



Suite informática de teoría algorítmica de grafos Graphvisualx

Alumno: Moisés Gautier Gómez

moisesgautiergomez@gmail.com

Director: Antonio J. Tomeu Hardasmal

Universidad de Cádiz
Escuela Superior de Ingeniería

Todo tiene un comienzo

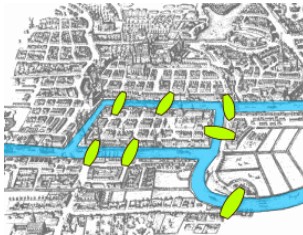
La teoría de grafos, es una rama importante de las matemáticas combinatorias que se ha estudiado intensamente durante cientos de años. Muchas propiedades importantes y útiles de los grafos se han demostrado, sin embargo, otros muchos problemas de mucha complejidad (problemas de complejidad NP o irresolubles en tiempo polinómico) siguen sin resolverse.

La investigación sobre teoría algorítmica de grafos es relativamente reciente. Aunque algunos de los algoritmos fundamentales son viejos, la mayoría de los más destacados han sido descubiertos en las últimas décadas.

Primeros tiempos

Historia

El primer artículo sobre teoría de grafos fue escrito por el famoso matemático suizo Euler, y apareció en 1736. Desde un punto de vista matemático, la teoría de grafos parecía, en sus comienzos, bastante insignificante, puesto que se ocupaba principalmente de pasatiempos y rompecabezas. Sin embargo, avances recientes en las matemáticas y, especialmente, en sus aplicaciones han impulsado en gran medida la teoría de grafos.



Generalidad

Un grafo es un conjunto de puntos y un conjunto de líneas donde cada línea une un punto con otro.

Llamaremos grafo, G , al par ordenado formado por un conjunto finito no vacío, V , y un conjunto, A , de pares no ordenados de elementos del mismo.

V es el conjunto de los vértices o nodos del grafo.

A será el conjunto de las aristas o arcos del grafo.

Utilizaremos la notación $G = (V, A)$ para designar al grafo cuyos conjuntos de vértices y aristas son, respectivamente, V y A .

A cualquier arista de un grafo se le puede asociar una pareja de vértices del mismo. Si u y v son dos vértices de un grafo y la arista a esta asociada con este par, escribiremos $a = uv$.

Estructura interna

Módulos y clases java.

- Envoltorio: Su estructura interna se basa en un HashMap sobre la que se asentarán todo los grafos del sistema que vayamos creando.
- CamposNodos y CamposArista: Clases bases para los objetos nodo y aristas del sistema.
- Algoritmos: Es la encargada de gestionar todos los algoritmos de trabajo de la aplicación.
- Grafo: Clase envoltorio de la propia clase Envoltorio para facilitar las operaciones con los grafos.
- LienzoGrafo: Clase encargada de la gestión de los lienzos de trabajo que son objetos de la clase Canvas de AWT.
- DibujaGrafo: Es la encargada de trabajar con el paquete graphviz y sus ficheros ".dot" en la generación de imágenes para los grafos editados o resultantes.
- Lec_Esc_File: Gestiona la lectura y escritura de ficheros que soportan la estructura de datos e información sobre el grafo.
- Grafico: Módulo que gestiona toda la interacción con el sistema: eventos, estructura de datos, algoritmos e interfaz gráfica. La interfaz visual ha sido creada combinando las bibliotecas Swing y AWT de java.

Diagrama de clases

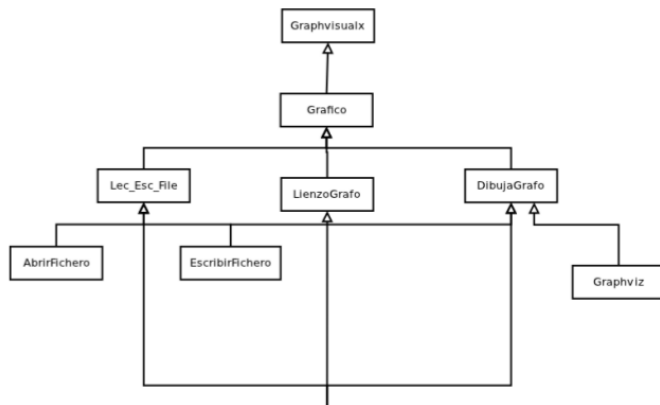
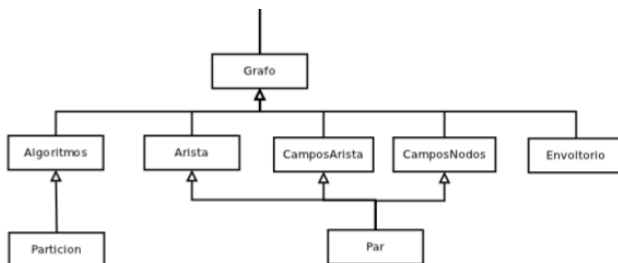


