PRÁCTICA MARP

GRAFOS
ORDEN TOPOLÓGICO
Y
COMPONENTES FUERTEMENTE CONEXAS

JAVIER CARRIÓN GARCÍA

ÍNDICE

FUNCIONAMIENTO

El programa ejecuta sobre grafos distintos el Orden Topológico y el procesamiento de las Componentes Fuertemente Conexas, pero en ambos casos parte de un grafo dirigido con 300 nodos y se le pregunta al usuario cuántos nodos como máximo quiere que tenga el grafo, entonces el programa irá añadiendo 300 nodos y a cada nodo se le añadirá entre 0 y 6 de aristas forma aleatoria cada vez y ejecutando el problema que toque. Cada problema se ejecutará 3 veces para tener 3 medidas de tiempo distintas y después se calculará la media para almacenarla en un fichero ".dat" que almacenará el tiempo de ejecución del problema y el tamaño del grafo, este fichero es el que utilizará el programa de creación de gráficas para generarlas al final de la ejecución. Todos los grafos generados se mostrarán por pantalla a medida que se vayan generando. El formato será: <id del vértice> → <lista de adyacencia>.
Dado que la cantidad de nodos la introduce el usuario al inicio de la ejecución y las aristas para cada vértice se calcula de forma aleatoria, no hay ningún fichero con casos de prueba.

ESTRUCTURA DE LA PRÁCTICA

La práctica está hecha en C++ y está en su totalidad escrita en Inglés debido a mi intención de publicar el código en un futuro. La práctica está estructurada en distintos módulos:

"Utilities.pp"

En este módulo se encuentra tanto una estructura propia de vértice como un sinónimo de string llamado Id que se utilizará para el identificador de un vértice. Aunque conocía de la existencia de una estructura "Vertex" predefinida, me parecía más oportuno crearme yo la mía propia.

"Graph.hpp"

Este módulo es una clase que representa a un grafo y tiene todas las funciones (necesarias para esta práctica) que se pueden aplicar sobre éstos.

"DFS.hpp"

Este módulo es la clase que representa el algoritmo de Búsqueda Primero en Profundidad. Incluye la búsqueda normal y una variante utilizada únicamente para el cálculo de las componentes fuertemente conexas.

"Main.cpp"

Aquí se encuentra la función que inicia toda la ejecución y genera las gráficas.

