Práctica 2

Pablo Gutiérrez Aguirre
pgutierrez2018@udec.cl

4 de abril 2022

Tabla de contenidos

Ejercicio 1a

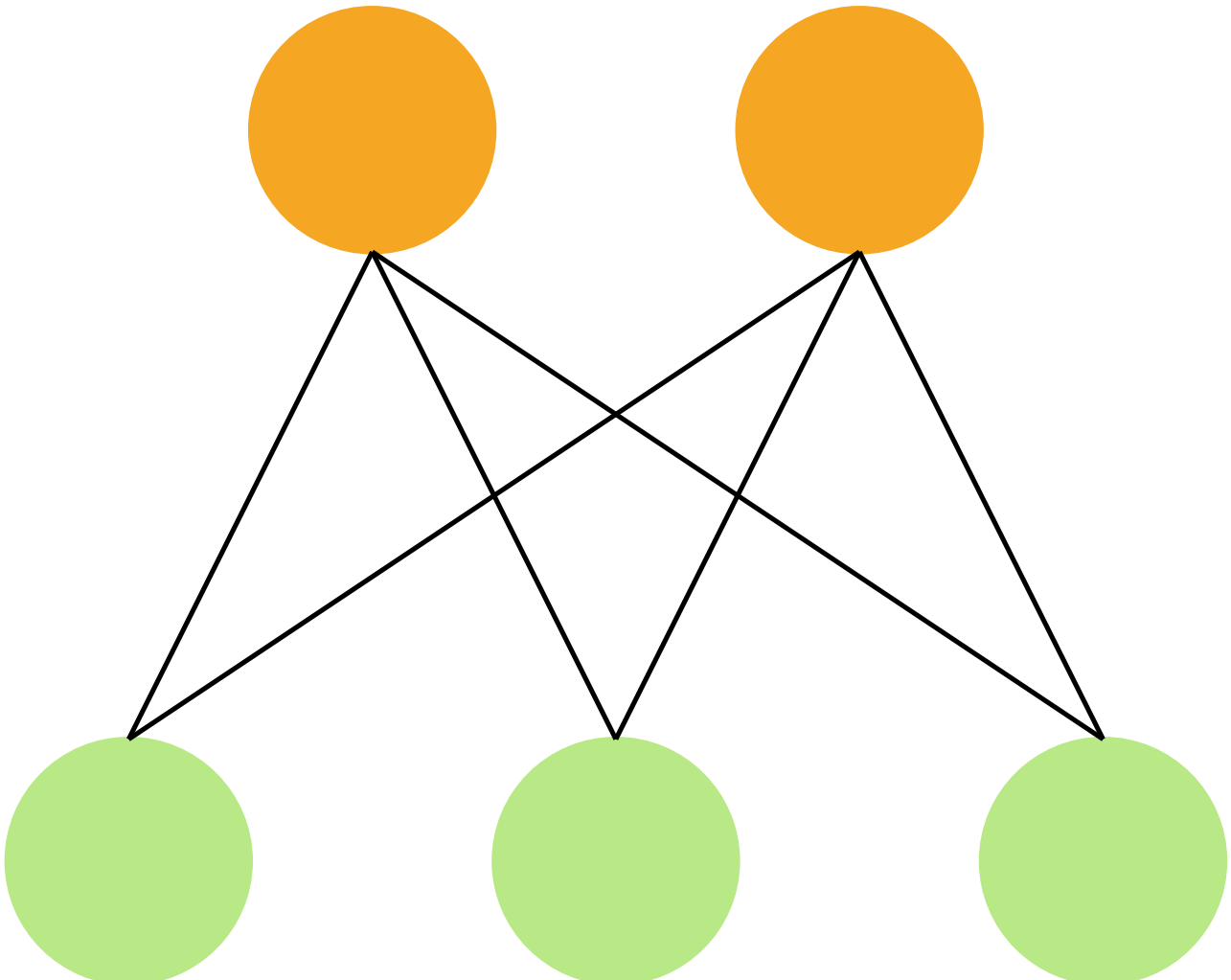
Responda las siguientes preguntas con verdadero o falso, de ser falso diga por qué o de un contraejemplo.

a. Un grafo bipartito completo puede tener un clique de tamaño $n \geq 3$

Ejercicio 1a

- También conocido como bipartido, es un grafo $G = (N, A)$ donde los vértices se pueden separar en dos conjuntos disjuntos U y V , es decir, tal que se cumple:
  - $U \cup V = N$
  - $U \cap V = \emptyset$
- Un clique C en un grafo no dirigido $G = (V, E)$, entonces, es un conjunto de vértices $C \subseteq V$, tal que todos los vértices son adyacentes entre sí. En otras palabras, el subgrafo inducido C es un grafo completo.

