PRÁCTICA 8: 1 sesión

(del 30 de mayo al 3 de junio de 2005)

BÚSQUEDA DE PUNTOS DE RUPTURA EN UN GRAFO CONEXO

A long time ago, in a galaxy far, far away... (Star Wars, George Lucas)

Tras las últimas batallas, la Confederación, el grupo separatista encabezado por el conde Dooku, ahora Lord Darth Tiranus, amenaza con diezmar definitivamente las tropas de la República.

En una operación secreta, un espía ha conseguido robar los planos de la red de comunicaciones de la Confederación con los datos de los nodos de transmisión y los enlaces entre ellos.

El objetivo del ejercito republicano es destruir esta red de comunicaciones. Con ello, el caos invadiría a la tropas robot de la Confederación y los ejércitos clon, liderados por los generales Jedi, conseguirían diezmar a las tropas confederadas.

El objetivo es buscar puntos críticos en la red de comunicaciones para poder, con un mínimo de riesgos y una máxima efectividad, destruir la red de comunicaciones confederada.

Los datos de los nodos planetarios están guardados en el fichero '**`Nodos.dat`**' y los enlaces existentes entre los nodos (las relaciones) se encuentran guardadas en el fichero '**`Relacion.dat`**'

0.- OBJETIVOS

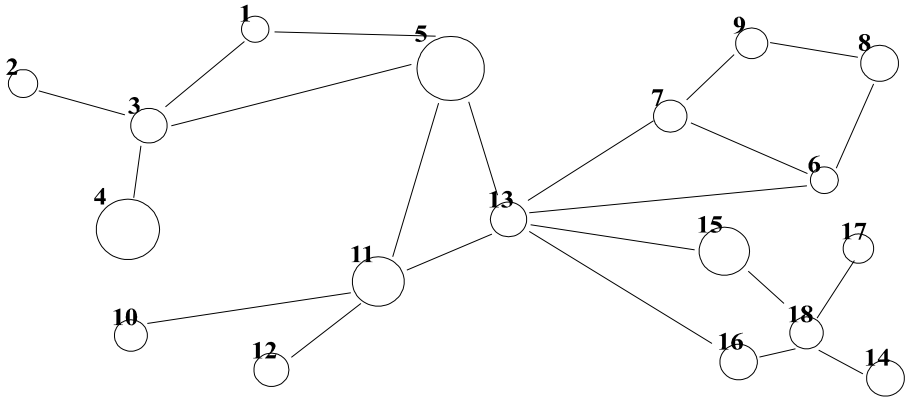
Por un lado ver la capacidad de los grafos para guardar representaciones de información compleja. Y por otro, la utilización del recorrido BFS sobre grafos para determinar los puntos de ruptura de la red de comunicaciones representada por el grafo.

1.- REPRESENTACIÓN DE UNA RED UTILIZANDO GRAFOS

Una red de comunicaciones planetaria se puede representar mediante la utilización de un grafo no dirigido, que será conexo, en el que cada nodo represente a una estación de comunicaciones planetaria, y las aristas los enlaces entre planetas.

Un punto crítico en una red de comunicaciones (punto de ruptura) es aquél que separa el grafo en distintos subgrafos conexos independientes.

Por ejemplo, dado el siguiente grafo:



Es conexo, porque desde cualquier nodo se puede acceder a cualquier otro nodo de la red.

El nodo 13 es un punto de ruptura porque sin él el grafo deja de ser conexo (por ejemplo no se puede acceder al nodo 5 desde el nodo 14).

2.- REALIZACIÓN DE LA PRÁCTICA

En esta práctica se trata de leer la información contenida en los ficheros '**`Nodos.dat`**' y '**`Relacion.dat`**' y guardarla en memoria en forma de matriz de adyacencia.

Para ello, realizaremos un método en la clase `Grafo` al que pasaremos un nombre de un fichero y a partir de la información leída del fichero se rellenará la información del grafo.

Una vez leída la información y puesta en la matriz de adyacencia, utilizaremos el recorrido BFS del grafo para determinar los puntos de ruptura de la red.

El algoritmo que utilizaremos para determinar los puntos de ruptura es el siguiente:

```
Para todos los nodos del grafo hacer
     Marcar como visitado el nodo
     Aplicar BFS desde el siguiente nodo
     Comprobar si todos los nodos del grafo han sido recorridos
     Si no han sido recorrido todos, el nodo era punto de ruptura (Guardarlo)
     Marcar de nuevo todos los nodos como no visitados
fin
```

Este algoritmo lo implementaremos, al igual que la lectura del grafo, como un método. El resultado del método será una lista con los nombres de los planetas que son punto de ruptura y su localización.

El programa finalizará mostrando la información de los planetas que son punto de ruptura de la red de comunicaciones por pantalla y generando un fichero binario que guarde los registros de estos planetas. El fichero se llamará '**ruptura.dat**'.

Formato de ficheros

Nodos.dat

En el fichero **nodos.dat** se guarda la información de cada una de las estaciones planetarias (nombre del planeta en una linea y en la siguiente el código y coordenadas espaciales del planeta). Por ejemplo:

```
Coruscant
0 20.5 44.8 32.4
```

Nombre del planeta: Coruscant / Código: 0 / Coordenadas: (20.5, 44.8, 32.4)

Relacion.dat

En el fichero **relacion.dat** se encuentran las relaciones entre los distintos nodos en forma de 'matriz', de manera que en cada fila, 'i' están las relaciones del nodo 'i' con los nodos 'j' de cada columna representados por un uno si existe la relación y un cero si no la hay. Por ejemplo, si tuviésemos 4 nodos, el siguiente fichero de relaciones, representaría las relaciones existentes en el grafo de la figura 2.

```
0 1 1 0
1 0 1 0
1 1 0 1
0 0 1 0
```

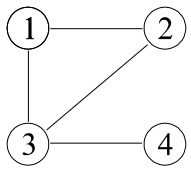


Figura 2

Ruptura.dat

En el fichero binario **ruptura.dat** se guarda la información de cada una de las estaciones planetarias (nombre del planeta, código y coordenadas espaciales) que son punto de ruptura de la red de comunicaciones.

