



Ejercicios de grafos

OBJETIVOS

El objetivo de estos ejercicios es repasar los conceptos sobre el TAD grafos.

En la página de UAMX de la asignatura dispones del esqueleto del código para poder trabajar con estos ejercicios. Las funciones están declaradas en el TAD grafo.

Ejercicio recorrido en anchura

Implementar una función que realice un recorrido en anchura desde el vértice con índice dado como parámetro. Esta función recibe como argumento un grafo y debe imprimir las aristas en el orden dado por un recorrido en anchura.

```
void imprimirEnAnchura(Grafo *g, int partida);
```

Ejercicio recorrido en profundidad

Implementar una función que realice un recorrido en profundidad desde el vértice con índice dado como parámetro. Esta función recibe como argumento un grafo y debe imprimir las aristas en el orden dado por un recorrido en profundidad.

```
void imprimirEnProfundidad(Grafo *g, int partida);
```

Ejercicio contar grados de entrada y salida

Realizar una función que cuente el número de arcos de entrada y de salida para cada vértice:

```
int *contarGradosVertices(Grafo *g);
```

El valor de retorno será un array de dimensión $2 \cdot n_{\text{vertices}}$ que contendrá en la posición $2i$ y $2i + 1$ el número de arcos de salida y de entrada respectivamente para el vértice $i = 0 \dots n_{\text{vertices}}$.

Ejercicio camino euleriano

Realizar una función compruebe si el grafo pasado como parámetro tiene un camino euleriano:

```
int tieneCaminoEuleriano(Grafo *g);
```

El valor de retorno será 0 si no tiene un camino euleriano, 1 si el grafo g tiene un camino euleriano, 2 si tiene un camino euleriano que además es un ciclo (es decir el camino acaba en el mismo vértice que empezó) y -1 si es un grafo **dirigido**. Un grafo **no dirigido** tiene un camino euleriano si y solo si hay 0 o dos vértices con un número impar de aristas. Se tratará de un ciclo cerrado si el número de vértices con número de aristas impar es 0. En caso de ser 2 se tratará de un camino euleriano no cerrado.

Una vez resuelto se debe aplicar al siguiente problema:

<https://www.aceptaelreto.com/problem/statement.php?id=212&cat=43>



Ejercicio recorrido orden topológico

Realizar una función realice el recorrido en orden topológico del grafo g:

```
int imprimirOrdenacionTopologica(Grafo *g);
```

La función debe imprimir el orden topológico del grafo. Si es posible hacer el recorrido completo del grafo, la función debe devolver 1 y si no 0.