GRÁFICAS

ORDENACIÓN TOPOLÓGICA

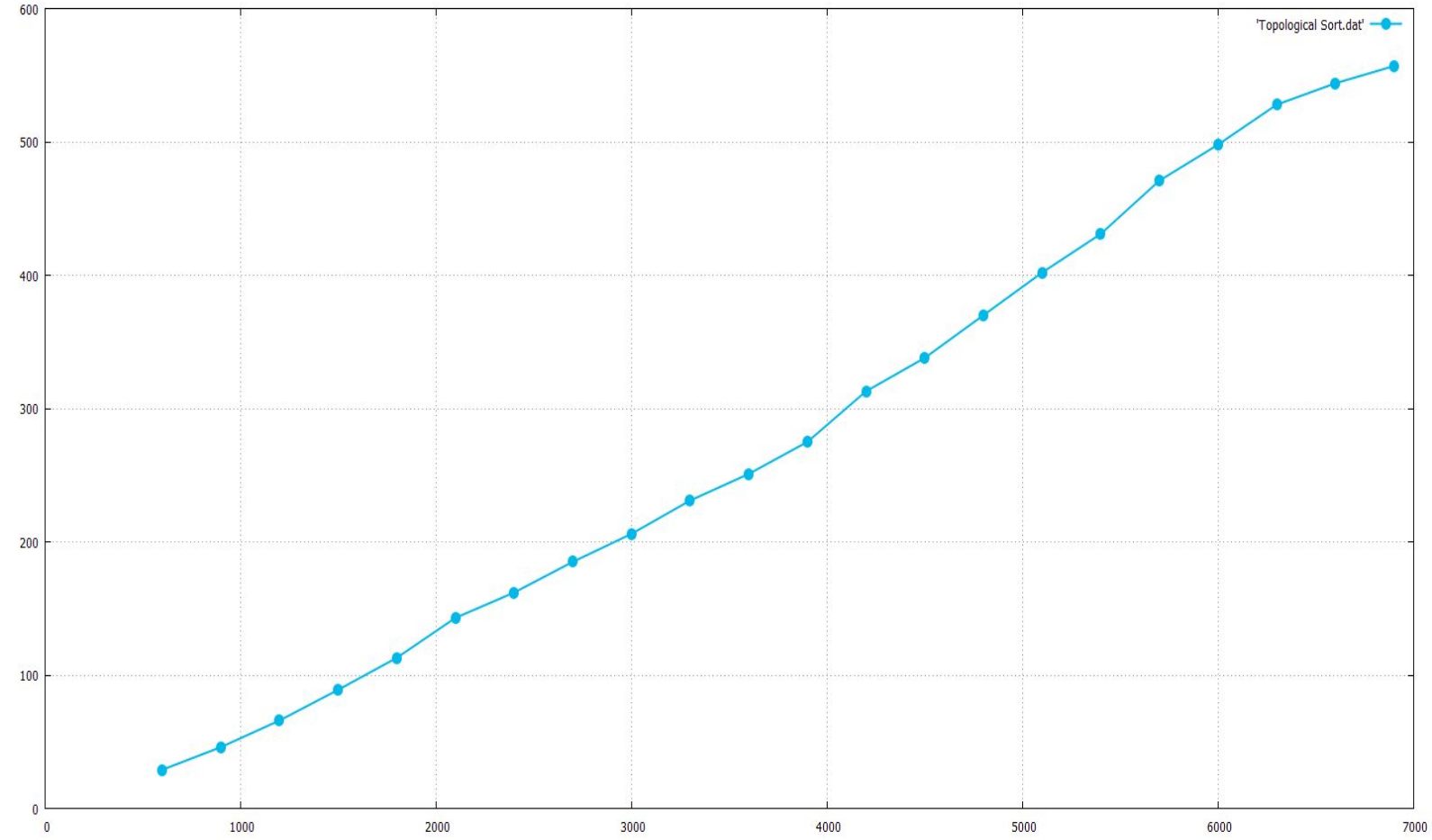

Imagen 1.1 Eje X: nº de vértices. Eje Y: tiempo (ms).