Diagrama de clases



Teoría algorítmica

Se analizarán en profundidad todos los algoritmos tratados en el proyecto de fin de carrera intentando en la medida de lo posible una fácil comprensión para el lector falto de conocimientos básicos y para el que tenga unas nociones mayores sobre la materia.

Los algoritmos a tratar serán muy variados:

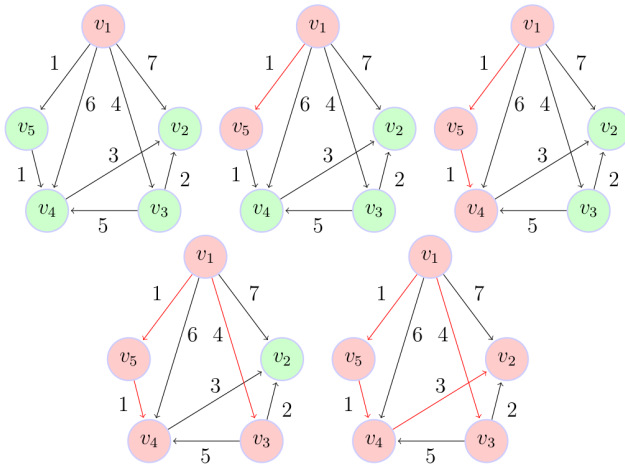
- Caminos y Ciclos de Euler.
- Grafos ponderados.
 - Caminos más cortos desde un vértice: Dijkstra.
 - Caminos más cortos desde un vértice: Bellman-Ford.
 - Caminos más cortos: Floyd.
 - Existencia de caminos: Warshall.
 - Árboles de expansión mínimos: Kruskal.
 - Árboles de expansión mínimos: Prim.
- Búsqueda en grafos: en profundidad (Depth First Search).
- Búsqueda en grafos: en anchura (Breadth First Search).
- Vértice coloración: número cromático.
- Ordenación Topológica.

Procedimiento

Para la implementación de los algoritmos se ha seguido la siguiente metodología de trabajo:

- Análisis e investigación con diferentes referencias bibliográficas sobre el algoritmo.
- Una vez realizada una comprensión del mismo se realiza una traza para comprobar su funcionamiento.
- Finalmente se perfila una aproximación en código java para su puesta en prueba con un grafo a través de las distintas clases del sistema.

Procedimiento



Procedimiento

Algoritmo de Dijkstra

```
/**  
 * Método observador (Algoritmo de Dijkstra).  
 * Calcula los caminos de coste mínimo entre origen y  
 * todos los vértices del grafo G.  
 * @param origen vértice desde el que se realiza la computación.  
 * @param G matriz de costes asociada al grafo G.  
 * @return Un vector de tamaño G.length con estos costes  
 * mínimos (fila 0 de la matriz) y un vector de tamaño G.length  
 * tal que vector[i] es el último vértice del camino origen a i  
 * (fila 1 de la matriz).  
 * @exception Exception  
 */  
  
public int[] [] algo_Dijkstra(int origen, int [] [] G) throws Exception  
{  
    int tama_G=G.length;
```

Diagrama de Gantt

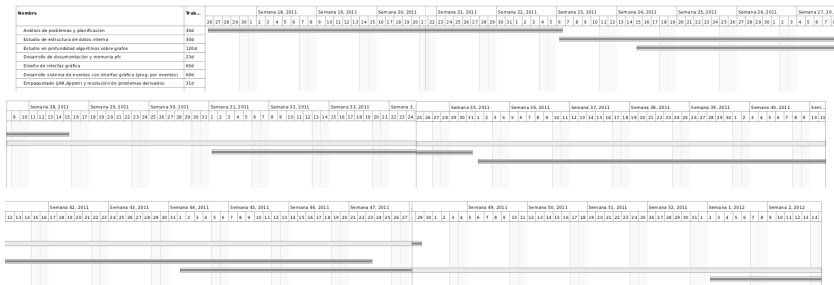
Para la elaboración de este proyecto de fin de carrera me ha sido necesario estudiar de nuevo el funcionamiento de muchos algoritmos sobre grafos que, previamente estudiados durante la titulación, había obviado en ciertos matices que habían que aclarar para acometer con precisión una implementación fina y eficiente.