Ejercicio 1a

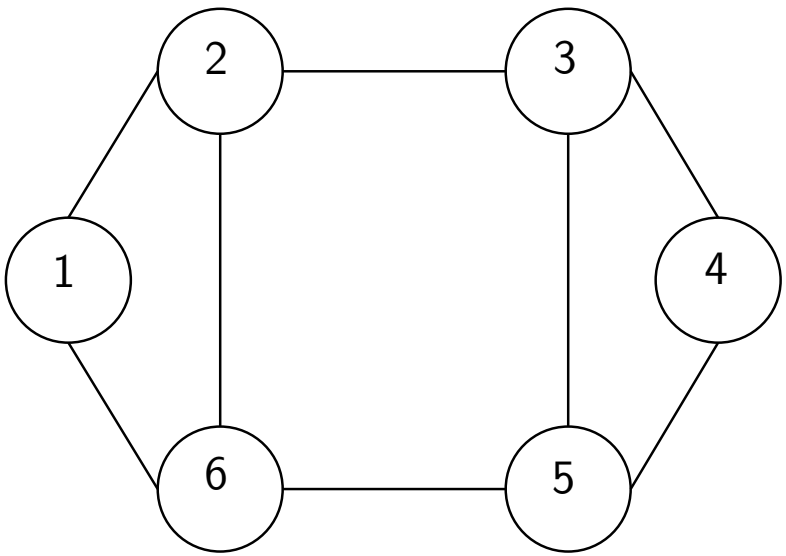


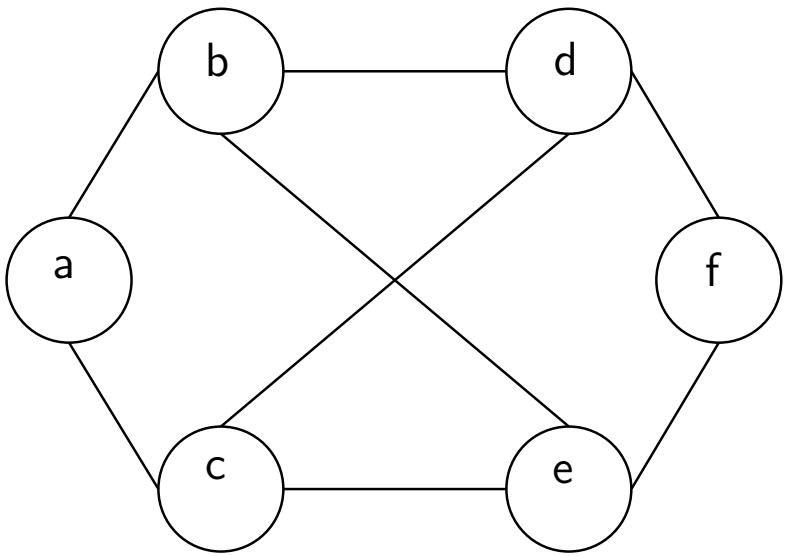
Ejercicio 1a

Dado que por definición, un grafo bipartito no tiene aristas entre vértices de un mismo conjunto, no puede existir un clique de tamaño $n \geq 3$, ya que un clique es un subgrafo inducido **completo**

Ejercicio 3a

Indique si los pares de grafos son isomorfos o no. Argumente su respuesta.



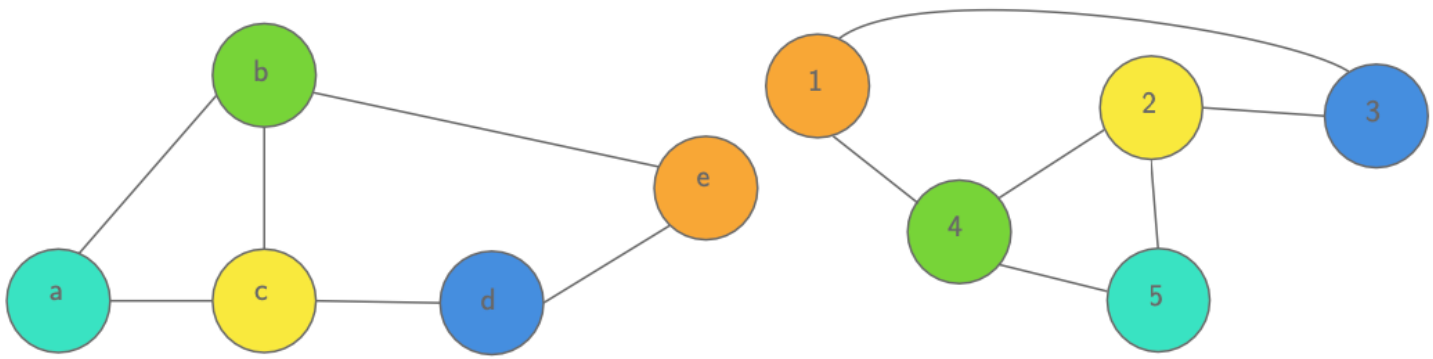


Ejercicio 3a

Dos grafos son isomorfos si existe una función entre los dos que respeta la **estructura de adyacencia** de una en la otra.

