MaGraDa 1.1

Manual de usuario y tutorial

Manuel Ángel Caballero Palomino

Violeta Migallón Gomis

José Penadés Martínez

Índice

Prólogo

Este manual explica el paquete de software MaGraDa (**Gra**fos para **M**atemática **D**iscret**a**). Esta aplicación se ha realizado en el Departamento de Ciencia de la Computación e Inteligencia Artificial, inicialmente como proyecto de la asignatura Sistemas Informáticos, por el Ingeniero Informático Manuel Ángel Caballero Palomino y bajo el asesoramiento de los profesores Violeta Migallón Gomis y José Penadés Martínez.

MaGraDa ha sido diseñado específicamente para fines docentes y cubre gran parte de los conceptos de la teoría de grafos que se estudian en la asignatura de Matemática Discreta, pretendiendo automatizar el trabajo con grafos y además que el usuario se familiarice con ellos de manera eficaz.

Este manual comienza con un capítulo introductorio en el que se describe MaGraDa y se dan algunas ideas básicas para su manejo. El segundo capítulo se centra en la utilización de MaGraDa para resolver problemas básicos de grafos, relacionados con sus fundamentos. En el capítulo tercero se estudiará la accesibilidad y conectividad de un grafo con MaGraDa. El capítulo cuarto está

dedicado al problema de recorrido de aristas. Los capítulos quinto y sexto están dedicados a los grafos ponderados. En el capítulo quinto se verá cómo introducir estos grafos con MaGraDa y la forma de obtener caminos críticos. Por último, en el capítulo sexto, se tratan los problemas de caminos más cortos y árboles generadores de mínimo peso.

Por último, mencionar que los autores han decidido que MaGraDa sea un software de dominio público, esperando que sea útil a toda la comunidad universitaria. Está disponible en http://www.dccia.ua.es/dccia/inf/asignaturas/MD/.

Capítulo 1

Introducción a MaGraDa

1.1 Introducción

El paquete de software MaGraDa (**Gra**fos para **M**atemátic**a** **D**iscret**a**) es una aplicación informática programada en lenguaje JAVA y diseñada específicamente para trabajar con grafos. MaGraDa trabaja con grafos tanto dirigidos como no dirigidos y ponderados como no ponderados, pero por el momento, sólo trataremos los grafos no ponderados. Más adelante, en el Capítulo 5, introduciremos los grafos ponderados. Según la filosofía de MaGraDa podríamos trabajar con un número ilimitado de vértices, pero hemos creido conveniente sólo permitir trabajar con grafos de hasta un máximo de 50 vértices. Este paquete es sencillo y cómodo de manejar, está basado en menús sobre pantalla y consta de dos pantallas de visualización:

(i) Modo texto: Llamémoslo de esta forma para diferenciarlo del otro. Permite trabajar con los grafos de forma analítica. Es decir, trabajaremos en todo momento con los datos del grafo, pero sin visualizarlo gráficamente.

(ii) Modo gráfico: Trabaja con los grafos de forma que pueden verse gráficamente.

Ambas pantallas de trabajo son prácticamente equivalentes en funcionalidad. Es decir, no hay ningún método que esté sólo implementado para una forma de trabajo exclusivamente. Sin embargo, la forma de ofrecer los resultados al usuario no es la misma en los dos modos. En cada uno de ellos se muestran los resultados intentando maximizar la comprensión de los mismos por el usuario. Básicamente, podemos agrupar las aplicaciones que nos ofrece MaGraDa en tres partes:

- **Manejo de grafos (Grafo)**: Ésta es la parte donde se pueden crear grafos nuevos o abrir grafos ya creados desde fichero, modificarlos, borrarlos de memoria, seleccionarlos o guardarlos en un fichero para su tratamiento posterior.

- **Cálculos Básicos**: Hay una serie de características o propiedades básicas de los grafos que se pueden averiguar fácilmente con esta serie de métodos, tales como grado de un vértice, matriz de adyacencia o pesos, ver aristas (o arcos) que pueda tener el grafo. También, en el caso de que el grafo sea dirigido, MaGraDa nos ofrece la utilidad de obtener su correspondiente grafo no dirigido asociado. Para grafos no dirigidos obtiene un árbol generador de los muchos que pueda tener. Podemos estudiar, también desde

este menú, si dos grafos son isomorfos, ver qué vértices alcanzan a otros, así como qué vértices son alcanzados por otros. Nos indica además si el grafo es simple, cíclico, completo o conexo. Otra aplicación de gran interés es el cálculo de componentes conexas.

- **Algoritmos**: Esta última parte es la más importante de la aplicación. Dispone de algoritmos muy conocidos en el mundo de los grafos, tales como Warshall, Fleury, Caminos más cortos en grafos acíclicos, PERT, Dijkstra, Floyd-Warshall, Kruskal y Prim. Sin duda lo más importante aquí es que MaGraDa los aplica sobre los grafos en curso, de manera que el usuario pueda ver los resultados intermedios para así entender mejor el funcionamiento del correspondiente algoritmo.

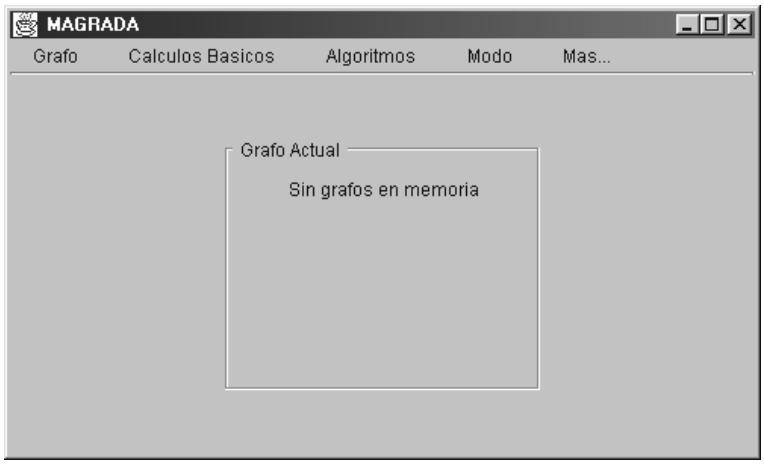
Una de las ventajas que ofrece MaGraDa es que puede tener en memoria varios grafos al mismo tiempo. De esta forma, el usuario puede seleccionar aquel que más le convenga en cada momento, sin tener que preocuparse por guardarlo en disco antes. La aplicación los mantendrá en memoria y al acabar la sesión, el mismo programa será quien recuerde al usuario si quiere guardar los grafos en archivo para un uso posterior. No obstante, es conveniente grabar los grafos en archivo nada más crearlos por posibles fallos inesperados en el sistema.

Llegados a este punto, lo mejor que podemos hacer es ir adentrándonos un poco más en cómo trabaja MaGraDa. Así nos daremos cuenta mucho mejor de sus posibilidades. En este capítulo, nos vamos a centrar en la parte correspondiente al manejo de datos. En la Sección 1.2 veremos la forma de introducir y manipular un grafo desde modo texto, en la Sección 1.3 estudiaremos la manera de hacer esto mismo desde modo gráfico. Pero antes de nada ejecutemos Ma-

GraDa. Recordemos que es una aplicación en lenguaje Java, por tanto, si no disponemos del icono correspondiente, tendremos que teclear java -jar magrada.jar. Aparecerá la siguiente pantalla:

1.2 Modo texto

Una vez situados en la portada de MaGraDa, pulsando *Comenzar*, aparecerá la siguiente pantalla principal del programa desde la que se trabaja cuando estamos en *Modo Texto*:



Como vemos, MaGraDa busca ya desde el principio y con su pantalla principal una de sus principales características, la sencillez. Claramente podemos observar, si desplegamos los distintos menús, cómo estructura MaGraDa su trabajo. Los tres primeros menús engloban toda la potencia de la aplicación. El cuarto menú *Modo* nos permitirá pasar a trabajar al ya mencionado *Modo Gráfico* del programa y el último hace referencia al aspecto de las ventanas de MaGraDa, al logotipo del programa y a sus autores.

Además de los menús, la pantalla principal también ofrece información básica de los datos del grafo que en cada momento se tenga seleccionado. Notemos que cuando no hay ningún grafo en memoria, prácticamente todas las opciones de los tres primeros menús están inhabilitadas, excepto aquellas que nos permiten agregar grafos o bien salir del programa.

1.2.1 Introducción de un grafo

A continuación vamos a ver cómo se crea un grafo desde modo texto. Para ilustrar el proceso consideraremos el siguiente grafo no dirigido:

$$G = \{V, A\}, \quad V = \{a, b, c, d\}, \; A = \{\{a, a\}, \{a, b\}, \{a, c\}, \{b, c\}, \{b, d\}, \{c, d\}\}.$$

Para la creación de un grafo desde modo texto debemos situarnos en la opción *Nuevo* del menú *Grafo*. Aparecerá una pantalla como ésta:

En ella insertamos los datos básicos de nuestro grafo. En una primera parte, le decimos si el grafo que se va a introducir es *Dirigido o No Dirigido* y *Ponderado o No Ponderado*. En nuestro ejemplo, por tanto, deberemos indicar que el grafo es no dirigido y no ponderado.

También debemos indicar el número de vértices que queremos que tenga el grafo (un número mayor que 0, lógicamente) y nos permitirá etiquetarlos (o nombrarlos), si queremos, con el botón *Nombrar*. En nuestro caso indicaremos que el grafo va a tener cuatro vértices y los llamaremos a, b, c y d, respectivamente.

Por último, según los primeros datos, nos dejará insertar *Aristas* o *Arcos* o bien acceder directamente a la *Matriz de Adyacencia* o *Pesos* para poder rellenarlas convenientemente. En nuestro caso, ya que se trata de un grafo no ponderado y no dirigido, podremos crear el grafo insertando las aristas o mediante la matriz de adyacencia. Por el momento nos limitaremos a insertar aristas. La introducción de un grafo mediante la matriz de adyacencia se verá más adelante. Si pulsamos el botón *Aristas* de la pantalla anterior nos saldrá algo así:

Se observa que hay dos cajas con todos los vértices en cada una de ellas. Sólo hay que ir seleccionando parejas de vértices. Como se trata de aristas, no importa el orden en el que pongamos los vértices de éstas. Una vez seleccionada una pareja, la agregamos e inmediatamente aparecerá en la ventana derecha

queriendo decir así, que ya pertenece al grafo. Si queremos eliminar alguna de las ya hechas, la formamos como si la fuéramos a introducir y pulsamos *Eliminar*.

En nuestro ejemplo, al final del proceso debe aparecer la pantalla de la siguiente forma:

Ahora si pulsamos *Aceptar*, volveremos a la pantalla inmediatamente anterior. Pulsando de nuevo *Aceptar* nos pedirá el nombre que le vamos a dar al grafo, por ejemplo grafo1. Una vez dado el nombre y pulsando de nuevo *Aceptar* nos aparecerá en pantalla un resumen de las características del grafo creado. Recordemos no obstante que aunque el grafo está disponible en memoria, no ha sido guardado en un fichero.

Si el grafo hubiera sido dirigido, la forma de proceder sería similar, no obstante tendríamos que indicárselo en la pantalla correspondiente y serían arcos lo que tendríamos que introducir en el grafo. La manera de hacerlo es igual que antes. La diferencia es que aquí sí que importa el orden de los vértices a la hora de formar el arco. Así, el vértice de origen será el vértice inicial del arco y el vértice de destino el vértice final del arco.

1.2.2 Borrando un grafo ya existente

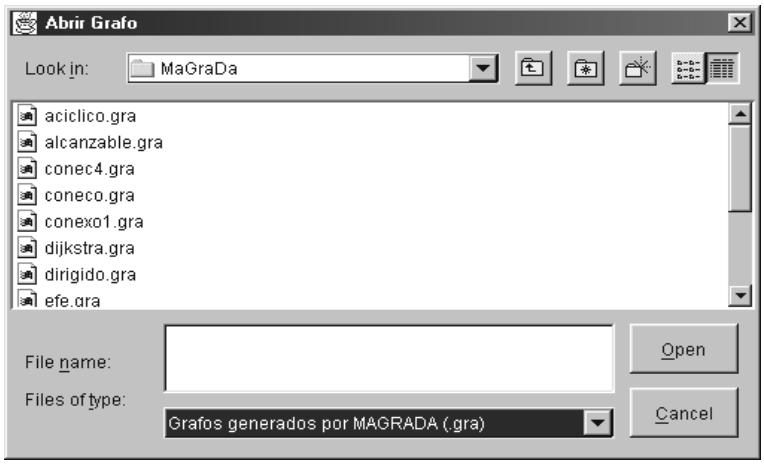
Cuando queramos borrar un grafo de memoria, sólo hay que situarse encima de la opción *Borrar* del menú *Grafo*. Aparecerá inmediatamente a la derecha un nuevo submenú con los nombres de todos los grafos que tengamos. No hay más que situarse encima del deseado y pulsar. Lo habremos eliminado. También habrá una opción suplementaria, *Todos*, por si queremos borrar todos de memoria.

1.2.3 Seleccionando un grafo

Para poder trabajar con un grafo, tiene que ser el actual. Para ello tenemos la opción *Seleccionar*. Es similar a la anterior. La diferencia está en que antes lo borrábamos y ahora lo seleccionamos para poder trabajar con él. Cuando seleccionemos un grafo distinto al actual, se actualizará la información que se presenta en la pantalla principal.

1.2.4 Abrir y guardar un grafo desde/en archivo

Estas opciones son muy útiles y nos ahorrarán bastante trabajo. Al pulsar la opción *Abrir* nos presenta un selector de ficheros análogo a éste:



Esto nos ofrece la posibilidad de seleccionar los archivos con extensión "gra", que son los generados por MaGraDa. Sólo hay que pinchar el que queramos y lo cargará en memoria al instante. Para *Guardar* un grafo en un archivo, se hará de forma análoga, salvo que tendremos que darle un nombre al archivo donde queremos guardar el grafo. Los nombres del grafo y del archivo donde lo guardamos no tienen, necesariamente, que llamarse igual. Además, también nos da la posibilidad de guardar todos los grafos que tengamos en memoria, aunque en este caso MaGraDa, por comodidad, toma el nombre del grafo para crear el fichero.