En este fichero binario guardaremos registros del tipo **Planeta**:

```
typedef float Coordenadas[3];

struct Planeta
{
    string nombre;
    int codigo;
    Coordenadas coord;
};
```

3. DETALLES DE IMPLEMENTACIÓN

Para la realización de la práctica necesitaremos la utilización de tres clases: La clase `Grafo`, la clase `Cola` y la clase `Lista`.

Clase `Grafo`

La interfaz de la clase `Grafo` ha de ser la siguiente, teniendo en cuenta que el máximo de planetas es 150, que la información referente a los arcos es sólo si existe o no una conexión entre dos nodos y que entre los métodos públicos es conveniente que estén, al menos, los siguientes métodos:

```
const int MAX_NODOS = 150;

typedef Planeta Valor_Nodo;

class Grafo
{
    public:
        // Constructor por defecto
        Grafo (void);
        // Metodo que a partir del nombre de dos ficheros (fichero de nodos y
        // fichero de relaciones) rellena la información del vector de nodos y de
        // la matriz de relaciones
        void LeerGrafo (string, string);
        // Metodo que determina los puntos de ruptura y los guarda en la lista
        // pasada como parametro
        Lista DeterminarPuntosDeRuptura ();

    private:
        struct Nodo
        {
            Valor_Nodo info;    /* Información asociada a cada nodo */
            //otros campos que sean necesarios
        };

        struct Arco
        {
            bool existe; /* hay arco o no hay arco */
        };

        int num_nodos;

        typedef Nodo Vect_Nodos[MAX_NODOS];
        typedef Arco Vect_Arcos[MAX_NODOS][MAX_NODOS];

        Vect_Nodos Nodos;
        Vect_Arcos Arcos;
};
```

Hay que señalar, ademas, que es posible que se necesite incluir alguna información adicional en la parte privada de la estructura del grafo para marcar los elementos que ya han sido recorridos en el recorrido BFS.

Clase `Lista`

Utilizaremos la clase lista proporcionada por el profesor en la práctica 5. Para poder utilizarla sólo tendremos que asegurarnos que el tipo Valor de la lista es '**`planeta`**':

```
typedef Planeta Valor;
```

Clase `Cola`

Para realizar el recorrido es necesaria la utilización de la clase **`Cola`**. Podemos utilizar la cola que ya realizamos en la práctica 4 o la publicada en la solución de la práctica 4.

4. DETALLES DE LA REALIZACIÓN DEL FORMULARIO

En el formulario referente a está práctica irán reflejados los siguientes detalles del programa:

- **Las dos clases que se van a utilizar en la práctica:**

  Clase Grafo / Clase Lista / Clase Cola (podemos reutilizar la parte del formulario realizado en las prácticas anteriores referentes a la clase Lista y clase Cola). Es importante que en la clase Grafo deis los detalles de la información y los métodos que pensais añadir.

- **Las funciones en que vamos a dividir el programa principal y que vamos a utilizar a lo largo de nuestro programa, especificando claramente las entradas y salidas de las mismas, así como la tarea que desarrolla explicada muy brevemente.**

  En esta práctica la carga principal de las tareas a desarrollar serán realizadas mediante la utilización de métodos propios de las clases 'Grafo' y 'Lista', a pesar de ello, sería interesante y conveniente la utilización de tantas funciones como partes esenciales tiene nuestro programa: Lectura de información, búsqueda de puntos de ruptura y recorrido de la lista para mostrar por pantalla los planetas punto de ruptura y guardarlos en un fichero binario.

- **Los diagramas de flujo tanto del programa principal como de aquellas funciones que consideremos más interesantes o complejas.**

  En esta práctica se recomienda la realización del diagrama de flujo de la función principal, así como del método que busca los puntos de ruptura.

5. ENTREGA DE PROGRAMAS

Cinco días después de realizada la sesión de prácticas se entregará al profesor la siguiente información:

1. Archivo con el programa (**`pr08##.cpp`**)
2. Archivos con la clase `Grafo` (**`Grafo##.h`**, **`Grafo##.cpp`**)
3. Archivos con la clase `Lista` (**`Lista##.h`**, **`Lista##.cpp`**)
4. Archivos con la clase `Cola` (**`Cola##.h`**, **`Cola##.cpp`**)

dónde **`##`** es el número asignado a la pareja (01, 02,...) en cada grupo de prácticas.

Al comenzar la siguiente sesión de prácticas se entregará al profesor el formulario con la documentación definitiva de la práctica reflejando todo el trabajo realizado realmente en la práctica.

Nota Muy Importante

Antes de poder empezar a realizar cualquiera de las prácticas es necesario presentar las hojas de especificación de programas (formulario de documentación de programas) con las tareas que se van a realizar en la práctica, explicando brevemente como se van a solucionarse los problemas que se plantean.

ENTREGA DE PROGRAMAS:

Los días 4, 5, 6, 7 y 8 de junio de 2005, correspondientes con las fechas de realización de prácticas 30 y 31 de mayo y 1, 2 y 3 de junio respectivamente.

ENTREGA DE FORMULARIOS DEFINITIVOS DE DOCUMENTACIÓN:

Los días 6, 7, 8, 9 y 10 de junio de 2005, correspondientes con las fechas de realización de prácticas 30 y 31 de mayo y 1, 2 y 3 de junio respectivamente.