Lab05: Subset_Sum and Shortest_Path

Subset_Sum

1. **Decidir si hay una subcolección de elementos que sumen una cantidad**

   En este ejercicio dispones del fichero (`lab05_subset_sum.py`), donde tienes que implementar la función `has_sum`. La función `has_sum`, dado un valor ≥ 0 y una colección de positivos, devuelve `True` si existe una subcolección que sume el valor de entrada. En otro caso, devuelve `False`.

   Para probar la función debes eliminar los comentarios de todas las instrucciones `assert` correspondientes a `has_sum` que se encuentran en `test()`.

2. **Encontrar una subcolección de elementos que sumen una cantidad**

   Ahora tienes que implementar la función `subset`, que dado un valor ≥ 0 y una colección de positivos, si existe una subcolección que sume el valor de entrada, la devuelve. En otro caso, devuelve la lista [`None`].

   Para probar la función debes eliminar los comentarios de todas las instrucciones `assert` correspondientes a `subset` que se encuentran en `test()`.

3. **Medir el tiempo de ejecución**

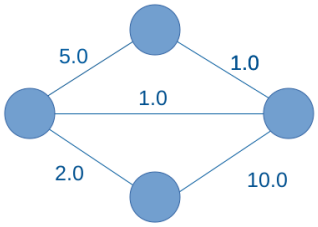
   Podemos medir el tiempo transcurrido durante la ejecución de una determinada función, a través del módulo `time`. Lo usaremos en el test de este laboratorio para calcular el tiempo de ejecución de tu función `subset`.

   Comenta todos los `assert` y descomenta la colección 6 junto con el `assert subset`. Cuando ejecutes el test, apunta el tiempo transcurrido. Añade un comentario a tu función `subset` indicando el tiempo que ha tardado la ejecución de la colección 6.

Shortest_Path

Implementa un algoritmo estudiado en el tema 5. En este ejercicio dispones del fichero (`lab05_Dijkstra.py`), donde tienes que implementar la función `Dijkstra`. Esta función, dado un grafo no dirigido con pesos en las aristas y un nodo, v, encuentra el camino más corto desde v a los demás nodos.

Por ejemplo, Si tenemos el siguiente grafo:



La matriz de adyacencia contiene los costes reales entre las aristas. El valor (∞) indica que no hay arista entre dos nodos distintos.

$$\begin{aligned} \text{graph} = \quad & [[0.0,\ 5.0,\ 1.0,\ \text{inf}], \\ & [5.0,\ 0.0,\ 1.0,\ 2.0], \\ & [1.0,\ 1.0,\ 0.0,\ 10.0], \\ & [\text{inf},\ 2.0,\ 10.0,\ 0.0]] \end{aligned}$$

La llamada a `Dijkstra`(graph, 3) debe devolver la lista $[4.0, 2.0, 3.0, 0.0]$.