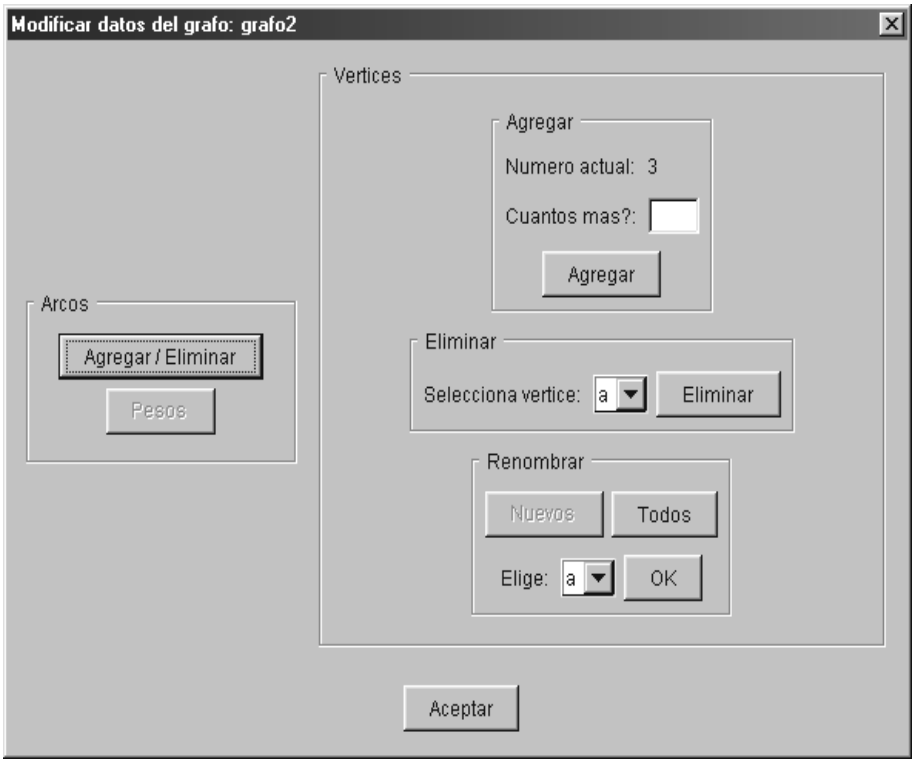
1.2.5 Modificación de un grafo

Una vez creado un grafo siempre existe la posibilidad de modificarlo. Previamente se selecciona el grafo que deseamos modificar si éste no es el actual, y se pulsa la opción *Modificar* del menú *Grafo*. Así por ejemplo, supongamos

que se desea modificar el siguiente grafo dirigido que previamente se ha creado y denominado grafo2:

$$G = \{V, A\}, \quad V = \{a, b, c\}, \ A = \{(a, b), (a, c), (b, c), (c, b)\}.$$

Observamos que se obtiene la siguiente pantalla:



Vemos que está dividida en dos partes:

- **Arcos** (o aristas en los no dirigidos): Nos dejará agregar nuevos arcos (aristas) o eliminarlos.
- **Vértices**: Nos dejará agregar nuevos vértices, eliminar aquellos que seleccionemos, así como renombrarlos o simplemente darles nombre si no tenían.

Así por ejemplo, supongamos que se desea modificar el grafo anterior (grafo2) añadiendo un nuevo vértice d y el arco (c, d), y eliminando el vértice b. En este caso, el grafo resultante sería: $G = \{V, A\}$, $V = \{a, c, d\}$, $A = \{(a, c), (c, d)\}$. Notemos que ahora es éste el grafo que posee MaGraDa en memoria como grafo2.

1.2.6 Salir de MaGraDa

Cuando queramos cerrar la sesión, ejecutamos la opción *Salir* del menú *Grafo*. Nos dirá si queremos salvar los grafos que tenemos antes de salir. Después, cerrará la aplicación hasta una nueva ocasión.

1.3 Modo gráfico

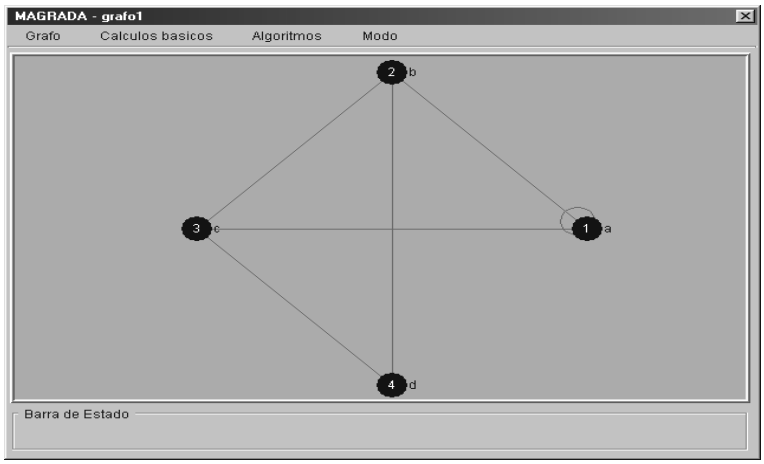
El modo gráfico es la segunda forma de trabajo que ofrece MaGraDa a sus usuarios. Las posibilidades que nos da son las mismas que en modo texto, aunque la forma de ver los resultados no siempre es la misma. Vamos a comentar un poco dichas posibilidades. Se tienen tres menús, además de un cuarto para poder retornar al modo texto:

- **Grafo**: Este menú se utiliza para manipular grafos. Comentaremos posteriormente aquellas opciones cuya forma de trabajo difiera del modo texto.
- **Cálculos Básicos**: Es análogo al menú *Cálculos Básicos* del modo texto. Pero aquí se tratan las soluciones de un modo más gráfico.

- **Algoritmos**: Aparecen los mismos algoritmos que en el modo texto, aunque como en el caso anterior la resolución de éstos se realiza de forma gráfica siempre que es posible.

En primer lugar, y para ponernos en situación vamos a ofrecer una pantalla que nos acerque lo antes posible a este modo de trabajo. Dicha pantalla muestra el grafo1 definido en la Sección 1.2.1, es decir, el grafo:

$$G = \{V, A\}, \quad V = \{a, b, c, d\}, \; A = \{\{a, a\}, \{a, b\}, \{a, c\}, \{b, c\}, \{b, d\}, \{c, d\}\}.$$



Podemos diferenciar en la pantalla tres zonas:

- **Barra de menús**: Nos permite acceder a los servicios que ofrece la aplicación.

- **Lienzo**: Llamaremos lienzo a la zona de dibujo donde aparecerán los grafos.

- **Barra de Estado**: Es la parte inferior. Mediante ella, MaGraDa intentará en todo momento informarnos y ayudarnos en el trabajo con los grafos.

1.3.1 El lienzo

Sobre el lienzo mostraremos los grafos de la siguiente forma:

- **Vértices**: Los representaremos mediante círculos negros. Dentro de cada uno colocaremos su número en blanco. Cuando apliquemos ciertos algoritmos, pueden cambiar de color para resaltarlos. También podrán alojar en su interior un segundo número correspondiente a pesos, cuando el grafo sea ponderado. Ya lo explicaremos más adelante.

- **Aristas**: Las representaremos mediante líneas rojas. Unirán dos vértices distintos, o bien el mismo formando un bucle. Entre dos vértices podrán existir varias aristas, apareciendo paralelas entre sí, conforme se van añadiendo.

- **Arcos**: Los representaremos mediante flechas azules. También podrá haber más de un arco entre el mismo par de vértices (en cualquier sentido).

- **Pesos**: Cuando el grafo sea ponderado, mostraremos los pesos de las aristas (o arcos) en blanco sobre cuadrados de color morado. Para evitar posibles colisiones, los cuadrados no se dibujarán en los centros de las aristas (o arcos). A esta cuestión volveremos cuando tratemos dichos grafos.

- **Rectángulo FIN**. Cuando activemos cualquier algoritmo de forma que exista interacción con el grafo, aparecerá en la esquina superior derecha del lienzo un rectángulo coloreado con la palabra FIN dentro (cada método o algoritmo tiene su propio color). Cuando queramos abandonar el método

o algoritmo actual, no tendremos más que pinchar sobre él para que el lienzo vuelva a la normalidad.

Existen algoritmos que resaltan de color blanco determinadas aristas (o arcos) del grafo con diferentes intenciones. Ya lo explicaremos más adelante. Otro aspecto importante es que muchos métodos o algoritmos son bloqueados al ser activados otros y no se podrá acceder a ellos hasta pulsar FIN.

1.3.2 Introducción de un grafo

Es desde el menú *Grafo*, como en el modo texto, desde donde se crearán o modificarán grafos en modo gráfico. Para crear un grafo nos situaremos en la opción *Nuevo*. Veamos una pantalla con dicha opción activada:

Como se puede observar, la pantalla que presenta el menú *Nuevo* del modo gráfico es análoga a su correspondiente en modo texto. Cabe destacar, no obstante, que en este caso, el usuario puede decidir donde poner los vértices o si prefiere, disponerlos de forma concéntrica en forma de círculo en el centro del lienzo (activando la opción *Distribuir uniformemente*). Veamos con un ejemplo cómo proceder: Supongamos que se desea introducir el siguiente grafo dirigido

$$G = \{V, A\}, \quad V = \{a, b, c, d, e\},$$

$$A = \{(a, b), (b, a), (c, b), (c, a), (c, d), (c, e), (d, a), (c, c)\}.$$

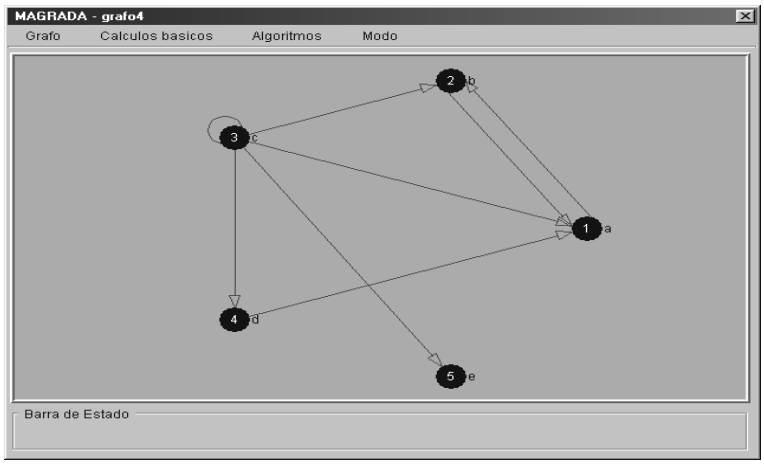
Para ello realizamos los siguientes pasos:

Paso 1. Se rellena la ventana anterior con los datos del grafo. Es decir, especificaremos el tipo de grafo que queremos crear y número de vértices que tendrá dicho grafo. En caso de que queramos que sea MaGraDa quien disponga los vértices en el lienzo activaremos la opción *Distribuir uniformemente*. Pulsaremos *Aceptar* e introduciremos el nombre del grafo, por ejemplo grafo4.

Paso 2. Si se ha activado la opción *Distribuir uniformemente*, aparecerán los vértices en el lienzo. En caso contrario nos situaremos sobre el lugar donde queramos que esté cada vértice y pulsaremos el botón izquierdo del ratón para que éste aparezca. Hacemos esto hasta introducir todos los vértices deseados. Como podréis observar, dentro de cada vértice aparece un número, que es el nombre que por defecto le da MaGraDa a cada vértice. Si hemos dado nombres a los vértices y queremos verlos hay que activar la opción *Vértices con nombre*. Si deseamos que vuelvan a desaparecer sus nombres activaremos la opción *Vértices sin nombre*.

Paso 3. Una vez introducidos todos los vértices, MaGraDa pasa directamente al modo de introducción de aristas o arcos, según sea el caso. Para especificar las aristas o arcos entre dos vértices nos situamos sobre un vértice, pulsamos el botón izquierdo del ratón y luego sobre el otro vértice, volviendo a pulsar el botón izquierdo del ratón. Repetiremos el proceso hasta especificar todas las aristas o todos los arcos deseados. Para finalizar el proceso pulsaremos FIN (aparece en el ángulo superior derecho).

Una vez realizados todos los pasos, y suponiendo que se ha elegido la opción de que sea MaGraDa quien distribuya los vértices del grafo del ejemplo anterior, el grafo quedaría dibujado de la siguiente forma:

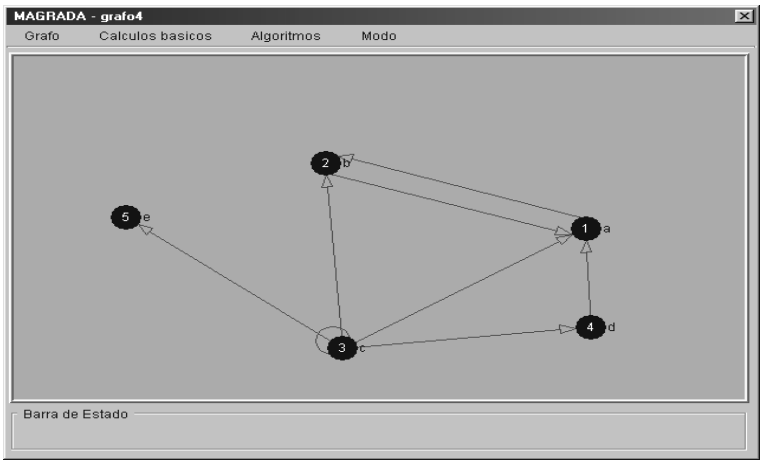


Notemos que se ha activado la opción que permite ver los nombres de los vértices.

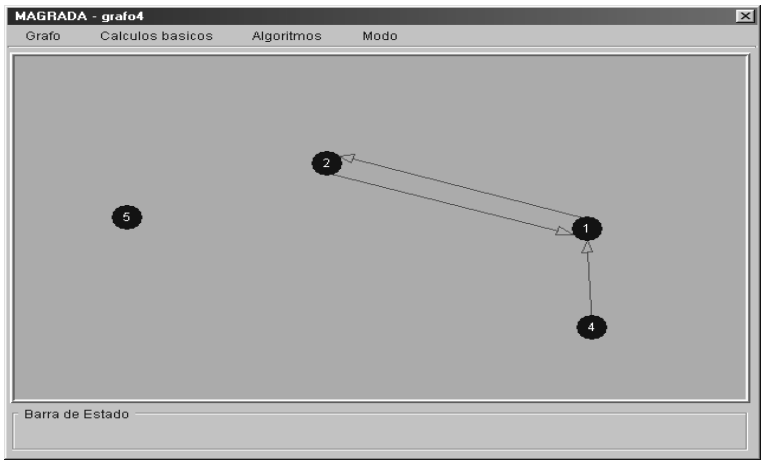
Desde modo gráfico y desde la opción *Modificar* del menú *Grafo* también se puede modificar el grafo. La forma de proceder es análoga a la de creación de un grafo, es decir, para añadir o eliminar vértices nos situamos en la opción correspondiente y situamos el ratón donde deseemos añadir el vértice o encima del vértice que se desea eliminar. Pulsamos entonces el botón izquierdo del

ratón y se realizará la operación deseada. La forma de eliminar arcos o aristas es similar a la de agregarlos, que se ha visto cuando se explicó la forma de introducir un grafo en modo gráfico. Desde esta misma opción también se puede cambiar el nombre de los vértices y recolocarlos. En ambos casos nos debemos situar, de nuevo, sobre el vértice con el que deseemos trabajar y pulsar el botón izquierdo del ratón. Si estamos en la opción *Renombrar vértices* nos pedirá que le demos el nuevo nombre. Si lo que queremos es cambiar el vértice de sitio, tendremos que mover el cursor hasta el sitio deseado y pulsar, de nuevo, el botón izquierdo del ratón.

Como ejemplo de lo explicado, mostramos a continuación, imágenes con resultados de aplicar algunas de estas opciones al grafo4. En la primera imagen hemos cambiado los vértices de sitio.



En la siguiente imagen hemos eliminado del grafo4 el vértice c y hemos indicado que los vértices aparezcan sin su nombre.



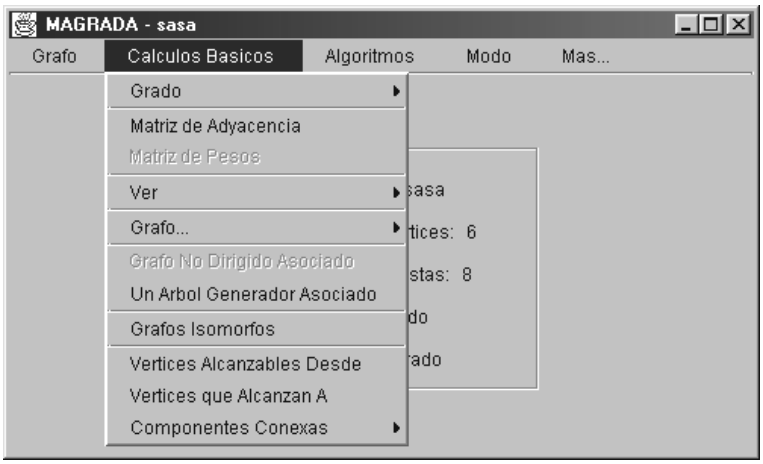
El resto de opciones que nos ofrece este primer menú (*Grafo*) relacionadas con la manipulación de un grafo son análogas a las explicadas en la sección correspondiente al modo texto. No obstante, para seleccionar o borrar un grafo lo hace ahora mediante nuevos diálogos y no con submenús como antes.

