Algoritmo de Prim
Árbol Generador de Costo Mínimo

Ejercicio de implementación de un algoritmo voraz

El algoritmo de Prim es un algoritmo de tipo voraz para obtener un árbol generador de costo mínimo de un grafo dado $G = (V, U)$, donde se especifica un costo c_e para cada $e \in U$.

En cada iteración del algoritmo se mantiene una partición del conjunto de nodos V en dos subconjuntos S y $\overline{S}$, es decir, $S \cup \overline{S} = V$ y $S \cap \overline{S} = \emptyset$. El conjunto S se inicializa con un nodo $s \in V$ **seleccionado de manera aleatoria** y el conjunto $\overline{S}$ con los nodos restantes del conjunto V, es decir, $\overline{S} \leftarrow V \setminus \{s\}$. En cada iteración del algoritmo voraz se selecciona del *conjunto factible de elementos* $\mathscr{F}$ la arista de menor costo. En el caso de empates, se puede seleccionar cualquiera de las aristas de menor costo. La arista seleccionada se añade al conjunto de aristas del árbol generador U', que se inicializa con el conjunto vacío. El *conjunto factible de elementos* $\mathscr{F}$ contiene todas las aristas que tienen uno de sus extremos en un nodo del conjunto S y el otro extremo en un nodo del conjunto $\overline{S}$. Se requiere:

- Implementar el algoritmo de Prim en mosel o en python. En ambos casos se debe almacenar el grafo **usando los conjuntos de adyacencia** de los nodos. Si se implementa en python **no se debe utilizar la clase NetworkX**.
- El algoritmo debe desplegar el costo de las árbol generador obtenido, así como las aristas del mismo.

Nota: Para generar un número aleatorio entero entre 1 y n en lenguaje mosel utilizar el comando:

```
NumAletorio:=integer(round(random*n+0.5)
```

Entregables

- Archivo de código de la implementación del algoritmo en lenguaje mosel o en python.
- Archivo de datos utilizado para probar la implementación en formato de texto.