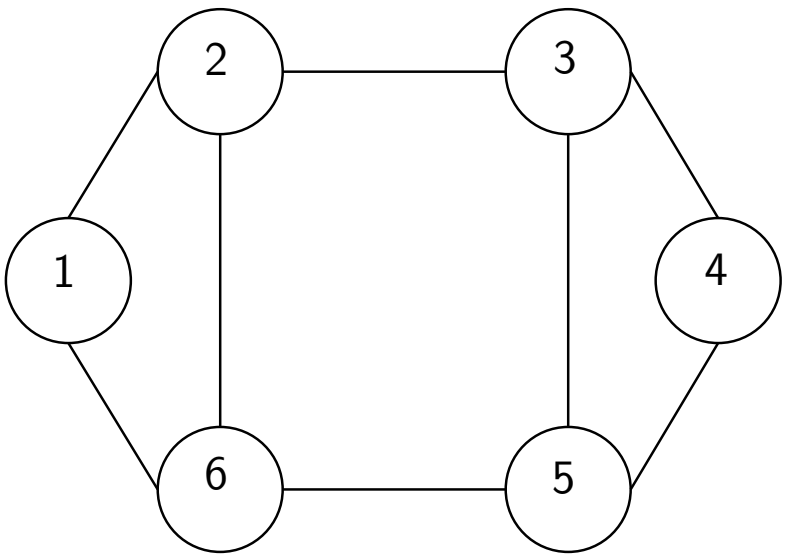
Definición

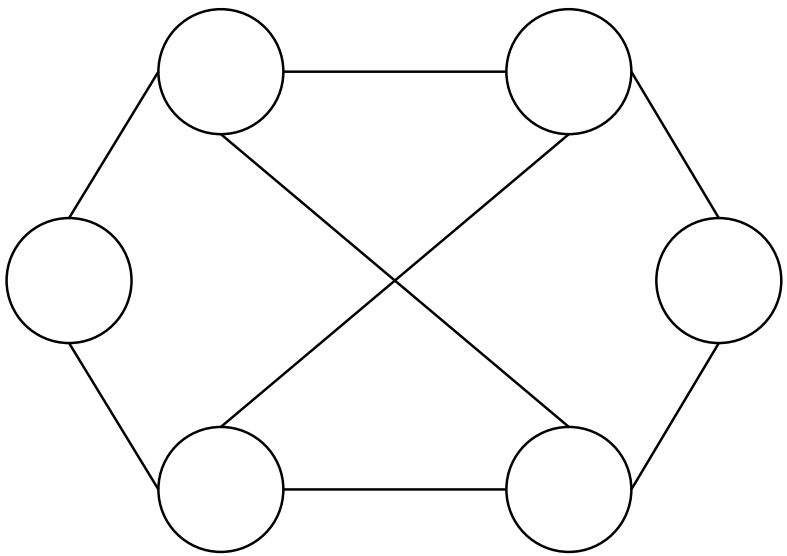
Sean dos grafos G y H, entonces un homomorfismo es una función $f : V(G) \to V(H)$ si y solo si u,v son cualquier par de vértices de G unidos por una arista, entonces $f(u)$ y $f(v)$ son vértices de H que también están unidos por una arista



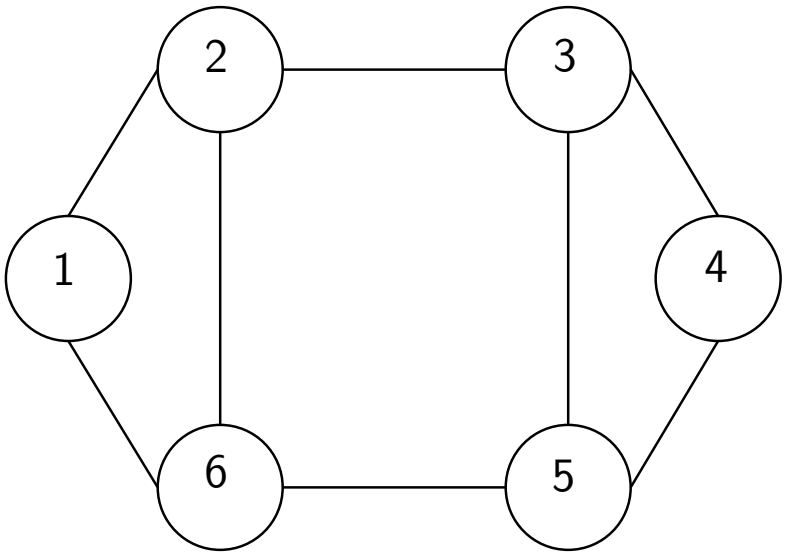
$f(a) = 5, f(b) = 4, f(c) = 2, f(d) = 3, f(3) = 1$