Capítulo 2

Fundamentos

2.1 Introducción

En este capítulo vamos a ver gran parte de las posibilidades que ofrece el menú *Cálculos Básicos* de MaGraDa, tanto desde modo texto como desde modo gráfico. Este menú es el segundo de los tres menús más importantes del programa. Como se puede apreciar en la siguiente figura, nos ofrece múltiples opciones.



La característica que comparten todas ellas es que las operaciones que tienen que hacer con los grafos son, en principio, más sencillas que si les aplicamos los algoritmos del tercer menú. En la Sección 2.2 comentaremos algunas de estas opciones con un poco de profundidad, y la Sección 2.3 estará dedicada a la matriz de adyacencia de un grafo.

2.2 Cálculos básicos

Las opciones del menú *Cálculos Básicos* que se van a estudiar en esta sección son:

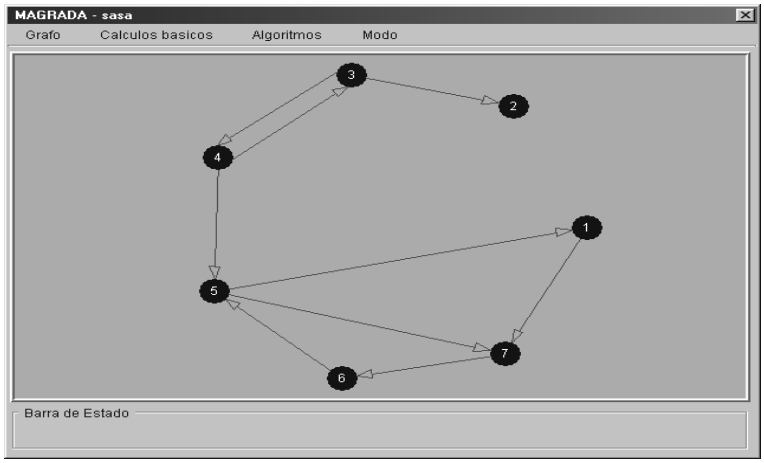
- Grado.
- Ver (aristas o arcos).
- Grafo (simple, cíclico, completo y conexo).

- Grafo no dirigido asociado.
- Grafos isomorfos.

En el resto de la sección vamos a comentar un poco cada una de estas opciones.

2.2.1 Grado

Para ilustrar esta opción dibujaremos con MaGraDa el siguiente grafo dirigido:

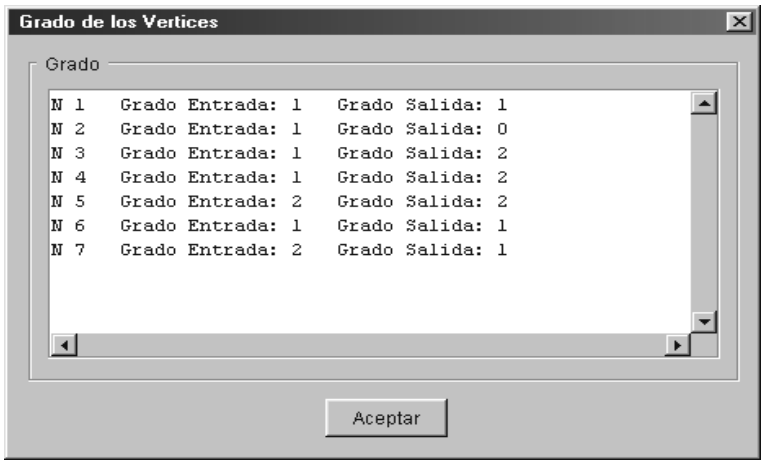


Recordemos que si el grafo es no dirigido hablaremos simplemente de *grado de un vértice*. Sin embargo, si como en nuestro caso, el grafo es dirigido habrá que diferenciar entre *grado de entrada* y *grado de salida* de un vértice. Esta opción está dividida en dos subopciones:

- **Grado de un sólo vértice**: Nos dirá el grado del vértice que queramos. Para ello, desde modo texto, nos presentará una pantalla de selección de vértices como ésta:

Notemos que si los vértices tuvieran nombre dado, en dicha pantalla aparecería el nombre. Como no es así, aparece sólo el número. En cuanto lo hayamos seleccionado, nos informará del grado del vértice en una pantalla de información.

- **Grado de todos los vértices**: Utilizamos esta opción, si queremos ver, directamente, desde modo texto, el grado de todos los vértices del grafo. Para nuestro ejemplo nos aparecerá algo así:



Aquí vemos cómo nos muestra el grado de todos los vértices del grafo de una manera sencilla y eficaz. Fijémonos, por ejemplo, en el vértice 3.

Su grado de entrada es 1. Por tanto, sólo hay un arco en el grafo cuyo vértice final sea 3. Sin embargo el grado de salida es dos, por lo tanto hay dos arcos en el grafo cuyo vértice inicial es el 3.

Desde modo gráfico podemos realizar estos mismos cálculos. La diferencia estriba en que cuando se quiere calcular el grado de un vértice debemos situarnos encima de él y pulsar el botón izquierdo del ratón. Para salir de la opción se debe pulsar el recuadro FIN situado en el ángulo superior derecho.

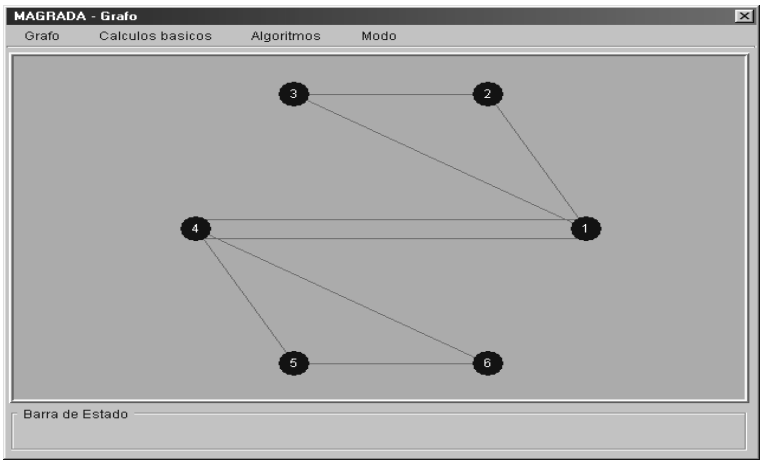
2.2.2 Ver

Cuando seleccionamos un grafo desde modo gráfico, dicho grafo aparece en el lienzo. Sin embargo, desde modo texto sólo nos aparece información general sobre él, pero no cuáles son las aristas o arcos del grafo. Para ver dichas aristas o arcos simplemente ejecutaremos, desde modo texto, la opción *Ver*. Las pantallas, que nos mostrará serán similares a las que el programa nos ofreció a la hora de introducir estos datos desde modo texto, aunque ahora son pantallas informativas. Si el grafo tiene nombrados sus vértices, MaGraDa nos mostrará la información usando estos nombres. En otro caso, usará los números.

2.2.3 Grafo (simple, cíclico, completo y conexo)

Son cuatro características importantes de los grafos. Cuando las ejecutamos, tanto desde modo texto como desde modo gráfico, el programa realizará cálculos

internos sobre el grafo actual para ofrecemos la información de forma inmediata. Si la respuesta es afirmativa, así nos lo hará saber en cada caso. Si no es así, también lo hará y además nos dirá la razón que le ha llevado a dar esa respuesta. Así por ejemplo, supongamos que se ha introducido con MaGraDa el siguiente grafo no dirigido:



Entonces MaGraDa nos dirá que el grafo no es simple, ya que hay más de una arista que une el mismo par de vértices, por ejemplo hay dos aristas uniendo los vértices 1 y 4.

Un grafo cíclico, es aquel que contiene algún ciclo (también denominado circuito en el caso dirigido). Por tanto, como indica MaGraDa, el grafo que nos ocupa lo es, ya que por ejemplo, el vértice 1 aparece al menos en un ciclo, por ejemplo en el ciclo $\{1,4\},\{4,1\}$. Animamos al lector a que observe el grafo y vea si dicho vértice aparece en algún otro ciclo.

Por otra parte los grafos completos son aquellos en los que hay al menos una arista o arco (según sea no dirigido o dirigido) uniendo cada par de vértices. Claramente el grafo que nos ocupa no es completo ya que, como indica MaGraDa,

por ejemplo, entre los vértices 1 y 5 no hay ninguna arista.

Un grafo es conexo, si todo par de vértices está conectado. En el grafo que nos ocupa, al ser no dirigido y relativamente pequeño, es fácil observar que esto ocurre, ya que no hay ningún vértice o vértices desconectados del resto, de hecho podríamos ir recorriendo, a lo largo de las aristas (con posibilidad de repetirlas), todos los vértices con un lápiz sin tener que levantar el lápiz del papel. No obstante, cuando el grafo es dirigido o de gran tamaño, esto no es tan fácil de ver. Para ello se dispone de algoritmos que se tratarán en el Capítulo 3. MaGraDa hace uso interno de estos algoritmos para dar solución a esta cuestión.

2.2.4 Grafo no dirigido asociado

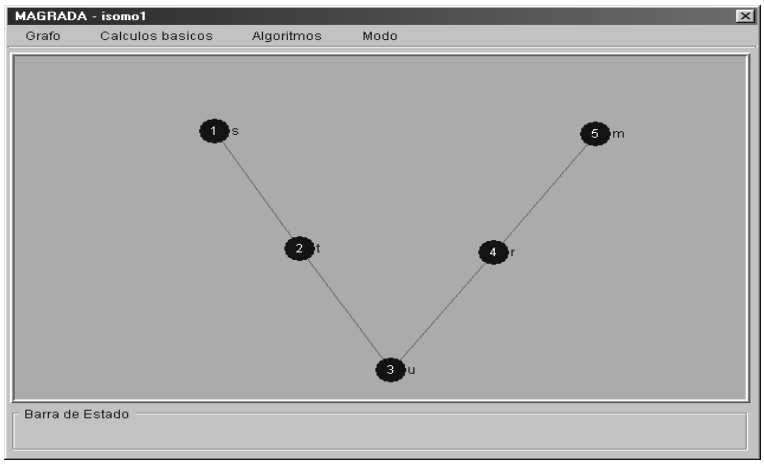
Esta es una opción sólo válida para grafos dirigidos. Se trata de que el programa genere un nuevo grafo, no dirigido, a partir del grafo dirigido que tengamos seleccionado. Conservará todas sus características con la diferencia de que los arcos los transformará en aristas. Cuando ejecutemos la opción, el programa nos preguntará qué nombre queremos darle al nuevo grafo, si dicho nombre es válido lo guardará en memoria y lo seleccionará como actual y a partir de ahí, si no hacemos otra cosa, trabajaremos con este grafo.

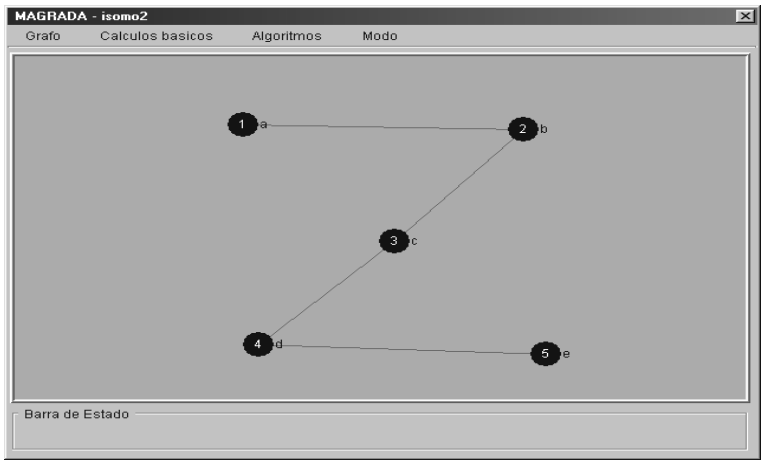
2.2.5 Grafos isomorfos

Supongamos que $G_1 = (V_1, A_1)$ y $G_2 = (V_2, A_2)$, son grafos no dirigidos (dirigidos). Se dice que G_1 y G_2, son grafos isomorfos si existe una aplicación biyectiva $f : V_1 \longrightarrow V_2$, tal que, $\{u, v\}$ es una arista de G_1, si y sólo si, $\{f(u), f(v)\}$ es una arista de G_2 ((u, v) es un arco de G_1, si y sólo si, $(f(u), f(v))$ es un arco de G_2).

Con MaGraDa podemos estudiar si dos grafos son isomorfos desde la opción *Grafos Isomorfos* del menú *Cálculos Básicos*. Ésta es la única opción del programa que permite trabajar con más de un grafo al mismo tiempo. Puede ser que hasta el grafo actual no sea usado. Eso sí, tienen que ser no ponderados. Para ello, tanto desde modo texto como gráfico, lo primero que hace MaGraDa es pedirnos dos grafos. Así, nos ofrecerá una pantalla como ésta:

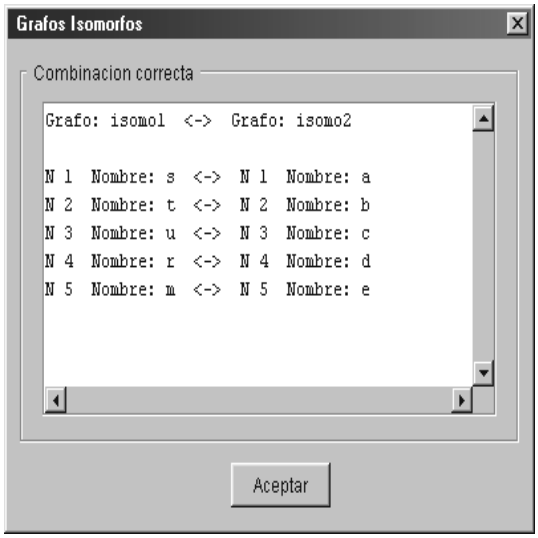
En ambas casillas aparecen los nombres de todos los grafos no ponderados de los que dispone en memoria el programa. Seleccionaremos dos grafos, por ejemplo los siguientes grafos, isomo1 e isomo2 dibujados previamente con MaGraDa, que representan la letra V y Z, respectivamente.





Si los grafos no son isomorfos por cualquier causa, nos lo dirá mediante un mensaje. Si por el contrario lo son, además de indicarlo, nos dará la aplicación biyectiva que ha dado lugar a que todas las condiciones para que sean isomorfos se den. Recordemos que se tienen que cumplir condiciones indispensables como que ambos grafos sean los dos dirigidos o los dos no dirigidos, que tengan el mismo número de vértices y que la sucesión de grados de los vértices sea la misma en ambos grafos.

En particular, ésta será la respuesta que nos dé MaGraDa, para el ejemplo considerado.



