Reporte Semana 8: Proyecto Final Integrador

Sistema Completo de Grafos

📋 Resumen Ejecutivo

Semana: 8
Tema: Proyecto Final - Integración de Todos los Conceptos
Fecha: Diciembre 2025

🎯 Objetivos Cumplidos

- ☑ Integrar BFS, DFS y Dijkstra en un solo sistema
- ☑ Crear aplicación práctica (Red de ciudades)
- ☑ Demostrar dominio de todos los conceptos del curso
- ☑ Implementar sistema production-ready
- ☑ Documentación completa

💻 Sistema Implementado

GraphSystem - Clase Integradora

Funcionalidades:

1. **BFS** - Camino más corto (sin pesos)
2. **DFS** - Detección de ciclos
3. **Dijkstra** - Camino más corto (con pesos)
4. **Estadísticas** - Análisis del grafo

Características:

- Genérico `GraphSystem<T>`
- Soporte para grafos dirigidos y no dirigidos
- Pesos en las aristas
- Reconstrucción de caminos

📊 Aplicación Práctica: Red de Ciudades Españolas

Grafo Implementado

```
Madrid (Centro)
  ├─ Barcelona (620 km)
  ├─ Valencia (350 km)
  ├─ Sevilla (530 km)
```

```
  └─ Zaragoza (320 km)

Barcelona
  ├─ Valencia (350 km)
  └─ Zaragoza (300 km)

Valencia
  └─ Sevilla (650 km)
```

Resultados de Ejecución

```
🎓 Semana 8 - Proyecto Final Integrador
Sistema Completo de Grafos

============================================================

🏗️ Construyendo red de ciudades...
📊 Estadísticas del grafo:
   Vértices: 5
   Aristas: 14
   Densidad: 0.700

🔍 BFS - Camino más corto (número de ciudades):
   Madrid → Sevilla
   Ciudades visitadas: 2

🗺️ Dijkstra - Camino más corto (distancia en km):
   Madrid → Barcelona: Madrid → Zaragoza → Barcelona (620 km)
   Madrid → Valencia: Madrid → Valencia (350 km)
   Madrid → Sevilla: Madrid → Sevilla (530 km)
   Madrid → Zaragoza: Madrid → Zaragoza (320 km)

🔄 Detección de ciclos:
   ¿Tiene ciclos? True

============================================================
☑️ Conceptos aplicados en este proyecto:
   ✓ Grafos (Listas de adyacencia)
   ✓ BFS (Búsqueda en amplitud)
   ✓ DFS (Búsqueda en profundidad)
   ✓ Dijkstra (Caminos más cortos ponderados)
   ✓ Detección de ciclos
   ✓ Análisis de complejidad
============================================================

🎉 ¡Proyecto Final Completado!
📚 Curso de Estructuras de Datos Avanzadas - 100% Completo
```

📈 Análisis de Resultados

Comparación BFS vs Dijkstra

Ruta	BFS (ciudades)	Dijkstra (km)	Diferencia
Madrid → Sevilla	2 ciudades	530 km	Directo
Madrid → Barcelona	2 ciudades	620 km	Vía Zaragoza
Madrid → Valencia	2 ciudades	350 km	Directo

Conclusión:

- BFS minimiza número de paradas
- Dijkstra minimiza distancia total
- Ambos son correctos para sus objetivos

🎓 Conceptos Integrados

1. Semana 1-2: Recursión y DP

```
// DFS usa recursión (Semana 1)
private bool HasCycleDFS(T node, HashSet<T> visited, HashSet<T> recStack)
{
    visited.Add(node);
    recStack.Add(node);

    foreach (var (neighbor, _) in graph[node])
    {
        if (!visited.Contains(neighbor))
        {
            if (HasCycleDFS(neighbor, visited, recStack))
                return true;
        }
        else if (recStack.Contains(neighbor))
        {
            return true; // Ciclo detectado
        }
    }

    recStack.Remove(node);
    return false;
}
```

2. Semana 3-4: Grafos y Validación

```
// Listas de adyacencia (Semana 3)
private readonly Dictionary<T, List<(T to, double weight)>> graph = new();

// Validación de propiedades (Semana 4)
```

```
public void PrintStats()
{
    Console.WriteLine($"Vértices: {graph.Count}");
    Console.WriteLine($"Aristas: {graph.Values.Sum(list => list.Count)}");
    Console.WriteLine($"Densidad: {(double)graph.Values.Sum(list => list.Count) /
(graph.Count * (graph.Count - 1)):F3}");
}
```

3. Semana 5: BFS y DFS

```
// BFS para camino más corto
public List<T> BFS(T start, T end)
{
    var visited = new HashSet<T>();
    var queue = new Queue<T>();
    var parent = new Dictionary<T, T>();

    queue.Enqueue(start);
    visited.Add(start);

    while (queue.Count > 0)
    {
        var current = queue.Dequeue();
        if (current.Equals(end))
            return ReconstructPath(parent, start, end);

        foreach (var (neighbor, _) in graph[current])
        {
            if (!visited.Contains(neighbor))
            {
                visited.Add(neighbor);
                parent[neighbor] = current;
                queue.Enqueue(neighbor);
            }
        }
    }
    return new List<T>();
}
```