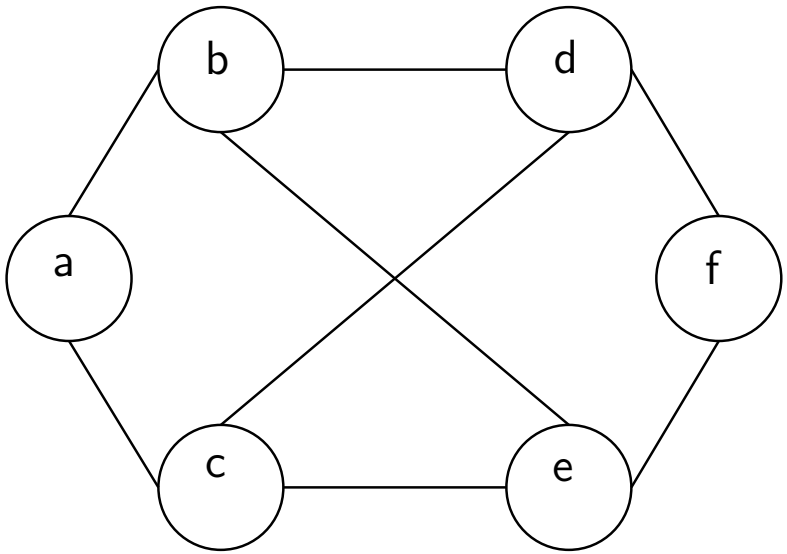
Ejercicio 3a





Ejercicio 3a



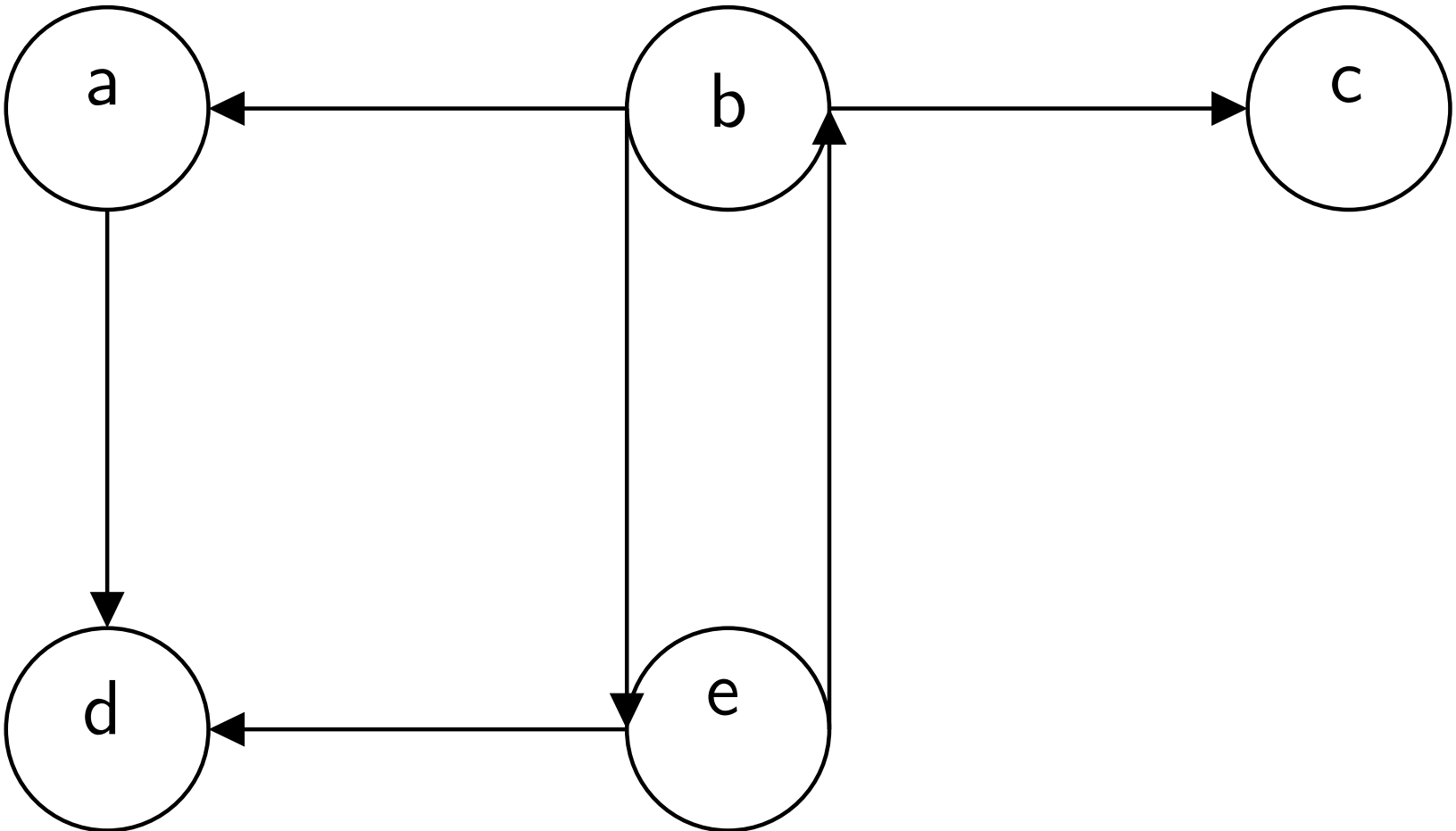


Dado que no se cumplen las reglas de adyacencias, los grafos **no son isomorfos**. Para que sean isomorfos, se debe cumplir que

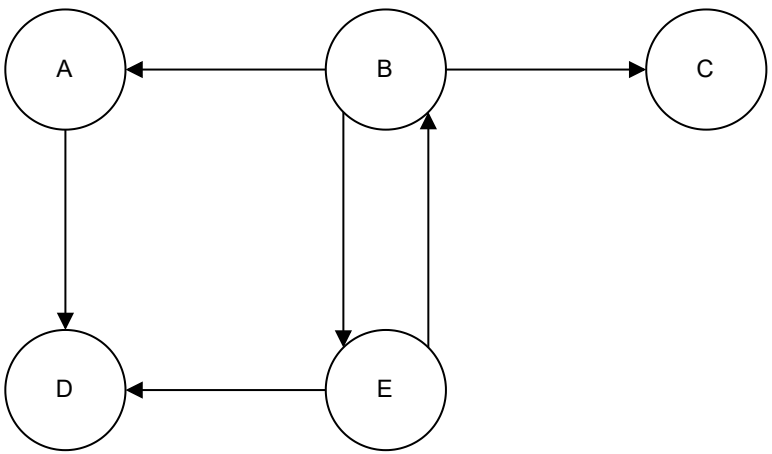
$$F(G_1) = G_2$$

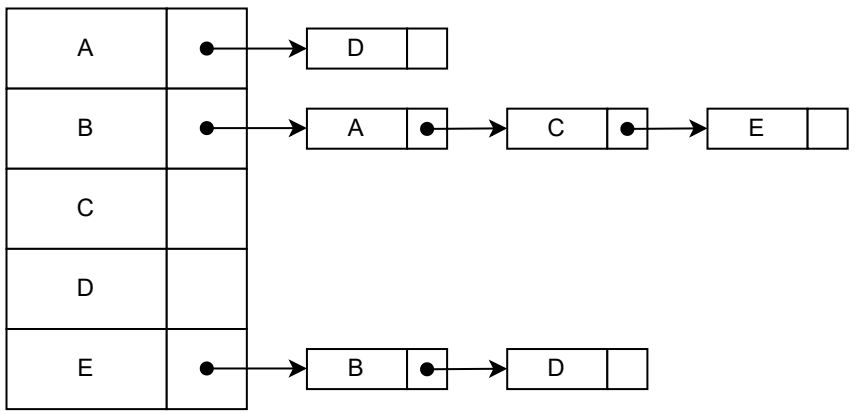
Ejercicio 8c

Represente los siguientes grafos utilizando: **lista de adyacencia**, lista de incidencia, matriz de adyacencia, matriz de incidencia.



