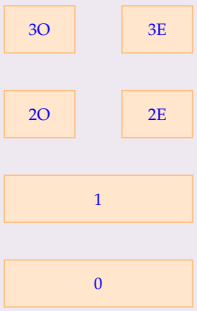
Pregunta 1

La facultad de Königsberg de arriba abajo

En esta facultad hay muchas escaleras. Llamaremos *tramo de escalera* al que lleva de un piso a otro. Sin aventurarnos en el sótano, nos proponemos recorrer todos esos tramos en un solo trayecto, sin repetir ninguno (ni usar los ascensores u otro tipo de trampa).

Para representar la información de partida (cuántos tramos de escalera hay y cómo están dispuestos), utilizaremos un grafo (no dirigido). Comencemos dibujándolo. Sus vértices son las seis «plantas» de la facultad (contamos como dos tanto la segunda planta como la tercera; además, excluimos el sótano), según el esquema de la derecha. Esos nodos tienen enlaces que permiten acceder a los planos de la instalación.
¡Atención! Es esta hoja trabajamos con grafos (no dirigidos y) no necesariamente simples: pueden aparecer múltiples aristas que unan la misma pareja de vértices.



1) *¿Es posible realizar un recorrido como el propuesto?* (*0.5 puntos*)

2) *¿Es posible recorrer sin repetir ninguno todos los tramos de escalera, comenzando y terminando además en el mismo lugar?* (*0.5 puntos*)

3) *Responde también estas dos preguntas sobre el grafo resultante de eliminar la planta baja.* (*0.5 puntos*)

Antes de continuar necesitamos definir un algoritmo para saber si un grafo no dirigido es conexo, es decir que siempre existe un camino entre dos nodos del grafo. Este problema se llama en general «dynamic connectivity» (conexión dinámica). Supongamos que tenemos un grafo con n nodos y los etiquetamos con números $1, \ldots, n$. La idea para saber si dos nodos están unidos es utilizar un representante de los nodos que están unidos entre si. Esto se representa como una lista que será iniciada a $1, 2, \ldots, n$, ya que cada nodo está conectado solo consigo mismo. Ahora, si tenemos $(p, q) \in E$ es una arista, entonces miramos que nodo representa la componente donde está p y cambiamos en la lista este número por el número que representa la componente de q. un pseudo-codigo podría ser asi:

```
def unir(p,q):
    pid = lista[p]
    qip = lista[q]
    para cada 0<= i < n:
     si (lista[i] == qid):
        lista[i] = pid
```

4) *Diseñe una estructura de datos, utilizando Scheme que represente este tipo de grafos. Esta estructura de datos tiene que ser programada con estilo «Message Passing» y tiene que permitir calcular adyacentes a un nodo, número de adyacentes a un nodo, quitar una arista y si un grafo es conexo.* (*3.5 puntos*)

Un camino en un grafo que recorre, sin repeticiones, todas sus aristas, se llama *camino euleriano*. Si, además, es circular, se denomina *ciclo* (o *circuito*) *euleriano*. Un grafo es *euleriano* cuando permite responder afirmativamente a la pregunta (2), es decir, cuando contiene un ciclo euleriano. Estas denominaciones se deben a que Euler dio, en un famosísimo estudio inspirado por los puentes regiomontanos, una caracterización sencilla que permite responder las dos preguntas, para cualquier grafo conexo, en función de la paridad de los grados de sus vértices:[1]

[1] L. Euler: «Solutio problematis ad geometriam situs pertinentis», §20 (1741).

> *Caſu ergo quocunque propoſito ſtatim facillime poterit cognoſci, vtrum tranſitus per omnes pontes ſemel inſtitui queat an non, ope huius regulae. Si fuerint plures duabus regiones, ad quas ducentium pontium numerus eſt impar, tum certo affirmari poteſt, talem tranſitum non dari. Si autem ad duas tantum regiones ducentium pontium numerus eſt impar, tunc tranſitus fieri poterit, ſi modo curſus in altera harum regionum incipiatur. Si denique nulla omnino fuerti regio, ad quam pontes numero impares conducant, tum tranſitus deſiderato modo inſtitui poterit, in quacunque regione ambulandi initium ponatur. Hac igitur data regula problemati propoſito pleniſſime ſatisfit.*

5) *Deduce (o traduce) una caracterización de los grafos que contienen un camino euleriano y una de los grafos eulerianos. (0.5 puntos)*

6) *Escribe una función que devuelva* `True` *o* `False` *según el grafo de entrada (que podría ser disconexo) sea o no euleriano. (1 punto)*

7) *Con las mismas condiciones, escribe una función que devuelva*

   - `False`*, si el grafo de entrada no tiene ningún camino euleriano;*
   - `True`*, si es euleriano;*
   - *una lista de los dos vértices que son los extremos de cualquier camino euleriano en el grafo, en otro caso.*

   (1 punto)

Es sencillo demostrar la necesidad de la condición que venimos estudiando para la existencia de un ciclo o camino euleriano. Para la suficiencia, es conveniente (y apropiado para una asignatura sobre algoritmos) una demostración constructiva. Como dice la memoria de Euler, *quaeſtio ſupereſt quomodo curſus ſit dirigendus.* Acto seguido, se propone describir el proceso de construcción: *Pro hoc ſequenti vtor regula*. Sin embargo, despacha la cuestión con unas indicaciones demasiado escuetas,[2] que podríamos parafrasear así:

> Suprimiendo provisionalmente todas las parejas de caminos que tengan los mismos extremos, por los caminos que queden se encuentra con facilidad un camino del tipo que nos interesa. No difiere mucho de la solución que buscamos.

A Euler no le pareció necesaria la descripción rigurosa de un algoritmo para esta tarea: *neque opus eſſe iudico plura ad curſus reipſa formados praecipere.* Siglo y medio más tarde, el interés por la cuestión algorítmica se mostraba más vivo: según Fleury, la descripción de Euler es, por una parte, equívoca e incompleta y, por otra, la mejor para asegurarse de no resolver el problema.[3]

8) *Programe en lenguaje Scheme un algoritmo que, tomando como entrada un grafo, construya un camino euleriano, si lo hay. (1.5 puntos)*

9) *Analiza la complejidad espacial y temporal de la implementación. Justifique si genera un proceso recursivo o iterativo. (1 punto)*

La búsqueda de un camino euleriano está en **P**, mientras que la de uno *hamiltoniano* (una única visita a cada vértice, en vez de a cada arista) es un problema **NP**-completo.

---

[2] «Solutio problematis ad geometriam situs pertinentis», §21.

[3] Fleury: «Deux problèmes de géométrie de situation». Journal de Mathématiques Élémentaires, 1883, p. 261:
La règle donnée par Euler, équivoque et imcomplète, d'une part, est, d'autre part, la meilleure pour assurer la non-réussite du problème.