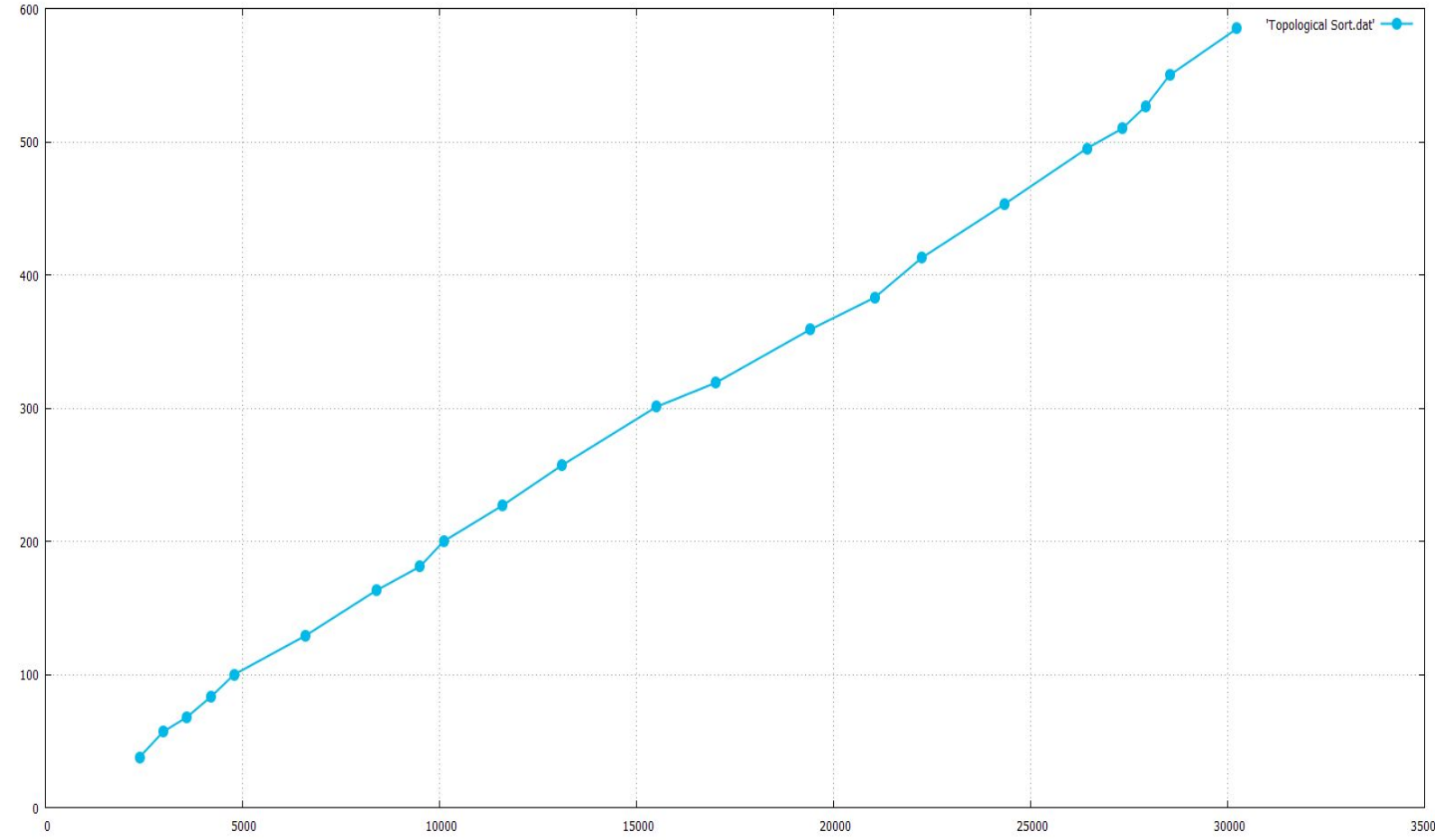

Imagen 1.2 Eje X vértices + aristas. Eje Y tiempo (ms).

El coste de esta operación se ajusta a una distribución lineal, corroborando así los costes teóricos estudiados en clase. En esta implementación he modificado un poco la implementación a la vista en clase para medir los tiempos del cálculo de la ordenación topológica: para calcular el orden topológico se realiza un DFS, y si inserto por delante en una lista el nodo que he terminado de explorar (el nodo que ponemos a Negro), tendré la ordenación topológica del grafo en caso de que este sea acíclico, o una lista ordenada por tiempos de finalización en caso de que sea cíclico, esto último sólo lo utilizará el grafo traspuesto para poder obtener las componentes fuertemente conexas. En cualquier caso el coste de este algoritmo será el mismo independientemente de si realizo la búsqueda sobre un grafo cíclico o acíclico.