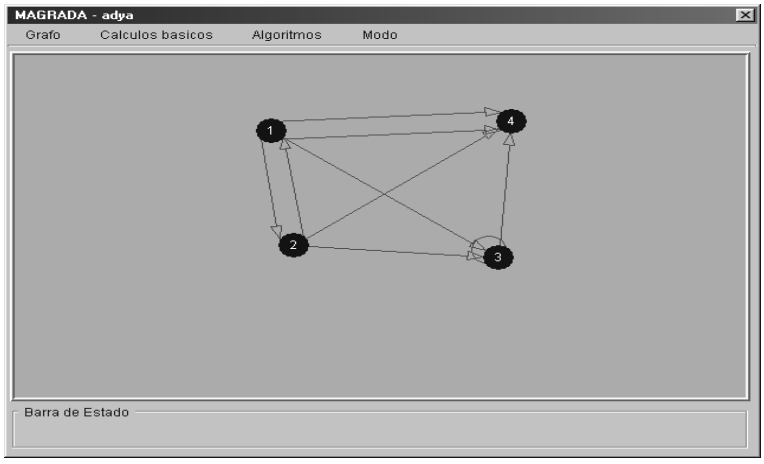
Por tanto, si nos fijamos en los nombres dados a los vértices de ambos grafos, obtenemos que una aplicación biyectiva $f : V_1 \longrightarrow V_2$, con $V_1 = \{s, t, u, r, m\}$, el conjunto de vértices de isomo1 y $V_2 = \{a, b, c, d, e\}$, el conjunto de vértices de isomo2, que cumple las condiciones para que los grafos sean isomorfos viene definida por $f(s) = a$, $f(t) = b$, $f(u) = c$, $f(r) = d$, $f(m) = e$, ya que cumple que:

- $\{s, t\}$ es arista de isomo1 $\Longleftrightarrow \{f(s), f(t)\} = \{a, b\}$ es arista de isomo2.
- $\{t, u\}$ es arista de isomo1 $\Longleftrightarrow \{f(t), f(u)\} = \{b, c\}$ es arista de isomo2.
- $\{u, r\}$ es arista de isomo1 $\Longleftrightarrow \{f(u), f(r)\} = \{c, d\}$ es arista de isomo2.
- $\{r, m\}$ es arista de isomo1 $\Longleftrightarrow \{f(r), f(m)\} = \{d, e\}$ es arista de isomo2.

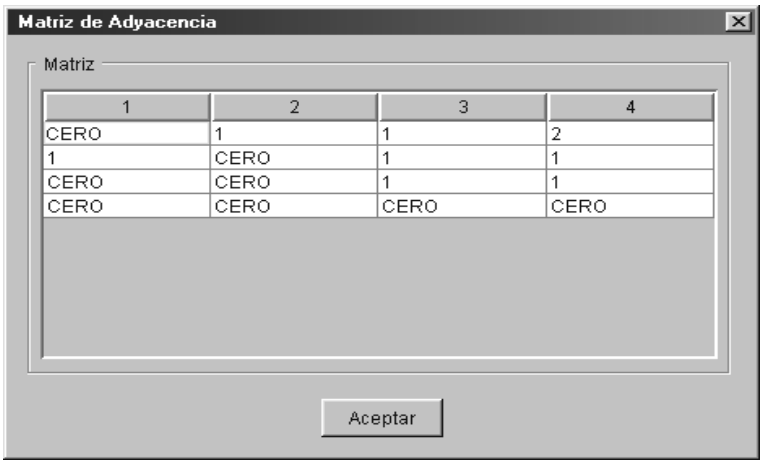
2.3 Matriz de adyacencia

Además de las formas de introducción de un grafo vistas en el capítulo anterior, MaGraDa nos permite introducirlo mediante su matriz de adyacencia. Esto se hará desde la opción *Nuevo* del Menú *Grafo* del modo texto, vista en el capítulo anterior, pulsando *Matriz de Adyacencia*, una vez indicado cómo es el grafo y el número de vértices que va a tener.

Ya haya sido introducido el grafo de esta forma o mediante la introducción de sus arcos o aristas, MaGraDa nos da la posibilidad de acceder a dicha matriz de una forma muy fácil, desde la opción *Matriz de adyacencia* del menú *Cálculos Básicos*. Vamos a ilustrar esto con un ejemplo. Supongamos que se ha introducido con MaGraDa el siguiente grafo dirigido:



Tanto desde modo texto como desde modo gráfico, la matriz de adyacencia obtenida para dicho grafo viene dada en la siguiente pantalla:



Una de las ventajas que MaGraDa ofrece en este método es que resalta de una manera significativa aquellas celdas de la matriz de adyacencia donde entre el par de vértices correspondientes no hay aristas (o arcos). Lo hace mediante la palabra CERO. De esta forma, es mucho más fácil ver de inmediato las casillas que nos interesan. No obstante, si es el usuario el que introduce el grafo mediante su matriz de adyacencia, deberá utilizar para ello números no negativos.

Capítulo 3

Estudio de la accesibilidad y conectividad

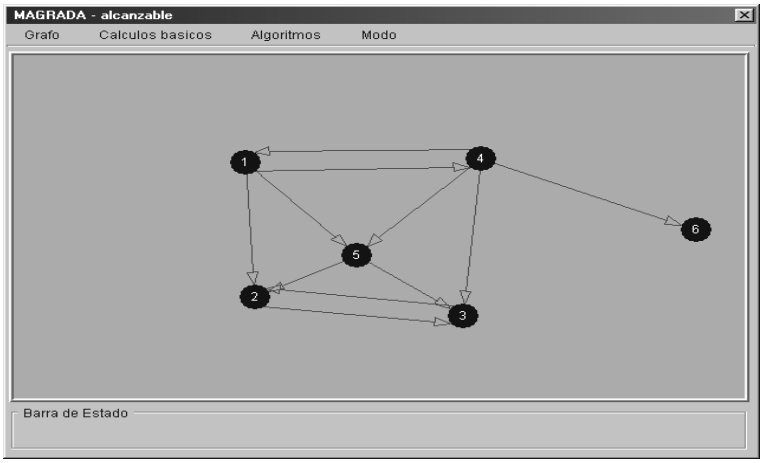
3.1 Introducción

Como se indicó en el capítulo anterior, cuando un grafo es pequeño y sobretodo si no es dirigido es fácil ver si dicho grafo es conexo. Sin embargo, conforme aumenta el tamaño del grafo, esto se hace más difícil. En este capítulo vamos a ver qué hace MaGraDa para ver si un grafo es conexo o no. En la Sección 3.3 estudiaremos dos algoritmos que permiten calcular las distintas componentes conexas de un grafo. Recordemos que si se obtiene una única componente conexa el grafo será conexo y en caso de obtener más de una, el grafo será no conexo. Previamente, en la Sección 3.2 estudiaremos algunos conceptos relacionados, como accesibilidad y acceso.

3.2 Matriz de accesibilidad y matriz de acceso

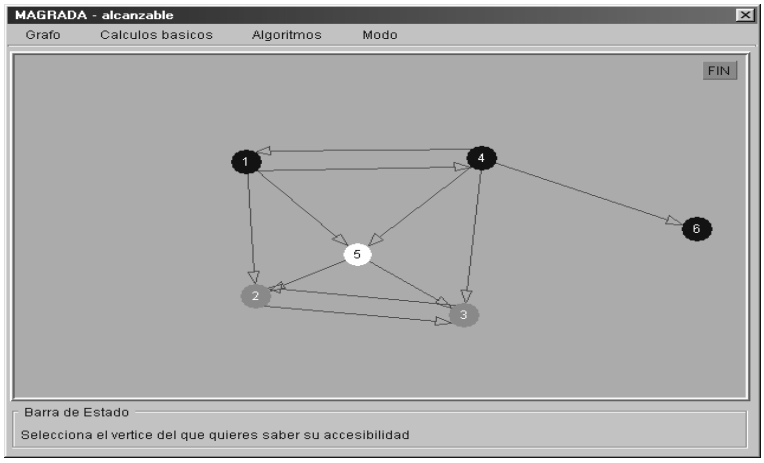
3.2.1 Vértices alcanzables desde otro

A partir de la matriz de adyacencia, se puede obtener el conjunto de vértices que son alcanzables desde uno dado, ya sea porque estén unidos directamente o porque exista un camino entre ellos. Con la ayuda de MaGraDa podemos saber cuál es dicho conjunto. Para ello debemos situarnos en la opción *Vértices Alcanzables Desde*, del menú *Cálculos Básicos*. Por ejemplo, supongamos que se ha introducido con MaGraDa el siguiente grafo dirigido:



Desde modo texto, MaGraDa nos ofrecerá una pantalla que nos da una relación de todos los vértices del grafo con los vértices a los que cada uno de ellos puede alcanzar. Si los vértices tuvieran nombre, éstos aparecerían referenciados con ellos. Si no es así muestra los números y ya está. Para salir, pulsaremos *Aceptar*.

Desde modo gráfico, al pinchar el vértice que se desea estudiar, cambiarán de color todos los vértices que son alcanzables desde dicho vértice. Así por ejemplo, en la siguiente pantalla se observa que los vértices alcanzables desde el vértice 5 son el 2, el 3 y el propio 5, por tanto, si denotamos por $R(v)$ el conjunto de vértices alcanzables desde el vértice v obtenemos $R(5) = \{2, 3, 5\}$.



Con el fin de familiarizarse con esta opción animamos al lector a que calcule $R(v)$, para el resto de vértices desde los dos modos, y a partir de los resultados obtenidos, la matriz de accesibilidad R, que se necesitará para ilustrar algunos métodos.

Recordamos que dicha matriz viene definida de la siguiente forma:

Definición 3.1. Sea $G = (V, A)$, un grafo dirigido tal que $V = \{x_i\}_{i=1}^{n}$. Llamamos matriz de accesibilidad asociada al grafo G a la matriz cuadrada de orden n definida por

$$R = [r_{ij}]_{1 \leq i,j \leq n} \quad / \quad r_{ij} = \begin{cases} 1 & \text{si } x_i \text{ alcanza a } x_j \\ 0 & \text{en otro caso} \end{cases}$$

3.2.2 Vértices que alcanzan a otro

Para ver qué vértices alcanzan a un vértice dado nos debemos situar en la opción *Vértices que Alcanzan A*. La forma en que MaGraDa presenta los resultados es similar a la de la opción anterior. La diferencia está en que ahora, para cada vértice nos dirá cuáles son los vértices que lo alcanzan a él, bien directamente o mediante un camino. Así por ejemplo, para el grafo del ejemplo anterior desde modo texto obtendríamos:

En este caso, si denotamos por $Q(v)$ el conjunto de vértices que alcanzan al vértice v obtenemos que $Q(1) = \{1, 4\}$, $Q(2) = \{1, 2, 3, 4, 5\}$, $Q(3) = \{1, 2, 3, 4, 5\}$, $Q(4) = \{1, 4\}$, $Q(5) = \{1, 4, 5\}$, $Q(6) = \{1, 4, 6\}$.

Recordemos ahora el concepto de matriz de acceso:

Definición 3.2. Sea $G = (V, A)$ un grafo dirigido tal que $V = \{x_i\}_{i=1}^{n}$. Llamamos matriz de acceso asociada al grafo G a la matriz cuadrada de orden n

definida por

$$Q = [q_{ij}]_{1 \leq i,j \leq n} \ / \ q_{ij} = \begin{cases} 1 & \text{si } x_i \text{ es alcanzable desde } x_j \\ 0 & \text{en otro caso} \end{cases}$$

De aquí se deduce por tanto que la matriz de acceso del grafo que nos ocupa es

$$Q = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

3.2.3 Accesibilidad y Algoritmo de Warshall

En la sección anterior se ha tratado la forma de calcular la matriz de accesibilidad de un grafo, a partir de la matriz de adyacencia. A continuación vamos a ver otra forma, más económica, de calcular la matriz de accesibilidad: el algoritmo de Warshall. Construiremos una sucesión de matrices $R_0, R_1, \ldots, R_n$, donde

- n es el número de vértices del grafo.
- R_0, es una matriz calculada a partir de la matriz de adyacencia A, del grafo, donde los elementos distintos de 0 se han cambiado por unos.

- R_n, es la matriz de accesibilidad pero sin considerar los caminos de longitud cero. Para obtener la matriz de accesibilidad, hay que añadir unos en la diagonal principal, y así incluir dichos caminos.

Si denotamos a los elementos de esta sucesión de matrices por

$$R_k = [r_{ij}^{(k)}]_{1\leq i,j\leq n}, \quad k = 0, 1, 2, \ldots, n,$$

estos elementos pueden ser calculados de acuerdo con la siguiente condición:

$$r_{ij}^{(k)} = 1 \Longleftrightarrow \begin{cases} r_{ij}^{(k-1)} = 1 \\ \text{ó} \\ r_{ik}^{(k-1)} = r_{kj}^{(k-1)} = 1 \end{cases}$$

$$k = 1, 2, \ldots, n.$$

Así, por ejemplo, consideremos un grafo simple G con 4 vértices, cuya matriz de adyacencia es:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} = R_0.$$

La sucesión de matrices que genera el algoritmo de Warshall es:

$$R_1 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad R_2 = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$$R_3 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad R_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

El procedimiento ilustrado en el ejemplo anterior produce el siguiente algoritmo para calcular la matriz R_n de un grafo con n vértices, a partir de su matriz de adyacencia A.

ALGORITMO DE WARSHALL

$R \leftarrow R_0$

Para $k = 1$ hasta n

  Para $i = 1$ hasta n

    Para $j = 1$ hasta n

      $R(i,j) \leftarrow R(i,j) \vee (R(i,k) \wedge R(k,j))$

MaGraDa realiza este algoritmo desde la opción *Warshall* del menú *Algoritmos*. Tanto desde modo texto como modo gráfico tiene dos posibilidades de uso:

- **Por Pasos**: Nos indica los distintos pasos, presentando la sucesión de matrices, correspondientes cada una de ellas, a cada iteración del algoritmo. El último paso consistirá en transformar en 1, los elementos diagonales nulos de R_n, para así, obtener la matriz de accesibilidad R tal y como se ha definido aquí, ya que por convenio todo vértice es alcanzado por él mismo por un camino de longitud 0. Para una mejor lectura, MaGraDa indica los elementos nulos de esta matriz con la palabra CERO.

- **Resultado Final**: Sólo muestra la matriz de accesibilidad, obtenida a partir de dicho algoritmo.

3.3 Cálculo de componentes conexas

En esta sección vamos a ver cómo calcula MaGraDa las componentes conexas de un grafo. Consideremos para ello un grafo dirigido $G = (V, A)$. Entonces, MaGraDa lo que hace es calcular $R \otimes Q$, donde $\otimes$ representa un producto elemento a elemento. Entonces la componente conexa de un vértice x_i se calcula viendo qué columnas tienen un 1 en la fila i.

Este algoritmo lo resuelve MaGraDa desde la opción *Componentes Conexas* del menú *Cálculos Básicos*. Desde modo texto, en esta opción, aparecen dos subopciones:

- **Por Pasos**: Nos mostrará cómo calcula las componentes conexas por pasos. Primero la matriz de accesibilidad R, luego su traspuesta Q, y luego la matriz $R \otimes Q$, resultante de multiplicar elemento a elemento las dos primeras matrices. Por último muestra las componentes conexas. Este resultado será la interpretación de la matriz $R \otimes Q$.

- **Resultado final**: Nos ofrece directamente las componentes conexas del grafo, es decir, la última parte de lo que nos ofrecía la subopción anterior.

Desde modo gráfico también aparecen dos opciones pero no exactamente iguales:

- **Por Pasos**: Nos mostrará cómo calcula las componentes conexas por pasos, de forma análoga a como lo hizo desde modo texto
- **Pinchando en el grafo**: Al pinchar un vértice enciende del mismo color todos los vértices que están en la misma componente conexa que el pinchado.

Así, si al grafo de nuestro ejemplo, le aplicamos este método, MaGraDa obtiene las matrices

$$R = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \qquad Q = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1, \end{bmatrix},$$

y

$$R \otimes Q = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Para calcular, por ejemplo, la componente conexa que contiene al vértice 1, nos situamos en la primera fila, como sólo aparecen unos en la primera y cuarta columna de dicha fila, la componente conexa estará formada por los vértices $\{1, 4\}$. De forma análoga se obtendrían el resto de componentes conexas, que son $\{2, 3\}, \{5\}$ y $\{6\}$.

