I. Diseño del experimento

1. **Algoritmo de Dijkstra**
   Trataremos de explicar brevemente el algoritmo de Dijkstra y del uso en el lenguaje R:
   Dado un grafo conexo, dirigido o no dirigido, con distancias no negativas; el algoritmo de Dijkstra empieza por marcar el nodo fuente e inicializar a 0 o a infinito las distancias acumuladas de cada nodo. Posteriormente itera, mientras queden nodos sin marcar, seleccionando en cada etapa el arco con la suma más pequeña de la distancia acumulada por uno de los nodos marcados más la distancia del arco que lo une a uno de los nodos no marcados, guardando este último nodo como marcado y actualizando su distancia acumulada con la suma antes calculada. Al final del proceso tendremos un árbol de expansión de camino más corto. La implementación del algoritmo tal y como está escrita en pseudocódigo no es casual y facilita portarlo a R. Existe varias implementaciones del algoritmo al margen de la original de Dijkstra. De hecho, la que utilizaremos en el paquete **optrees** modifica un poco su funcionamiento para adaptarlo a nuestras necesidades. Por otra parte, es necesario, eso sí, indicar si el grafo que introducimos en la función es o no dirigido, pues en este segundo caso deberemos realizar el paso previo de duplicar los arcos. El resto de datos del grafo de entrada lo constituyen el conjunto de nodos y arcos, los cuales se introducen en R de la misma forma que hemos visto hasta ahora, por lo demás, las instrucciones del algoritmo de Dijkstra se pueden trasladar directamente manteniendo el uso de un único bucle, lo que permite evitar aumentar la complejidad del problema. La propuesta original de Dijkstra funciona en un tiempo computacional teórico de $O(|V|^2)$. Con el tiempo se han desarrollado mejores implementaciones que requieren una cola de prioridad y reducen la complejidad del algoritmo, pero complican portal el código a R. En nuestro caso, aunque con diferencias, mantenemos un formato muy similar al modelo original de Dijkstra, que es de por sí lo suficientemente rápido como para obtener resultados por debajo del segundo en grafos con cientos de nodos y miles de arcos.

I-A. Optrees

Este paquete en R, encuentra árboles óptimos ponderados. En particular, este paquete proporciona herramienta de resolución de problemas de árbol de expansión de costo mínimo, problemas de arborescencia de costo mínimo, problemas de árbol de ruta más corto y problema de árbol de corte mínimo.

2. **Algoritmo de Bellman-Ford** Este segundo algoritmo, para el problema del árbol de camino más corto es el conocido como algoritmo de Bellman-Floyd. El algoritmo empieza inicializando a 0 la distancia asociada a la fuente y a infinito las distancias acumuladas del resto de nodos. Posteriormente, lleva a cabo el proceso de relajación iterando en todos los nodos distintos de la fuente, comprobando en cada arco que sale de él si la distancia hacia otro nodo es mayor que la suma de la distancia acumulada del primero más el peso del arco que los une, en caso afirmativo establece este resultado como la distancia acumulada por el nodo de llegada y guarda el nodo de partida como su predecesor. Al final termina comprobando si se han encontrado ciclos negativos, revisando arco por arco si se puede reducir la distancia acumulada del nodo de llegada. Si no se topa con uno, el resultado final es un árbol de expansión del camino más corto. El proceso de programar el algoritmo de Bellman-Floyd en R es equivalente al pseudocódigo aquí desarrollado, aunque con ligeros añadidos. En este caso, también es necesario introducir un parámetro de entrada que permita señalar si estamos ante un grafo dirigido o no, para duplicar los arcos ante eseta segunda situación, pero al final es necesario incorporar un bucle adicional que reconstruya el árbol del camino más corto a partir del conjunto de nodos y los predecesores que hemos ido almacenando durante el proceso de relajación. Este último bucle ralentiza un poco nuestra función, tiene una complejidad de $O(|V||A|)$. Es, por tanto, más lento que el de Dijkstra, siendo este el precio a pagar por permitir trabajar con pesos negativos.