COMPONENTES FUERTEMENTE CONEXAS

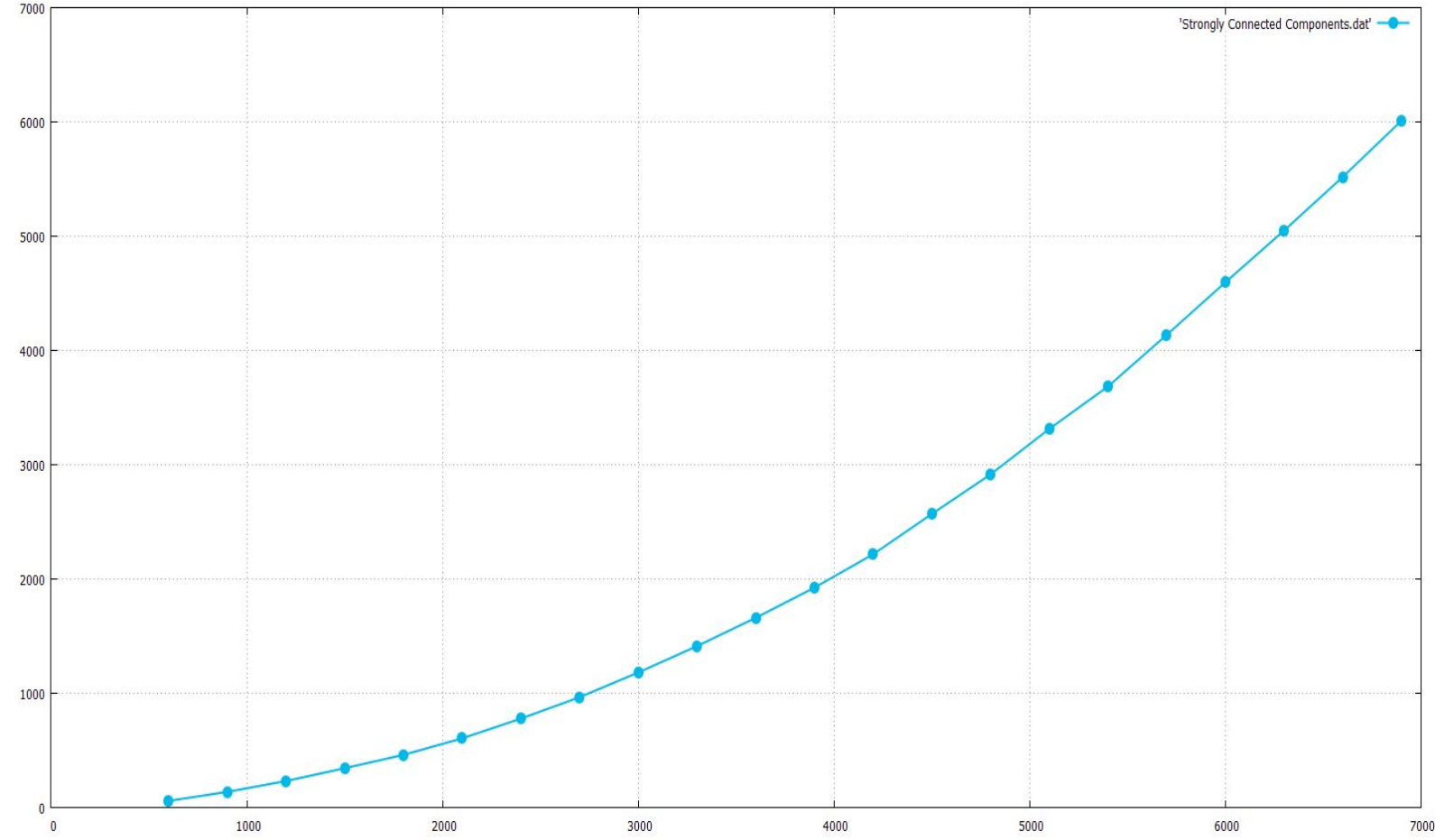

Imagen 2.1 Eje X vértices. Eje Y tiempo (ms).