Capítulo 4

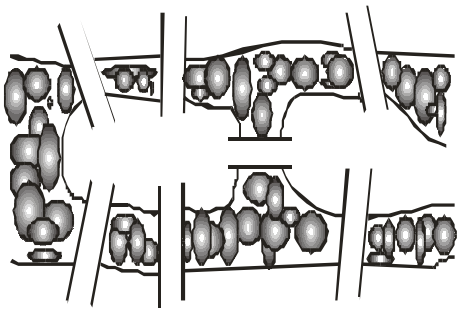
Problema de recorrido de aristas

4.1 Introducción

En este capítulo, vamos a ver cómo utilizar MaGraDa, para estudiar el problema de recorrido de aristas. Concretamente, la Sección 4.2 está dedicada al *algoritmo de Fleury*.

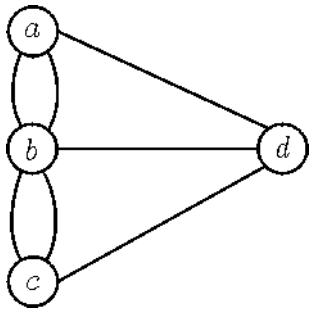
4.2 Algoritmo de Fleury

Durante el siglo XVIII, Königsberg fue una populosa y rica ciudad de la zona oriental de Prusia. Esta ciudad estaba dividida en cuatro zonas por el río Pregel. Siete puentes comunicaban estas zonas, como se muestra en la figura:



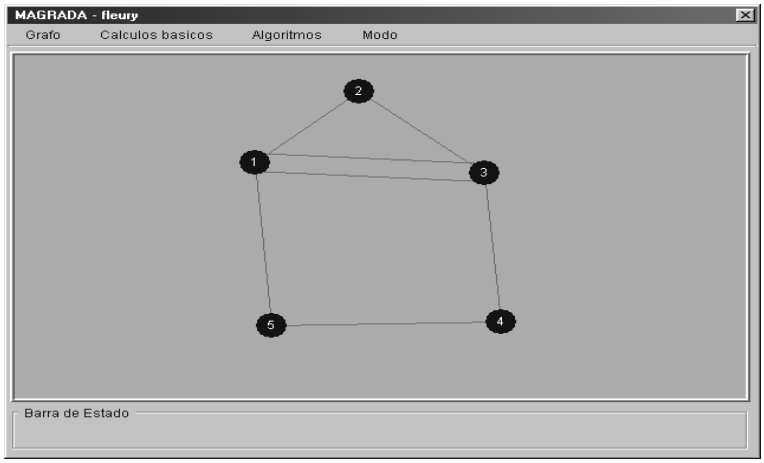
Se cuenta que los habitantes destinaban sus paseos dominicales intentando hallar un punto inicial para poder pasear por toda la ciudad, cruzando cada puente exactamente una vez y regresando a dicho punto.

Leonardo Euler, un matemático suizo natural de Basilea dio la solución a este problema. Para ello representó las cuatro zonas de la ciudad y los siete puentes como el multigrafo de la siguiente figura:

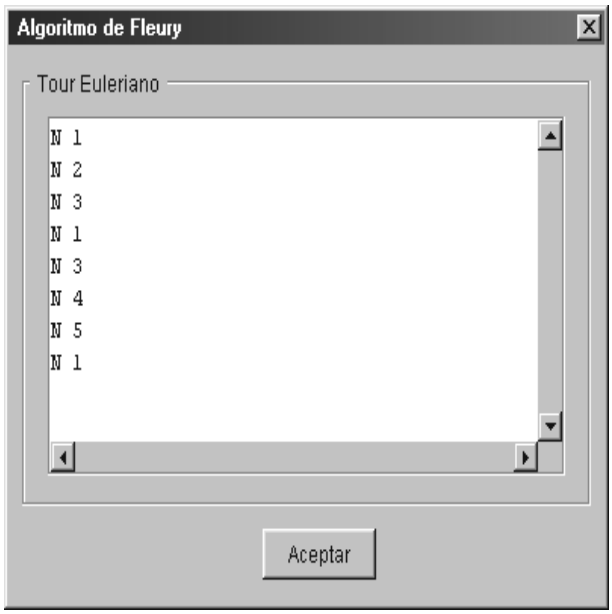


Entonces, Euler planteó el problema en los siguientes términos. ¿Es posible, partiendo de un vértice cualquiera, dibujar la figura volviendo siempre al mismo punto de partida, sin repasar ninguna línea ya trazada y sin levantar el lápiz del papel? La respuesta fue no, ya que la figura representa un grafo no dirigido que tiene vértices de grado impar, de hecho todos los vértices de este grafo tienen grado impar. Éste fue el comienzo de la teoría de grafos relativa a los caminos y tours eulerianos.

A continuación vamos a tratar la obtención de caminos y tours eulerianos en un grafo. Para ello consideremos el siguiente grafo no dirigido introducido desde modo gráfico con MaGraDa:



Este grafo es conexo y no tiene vértices de grado impar, por tanto es euleriano. Para la obtención del tour euleriano con MaGrada, tanto desde modo texto como gráfico, nos situaremos en la opción *Algoritmo de Fleury* del menú *Algoritmos*. Esta opción nos permite aplicar el algoritmo de Fleury para obtener tours o caminos eulerianos. Cuando ejecutemos esta opción MaGraDa verá si el grafo actual es euleriano, o tiene algún camino euleriano. Si es así, desde modo texto nos ofrecerá una pantalla con una sucesión de vértices que indican la forma de recorrer las aristas del grafo para formar el tour o camino. Así, en nuestro ejemplo, el tour obtenido por MaGraDa viene indicado de la siguiente forma:



Es decir, el tour encontrado, empieza en el vértice 1 y recorre todas las aristas del grafo terminando de nuevo en el vértice 1.

Desde modo gráfico, si el grafo tiene algún tour o camino euleriano no es el programa quien lo calcula y muestra al usuario, sino que es el propio usuario quien lo compone. MaGraDa generará una copia del grafo y el usuario irá pinchando de vértice en vértice hasta acabar. Si la arista o arco recorrido es legal, lo eliminará (del grafo copia, el original se lo guarda). Si no es legal, avisará al usuario y éste tendrá que elegir otra alternativa. Cuando esta copia del grafo se quede sin aristas (o arcos), se habrá confeccionado el tour o camino y se mostrará al usuario una pantalla con el orden de los vértices que él mismo eligió. Cuando esto acabe, la copia del grafo se borrará y se mostrará de nuevo el grafo original con la totalidad de aristas (o arcos).

Capítulo 5

Grafos ponderados. Caminos críticos en grafos acíclicos

5.1 Introducción

En este capítulo vamos a introducir los grafos ponderados. En la Sección 5.2 veremos cómo construir un grafo ponderado con MaGraDa. Posteriormente en la Sección 5.3 veremos cómo obtiene MaGraDa caminos más cortos y el algoritmo PERT, para grafos acíclicos, basándose en las ecuaciones de Bellman.

5.2 Creación de un grafo ponderado

Dado un grafo simple, se dice que dicho grafo es ponderado si las aristas o arcos del grafo tienen asociado un peso. Como en el caso de grafos no ponderados, los grafos ponderados pueden crearse tanto desde modo texto como desde modo gráfico. Vamos a explicar la forma de hacerlo desde ambos modos:

- **Modo texto**: La creación de un grafo ponderado desde modo texto se realiza, como en el caso de los grafos no ponderados, desde la opción *Nuevo*. Al indicar que el grafo es ponderado, nos permitirá crear este grafo de dos maneras, mediante la introducción de los arcos o aristas, o mediante su matriz de pesos. Si lo hacemos de la primera forma, además de pedirnos los vértices extremos de cada arista o arco, nos pedirá también su peso. Queremos hacer notar que esta versión de MaGraDa sólo trabaja con pesos enteros no negativos. Además, la introducción de un grafo mediante su matriz de pesos, está limitada a grafos que no tienen peso cero en ninguna arista o arco, ya que internamente utiliza ese valor para indicar que no existe arista o arco entre el correspondiente par de vértices. De hecho, si se crea el grafo de esta forma y luego se desea visualizar la *matriz de pesos* desde el menú de *Cálculos Básicos*, observaremos que los ceros los ha transformado en INFINITO. Cabe mencionar también que, debido al tipo de algoritmos con los que MaGraDa trabaja dentro del contexto de grafos ponderados, es irrelevante tener más de una arista o arco uniendo el mismo par de vértices y con pesos distintos, por lo que en el caso de grafos ponderados MaGraDa sólo permite una única arista o arco entre el

mismo par de vértices. Obviamente, en el caso dirigido, se entiende que puede haber dos arcos entre un mismo par de vértices pero con sentidos contrarios.

- **Modo gráfico**: La forma de introducir un grafo ponderado desde modo gráfico es análoga al caso de grafos no ponderados. La diferencia estriba en que hay que decir que es ponderado, cuando especifiquemos el tipo de grafo. Además como el grafo es ponderado, a cada arista o arco tendremos que asociarle un peso. Cuando se introduce cada arista o arco, el propio MaGraDa os preguntará su peso. Como ya se adelantó en el Capítulo 1 el peso se muestra en blanco sobre cuadrados de color morado. Hacemos notar que, al igual que desde modo texto, sólo se pueden utilizar pesos enteros no negativos y que MaGraDa sólo admite una única arista o arco entre un mismo par de vértices.

5.3 Ecuaciones de Bellman

En esta sección vamos a estudiar cómo utilizar las ecuaciones de Bellman para obtener caminos más cortos y más largos en un grafo acíclico.

5.3.1 Caminos más cortos

En un grafo ponderado se llama camino más corto entre dos vértices dados, al camino de peso mínimo entre dichos vértices. Supondremos que el grafo es dirigido y que los pesos asociados a los arcos son todos no negativos. En caso de que el grafo fuera no dirigido se transformaría cada arista en dos arcos (una en cada sentido) y se resolvería el problema como si el grafo fuera dirigido.

MaGraDa obtiene este camino para grafos acíclicos usando internamente las ecuaciones de Bellman. Por tanto, el método que vamos a tratar aquí no va a ser válido para grafos no dirigidos.

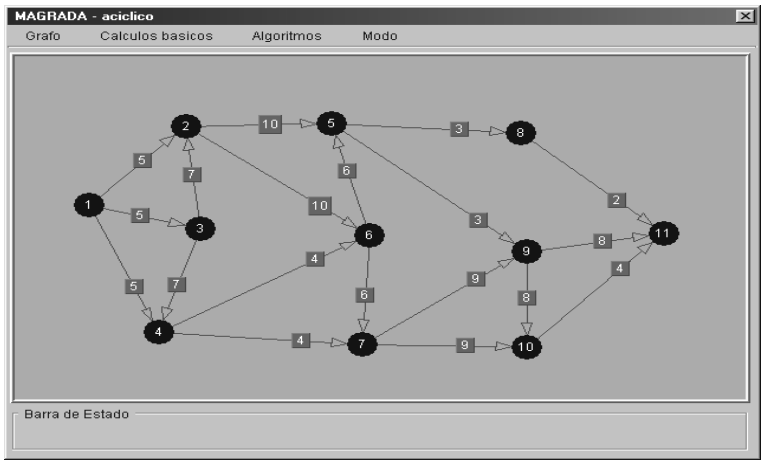
Para ello se suponen los vértices numerados del 1 al n, de forma que ω_{ij} representa el peso del arco (i, j) y que el vértice 1 es el origen del camino. Por último denotaremos por u_j el peso del camino más corto del vértice 1 al vértice j. Entonces, las ecuaciones de Bellman vienen dadas por la siguiente expresión:

$$u_1 = 0,$$

$$u_j = min_{k<j}\{u_k + \omega_{kj}\}, \quad j = 2, 3, \ldots, n.$$

Recordemos que dichas ecuaciones se pueden aplicar si la numeración considerada de los vértices del grafo cumple que si (i, j) es un arco del grafo, entonces $i < j$ y esta numeración o reetiquetación de vértices es posible encontrarla si y sólo si el grafo es acíclico. Es por ello que las ecuaciones de Bellman no son aplicables a grafos no dirigidos.

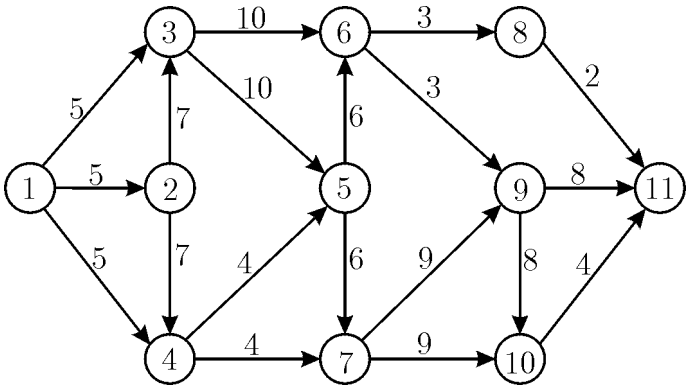
Para entender el uso de las ecuaciones de Bellman para la obtención de caminos más cortos, utilizaremos el siguiente grafo ponderado:



Este grafo, como se puede comprobar con MaGraDa es acíclico, sin embargo la numeración actual de los vértices no permite aplicar las ecuaciones de Bellman. Así por ejemplo, existe un arco del vértice 3 al 2 y sin embargo 3 no es menor que 2. Por tanto, para aplicar dichas ecuaciones debemos renumerar los vértices. Esto lo hace MaGraDa, tanto desde modo gráfico como desde modo texto en la opción *Renumerar* de *Caminos Más Cortos en Grafos Acíclicos*. Si aplicamos dicha opción, MaGraDa presenta una pantalla con la renumeración que va a usar y el vértice origen de los caminos que va a buscar, que será el 1, según su numeración.

Si observamos los resultados que aparecen en dicha pantalla, vemos que ha realizado sólo los siguientes cambios: El vértice 2 ha pasado a ser el 3 y viceversa, y el vértice 5 ha pasado a ser el 6 y viceversa. Notemos que sin embargo esto lo hace internamente. Es decir el grafo que presenta MaGraDa en pantalla sigue siendo el mismo.

Si ahora aplicamos el algoritmo al grafo con la nueva renumeración, que viene representada en la siguiente figura:



se obtiene lo siguiente:

$$
\begin{aligned}
u_1 &= 0.\\
u_2 &= min\{u_1+\omega_{12}\} = min\{0+5\} = 5.\\
u_3 &= min\{u_1+\omega_{13}\} = min\{0+5\} = 5.\\
u_4 &= min\{u_1+\omega_{14}\} = min\{0+5\} = 5.\\
u_5 &= min\{u_3+\omega_{35}, u_4+\omega_{45}\} = min\{5+10, 5+4\} = 9.\\
u_6 &= min\{u_3+\omega_{36}, u_5+\omega_{56}\} = min\{5+10, 9+6\} = 15.\\
u_7 &= min\{u_4+\omega_{47}, u_5+\omega_{57}\} = min\{5+4, 9+6\} = 9.
\end{aligned}
$$

$$
\begin{aligned}
u_8 &= min\{u_6+\omega_{68}\} = min\{15+3\} = 18.\\
u_9 &= min\{u_6+\omega_{69}, u_7+\omega_{79}\} = min\{15+3, 9+9\} = 18.\\
u_{10} &= min\{u_7+\omega_{7,10}, u_9+\omega_{9,10}\} = min\{9+9, 18+8\} = 18.\\
u_{11} &= min\{u_8+\omega_{8,11}, u_9+\omega_{9,11}, u_{10}+\omega_{10,11}\}\\
&= min\{18+2, 18+8, 18+4\} = 20.
\end{aligned}
$$

Hacemos notar que, según las ecuaciones de Bellman, por ejemplo, $u_5 = min\{u_1 + \omega_{15}, u_2 + \omega_{25}, u_3 + \omega_{35}, u_4 + \omega_{45}\}$, pero como $\omega_{15} = \omega_{25} = \infty$, hemos obviado los correspondientes valores $u_1 + \omega_{15}$ y $u_2 + \omega_{25}$, en el cálculo del mínimo realizado anteriormente. Esto es válido en todos los casos.

