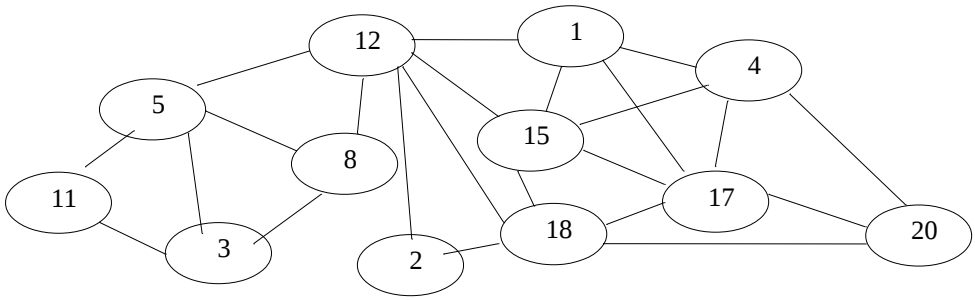
4. Buscar camino (2,5 puntos)

Tenemos un grafo:



Queremos desarrollar un algoritmo que, dada una lista de valores, nos diga si existe un camino en el grafo que conecta los elementos de la lista. Es decir, por cada pareja de elementos consecutivos de la lista (x, y), existe un arco en el grafo desde x hasta y.

Por ejemplo:

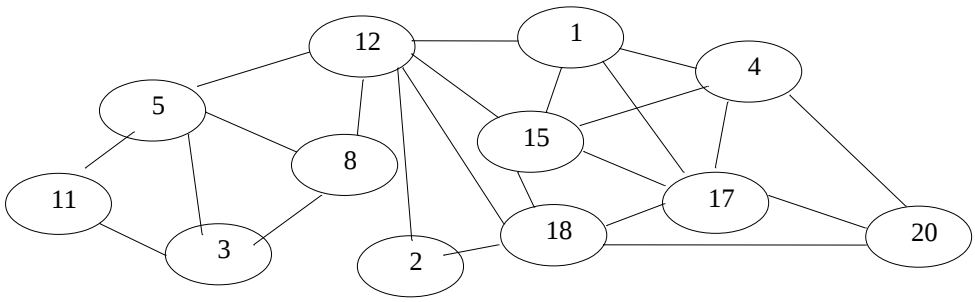
- el resultado con la lista <12, 1, 4, 17, 18, 15> será true, ya que ese camino sí existe en el grafo, que contiene los arcos (12, 1), (1, 4), (4, 17), (17, 18), y (18, 15)
- el resultado con la lista <12, 1, 20, 17, 18, 15> será false, ya que no se puede ir del nodo 1 al 20 (no hay una conexión directa)

```
public class Grafo
{

   protected int numVertices;   // number of vertices in the graph
   protected int[][] adjMatrix;   // adjacency matrix


   public boolean existeCamino(ArrayList<Integer> lista)
}
```

Se pide:

- Implementar el algoritmo
- Establecer de manera razonada el coste del algoritmo.

4. Bidea bilatu (2,5 puntu)
Ondoko irudiak grafo bat adierazten du:



Balioen zerrenda bat emanda, zerrendako elementuak konektatzen duen bidea dagoen jakin nahi da. Hau da, zerrendan jarraian doazen elementuen (x, y) bikote bakoitzeko, grafoak arku bat du x-etik y-raino.

Adibidez:

- <12, 1, 4, 17, 18, 15> zerrendarekin emaitza true izango da, bide hori grafoan baitago, eta arku hauek daudelako: (12, 1), (1, 4), (4, 17), (17, 18), eta (18, 15)
- <12, 1, 20, 17, 18, 15> zerrendarekin emaitza false izango da, ezin delako 1-etik 20-ra joan (ez dago arku zuzena)

```
public class Grafo
{

   protected int numVertices;   // number of vertices in the graph
   protected int[][] adjMatrix;   // adjacency matrix


   public boolean bideaDago(ArrayList<Integer> lista)
}
```

Hau eskatzen da:
- Algoritmoa inplementatu
- Algoritmoaren kostua kalkulatu, modu arrazoituan.