Difusión del mensaje mediante palomas mensajeras - Informe técnico

Introducción

En este trabajo se desarrolló un sistema para simular la **difusión eficiente de mensajes entre aldeas** utilizando palomas mensajeras. Con el objetivo de minimizar la distancia total recorrida por las palomas, se utilizó un **Árbol de Expansión Mínima (MST)** generado mediante el **algoritmo de Prim**, apoyado por una **cola de prioridad implementada con un montículo binario**.

El programa fue diseñado de forma modular, separando las responsabilidades de lectura de datos, procesamiento del grafo y presentación de resultados.

---

1. Estructura de datos utilizada

a) Grafo no dirigido y ponderado

Se utilizó una estructura de grafo definida en la clase Grafo, contenida en el módulo modules.grafo. Este grafo es:

- **No dirigido**: las conexiones entre aldeas son bidireccionales.
- **Ponderado**: cada arista tiene un peso asociado (distancia en leguas).

Cada vértice representa una aldea, y cada arista representa un camino entre dos aldeas, con una distancia determinada.

b) Montículo binario

Para implementar la cola de prioridad utilizada en el algoritmo de Prim, se usó un **montículo binario mínimo**, una estructura eficiente que permite:

- Insertar elementos con prioridad en O(log n)
- Extraer el mínimo en O(log n)
- Acceder al mínimo en O(1)

Esto permite que el algoritmo de Prim mantenga siempre al frente el vértice más prometedor (el que tiene la menor distancia conocida al MST parcial).

c) Cola de prioridad

Internamente, el algoritmo de Prim utiliza una cola de prioridad implementada con un montículo binario. Cada elemento de la cola representa una aldea y se prioriza por la menor distancia posible al MST.

---

2. Algoritmo de Prim

El algoritmo de **Prim** se utilizó para generar un Árbol de Expansión Mínima desde una aldea inicial, asegurando que todas las aldeas estén conectadas con el mínimo costo total.

Este algoritmo:

1. Comienza con un solo nodo (la aldea inicial, en este caso "Peligros").
2. Agrega iterativamente la arista de menor peso que conecta una aldea ya visitada con una no visitada.
3. Mantiene estructuras para:
    - Registrar la distancia mínima de cada aldea al MST (distancias).
    - Registrar el padre de cada aldea en el árbol (padres).
    - Actualizar prioridades mediante el montículo binario.

La implementación del algoritmo está encapsulada en la función prim() dentro del módulo modules.algoritmo_prim.

---

3. Descripción detallada del código

El programa principal está compuesto por las siguientes funciones:

cargar_grafo_desde_archivo(ruta_archivo)

- Lee un archivo .txt con el formato origen, destino, peso.
- Construye el grafo mediante llamadas a grafo.agregar_arista().

invertir_diccionario(padres)

- A partir del diccionario de padres generado por Prim, crea un diccionario inverso que asocia a cada aldea con la lista de aldeas a las que envía el mensaje.

mostrar_resultado(padres, distancias)

- Imprime en pantalla la lista de aldeas en orden alfabético.
- Para cada aldea, muestra de dónde recibe el mensaje y a quién se lo envía.
- Calcula la **distancia total recorrida por las palomas**, como la suma de las distancias de todas las aristas del MST (omitiendo la raíz).

Bloque main:

- Define la ruta del archivo.
- Carga el grafo.
- Ejecuta el algoritmo de Prim desde "Peligros".
- Llama a mostrar_resultado() con los resultados obtenidos.

4. Salida esperada

El sistema genera una salida detallada con:

- Listado alfabético de todas las aldeas.
- Para cada aldea:
    - La aldea de la que recibe el mensaje (siendo la raíz la única que no recibe).
    - Las aldeas a las que envía el mensaje.
- La **distancia total recorrida** por las palomas en leguas.

Esto simula un esquema de comunicación jerárquico, eficiente y sin redundancias ni ciclos, tal como se espera de un árbol generador mínimo.

---

5. Conclusiones

El desarrollo implementado demuestra una correcta aplicación de algoritmos clásicos de teoría de grafos en un contexto aplicado. La utilización de un grafo ponderado, combinado con el algoritmo de Prim y el uso de estructuras eficientes como el **montículo binario**, permitió generar una solución óptima y de bajo costo computacional para el problema de difusión de mensajes.

El código resultante es claro, modular y extensible. Se garantiza que el mensaje llega a todas las aldeas de manera eficiente, minimizando el recorrido total, tal como lo haría un buen sistema de mensajería ancestral basado en palomas.