**INFORME:**

En primera instancia para la resolución del problema, definimos un módulo llamado aldeas, dedicado a extraer la información de aldeas.txt, documento brindado por la cátedra, con la información base sobre las aldeas emisoras, las que recibían las noticias y las leguas que se deben recorrer, esto nos dio por resultado una tupla con la información del origen, destino y ponderación, en ese orden.

Estos son los datos que luego utilizamos en el montículo binario de mínima, el cual está en un módulo llamado montículo binario; no vimos la necesidad de implementar una cola de prioridad (Esta aporta modularidad y orden al código, pero no es esencial), ya que la misma información puede ser procesada por el algoritmo prim desde el montículo, con un menor costo, incluso si la diferencia es ínfima.

En un módulo aparte, construimos un grafo cuya representación es del tipo lista de adyacencias (más específicamente implementada como diccionario de adyacencias, esta es más eficiente para almacenar los vecinos de cada vértice, ya que almacena sólo los que existen (no hay lugares vacíos); y permite acceder más rápido a los vecinos y sus ponderaciones, dado que cada vértice tiene un conjunto de vecinos, y cada vecino tiene una ponderación asociada; en el diccionario, acceder a los vecinos tiene una complejidad O(1) y en el caso de una lista de adyacencias tradicional, es O(n)); en él mismo (grafo), los vértices representan a las aldeas y los arcos representan el costo de la trayectoria en leguas.

Con el algoritmo prim, utilizamos el grafo y la aldea de inicio, para dar lugar al recorrido del algoritmo por el grafo y determinar el camino que represente el costo mínimo utilizando un árbol de expansión mínima (aem). Dentro del algoritmo incluimos un

visitados = set()

con la finalidad de establecer qué vértices (aldeas) ya fueron visitados y evitar que se almacene más de una vez la misma aldea, haciendo más eficiente la verificación de las aldeas visitadas; prim devuelve el aem.

Para cumplir con las consignas del trabajo, se realizó un archivo aplicación en la carpeta app, la cual implementa el algoritmo Prim (que, a su vez, utiliza el montículo binario), el grafo y la función para cargar la información de las aldeas al mismo, usa print() para mostrar al usuario la siguiente información sobre el recorrido: Aldea en la que se encuentra (cual es la clave del vértice), de qué aldea viene la noticia (vértice predecesor), y a qué aldea (vértice) se envía desde el nodo actual. Al final de la comunicación del recorrido, se muestra la suma de las ponderaciones recorridas del grafo. Una consideración para tomar en cuenta al observar el código, es que el recorrido inicia en la aldea “Peligro” para cumplir con la consigna del trabajo práctico, sin embargo, inicializando el recorrido desde cualquier aldea, se recorrerán todas con el costo total mínimo, esto nos asegura el uso del árbol de expansión mínima. El recorrido más corto es de 94 leguas y que siempre que se implemente este algoritmo con las mismas ponderaciones, el orden de las aldeas es irrelevante (en parte esto está justificado, ya que el grafo es no dirigido) y se mantendrá el costo total.

Para visualizar el grafo, y corroborar el correcto funcionamiento del código se realizó el siguiente esquema en base a la información de las aldeas. En el esquema se puede observar en celeste el grafo y en azul el aem.

