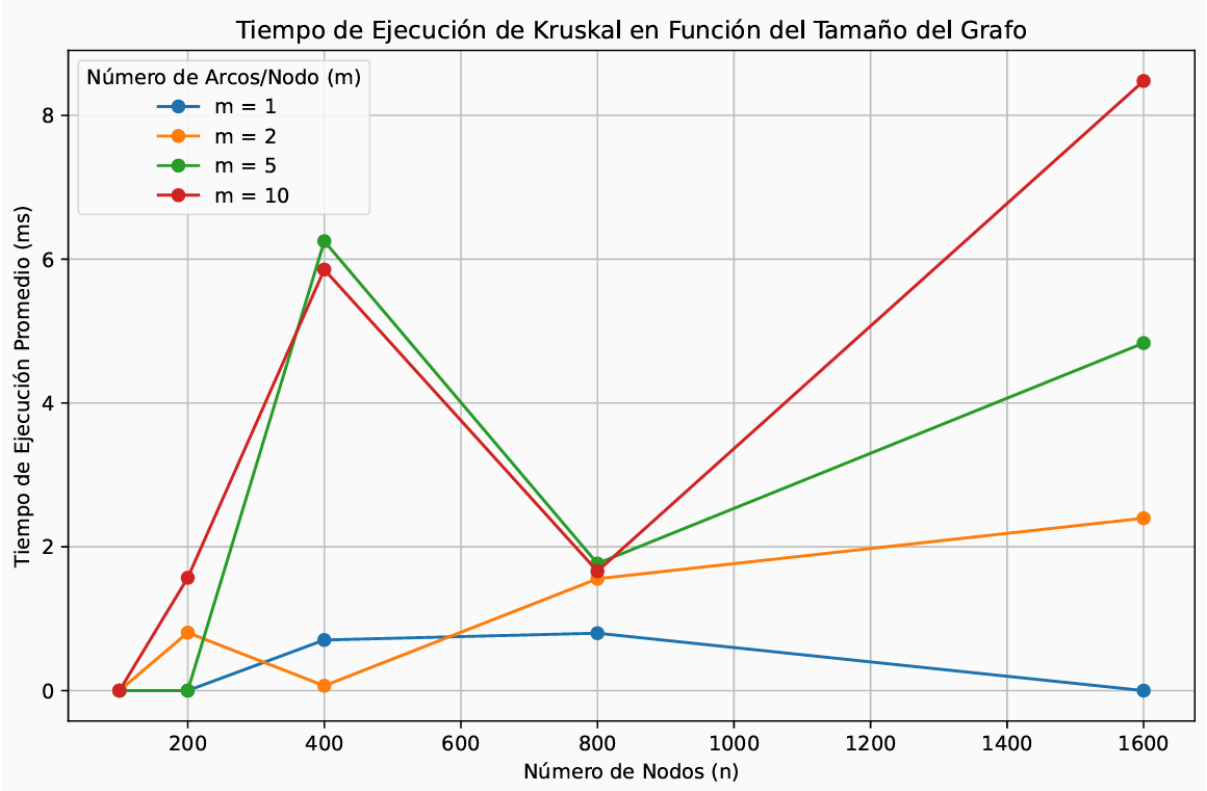
ANÁLISIS DE LA GRÁFICA DE TIEMPO DE EJECUCIÓN DE KRUSKAL

Iván Sánchez Bonacasa
Christian Grosso



La gráfica que observamos muestra cómo varía el tiempo de ejecución del algoritmo de Kruskal en función del tamaño del grafo (medido en el número de nodos) y de su densidad (medida por el número de arcos promedio por nodo, representado por **m**). El eje X, que representa el número de nodos **n**, se extiende desde un pequeño tamaño de grafo hasta valores mucho mayores, mientras que el eje Y muestra el tiempo promedio de ejecución en milisegundos. En la leyenda hemos indicado los distintos valores de m que se han evaluado: 1, 2, 5 y 10 arcos por nodo.

En general, podemos observar que, a medida que el número de arcos promedio por nodo aumenta, el tiempo de ejecución también tiende a incrementarse. Esto se debe a que Kruskal es un algoritmo que depende del número de arcos en el grafo, ya que primero los ordena y luego realiza uniones entre conjuntos de nodos, lo cual se vuelve más costoso a medida que la densidad de arcos aumenta.

Analizando cada curva de forma más detallada:

Para **m = 1** (la curva azul), vemos que el tiempo de ejecución es bajo y se mantiene bastante estable, incluso conforme aumenta el número de nodos en el grafo. Esto indica que el algoritmo de Kruskal es muy eficiente en grafos con pocos arcos, ya que en estos casos la cantidad de trabajo que debe realizar es mínima. Esta baja densidad de conexiones permite que el algoritmo procese rápidamente, incluso en grafos con un número alto de nodos, debido a que la cantidad de arcos (o conexiones) a considerar sigue siendo baja.

En la curva para **m = 2** (en color naranja), el tiempo de ejecución aumenta ligeramente en comparación con m = 1 pero aún así sigue siendo relativamente bajo. Aunque hay un pequeño pico en torno a los 800 nodos, en general el crecimiento del tiempo de ejecución no es pronunciado. Esto nos muestra que, con solo el doble de arcos por nodo, el algoritmo de Kruskal sigue funcionando eficientemente, aunque con una carga de trabajo ligeramente mayor debido al aumento en el número de arcos.

Para valores de m mayores, como **m = 5** (curva verde), ya notamos un crecimiento importante en el tiempo de ejecución, especialmente en los primeros tamaños de grafo (hasta aproximadamente 400 nodos). A medida que la densidad de arcos aumenta, el algoritmo necesita procesar más conexiones, lo cual incrementa el tiempo de ejecución de forma notable en estos tamaños de grafo. A partir de cierto punto, el crecimiento se vuelve un poco más moderado, pero el tiempo de ejecución sigue siendo significativamente mayor que en los grafos más dispersos (con m = 1 o m = 2).

Por último, la curva para **m = 10** (en color rojo) muestra el mayor tiempo de ejecución entre todas las series. Aquí vemos un aumento continuo y drástico a medida que incrementa el número de nodos en el grafo.

A partir de esta observación, podemos concluir que el algoritmo de Kruskal es una excelente opción para grafos con baja densidad de arcos, donde mantiene un rendimiento eficiente y un tiempo de ejecución bajo. Sin embargo, a medida que el grafo se vuelve más denso, el tiempo de ejecución aumenta de manera significativa. En estos casos, como el algoritmo debe manejar una mayor carga de arcos entendemos que incremente el trabajo que realiza y hace que el tiempo de ejecución crezca proporcionalmente.