E.T.S.I.I.T - Grado en Ingeniería de Tecnologías de Telecomunicación

Dimensionado y Planificación de Redes
Curso 2019/2020

Práctica 1 - Representación y algoritmos sobre grafos.

Los alumnos deberán realizar el proyecto asignado y presentar el funcionamiento del programa antes del 16 de diciembre. Se deberá entregar el código correspondiente y un pequeño informe (máximo de 10 páginas) comentando la implementación llevada a cabo, resultados, etc. La entrega se realizará a través del Aula Virtual de la UC (`moodle`). Se recomienda que se utilicen todos los recursos a disposición de los estudiantes, pero que se adapte en todo caso el código fuente al estilo propio de cada grupo.

La práctica se realizará en una máquina virtual sobre `VMWare`. El SO *host* será *Windows*, y los alumnos utilizarán la cuenta `alumnos`, con la contraseña `telematica`. Como no se puede garantizar el almacenamiento permanente de los ficheros entre sesiones, se recomienda que los alumnos se encarguen de, tras cada sesión, guardar sus desarrollos.

En la elaboración de la memoria es recomendable que se representen (en ciertos casos especialmente) los grafos correspondientes, para lo que se puede utilizar el programa `graphviz`[1].

Proyecto 1. Algoritmo de Misra & Gries

Desarrollar un programa en el lenguaje de programación C que utilice el Algoritmo Misra & Gries para colorear los enlaces de un grafo no dirigido. El programa deberá leer el grafo utilizando un fichero con el formato que se muestra en el Anexo, y guardarlo como una lista de adyacencia.

Como resultado, el algoritmo deberá indicar los colores de los diferente enlaces del grafo.

Proyecto 2. Algoritmo de Borůvka

Desarrollar un programa en el lenguaje de programación C que utilice el Algoritmo de Borůvka para establecer el *Minimum Spanning Tree* (MST) de un grafo no dirigido conectado. El programa deberá leer el grafo utilizando un fichero con el formato que se muestra en el Anexo, y guardarlo como una lista de adyacencia.

Como resultado, el algoritmo deberá indicar el MST correspondiente.

Proyecto 3. Algoritmo de Fleury

Desarrollar un programa en el lenguaje de programación C que utilice el Algoritmo de Fleury para establecer el ciclo de Euler en un grafo no dirigido. El programa deberá leer el grafo utilizando un fichero con el formato que se muestra en el Anexo, y guardarlo como una lista de adyacencia. El código deberá comprobar, antes de ejecutar el algoritmo, si el grafo cumple los requisitos para poder establecer el ciclo de Euler.

Como resultado, el algoritmo deberá presentar el ciclo encontrado.

[1] `http://www.graphviz.org/`

Proyecto 4. Algoritmo de Bron–Kerbosch

Implementar un programa que utilice el algoritmo de Bron–Kerbosch para encontrar *cliques* máximos en un grafo no dirigido. El grafo se leerá de un fichero de entrada, con el formato que se muestra en el Anexo, y se guardará como una lista de adyacencia.

Como resultado, el algoritmo deberá presentar los *cliques* encontrados.

Proyecto 5. Algoritmo de Dinic

Desarrollar un programa en el lenguaje de programación C que utilice el Algoritmo de Dinic para establecer el máximo flujo en un grafo dirigido entre dos nodos, s y d. El programa deberá leer el grafo utilizando un fichero con el formato que se muestra en el Anexo (utilizando capacidades en lugar de costes), y guardarlo como una lista de adyacencia. Los nodos s y d se especificarán por teclado (o como argumentos a la hora de ejecutar el programa).

Como resultado, el algoritmo deberá indicar el máximo flujo que se puede transmitir, así como el que va por cada uno de los enlaces.

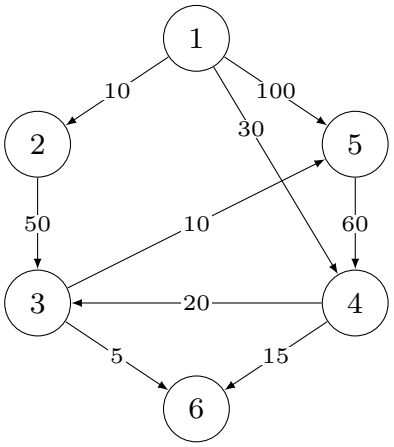
Anexo: distribución de proyecto por grupos

Grupo	Proyecto
1	Proyecto 1. Misra & Gries
2	Proyecto 5. Dinic
3	Proyecto 2. Borůvka
4	Proyecto 4. Bron-Kerbosh
5	Proyecto 3. Fleury

Anexo: formato fichero lista adyacencia

La primera línea tiene el número de nodos del grafo y, a partir de la segunda se utiliza una línea por enlace, situando el nodo origen, el nodo destino y el coste, separados por comas.

A continuación se muestra una red ejemplo y cuál sería el fichero correspondiente.



grafo.ady

```
6
1,2,10
1,4,30
1,5,100
2,3,50
3,5,10
3,6,5
4,3,20
4,6,15
5,4,60
```