Programación Avanzada (1113) **UNLaM**

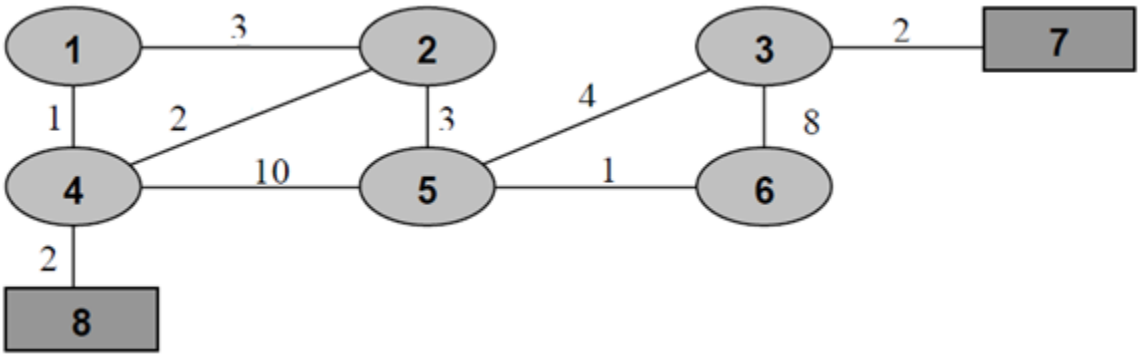
Fecha: 25/11/2021 **Instancia:** 2do Parcial individual

Consigna general

- Responder todas las preguntas en un documento .pdf.
- El nombre de los archivos debe ser: DNI-ApellidosNombres.
- La entrega se realizará a través de la solapa “Prácticas” de la plataforma MIeL.

1. Grafos

Los marcianos han decidido invadir el planeta, otra vez. Su plan es el siguiente. Inicialmente invadirán un conjunto de ciudades seleccionadas. A partir de ahí, moverán sus naves por todos los caminos posibles, invadiendo las ciudades que encuentren.Se desea saber cuál es el tiempo mínimo que tardan en invadir todo el planeta, y por qué caminos se moverán
Ejemplo: ciudades de 1 a 8, ciudades invadidas inicialmente 7 y 8. El peso de las aristas indica tiempo en ir de una ciudad a la otra.



No escriba el algoritmo en java, responder las siguientes preguntas

1. ¿Qué algoritmo/s de los vistos puede aplicar para resolver este problema?.
2. Indique el más conveniente si lo hay y justifique la respuesta
3. ¿Cuál es la complejidad computacional del algoritmo?
4. ¿Cúal es el resultado de aplicar el algoritmo en el ejemplo dado?

Programación Avanzada (1113) **UNLaM**

Fecha: 25/11/2021 **Instancia:** 2do Parcial individual

2. Prolog

En una organización se distribuyen las órdenes desde cada empleado a sus subordinados directos, estos la informan a su vez a sus subordinados y así sucesivamente hasta llegar a los empleados de menor nivel.
Se tiene un predicado llamado "reporta_a" , que relaciona cada persona con su superior directo, utilizando sus nombres. En este predicado, el segundo es el jefe del primero. No pueden existir dos empleados con el mismo nombre.
Por ejemplo:

```
reporta_a(juan, pedro).
reporta_a(carlos, juan).
reporta_a(jose, carlos).
```

Se pide definir un predicado "recibeOrdenesDe", dados dos nombres de persona, determinar si el primero depende del segundo directa o indirectamente)
Por ejemplo recibeOrdenesDe(jose, pedro) deberias dar true.
No se sabe la cantidad de niveles del organigrama.

3. Haskell

```
-- sumar los elementos situados en posiciones pares de una lista,
-- sin utilizar la función "sum"
-- ejemplo:
-- sump [1, 2, 3] -> 4
-- sump [2, 1, 5, 8] -> 7

sump :: [Int] -> Int
```