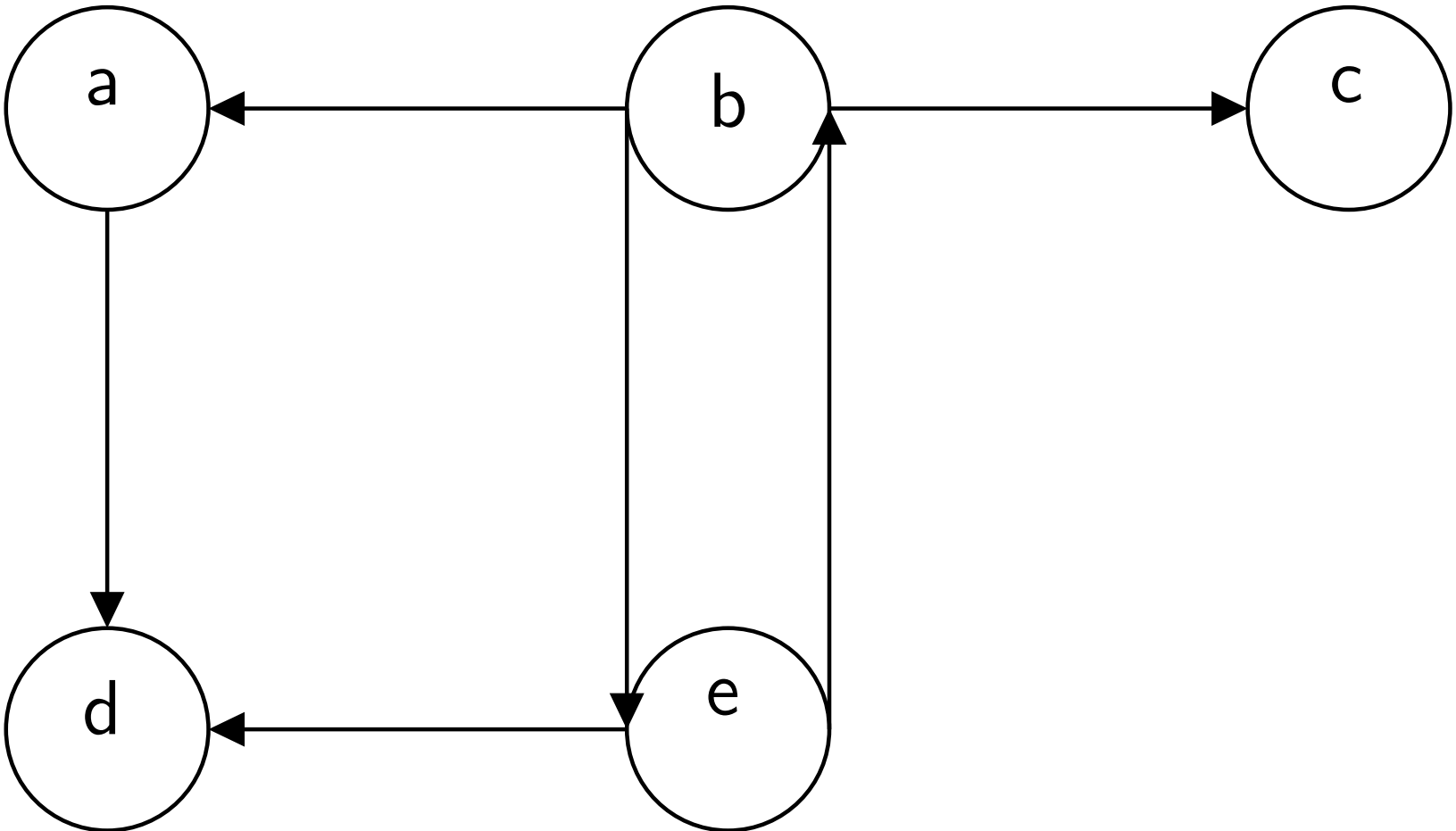
Ejercicio 8c: Lista de adyacencia



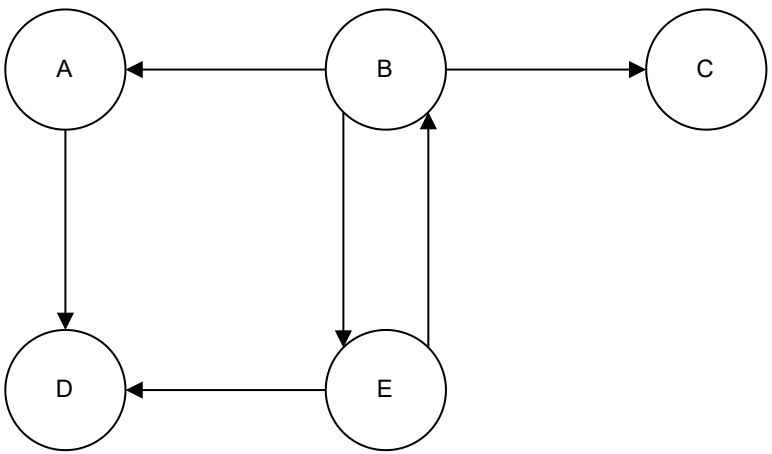


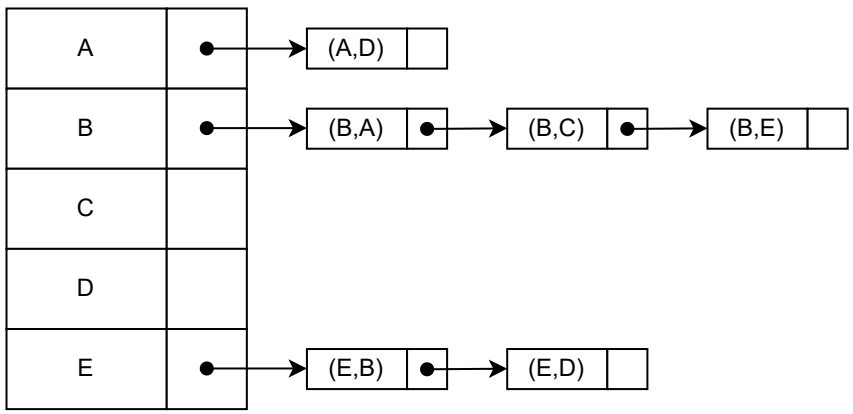
Ejercicio 8c

Represente los siguientes grafos utilizando: lista de adyacencia, **lista de incidencia**, matriz de adyacencia, matriz de incidencia.