Una vez que tenemos los pesos de cada uno de los caminos del vértice 1 a los restantes vamos a identificar los caminos:

- $u_1 = 0$, el peso del camino más corto del vértice 1 al 1 tiene peso cero porque es de longitud cero.

- $u_2 = 5$, el peso del camino más corto del vértice 1 al 2, es 5 y dicho camino está formado por el arco $(1, 2)$.

- $u_3 = 5$, el peso del camino más corto del vértice 1 al 3, es 5 y dicho camino está formado por el arco $(1, 3)$.

- $u_4 = 5$, el peso del camino más corto del vértice 1 al 4, es 5 y dicho camino está formado por el arco $(1, 4)$.

- $u_5 = 9$, el peso del camino más corto del vértice 1 al 5, es 9 y dicho camino está formado por el camino más corto del vértice 1 al 4 más el arco $(4, 5)$. Es decir, por los arcos $(1, 4)$ y $(4, 5)$.

- $u_6 = 15$, el peso del camino más corto del vértice 1 al 6, es 15 y dicho camino está formado, por ejemplo, por el camino más corto del vértice 1 al 3 más el arco $(3, 6)$. Es decir, por los arcos $(1, 3)$ y $(3, 6)$. Notemos que en este caso podríamos haber elegido también el camino más corto del vértice 1 al 5 más el arco $(5, 6)$, porque el peso es el mismo.

- $u_7 = 9$, el peso del camino más corto del vértice 1 al 7, es 9 y dicho camino está formado por el camino más corto del vértice 1 al 4 más el arco $(4, 7)$. Es decir, por los arcos $(1, 4)$ y $(4, 7)$.

- $u_8 = 18$, el peso del camino más corto del vértice 1 al 8, es 18 y dicho camino está formado por el camino más corto del vértice 1 al 6 más el arco $(6, 8)$. Es decir, por los arcos $(1, 3)$, $(3, 6)$ y $(6, 8)$.

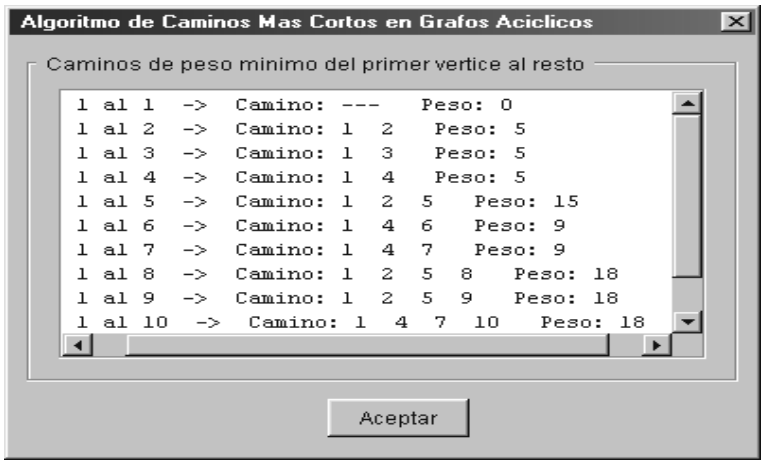
- $u_9 = 18$, el peso del camino más corto del vértice 1 al 9, es 18 y dicho camino está formado, por ejemplo, por el camino más corto del vértice 1 al 6 más el arco $(6, 9)$. Es decir, por los arcos $(1, 3)$, $(3, 6)$ y $(6, 9)$. Notemos que en este caso podríamos haber elegido también el camino más corto del vértice 1 al 7 más el arco $(7, 9)$, porque el peso es el mismo.

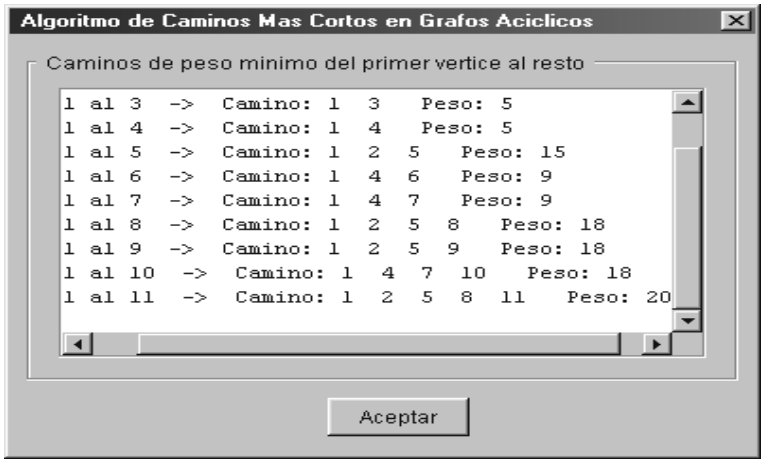
- $u_{10} = 18$, el peso del camino más corto del vértice 1 al 10, es 18 y dicho camino está formado por el camino más corto del vértice 1 al 7 más el arco $(7, 10)$. Es decir, por los arcos $(1, 4)$, $(4, 7)$ y $(7, 10)$.

- $u_{11} = 20$, el peso del camino más corto del vértice 1 al 11, es 20 y dicho camino está formado por el camino más corto del vértice 1 al 8 más el arco $(8, 11)$. Es decir, por los arcos $(1, 3)$, $(3, 6)$, $(6, 8)$ y $(8, 11)$.

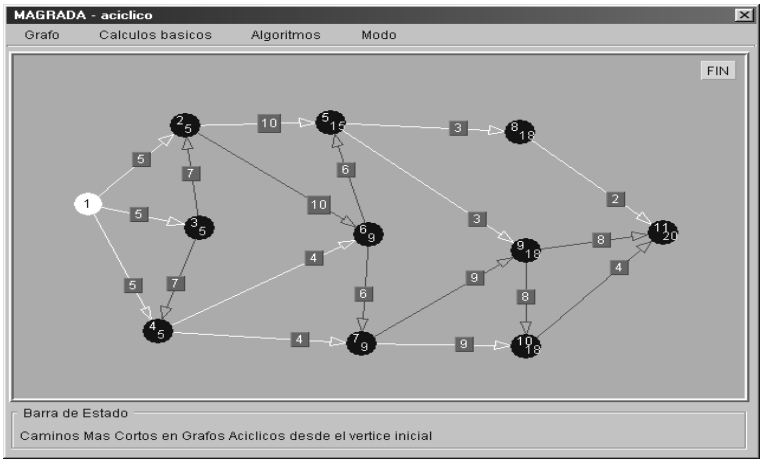
Una vez obtenidos los caminos más cortos del vértice 1 a los restantes en el grafo renumerado, hay que cambiar de nuevo la renumeración para obtener dichos caminos en el grafo inicial. Es decir, hay que cambiar de nuevo el 3 por el 2 y viceversa, y el 6 por el 5 y viceversa, en todos los resultados. De está forma se obtiene el resultado final, que presenta MaGraDa, cuando se utiliza

la opción *Aplicar* del algoritmo. Es decir, MaGraDa desde modo texto da la pantalla explicativa que mostramos a continuación en dos partes:





Desde modo gráfico, MaGraDa da la información de la siguiente forma:



Como se puede observar los caminos más cortos se indican con flechas blancas, y dentro de cada vértice aparece, además de su número, el peso del camino más corto del vértice 1 a dicho vértice.

5.3.2 Secuenciación de actividades (PERT)

Se denomina camino más largo entre dos vértices dados, al camino de peso máximo entre dichos vértices. Un ejemplo típico de obtención de un camino más largo en un grafo dirigido, es la secuenciación de actividades, también llamada red de actividades.

Así por ejemplo, un proyecto de construcción grande se suele dividir en varias actividades menores; estas actividades están relacionadas, en el sentido de que algunas no pueden comenzar hasta que otras hayan finalizado. Por ejemplo, para la construcción de una casa son necesarios cimientos, muros, carpintería, tejados, instalación eléctrica, ... Una actividad como la instalación eléctrica,

no puede empezar hasta haber completado otras actividades. La realización de este tipo de proyectos hace necesaria una planificación racional de las actividades a realizar que se denomina PERT (Program Evaluation and Review Technique). Para planificar un proyecto de esta clase podemos representar cada actividad de las que se compone el proyecto por un vértice. Si para realizar una actividad i es necesario haber realizado antes la actividad j, se incluirá un arco de j a i. A cada arco (j, i), se le asociará un peso ω_{ji}, que indica el tiempo que debe transcurrir desde el comienzo de la actividad j al comienzo de la actividad i. En este grafo se incluyen dos vértices artificiales, que representan respectivamente el inicio y el final del proyecto que se unirán, respectivamente, con los vértices con grado de entrada cero y con los vértices con grado de salida cero. Es evidente que el camino más largo (de mayor peso), en el grafo construido, representa el mínimo tiempo necesario para completar el proyecto en su totalidad. A este camino se le denomina *camino crítico* ya que las actividades que incluye determinan el tiempo total de realización del proyecto y cualquier retraso en la ejecución de una de ellas, implica un retraso en la terminación de proyecto. Hacemos notar que el grafo de un PERT tiene que ser acíclico, ya que de lo contrario el proyecto sería impracticable. Por tanto, podemos aplicar las ecuaciones de Bellman, reemplazando las operaciones de minimización por maximización y así obtener el camino crítico y su peso, de la siguiente forma:

$$u_1 = 0,$$

$$u_j = max_{k<j}\{u_k + \omega_{kj}\}, \quad j = 2, 3, \ldots, n.$$

MaGraDa resuelve este problema desde la opción PERT del menú *Algoritmos*. Tanto desde modo texto como modo gráfico ésta se divide en dos subopciones

similares a las de la sección anterior:

- **Renumerar**: Se ha de ejecutar en primer lugar para tener acceso a la segunda opción. Una vez que se sabe que el grafo es acíclico tenemos que prepararlo (renumerarlo) para poder ejecutar el algoritmo. Es decir, hay que darle una nueva renumeración a los vértices asegurando que ningún vértice sea extremo final de un arco cuyo vértice inicial tenga un número superior a él.

- **Aplicar**: Una vez ejecutada la opción anterior, el grafo está preparado para aplicar el algoritmo PERT. MaGraDa nos dará el camino más largo del vértice inicial al resto y sus pesos y en particular el camino crítico.

Queremos hacer notar que la forma de presentar los resultados es análoga a la vista en la sección anterior. Es decir, desde modo texto al ejecutar *Renumerar* nos presenta una pantalla explicativa de la renumeración utilizada para aplicar el PERT y al ejecutar *Aplicar* nos da los caminos más largos del vértice 1 al resto y su peso, en el grafo inicial (es decir con la numeración original). Desde modo gráfico nos resalta en blanco el camino crítico de dicho grafo.

Capítulo 6

Caminos más cortos y árboles generadores de mínimo peso

6.1 Introducción

En el Capítulo 5 se estudió una forma de encontrar caminos más cortos entre dos vértices, en grafos acíclicos, usando para ello, las ecuaciones de Bellman. En este capítulo y concretamente en la Sección 6.2 veremos dos algoritmos que permiten obtener estos caminos sin la restricción de que el grafo sea acíclico: el algoritmo de Dijkstra y el algoritmo de Floyd-Warshall. La Sección 6.4 está dedicada a la obtención de árboles generadores de mínimo peso, utilizando para ello los algoritmos de Kruskal y Prim. Previamente, en la Sección 6.3 introduciremos el concepto de árbol.

6.2 Caminos más cortos

6.2.1 Algoritmo de Dijkstra

El algoritmo de Dijkstra encuentra los caminos más cortos y sus pesos desde un vértice cualquiera, que denotaremos por comodidad como vértice 1, al resto. Recordemos que a partir de las ecuaciones de Bellman podíamos calcular caminos más cortos, siempre y cuando el grafo fuera acíclico. Con el algoritmo de Dijkstra, no es necesaria esta restricción. Esto permite, por tanto, calcular caminos más cortos tanto en grafos dirigidos como en no dirigidos. En el caso no dirigido podemos trabajar transformando cada arista en dos arcos, uno en cada sentido, y resolver el algoritmo de Dijkstra como si de un grafo dirigido se tratase. Esto no lo podríamos haber hecho con las ecuaciones de Bellman porque el grafo dirigido resultante sería cíclico. Recordamos también que se supone que los vértices están numerados del 1 al n y que los pesos asociados a los arcos o aristas deben ser no negativos ($w_{ij} \geq 0$).

En el algoritmo de Dijkstra se asignan varias etiquetas a los vértices del grafo. En algún momento algunos vértices podrán tener etiquetas variables y el resto etiquetas fijas. Denotaremos al conjunto de vértices con etiqueta fija por P y al conjunto de vértices con etiqueta variable por T. Con estas consideraciones dicho algoritmo puede ser definido de la siguiente forma:

ALGORITMO DE DIJKSTRA

Paso 1. Inicialización.

$$P = \{1\} \quad T = \{2, 3, \ldots, n\}$$
$$u_1 = 0$$
$$u_j = w_{1j} \quad j \in \Gamma(1)$$
$$u_j = \infty \quad j \notin \Gamma(1)$$

Paso 2. Designación de etiqueta variable como fija.

Determinar $k \in T$ / $u_k = min_{j \in T}\{u_j\}$

Hacer $T := T \sim \{k\}$ y $P := P \cup \{k\}$

Si $T = \emptyset$, STOP; u_j es el peso del camino más corto de 1 a j,

$j = 2, 3, \ldots, n$

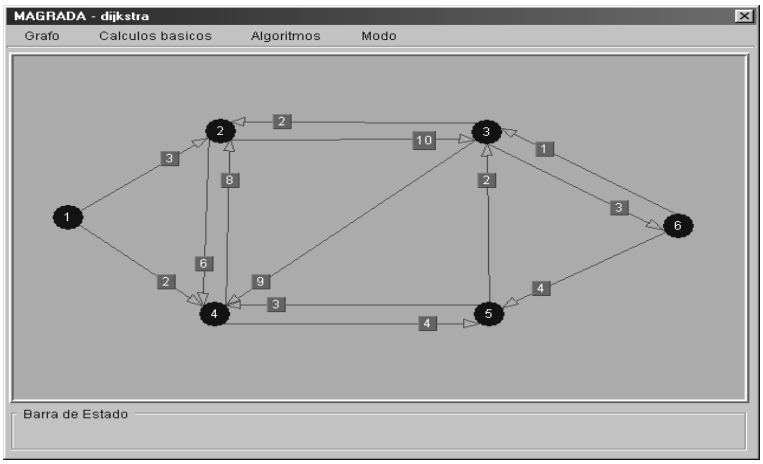
Paso 3. Actualización.

$\forall j \in \Gamma(k) \cap T \quad u_j := min\{u_j, u_k + w_{kj}\}$

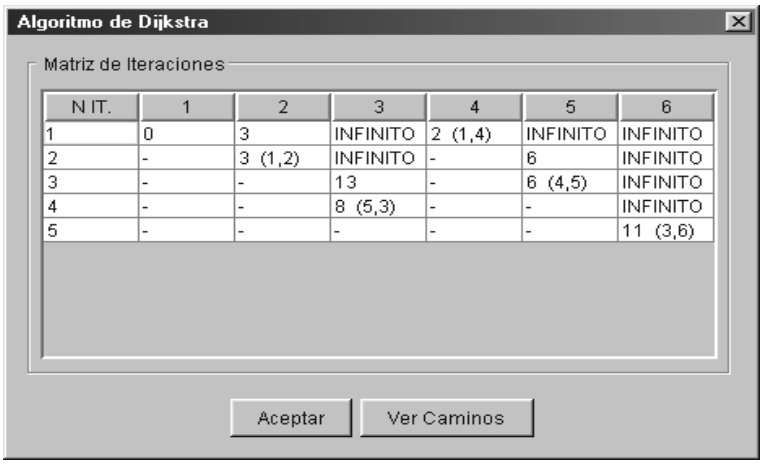
Ir al Paso 2.