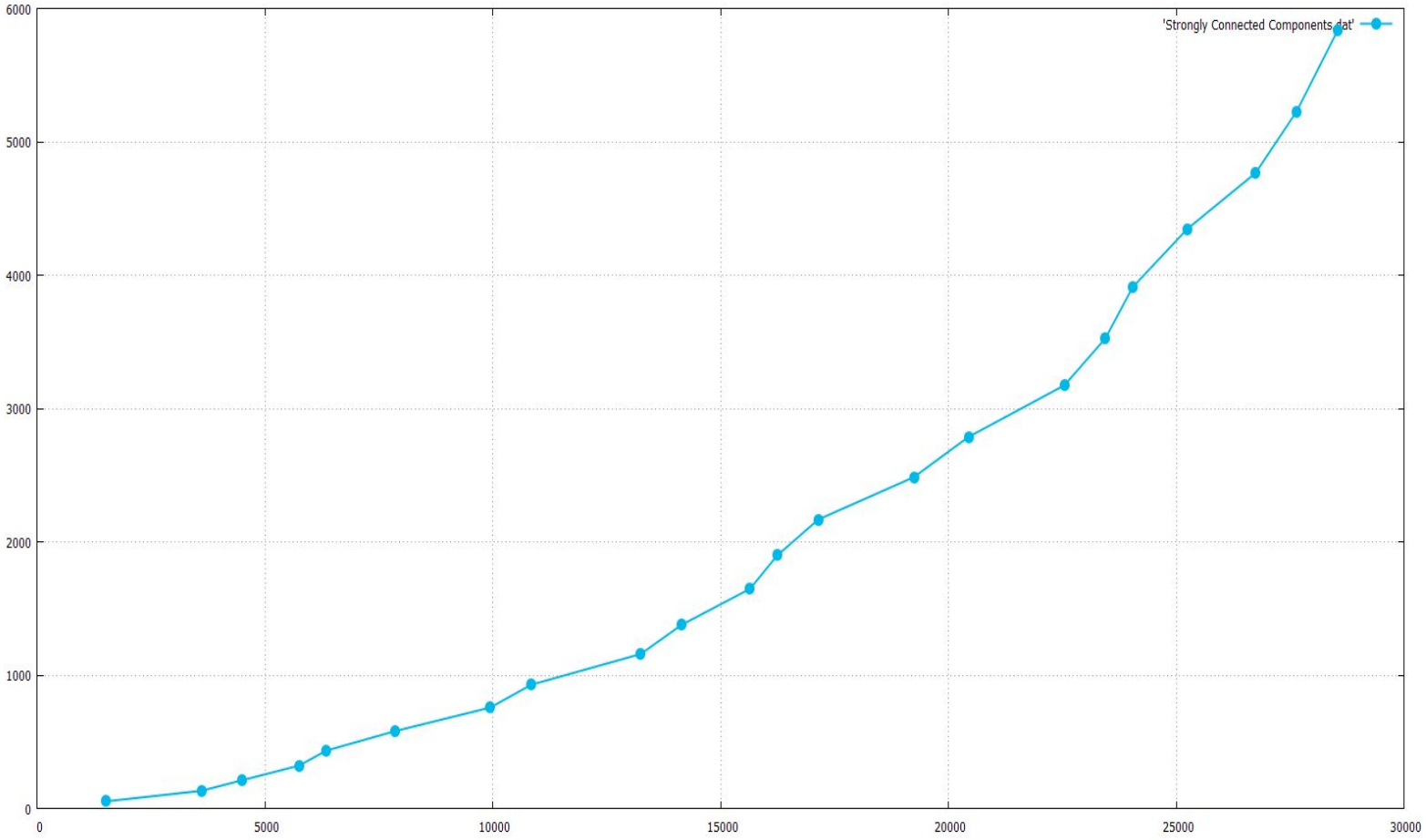

Imagen 2.2 Eje X vértices + aristas. Eje Y tiempo (ms).

En este caso, si nos fijamos sólo en la Imagen 2.1 vemos que el coste se desvía ligeramente del coste teórico, pero al ver la la Imagen 2.2 vemos que los vértices más las aristas crece en orden cuadrático y esto está bien puesto que el coste exacto del algoritmo es O(n+m), siendo n el nº total de vértices y m el nº total de aristas, y hemos visto que en el caso peor esto podría ser $O(n^2)$. Como estas gráficas se alejan del coste lineal pero no pasan del coste cuadrático, con lo cual se puede afirmar que el coste que representan estas gráficas es O(n+m). El algoritmo utilizado es el visto en Cormen (2001, second edition). A la hora de medir los tiempos, sólo he tenido en cuenta el tiempo a partir de la creación del grafo traspuesto, dado que considero que tener a disposición la matriz traspuesta es una precondición y su cálculo no debería afectar al cálculo del tiempo.