Además, acometer ciertos problemas de programación por eventos, el trabajo con bibliotecas gráficas como AWT y Swing, como empaquetar todo el sistema completo de módulos en un formato ejecutable “.jar” ha supuesto una gran demora añadida en muchos puntos del proyecto.

WBS	Nombre	Inicio	Fin	Trabajo	Duración
1	Análisis de problemas y planificación	abr 26	jun 6	30d	30d
2	Estudio de estructura de datos interna	jun 6	jul 15	30d	30d
3	Estudio en profundidad algoritmos sobre grafos	jun 15	nov 29	120d	120d
4	Desarrollo de documentación y memoria pfc	ago 1	ago 31	23d	23d
5	Diseño de interfaz gráfica	sep 1	nov 23	60d	60d
6	Desarrollo sistema de eventos con interfaz gráfica (prog. por eventos)	nov 1	ene 23	60d	60d
7	Empaquetado (JAR,Applet) y resolución de problemas derivados	ene 2	feb 13	31d	31d

Planificación

Diagrama de Gantt



Finalidad del proyecto

El objetivo es elaborar una aplicación informática que sea capaz de representar un grafo y los posibles algoritmos de procesamiento que se pueden aplicar a los grafos.

- Se mostrará en una primera instancia la representación del grafo una vez se haya introducido los valores asociados para las aristas o vértices.
- Una vez seleccionado el algoritmo de procesamiento o la operación pertinente se mostrará el grafo origen y el resultado de la operación en la misma ventana para resaltar los cambios efectuados por el procesamiento del algoritmo.
- Habrá un listado con los algoritmos más reseñables de la historia sobre teoría de grafos para aplicar sobre la estructura de grafos creada previamente.

Dicha aplicación podría estar enfocada como complemento al material docente de asignaturas de la titulación así como laboratorio de pruebas de algoritmos sobre grafos de ejemplo. También puede utilizarse como soporte ágil para realizar comprobaciones sobre las distintas funcionalidades de los algoritmos disponibles.

Resultado final

El proyecto Suite informática de Teoría Algorítmica de Grafos dará como resultado la aplicación Graphvisualx que cumplirá con todos los objetivos y especificaciones indicados.

La aplicación Graphvisualx se distinguirá en varias partes: el contenido teórico de cada algoritmo a desarrollar como posible comprensión del desarrollo del mismo, el desarrollo de los algoritmos con una entrada ya sea bien en formato fichero o introducida desde la entrada de estándar en el momento de ejecución de la aplicación y contenidos bibliográficos haciendo hincapié en algoritmos clásicos de la materia y su resolución desarrollada.



Software utilizado

En la realización de este proyecto se ha empleado el lenguaje de programación Java (J2SE) en su versión 1.6, empleando además la utilidad de openJDK para dicha plataforma Java.


Para la realización y estructuración del proyecto se ha empleado la herramienta IDE NetBeans v. 7.0.

La documentación y las figuras expuestas en este documento y derivados se han creado mediante el lenguaje de interpretación \LaTeX y para las figuras el paquete de opciones de tikz.

La aplicación gráfica Graphvisualx es software libre: usted puede redistribuirlo y / o modificar bajo los términos de la Licencia Pública General de GNU según lo publicado por la Free Software Foundation, ya sea la versión 3 de la Licencia, o (a su elección) cualquier versión posterior.



Formato de aplicación

La aplicación final Graphvisualx estará disponible a través de un fichero  que será ejecutado desde la distribución favorita del usuario. [Para más información consultar [guía de usuario](#). (7.7 Ingreso al sistema)]

Además estará disponible una versión en formato Applet en la web para su interacción sin necesidad de instalación de la máquina virtual de java.



