Reporte Semana 5: BFS y DFS

Algoritmos de Exploración y Búsqueda en Grafos

📋 Resumen Ejecutivo

Semana: 5
Tema: Búsqueda en Amplitud (BFS) y Búsqueda en Profundidad (DFS)
Fecha: Diciembre 2025

🎯 Objetivos Cumplidos

- ☑ Implementar BFS (Búsqueda en Amplitud)
- ☑ Implementar DFS (Búsqueda en Profundidad - Recursivo e Iterativo)
- ☑ Detección de ciclos en grafos dirigidos
- ☑ Encontrar caminos más cortos (BFS)
- ☑ Análisis de complejidad temporal y espacial

💻 Implementación

Algoritmos Implementados

1. **BFS (Breadth-First Search)**
    - Exploración por niveles
    - Usa cola (Queue) - FIFO
    - Garantiza camino más corto
2. **DFS Recursivo**
    - Exploración en profundidad
    - Usa pila implícita (recursión)
    - Elegante y conciso
3. **DFS Iterativo**
    - Exploración en profundidad
    - Usa pila explícita (Stack)
    - Evita stack overflow
4. **Detección de Ciclos**
    - Usa DFS con estados
    - Detecta ciclos en grafos dirigidos
    - Estados: NO_VISITADO, EN_PROCESO, COMPLETADO

5. **Camino Más Corto**
    - Usa BFS
    - Reconstrucción de camino con padres
    - Óptimo para grafos no ponderados

📊 Resultados de Ejecución

Grafo de Prueba

```
A --- B --- D
|     |
C --- E --- G
      |
      F
```

Salida del Programa

```
🔍 Semana 5 - BFS y DFS

📊 Grafo: A-B-C-D-E-F-G

BFS desde A: A → B → C → D → E → F → G
DFS desde A: A → B → D → E → G → F → C
DFS Iterativo: A → B → D → E → G → F → C

Camino más corto A → G: A → B → E → G
Distancia: 3 aristas

¿Tiene ciclos? False

Grafo 1→2→3→1 tiene ciclos? True

☑ Programa completado!
```

📈 Análisis de Complejidad

BFS (Búsqueda en Amplitud)

Aspecto	Complejidad
Temporal	O(V + E)
Espacial	O(V)
Estructura	Cola (Queue)

Ventajas:

- ☑ Encuentra camino más corto
- ☑ Explora por niveles
- ☑ Útil para distancias mínimas

Desventajas:

- ❌ Usa más memoria en grafos anchos
- ❌ No óptimo para grafos profundos

DFS (Búsqueda en Profundidad)

Aspecto	Complejidad
Temporal	O(V + E)
Espacial	O(V) peor caso, O(h) promedio
Estructura	Pila (Stack) o Recursión

Ventajas:

- ☑ Eficiente en memoria para grafos profundos
- ☑ Ideal para detección de ciclos
- ☑ Útil para ordenamiento topológico

Desventajas:

- ❌ No garantiza camino más corto
- ❌ Puede causar stack overflow (versión recursiva)

🔍 Comparación BFS vs DFS

Criterio	BFS	DFS
Estructura	Cola (FIFO)	Pila (LIFO)
Exploración	Por niveles	En profundidad
Camino corto	☑ Garantizado	❌ No garantizado
Memoria	O(V)	O(h)
Ciclos	Posible	☑ Excelente
Uso típico	Distancias, niveles	Ciclos, topología

🎓 Aplicaciones Prácticas

BFS - Búsqueda en Amplitud

1. **Redes Sociales**

- "Personas que quizás conozcas"
- Grados de separación
- Difusión de información

2. **Navegación**

- Camino más corto (sin pesos)
- Rutas en mapas
- Laberintos

3. **Sistemas**

- Crawlers web
- Análisis de redes
- Broadcasting

DFS - Búsqueda en Profundidad

1. **Análisis de Código**

- Detección de dependencias circulares
- Ordenamiento topológico
- Análisis de imports

2. **Juegos**

- Backtracking (Sudoku, N-Queens)
- Exploración de árboles de decisión
- Pathfinding

3. **Sistemas de Archivos**

- Búsqueda recursiva de archivos
- Cálculo de tamaño de directorios
- Eliminación recursiva

Conceptos Clave Aprendidos

1. Exploración Sistemática

- BFS explora nivel por nivel
- DFS explora rama por rama
- Ambos visitan todos los nodos alcanzables

2. Estructuras de Datos

- Cola para BFS (FIFO)
- Pila para DFS (LIFO)
- Set para nodos visitados

3. Detección de Ciclos

- Usar estados (NO_VISITADO, EN_PROCESO, COMPLETADO)
- Si encontramos un nodo EN_PROCESO, hay ciclo
- Crucial para validar grafos dirigidos

4. Reconstrucción de Caminos

- Guardar punteros "padre" durante exploración
- Reconstruir camino desde destino a origen
- Invertir para obtener camino correcto

Casos de Prueba

Caso 1: Grafo Simple (Sin Ciclos)

```
Entrada: A-B-C-D-E-F-G
BFS desde A: A → B → C → D → E → F → G
DFS desde A: A → B → D → E → G → C → F
Ciclos: No
```

Caso 2: Grafo con Ciclo

```
Entrada: 1→2→3→1
Ciclos: Sí (detectado en paso 4)
```

Caso 3: Camino Más Corto

```
Entrada: A a G
BFS: A → B → E → G (3 aristas)
DFS: A → B → D → E → G (4 aristas)
Ganador: BFS ☑
```

Conclusiones

Logros

1. ☑ Implementación correcta de BFS y DFS
2. ☑ Detección de ciclos funcional
3. ☑ Caminos más cortos con BFS
4. ☑ Complejidad O(V + E) para ambos

Lecciones Aprendidas

1. **BFS es mejor para:**

- Caminos más cortos (sin pesos)
- Exploración por niveles
- Distancias mínimas

2. **DFS es mejor para:**

- Detección de ciclos
- Ordenamiento topológico
- Backtracking

3. **Ambos son fundamentales:**

- Base para algoritmos avanzados
- Complejidad óptima O(V + E)
- Aplicaciones en múltiples dominios

Referencias

- Cormen, T. H., et al. (2009). *Introduction to Algorithms* (3rd ed.)
- Sedgewick, R., & Wayne, K. (2011). *Algorithms* (4th ed.)
- Material del curso - Semana 5

Fecha: Diciembre 2025
Curso: Estructuras de Datos Avanzadas
Semana: 5 - BFS y DFS
Estado: ☑ Completado