observatorio", "Juanacatlan", "Constituyentes"],

...

}

Justificación

El Metro es un grafo disperso, ya que:

- Cada estación tiene en promedio entre 2 y 4 conexiones.
- El número de aristas es mucho menor que $|V|^2$.

Por lo tanto, la lista de adyacencia es más eficiente que la matriz de adyacencia.

Ventajas

- Recorrer vecinos es eficiente.
- Ideal para BFS y DFS.

4. Algoritmos Implementados

4.1 BFS (Breadth-First Search)

- Utiliza una cola (queue).
- Marca nodos visitados.
- Guarda padres para reconstruir la ruta.
- Explora por niveles.

BFS encuentra primero todas las estaciones a distancia 1, luego distancia 2, etc.

4.2 DFS (Depth-First Search)

- Implementado de forma iterativa con pila (stack).
- Marca nodos visitados.
- Guarda padres.
- Explora lo más profundo posible antes de retroceder.

5. Teoría

5.1 Completitud

BFS

Un algoritmo es completo si garantiza encontrar solución cuando existe.

BFS es completo si:

1. El grafo es finito.
2. Se utiliza estructura de visitados para evitar ciclos.

En este problema:

- El número de estaciones del Metro es finito.
- Se implementó conjunto de visitados.

Por lo tanto, BFS es completo en este problema.

En grafos infinitos, BFS sigue siendo completo si la solución está a profundidad finita**.**

DFS

DFS también es completo en grafos finitos si:

- Se utiliza conjunto de visitados.

Sin visitados, DFS podría ciclar indefinidamente.

En grafos infinitos, DFS no es completo, ya que puede profundizar indefinidamente sin encontrar la solución, aunque exista en otra rama.

Conclusión:

- En el Metro CDMX (grafo finito), DFS es completo.
- En grafos infinitos, no necesariamente.

5.2 Optimalidad

BFS

En grafos no ponderados:

- Cada arista tiene costo 1.
- BFS explora por niveles.
- La primera vez que alcanza el objetivo es a profundidad mínima.

Por lo tanto:

BFS es óptimo en grafos no ponderados.

Encuentra el camino con menor número de aristas.

DFS

- No explora por niveles.
- Puede encontrar una ruta larga antes que una más corta.

Por lo tanto:

DFS no es óptimo.

No garantiza el menor número de aristas.

5.3 Complejidad en Tiempo

Con lista de adyacencia:

O(|V|+|E|)

Justificación:

- Cada nodo se visita una vez.
- Cada arista se examina como máximo dos veces (grafo no dirigido).

Esto aplica tanto para BFS como para DFS.

BFS

O(b^d)

Explora todos los nodos hasta profundidad d.

DFS

O(b^m)

Puede explorar hasta profundidad máxima.

5.4 Complejidad en Espacio

Ambos algoritmos requieren:

O(|V|)

Diferencias importantes

BFS

- La cola puede crecer hasta O(b^d)
- Puede consumir mucha memoria si la solución está lejos.

En el contexto del Metro:

- Si la ruta cruza muchas líneas, la frontera puede incluir muchas estaciones simultáneamente.

DFS

- La pila suele contener un solo camino activo.
- Espacio aproximado: O(m)

Generalmente usa menos memoria que BFS.

6. Resultados Experimentales

Se ejecutaron los siguientes casos obligatorios:

1. Observatorio → Ciudad Azteca
2. Indios Verdes → Velódromo
3. UAM-I → El Rosario

Para cada caso se reportó:

- Ruta encontrada
- Longitud en aristas
- Número de nodos visitados

7. Conclusiones

1. El Metro CDMX puede modelarse correctamente como un grafo no dirigido y no ponderado.
2. BFS es completo y óptimo en este contexto.
3. DFS es completo en grafos finitos, pero no es óptimo.
4. Ambos algoritmos tienen complejidad O(|V| + |E|) con lista de adyacencia.
5. BFS puede consumir más memoria debido al crecimiento de la frontera.
6. DFS depende fuertemente del orden de exploración.

En problemas donde se necesita la ruta más corta en número de estaciones, BFS es la mejor opción.

En problemas donde la memoria es limitada, DFS puede ser más conveniente.