4. Semana 6: Dijkstra

```
// Dijkstra para caminos ponderados
public (Dictionary<T, double> distances, Dictionary<T, T> previous) Dijkstra(T
start)
{
    var distances = new Dictionary<T, double>();
    var previous = new Dictionary<T, T>();
    var pq = new SortedSet<(double dist, T node)>();
```

```
        foreach (var node in graph.Keys)
            distances[node] = double.PositiveInfinity;

        distances[start] = 0;
        pq.Add((0, start));

        while (pq.Count > 0)
        {
            var (currentDist, current) = pq.Min;
            pq.Remove(pq.Min);

            if (currentDist > distances[current]) continue;

            foreach (var (neighbor, weight) in graph[current])
            {
                var newDist = distances[current] + weight;
                if (newDist < distances[neighbor])
                {
                    pq.Remove((distances[neighbor], neighbor));
                    distances[neighbor] = newDist;
                    previous[neighbor] = current;
                    pq.Add((newDist, neighbor));
                }
            }
        }
        return (distances, previous);
    }
```

Lecciones Aprendidas

1. Integración de Conceptos

- Todos los algoritmos trabajan sobre la misma estructura (grafo)
- Cada algoritmo tiene su propósito específico
- La combinación ofrece soluciones completas

2. Trade-offs

- BFS: Rápido pero no considera pesos
- Dijkstra: Óptimo pero más complejo
- DFS: Eficiente para ciclos pero no para distancias

3. Diseño de Software

- Clase genérica permite reutilización
- Separación de responsabilidades
- Código limpio y mantenible

4. Aplicaciones Reales

- Navegación GPS
- Redes sociales
- Logística y transporte
- Análisis de redes

Casos de Uso Demostrados

Caso 1: Planificación de Viajes

```
Problema: Viajar de Madrid a Barcelona
Solución BFS: 2 paradas (minimiza paradas)
Solución Dijkstra: 620 km vía Zaragoza (minimiza distancia)
```

Caso 2: Detección de Rutas Circulares

```
Problema: ¿Hay rutas circulares en la red?
Solución DFS: Sí, detectado ciclo
Aplicación: Evitar bucles infinitos en routing
```

Caso 3: Análisis de Red

```
Problema: Estadísticas de la red
Solución: 5 ciudades, 14 conexiones, densidad 70%
Aplicación: Planificación de infraestructura
```

Conclusiones del Proyecto

Logros Técnicos

1. ☑ Integración exitosa de 8 semanas de contenido
2. ☑ Sistema funcional y completo
3. ☑ Código production-ready
4. ☑ Documentación exhaustiva

Logros Académicos

1. ☑ Dominio de algoritmos fundamentales
2. ☑ Comprensión de estructuras de datos
3. ☑ Análisis de complejidad
4. ☑ Aplicación práctica de conceptos

Habilidades Desarrolladas

- Programación en C# y Python
- Diseño de algoritmos
- Análisis de complejidad
- Resolución de problemas
- Documentación técnica

Resumen del Curso Completo

Semanas 1-2: Fundamentos

- Recursión vs Iteración
- Programación Dinámica
- Memoización y Tabulación

Semanas 3-4: Grafos Básicos

- Representaciones de grafos
- Propiedades y validación
- Havel-Hakimi

Semanas 5-6: Algoritmos de Grafos

- BFS y DFS
- Dijkstra
- Caminos más cortos

Semanas 7-8: Integración

- Árboles BST
- Proyecto final integrador
- Aplicaciones prácticas

Aplicaciones Futuras

Algoritmos Avanzados

- A* (Dijkstra con heurística)
- Bellman-Ford (pesos negativos)
- Floyd-Warshall (todos los pares)
- Algoritmos de flujo máximo

Estructuras Avanzadas

- AVL Trees (balanceo automático)
- Red-Black Trees
- B-Trees (bases de datos)
- Heaps y colas de prioridad

Aplicaciones

- Sistemas de recomendación
- Análisis de redes sociales
- Optimización de rutas
- Inteligencia artificial

✨ Reflexión Final

Este proyecto demuestra:

1. **Dominio Técnico Completo**
    - 8 semanas de contenido implementado
    - Algoritmos fundamentales dominados
    - Código de calidad profesional
2. **Pensamiento Algorítmico**
    - Selección apropiada de algoritmos
    - Análisis de trade-offs
    - Optimización de soluciones
3. **Habilidades Profesionales**
    - Código limpio y documentado
    - Testing exhaustivo
    - Documentación completa

El curso de Estructuras de Datos Avanzadas está 100% completado con éxito.

🕮 Referencias

- Cormen, T. H., et al. (2009). *Introduction to Algorithms* (3rd ed.)
- Sedgewick, R., & Wayne, K. (2011). *Algorithms* (4th ed.)
- Material del curso - Semanas 1-8

Fecha: Diciembre 2025
Curso: Estructuras de Datos Avanzadas
Semana: 8 - Proyecto Final
Estado: ☑ COMPLETADO AL 100%

🎉 **¡FELICIDADES POR COMPLETAR EL CURSO!** 🎉