E.T.S.I.I.T - Grado en Ingeniería de Tecnologías de Telecomunicación
Dimensionado y Planificación de Redes
Curso 2023/2024

Práctica 1 - Representación y algoritmos sobre grafos.

Los alumnos deberán realizar el proyecto asignado y presentar el funcionamiento del programa antes del **1 de diciembre**. Se deberá entregar el código correspondiente y un pequeño informe (aproximadamente con una extensión 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 `VirtualBox`. 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.

La máquina virtual se podrá descargar con un enlace que se habilitará en el Moodle.

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 Karger

Implementar un programa que ejecute el Algoritmo de Karger, para encontrar el corte mínimo (menor número de enlaces) 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.

Proyecto 2. Algoritmo de Kosaraju

Implementar un programa que utilice el algoritmo de Kosaraju para encontrar los compenentes fuertemente conectados de un grafo 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.

Proyecto 3. Algoritmo de Yen

Implementar un programa que encuentre las k rutas de coste mínimo entre un nodo origen y un destino en un grafo dado, utilizando el Algoritmo de Yen.

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.

Proyecto 4. 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.

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

Proyecto 5. Algoritmo de Dinic

Desarrollar un programa 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.

Proyecto 6. Componentes conectados en un grafo no-dirigido

Implementar un programa que, utilizando el algoritmo BFS, determine el número de componentes aislados de un grafo no dirigido (un componente aislado es un conjunto de nodos conectados entre sí, pero aislados del resto de la red).

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. El programa generará un fichero de salida en el que se mostrará el número de componentes que tiene el grafo, en función del número de nodos que tienen.

Proyecto 7. Matching en grafos bipartitos: Algoritmo Hopcroft-Karp

Implementar el algoritmo de Hopcroft-Karp, que permite resolver el problema de *matching* de cardinalidad máxima en un grafo bipartito.

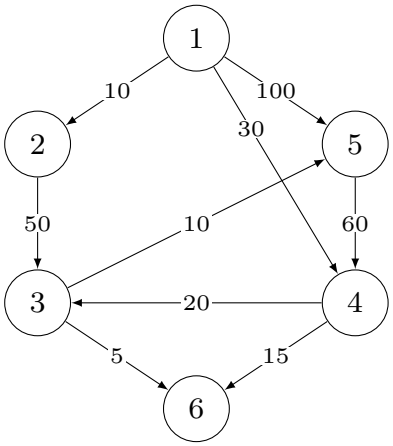
El grafo (siempre será bipartito) se leerá de un fichero de entrada, con el formato que se indica en el anexo, y se guardará como una lista de adyacencia.

Anexo: distribución de proyecto por grupos

Grupo	Proyecto
1	Proyecto 1. Karger
2	Proyecto 2. Kosaraju
3	Proyecto 6. BFS. Componentes conectados
4	Proyecto 7. Hopcroft-Karp
5	Proyecto 4. Buruvka
6	Proyecto 5. Dinic
7	Proyecto 3. Yen

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
```