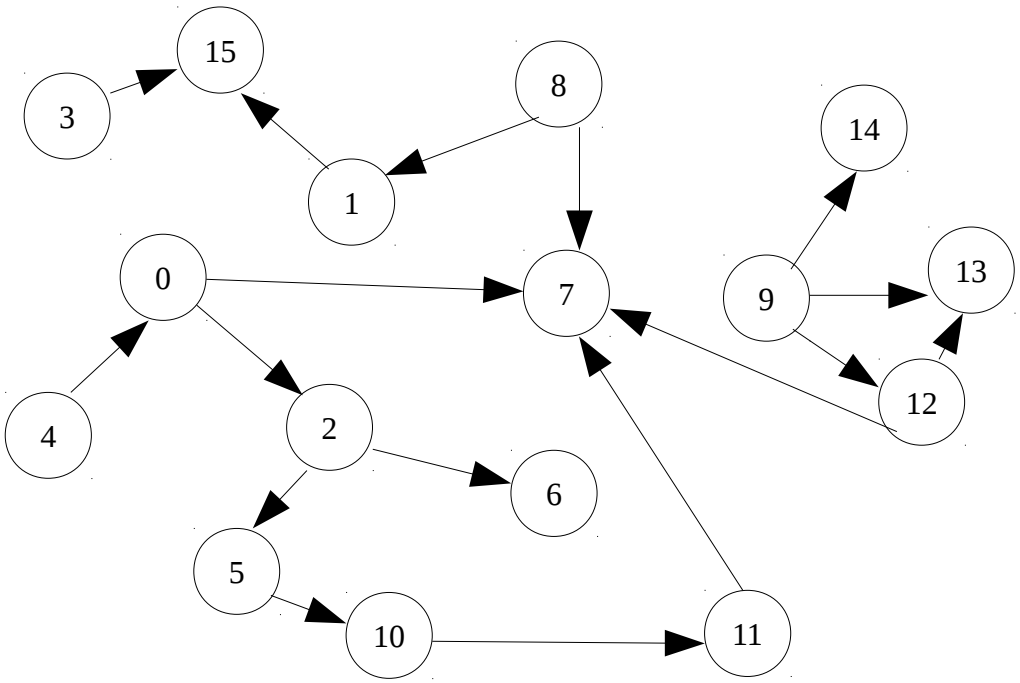
Ejercicio 4: El contacto que más puede ayudar (1,5 puntos)

En el siguiente grafo, los nodos representan personas y los arcos indican los contactos de cada persona en redes sociales.



Cuando una persona necesite un transplante, pondrá un mensaje en sus redes sociales para buscar personas compatibles con ella, de manera que sus contactos puedan ver este mensaje y difundirlo a otros contactos, y así sucesivamente.

Debéis implementar el método *elQueMasPuedeAyudar* e indicar su coste. Este método recibe un ArrayList con personas que necesitan un transplante, y debe buscar la persona del grafo que puede ayudar a más personas de dicha lista (en caso de empate la de menor índice). Una persona X podrá ayudar a una persona Y si X e Y son compatibles para el transplante y además X recibe por redes sociales el mensaje de Y, sabiendo que los mensajes se difunden según lo explicado en el párrafo anterior. Para saber si dos personas son compatibles para un transplante debéis utilizar la función *sonCompatibles* (no hay que implementarla).

```
public class RedDeContactos {
    private boolean[][] adjMatrix; //adjacency matrix

     public int elQueMasPuedeAyudar(ArrayList<Integer> personas){
    . . .
    }
    private boolean sonCompatibles(Integer p1, Integer p2){
    //POST: Devuelve si p1 y p2 son compatibles. NO HAY QUE IMPLEMENTARLA
    }
}
```

Por ejemplo, supongamos que la lista de entrada es <4, 8, 2, 9> y que éstas son las parejas compatibles: (4, 7), (4, 10), (2, 10), (8, 7), (8, 10) y (9, 7).

En este ejemplo 7 sería el que más puede ayudar (porque es accesible desde 4, 8 y 9 y compatible con ellos, es decir, puede ayudar a 3 contactos). La persona con el número 10 sin embargo sólo puede ayudar a dos personas (4 y 2).