Applet

Como empezar

Una interfaz gráfica de usuario (GUI) presenta un mecanismo amigable al usuario para interactuar con una aplicación. Una GUI proporciona a una aplicación una “aparición visual” única. Al proporcionar distintas aplicaciones en las que los componentes de la interfaz de usuario sean consistentes e intuitivos, los usuarios pueden familiarizarse en cierto modo con una aplicación, de manera que pueden aprender a utilizarla en menor tiempo y con mayor productividad.

7.8. Operaciones de la suite

7.8.1. Archivo

Icono representativo: 

Dentro de este apartado se definirán los items del menú de archivo. En él se encuentran cuatro posibles operaciones a realizar por el usuario según haya o no:

- Creado un grafo mediante su edición o cargado en el sistema.
- Procesado dicho grafo mediante algún algoritmo.
- En caso contrario, no haber realizado ninguna operación con el sistema.

Para el caso de que el usuario haya simplemente cargado o editado un grafo en el sistema, tendrá las siguientes operaciones disponibles (o items) del menú archivo.

- Guardar grafo inicial (Atajo de teclado Ctrl+g). 

Objetivos logrados

Durante la realización de la aplicación *Graphvisualx: Suite informática de Teoría Algorítmica de Grafos* como trabajo de fin de carrera, he conseguido profundizar mi conocimiento en estos campos:

- Ampliación de conocimientos sobre \LaTeX paquete gráfico PFG-tikz.
- Profundización sobre el lenguaje java.
- Aprendizaje sobre programación por eventos.
- Diseño y creación de interfaces gráficas con el entorno visual de la bibliotecas Swing y AWT de java.
- Empaquetación de los módulos de lo que se componen el sistema y su obtención final de un fichero ejecutable “.jar”. Además de la subsanación de problemas derivados de la creación del fichero ejecutable.
- Aprendizaje sobre herramientas de documentación de clases y generación de estadísticas del sistema de repositorios donde se aloja mi proyecto.
- Estudio y análisis de nuevos algoritmos de grafos que desconocía.
- Finalmente, elaboración y empaquetación de la aplicación en formato Applet.

Expectativas de futuro

El producto obtenido de este proyecto de fin de carrera es un producto final. Por su naturaleza está ideado en principio para la trata y uso de los posibles algoritmos implementados en un principio, dado que no se facilita ningún soporte directo para la implementación de un mayor contenido algorítmico debido a ausencia del mismo o recientes descubrimientos de nuevas teorías algorítmicas.

Se podrían implementar algoritmos de teoría de grafos sobre redes de comunicaciones o relaciones de ontología de la web que dado a la falta de conocimientos en la materia se han obviado para una posible actualización futura.

Referencias



[BRA, 2008] G. Brassard and P. Bratley.
Fundamentos de algoritmia, 2008



[ECK, 2002] Eckel Bruce
Piensa en Java, 2002



[GIB, 1985] Alan Gibbons
Algorithmic graph theory, 1985



[INT, 2009] Thomas H. Cormen
Introduction to algorithms 3rd ed., 2009



[EST, 1998] Alfred V. Aho, John E. Hopcroft, Jeffrey D. Ullman
Estructuras de datos y algoritmos, 1998



[BER, 1989] Claude Berge
Graphs, 1989



[GAR, 2003] Michael R. Garey, David S. Johnson
Computers and intractability: a guide to the Theory of NP-Completeness [26th print.], 2009

Referencias



[HAR, 1996] Frank Haray
Graph theory, 1996



[HOP, 2008] John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman
Introducción a la teoría de autómatas, lenguajes y computación 3ª ed., 2008



[ROS, 2004] Kenneth H. Rosen
Matemática discreta y sus aplicaciones, 2004



[LEV, 2007] Anany Levitin
Introduction to the design & analysis of algorithms 2nd ed., 2007



[NEA, 2011] Richard Neapolitan, Kumarss Naimipour
Foundations of algorithms, 2011



[SED, 2003] Robert Sedgewick
Algorithms in C++ 3th ed., 2003



[SKI, 2008] Steven S. Skiena
The algorithm design manual 2nd ed., 2008

Final

¿Preguntas?



Para más información: [Página web del proyecto](#)

Si quieres ir al grano: moisesgautiergomez@gmail.com