Queremos hacer notar que con este algoritmo, realmente se pueden calcular los caminos más cortos entre todos los pares de vértices tomando como origen, en cada caso, cada uno de los vértices del grafo. Por otro lado, cabe mencionar que cuando un vértice j está etiquetado con etiqueta fija, la correspondiente variable u_j, ya indica el peso del camino más corto del vértice 1 al j. Así, si quisieramos aplicar este algoritmo para calcular un sólo camino más corto entre, por ejemplo, los vértices 1 y j, el algoritmo finalizaría cuando $j \in P$.

MaGraDa resuelve este algoritmo tanto desde modo texto como desde modo gráfico, en la opción *Dijkstra* del menú *Algoritmos*. Para comprender la forma en que MaGraDa presenta los resultados, vamos a considerar el siguiente grafo que ha sido previamente creado con MaGrada.



Desde modo texto, al aplicar el algoritmo de Dijkstra, mediante MaGraDa, éste nos pide que seleccionemos el vértice inicial de los caminos más cortos que vamos a calcular. Consideraremos en este caso que se ha elegido el vértice 1. MaGraDa ofrecerá, en primer lugar, la siguiente pantalla:



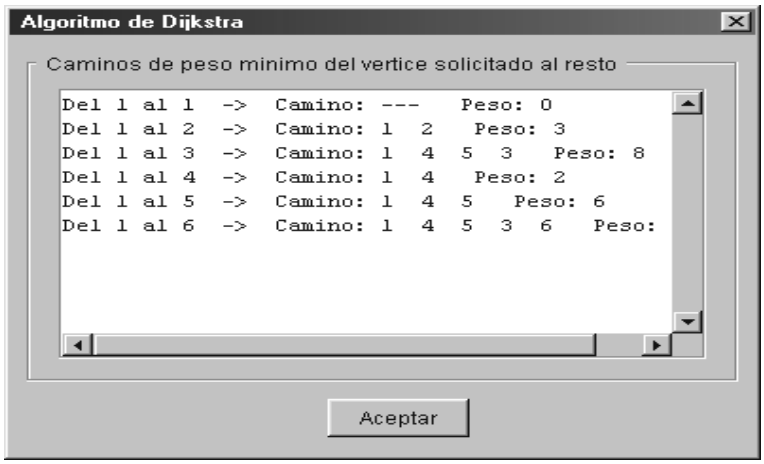
N IT.	1	2	3	4	5	6
1	0	3	INFINITO	2 (1,4)	INFINITO	INFINITO
2	-	3 (1,2)	INFINITO	-	6	INFINITO
3	-	-	13	-	6 (4,5)	INFINITO
4	-	-	8 (5,3)	-	-	INFINITO
5	-	-	-	-	-	11 (3,6)

Veamos a qué corresponde la primera iteración: En ella empezamos realizando el Paso 1, es decir, se ha tomado el 1 como etiqueta fija y el resto variables. En la fila 1, para cada columna j, aparece el peso $u_j = \omega_{1j}$, es decir, se ha tomado $P = \{1\}$, $T = \{2, 3, 4, 5, 6\}$, y los pesos obtenidos han sido $u_1 = 0$, $u_2 = 3$, $u_4 = 2$, y el resto ∞.

A continuación, en el Paso 2, el algoritmo designaba una etiqueta variable $k \in T$ como fija, que era aquella que satisfacía que $u_k = min_{j \in T}\{u_j\}$. Esta etiqueta viene determinada por MaGraDa en cada iteración por la columna en la que aparece entre paréntesis un arco. Este arco corresponde al último arco del camino del vértice 1 al k. Así en nuestro ejemplo, se ha obtenido $u_4 = min\{2, 3\} = 2$, y como el arco (1,4) que aparece en la casilla, tiene como vértice inicial el 1, el camino más corto del vértice 1 al 4 está formado únicamente por el arco $(1, 4)$ y su peso es 2. Con esta información tendríamos que $P = \{1, 4\}$ y $T = \{2, 3, 5, 6\}$. Como $T \neq \emptyset$, debemos continuar con el Paso 3 de actualización. Para ello debemos calcular para todo $j \in \Gamma(k) \cap T$, $u_j = min\{u_j, u_k + w_{kj}\}$. Como $\Gamma(k) \cap T = \Gamma(4) \cap T = \{2, 5\} \cap \{2, 3, 5, 6\} = \{2, 5\}$, sólo tenemos que actualizar u_2 y u_5. Los pesos del resto de etiquetas variables (3 y 6) se mantienen igual, por tanto en este caso, seguirán siendo ∞. Observamos en la segunda iteración que los pesos obtenidos son $u_2 = min\{u_2, u_4 + w_{42}\} = min\{3, 2 + 8\} = 3$ y $u_5 = min\{u_5, u_4 + w_{45}\} = min\{\infty, 2 + 4\} = 6$. Por tanto, como muestra MaGraDa, en la segunda iteración, si volvemos al Paso 2, la etiqueta que ahora va a pasar a fija es $k = 2$ y en este caso, el peso $u_2 = 3$, se ha obtenido con el arco $(1, 2)$. Por tanto, el camino más corto del vértice 1 al 2 es el arco $(1, 2)$. Siguiendo el proceso, ahora tendríamos $P = \{1, 4, 2\}$ y $T = \{3, 5, 6\}$. La actualización de los pesos aparece ahora en la iteración 3. De ella se deduce

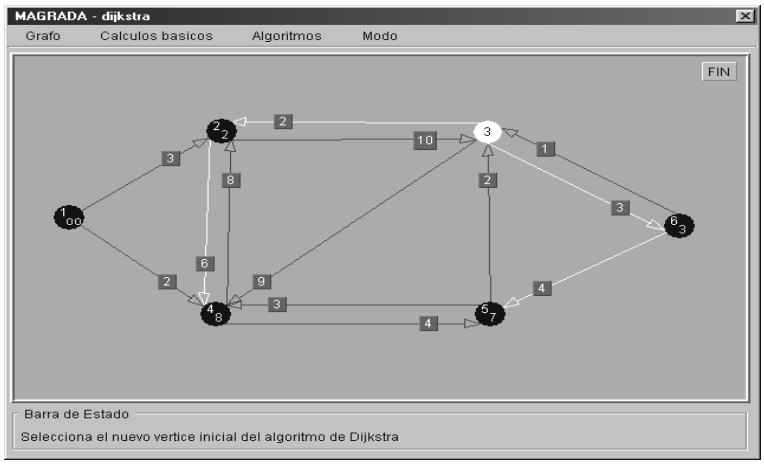
que la nueva etiqueta que va a pasar de variable a fija es $k = 5$ y $u_5 = 6$. Para obtener el camino más corto del vértice 1 al 5 debemos fijarnos en el par que aparece al lado del número. En este caso dicho arco es el $(4, 5)$. Como el vértice inicial del arco no es 1, para obtener el camino más corto debemos ir hacia atrás y encontrar el camino más corto entre 1 y 4, que ya vimos que era el arco $(1, 4)$. Por tanto, resumiendo, el camino más corto del vértice 1 al 4 está formado por los arcos $(1, 4)$ y $(4, 5)$. De forma análoga seguiríamos el algoritmo obteniendo en la iteración 4 la actualización de los pesos que da lugar a que la nueva etiqueta que se transformará en fija es $k = 3$ y que el camino más corto del vértice 1 al 3 está formado por los arcos $(1, 4), (4, 5)$ y $(5, 3)$ y su peso es 8. Por último, en la iteración 5, una vez actualizados los pesos se obtiene que el camino más corto del vértice 1 al 6 tiene peso 11 y está formado por los arcos $(1, 4), (4, 5), (5, 3)$ y $(3, 6)$. Como ya no quedan etiquetas variables se ha finalizado el algoritmo, obteniendo todos los caminos más cortos entre el vértice 1 y el resto.

Estos caminos y sus pesos vienen dados por MaGraDa, pulsando en la pantalla anterior *Ver Caminos*. Concretamente MaGraDa presentará la siguiente pantalla:



Queremos hacer notar que puede ocurrir que no siempre haya caminos entre cada par de vértices.

Desde modo gráfico, MaGraDa actúa de forma análoga. Sin embargo, para obtener los caminos más cortos de un vértice al resto habrá que pinchar dicho vértice. Así, MaGraDa nos presentará una pantalla con las iteraciones igual que la del modo texto. Si pulsamos *Ver Caminos*, éstos nos los presentará de forma gráfica en el grafo en color blanco, como muestra la siguiente pantalla, en la que aparecen los caminos más cortos del vértice 3 a los restantes junto con su peso. Este peso aparece dentro del vértice final de cada camino.



6.2.2 Algoritmo de Floyd-Warshall

Aunque se pueden calcular los caminos más cortos entre todos los pares de vértices aplicando Dijkstra tomando como origen, en cada caso, cada uno de los vértices del grafo, el método de Floyd-Warshall es más eficiente en general para este propósito.

Llamaremos u_{ij} al peso del camino más corto de i a j. Utilizaremos las variables $u_{ij}^{(m)}$, que indican el peso del camino más corto del vértice i al j con la restricción de que dicho camino no contenga los vértices $m, m+1, \ldots, n$ (exceptuando a los extremos i y j en su caso). Es decir, el camino que representa la variable $u_{ij}^{(m)}$ no debe contener **como internos** ninguno de los vértices $m, m+1, \ldots, n$.

Estas variables pueden calcularse recursivamente utilizando las ecuaciones:

$$\begin{aligned} u_{ij}^{(1)} &= w_{ij}, \quad \forall i,j \\ u_{ij}^{(m+1)} &= min\left\{u_{ij}^{(m)}, u_{im}^{(m)} + u_{mj}^{(m)}\right\}, \quad \forall i,j, \\ & \quad m = 1,2,\ldots n. \end{aligned}$$

Es posible ver que

$$u_{ij} = u_{ij}^{(n+1)},$$

con lo que tendremos los pesos de los caminos más cortos entre todos los pares de vértices.

Para facilitar la construcción de los caminos más cortos una vez calculados sus pesos, se puede utilizar otra matriz

$$\Theta^{(m)} = [\theta_{ij}^{(m)}]_{1 \le i,j \le n},$$

donde $\theta_{ij}^{(m)}$ representa el vértice anterior al j en el camino más corto de i a j en la iteración m.

Inicialmente $\theta_{ij}^{(1)} = i$, si $u_{ij}^{(1)} < +\infty$, y

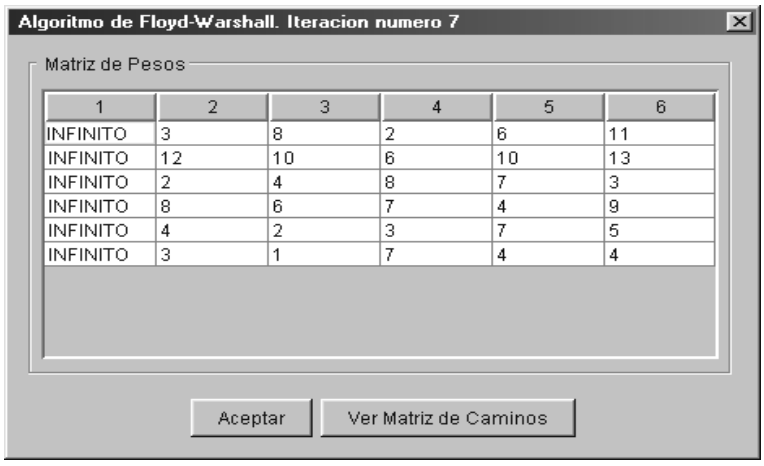
$$\theta_{ij}^{(m+1)} = \begin{cases} \theta_{ij}^{(m)} & \text{si } u_{ij}^{(m+1)} = u_{ij}^{(m)} \\ \theta_{mj}^{(m)} & \text{si } u_{ij}^{(m+1)} < u_{ij}^{(m)} \end{cases}$$

Tanto desde modo texto como gráfico, MaGraDa resuelve este algoritmo desde la opción *Floyd-Warshall* del menú *Algoritmos*. Dentro de dicha opción tenemos dos subopciones:

- **Por pasos:** Este método muestra cómo se van calculando las matrices $[u_{ij}^{(m)}]$ y $\Theta^{(m)}$, para cada iteración m.

- **Entre dos vértices:** Nos da directamente el camino más corto entre el par de vértices seleccionado. Desde modo texto nos da una pantalla donde debemos seleccionar dichos vértices y desde modo gráfico tendremos que pinchar los vértices que van a ser extremos del camino.

Vamos a ver cómo interpretar los resultados obtenidos con MaGraDa. Supongamos que deseamos calcular los caminos más cortos entre cada par de vértices para el grafo de este capítulo. Una vez realizadas todas las iteraciones del algoritmo obtenemos las siguientes matrices de pesos y caminos, respectivamente:



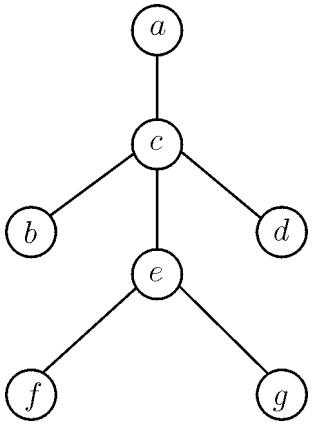
1	2	3	4	5	6
-	1	5	1	4	3
-	3	2	2	4	3
-	3	6	2	6	3
-	4	5	5	4	3
-	3	5	5	4	3
-	3	6	5	6	3

En la primera matriz aparece el peso de los caminos más cortos entre cada par de vértices. Así, por ejemplo, para calcular el peso del camino más corto del vértice 4 al vértice 6, nos debemos situar en la cuarta fila y sexta columna de la primera matriz, obteniendo que el peso es 9. Para identificar el camino nos situamos en la misma posición de la siguiente matriz (matriz de caminos), como aparece un 3, eso significa que el último arco del camino es el $(3, 6)$. Ahora, siguiendo en la cuarta fila nos situamos en la columna 3, obtenemos un 5, por tanto ya tenemos los arcos $(5, 3)$ y $(3, 6)$. De nuevo en la cuarta fila, nos situamos en la columna 5 y obtenemos un 4. Lo que quiere decir que hemos finalizado el proceso y el camino más corto está formado por los arcos $(4, 5)$, $(5, 3)$ y $(3, 6)$.

