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

Dimensionado y Planificación de Redes

Curso 2024/2025

Práctica 1 - Representación y algoritmos sobre grafos.

Los alumnos deberán realizar el proyecto asignado y presentar el funcionamiento del programa antes del **29 de noviembre**. 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. Determinar si un grafo es bipartito

Implementar un programa que, utilizando el algoritmo BFS, determine si un grafo es bipartito o no.

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. Orden topológico mediante DFS

Implementar un programa que, utilizando el algoritmo DFS, establezca el orden topológico de los nodos en un grafo acíclico.

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. Se deja a elección de los alumnos el comprobar que el grafo es acíclico o asumir que lo es.

Proyecto 3. Algoritmo DSATUR para colorear grafos

El algoritmo DSATUR es una técnica heurística, que da buenos resultados para colorear grafos. En este proyecto se pide que se implemente dicho algoritmo y que se pruebe sobre un grafo no-dirigido, que se leerá de un fichero de entrada con el formato que se especifica en el anexo, y que se guardará como una lista de adyacencia.

Proyecto 4. Algoritmo de Kruksal

Implementar un programa que utilice el algoritmo de *Kruksal* para determinar el *Minimum Spanning Tree* (MST) de 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. El programa deberá generar el árbol en un formato que se deja a elección de los alumnos.

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

Proyecto 5. Algoritmo de Prim

Implementar un programa que utilice el algoritmo de Prim para determinar el *Minimum Spanning Tree* (MST) de 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. El programa deberá generar el árbol en un formato que se deja a elección de los alumnos.

Proyecto 6. Algoritmo de Edmonds-Karp

Se trata de una implementación particular del algoritmo de *Ford-Fulkerson*, en el que la búsqueda del camino en el grafo residual se realiza utilizando BFS, asumiendo que el coste de todos los enlaces es el mismo.

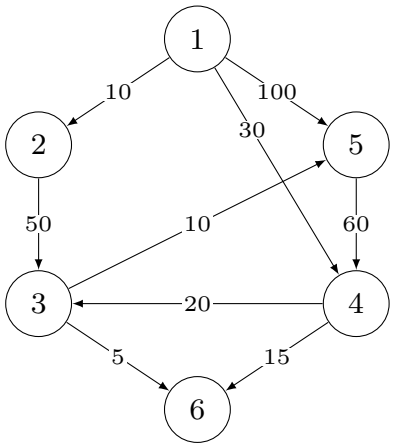
Se partirá de un grafo dirigido, que se leerá de un fichero de entrada con el formato indicado en el Anexo (en este caso se proporcionará la capacidad por enlace, y no su coste), guardándolo como una lista de adyacencia. Se proporcionará, como salida, los caminos que tendrían que usarse para lograr enviar la máxima capacidad posible entre dos nodos del mismo.

Anexo: distribución de proyecto por grupos

Grupo	Proyecto
1	Proyecto 2. DFS: Orden topológico
2	Proyecto 6. Edmonds-Karp
3	Proyecto 4. Kruksal
4	Proyecto 5. Prim
5	Proyecto 1. BFS: Grafos bipartitos
6	Proyecto 3. DSATUR

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
```