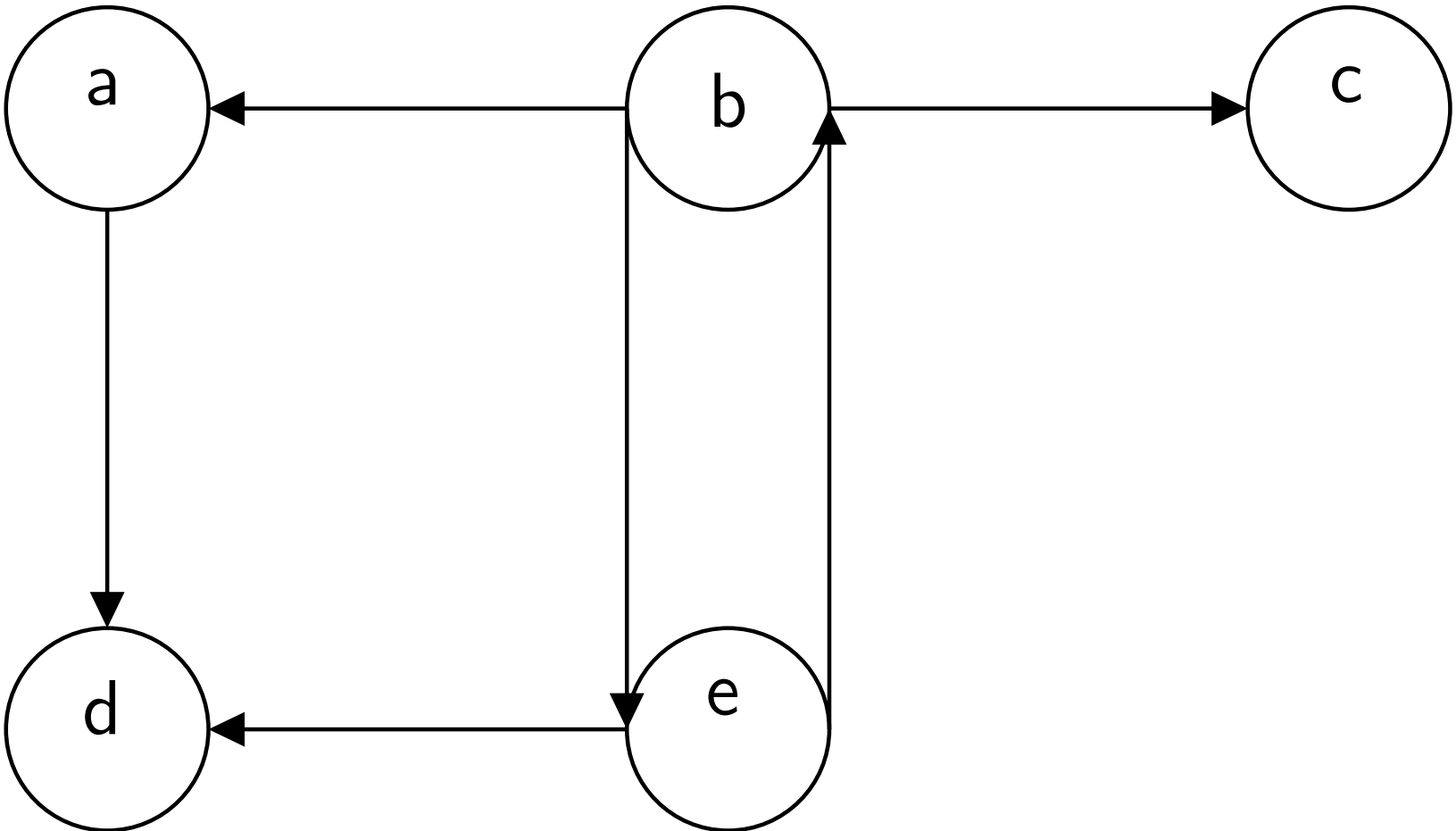
Ejercicio 8c: Lista de incidencia



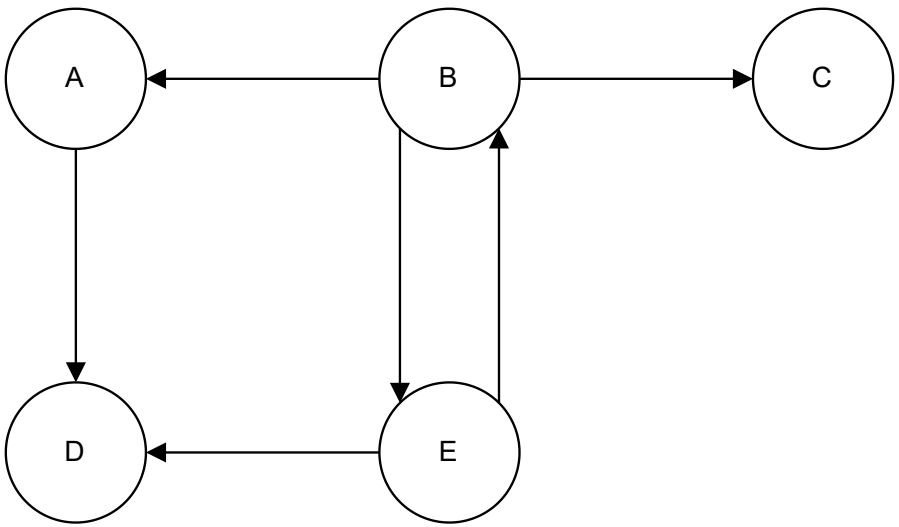


Ejercicio 8c

Represente los siguientes grafos utilizando: lista de adyacencia, lista de incidencia, **matriz de adyacencia**, matriz de incidencia.