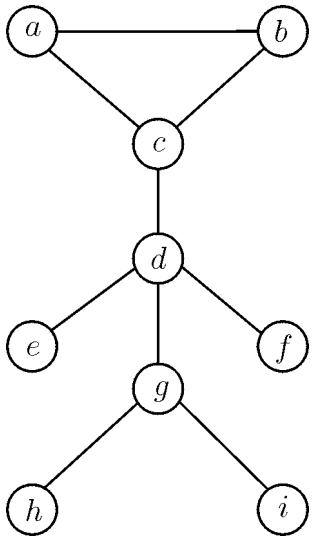
6.3 Estructura de árbol

Recordemos que un árbol es un grafo no dirigido conexo y acíclico. Por tanto podemos ayudarnos de MaGraDa para ver si un grafo es árbol. Para ello utilizaremos las opciones *cíclico* y *conexo* del menú *Grafo* de *Cálculos Básicos*. Veamos un ejemplo. Consideremos los grafos no dirigidos:

(i) Grafo a:



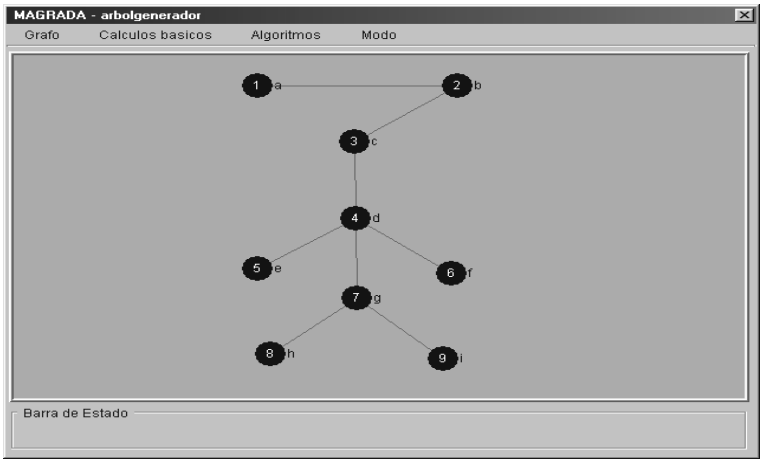
(ii) Grafo b:



Como se puede comprobar con MaGraDa ambos grafos son conexos, sin embargo el Grafo a es acíclico y el Grafo b no lo es, por tanto sólo es árbol el Grafo a.

Por otro lado, dado un grafo no dirigido, se llama árbol generador de dicho grafo a un subgrafo generador del grafo que además es árbol. Con MaGrada se puede obtener un árbol generador de un grafo desde la opción *Un Árbol Gene-*

rador Asociado del menú *Cálculos Básicos*. Así por ejemplo, el árbol generador que obtiene MaGraDa para el Grafo b viene dibujado en la siguiente pantalla:



6.4 Árboles generadores de mínino peso

Sea G un grafo ponderado y no dirigido. Diremos que T es un árbol generador de mínimo peso, si T es un árbol generador tal que la suma de los pesos asociados a sus aristas es mínima.

En esta sección vamos a ver cómo utilizar los algoritmos de Kruskal y Prim, para obtener árboles generadores de mínimo peso. Este tipo de algoritmos nos permiten resolver problemas reales cuya solución sea la obtención de dicho árbol, como por ejemplo la construcción de un sistema de carreteras de coste mínimo que enlace una serie de ciudades.

6.4.1 Algoritmo de Kruskal

Dado $G = (V, A)$, un grafo no dirigido con n vértices y con pesos w_i asociados a cada arista $e_i \in A,\ i = 1, 2, \ldots, m$, el algoritmo de Kruskal, para la obtención de árboles generadores de mínimo peso, consiste en lo siguiente:

Paso 1. $T = \emptyset$.

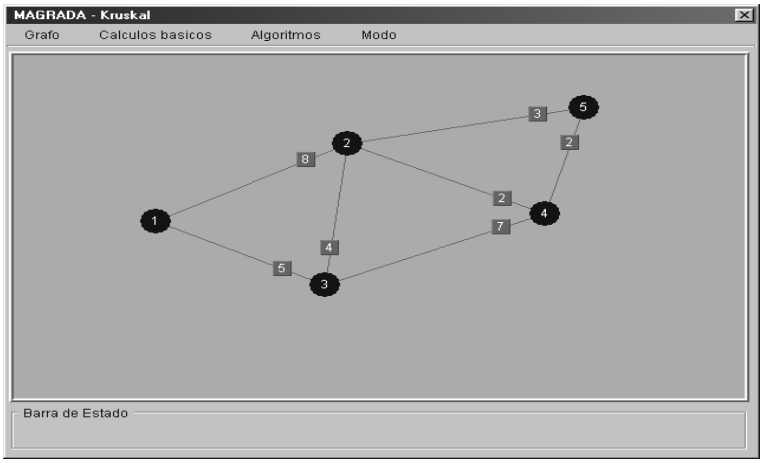
Paso 2. Ordenar en orden creciente las aristas de G atendiendo a sus pesos, es decir,

$$e_1, e_2, \ldots, e_m \ / \ w_1 \leq w_2 \leq \cdots \leq w_m.$$

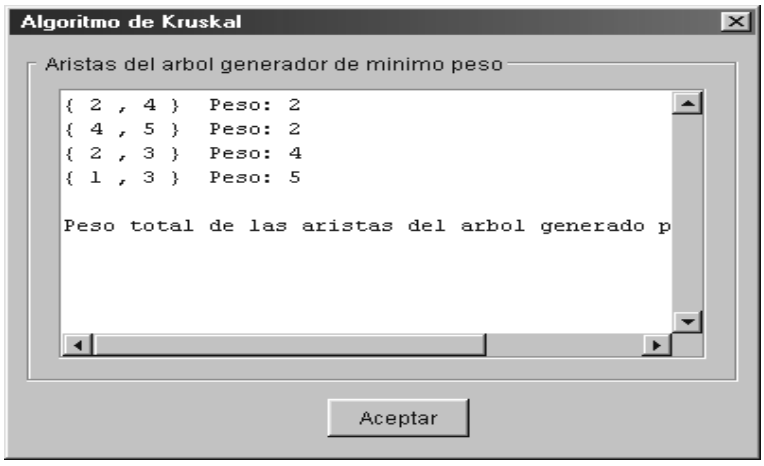
Paso 3. Añadir aristas en T de forma ordenada siempre que no se formen ciclos hasta tener en T, $n - 1$ aristas.

MaGraDa resuelve este algoritmo desde la opción *Kruskal* del menú *Algoritmos*. La solución que nos da es clara. Nos dirá qué aristas del grafo son las que pertenecen al árbol generador de mínimo peso. Desde modo texto lo hace con dos pantallas. En una nos ofrece una relación de aristas del grafo con sus pesos ordenados de menor a mayor y nos dice de cada una de ellas si la puede coger para el árbol o no, porque forma ciclo. En la otra pantalla nos dice claramente qué aristas va a coger y cuál es el peso del árbol obtenido. Desde modo gráfico la primera pantalla es la misma pero la segunda presenta el árbol de forma gráfica resaltando las aristas en color blanco.

Así por ejemplo, para el siguiente grafo creado con MaGraDa,



desde modo texto obtenemos las siguientes pantallas:



En la primera pantalla vemos que hay 3 aristas que formarían ciclo en el grafo, por tanto no las colocará en el árbol. En la segunda pantalla vemos las aristas definitivas que van a formar el árbol generador de mínimo peso.

6.4.2 Algoritmo de Prim

Sea G un grafo no dirigido ponderado con n vértices, entonces el algoritmo de Prim para la obtención de un árbol generador de mínimo peso consiste en realizar los siguientes pasos:

Paso 1. $T = \emptyset$, $U = \{v^\star\}$, $\quad v^\star \in V(G)$,

$$L(u) = w_{uv^\star} \ (\infty \text{ si } \nexists \text{ arista}) \ \forall u \in V(G).$$

Paso 2. Encontrar $u^\star \in V(G)$, tal que

$$L(u^\star) = min_{u \notin U}\{L(u)\}.$$

Paso 3. Añadir $u^\star$ a U, es decir, $U := U \cup \{u^\star\}$.

Añadir la arista e incidente con $u^\star$ con peso $L(u^\star)$ a T, es decir,

$T := T \cup \{e\}$.

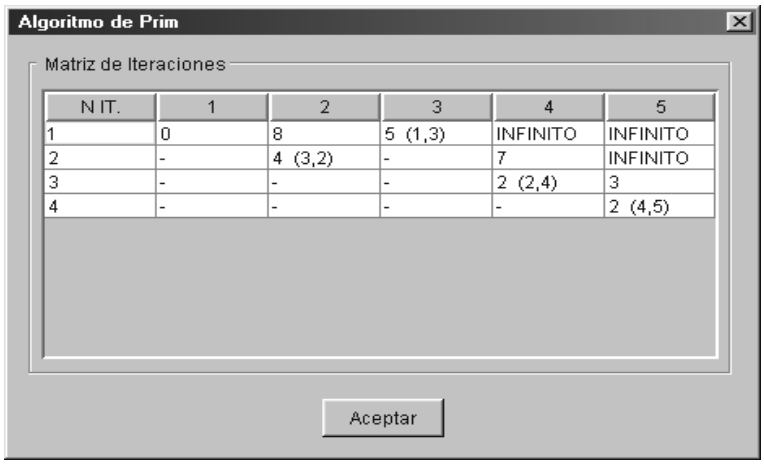
Paso 4. Si $card(U) = n$, STOP.

Si $card(U) < n$, hacer

$$L(u) := min\{L(u), w_{u^\star u}\}, \quad \forall u \notin U,$$

e ir al Paso 2.

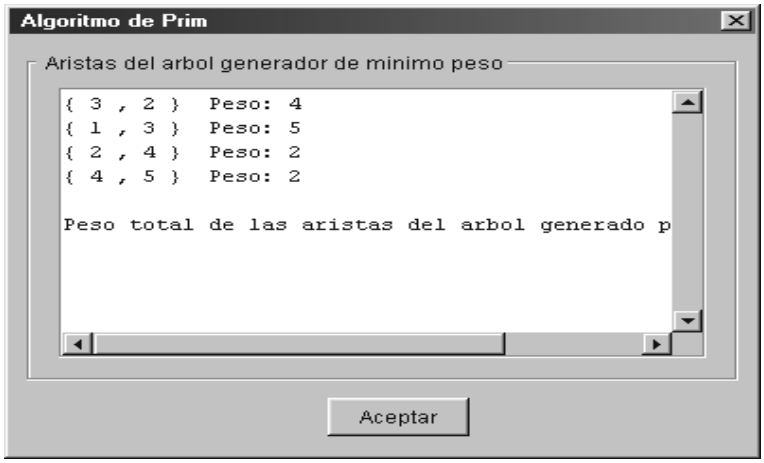
Para resolver este algoritmo mediante MaGraDa debemos seleccionar la opción *Prim* del menú *Algoritmos*. Tanto desde modo texto como desde modo gráfico nos permite elegir el vértice a partir del cual vamos a construir el árbol generador de mínimo peso, es decir, el vértice $v^\star$. La solución nos la presenta en dos pantallas. La primera, la *matriz de iteraciones*, resume los pasos del algoritmo mediante los que se va a obtener el árbol. Así, para nuestro ejemplo, partiendo del vértice 1 obtendríamos la siguiente pantalla:



Notemos que esta pantalla tiene mucha similitud con la que se obtenía en el algoritmo de Dijkstra para la obtención de caminos más cortos. No en vano, los algoritmos de Dijkstra y Prim son muy similares. En cada iteración, cuando se alcanza el mínimo, se pone la arista que lo ha conseguido. Vamos a explicarlo de forma algo más detallada. Como hemos partido del vértice 1, en la primera iteración se obtiene $L(2) = \omega_{12} = 8$, $L(3) = \omega_{13} = 5$, y $L(4) = L(5) = \infty$. Claramente el mínimo se obtiene para $u^\star = 3$. Por tanto, la primera arista que debemos introducir en el árbol es la arista $\{1, 3\}$. Tenemos entonces, $T = \{\{1, 3\}\}$ y $U = \{1, 3\}$. Si ahora pasamos a la segunda iteración, obtenemos $L(2) = min\{L(2), \omega_{32}\} = min\{8, 4\} = 4$, $L(4) = min\{L(4), \omega_{34}\} = min\{\infty, 7\} = 7$ y $L(5) = min\{L(5), \omega_{35}\} = min\{\infty, \infty\} = \infty$. Por tanto, en esta iteración, el mínimo se obtiene para $u^\star = 2$ y la arista que debemos introducir en el árbol es la arista $\{3, 2\}$. Tenemos entonces, $T = \{\{1, 3\}, \{3, 2\}\}$ y $U = \{1, 3, 2\}$. Pasamos ahora a la tercera iteración y de forma análoga obtenemos $L(4) = min\{L(4), \omega_{24}\} = min\{7, 2\} = 2$ y $L(5) = min\{L(5), \omega_{25}\} = min\{\infty, 3\} = 3$. Por tanto, en esta iteración, el mínimo se obtiene para $u^\star = 4$, y la arista que debemos introducir en el árbol es la arista $\{2, 4\}$. Tenemos entonces, $T = \{\{1, 3\}, \{3, 2\}, \{2, 4\}\}$ y $U = \{1, 3, 2, 4\}$. Por último, en la iteración cuarta se obtiene $L(5) = min\{L(5), \omega_{45}\} = min\{3, 2\} = 2$. Por tanto, en esta iteración, el mínimo se obtiene en el único vértice que nos queda $u^\star = 5$, y la arista que debemos introducir en el árbol es la arista $\{4, 5\}$. Tenemos entonces, $T = \{\{1, 3\}, \{3, 2\}, \{2, 4\}, \{4, 5\}\}$ y $U = \{1, 3, 2, 4, 5\}$. Por tanto, se ha finalizado el proceso y en T aparecen las aristas que forman el árbol.

Estas aristas, que forman el árbol generador de mínimo peso se mostrarán luego de forma más clara desde modo texto en otra pantalla de la siguiente

forma:



Si estamos en modo gráfico presentará dicho árbol de forma gráfica.

Bibliografía

[1] M. Abellanas y D. Lodares. *Análisis de algoritmos y teoría de grafos.* Ra-ma, 1990.

[2] M. Abellanas y D. Lodares. *Matemática discreta.* Ra-ma, 1990.

[3] N. L. Biggs. *Matemática discreta.* Vicens Vives, 1994.

[4] J. A. Bondy y U. S. Murty. *Graph theory with applications.* Mc Millan Press, 1976.

[5] N. Christofides. *Graph theory. An algorithmic approach.* Academic Press, 1975.

[6] M. A. Caballero, V. Migallón y J. Penadés. *Prácticas de Matemática Discreta con MaGraDa.* Disponible en http://www.dccia.ua.es/dccia/inf/asignaturas/MD/pracMaGraDa.html

[7] P. M. Cuenca. *Programación en Java.* Anaya, 1998.

[8] P. F. Dierker y W. L. Voxman. *Discrete mathematics.* HBJ, 1986.

[9] S. Even. *Graph algorithms.* Computer Science Press, 1979.

[10] A. Froufe. *JAVA 2: Manual de usuario y tutorial.* Ra-Ma, Segunda edición, 2000. Disponible en http://usuarios.tripod.es/froufe/.

[11] R. Gould. *Graph theory.* The Benjamin/Cummings Publishing Company, INC., 1988.

[12] R. P. Grimaldi. *Matemáticas discreta y combinatoria.* Addison-Wesley Iberoamericana, 1989.

[13] J. Gross y J. Yellen. *Graph theory and its applications.* CRC Press, 1999.

[14] R. Johnsonbaugh. *Matemáticas discretas.* Grupo Editorial Iberoamericana, 1988.

[15] B. Kolman y R. Busby. *Estructuras de matemática discreta para la computación.* Prentice Hall Hispanoamericana, 1986.

[16] S. Lipschutz. *Matemática discreta.* McGraw-Hill, 1990.

[17] R. Merris. *Graph theory.* Wiley-Interscience, 2000.

[18] Sun Microsystems. The Java Tutorial. Tutorial on line, disponible en http://java.sun.com/docs/books/tutorial/.

[19] M. Morgan. *Descubre Java 1.2.* Prentice Hall, 1998.

[20] J. L. Mott, A. Kandel, y T. P. Baker. *Discrete mathematics for computer scientists and mathematicians.* Prentice-Hall, 1986.

[21] J. Wilson Robin. *Introduction to graph theory.* Longman, 1985.

[22] K. A. Ross y C. R. Wright. *Matemáticas discretas.* Prentice Hall, 1990.

[23] M. N. S. Swamy y K. Thulasiraman. *Graphs, networks and algorithms.* John Wiley, 1981.

[24] J. P. Tremblay y R. Manohar. *Discrete mathematical structures with applications to computer Science.* MacGraw-Hill, 1975.