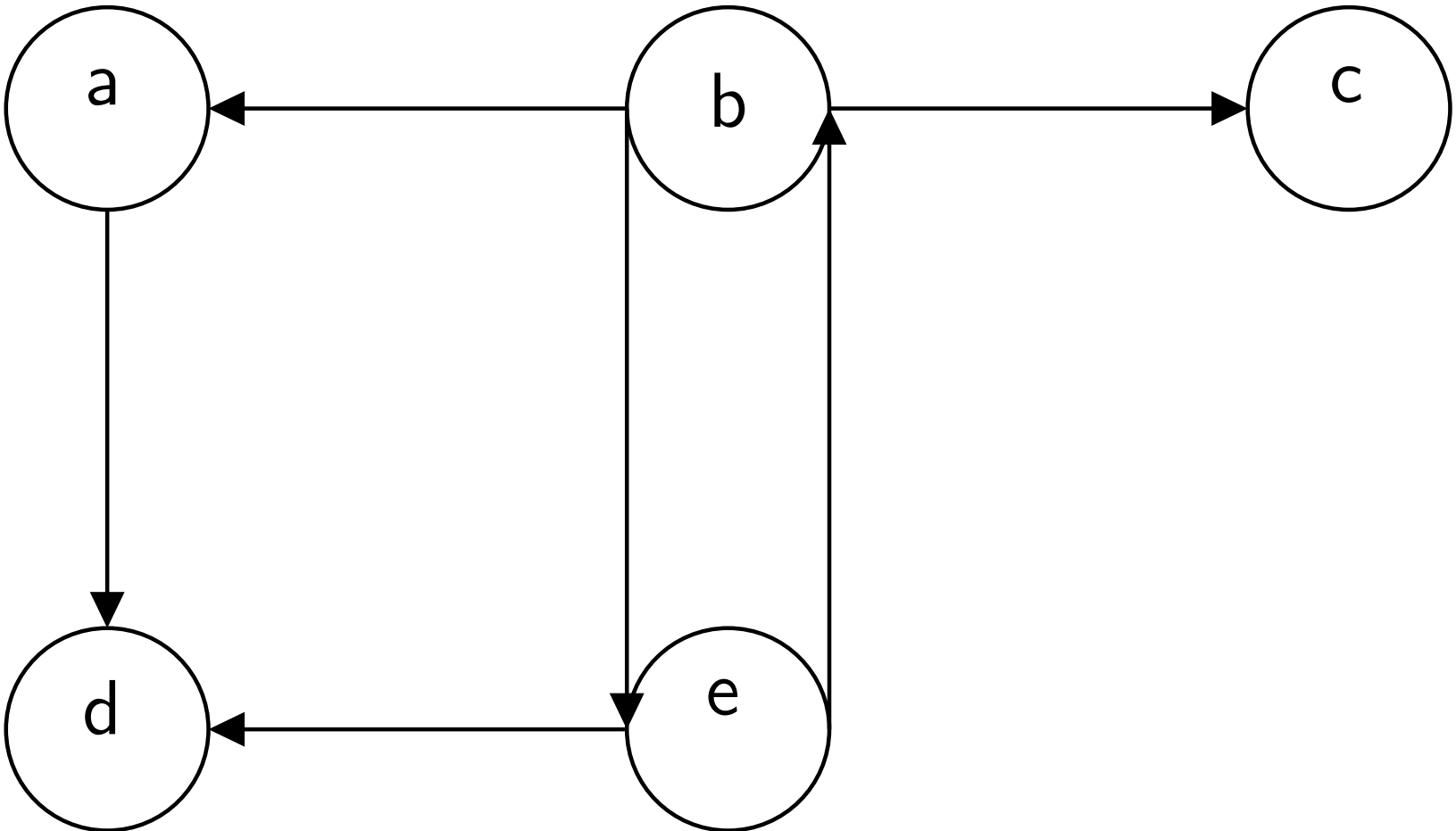
Ejercicio 8c: Matriz de adyacencia



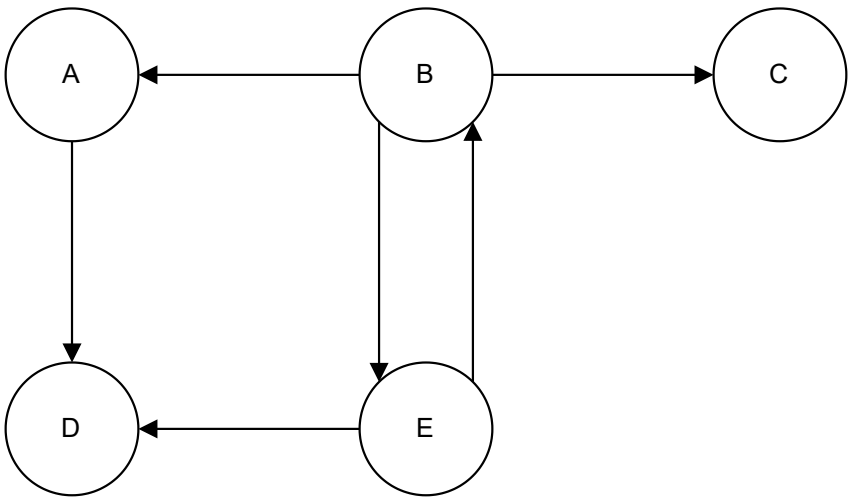
	A	B	C	D	E
A	0	0	0	1	0
B	1	0	1	0	1
C	0	0	0	0	0
D	0	0	0	0	0
E	0	1	0	1	0

Ejercicio 8c

Represente los siguientes grafos utilizando: lista de adyacencia, lista de incidencia, matriz de adyacencia, **matriz de incidencia**.



Ejercicio 8c: Matriz de incidencia



	AD	BA	BC	BE	EB	ED
A	1	-1	0	1	0	0
B	0	1	1	1	-1	0
C	0	0	-1	0	0	0
D	-1	0	0	0	0	-1
E	0	0	0	-1	1	1

Ejercicio 12

Escriba un script en Python, que realice las siguientes tareas:

a. Lea un archivo de texto (.txt) que contiene un grafo no dirigido. Cada línea del archivo corresponde a una arista del grafo, con la siguiente estructura:

Nodo Nodo Peso

b. Usando la librería *networkx* almacene el grafo y que muestre en pantalla: matriz de adyacencia, matriz de incidencia, lista de adyacencia y representación gráfica del grafo.

Ejercicio 12: A

```
1  import networkx as nx
2  import matplotlib.pyplot as plt
3
4  def leer_archivo(nombre):
5      archivo = open(nombre,"r")
6      matriz = [linea.split(" ") for linea in archivo.
       read().split("\n")]
7      archivo.close()
8      for i in range(len(matriz)):
9          for j in range(len(matriz[i])):
10             matriz[i][j] = int(matriz[i][j])
11     return matriz
```

Ejercicio 12: A

```
1  nombre = "grafo3.txt"
2  matriz = leer_archivo(nombre)
3  nodos = []
4  for i in matriz:
5      for j in i[0:2]:
6          if j not in nodos:
7              nodos.append(j)
8  G = nx.Graph()
9  for i in nodos:
10     G.add_node(i)
11
12 for i in matriz:
13     G.add_edge(i[0],i[1],weight=i[2])
```

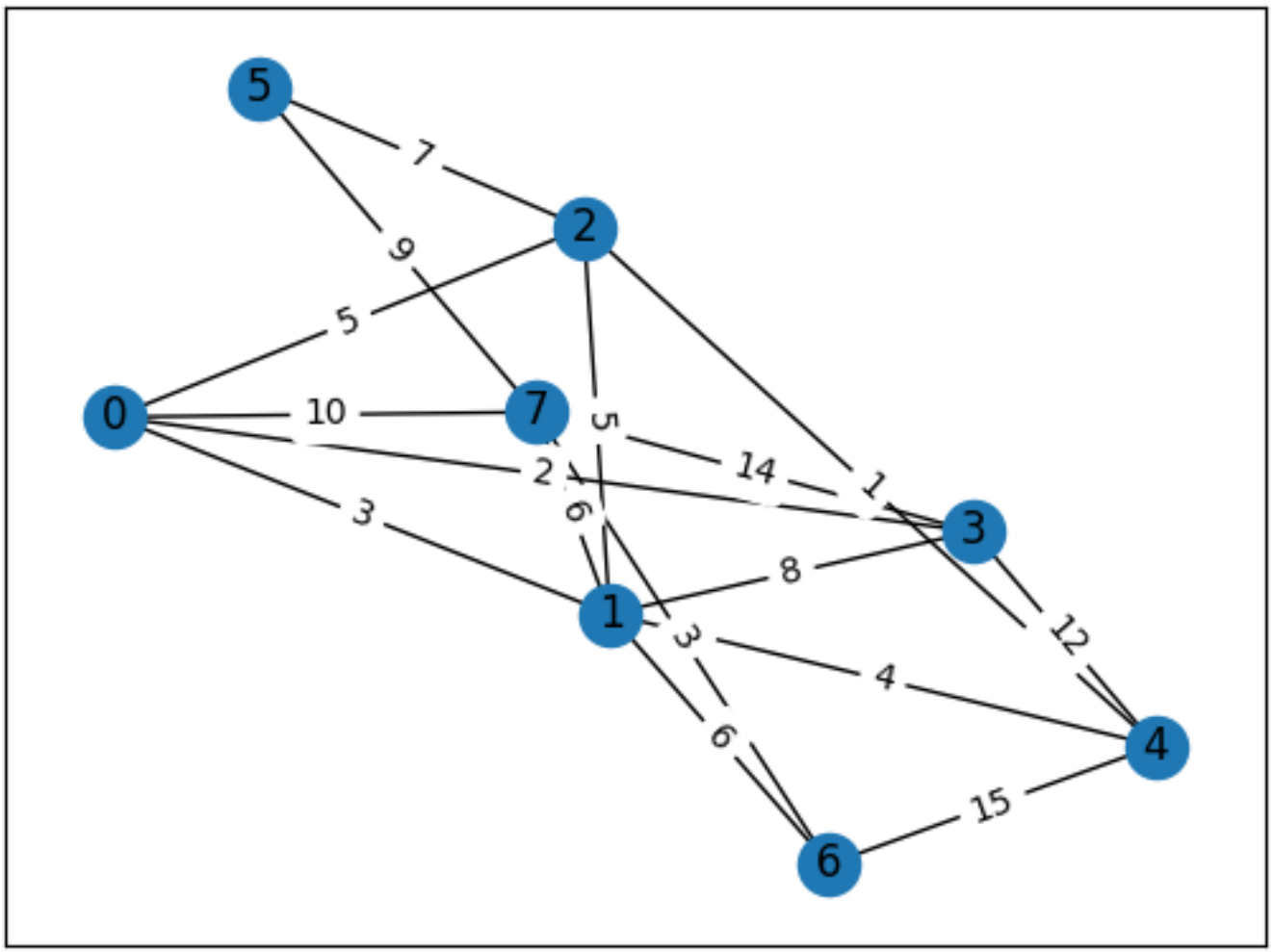
Ejercicio 12: A

```python
#Matriz de adyacencia
A = nx.adjacency_matrix(G)
print("Matriz de adyacencia")
print(A.todense(),"\n")

#Matriz de incidencia
print("Matriz de incidencia")
IM = abs(nx.incidence_matrix(G, oriented=True))
print(IM.toarray(),"\n")

#Lista de adyacencia
print("Lista de adyacencia")
for n,nbrdict in G.adjacency():
    print(str(n)+" "+str(nbrdict))
```

Ejercicio 12: A

```
#Mostrar pesos
pos=nx.spring_layout(G)
nx.draw_networkx(G,pos)
labels = nx.get_edge_attributes(G,'weight')
nx.draw_networkx_edge_labels(G,pos,edge_labels=labels)
plt.show()

#No mostrar pesos
# nx.draw(G, with_labels=True)
# plt.show()
```

Ejercicio 12: A



Gracias por su atención

Dudas o consultas:

pgutierrez2018@udec.cl