Tarea 1

Jesus Angel Patlán Castillo

12 de febrero de 2019

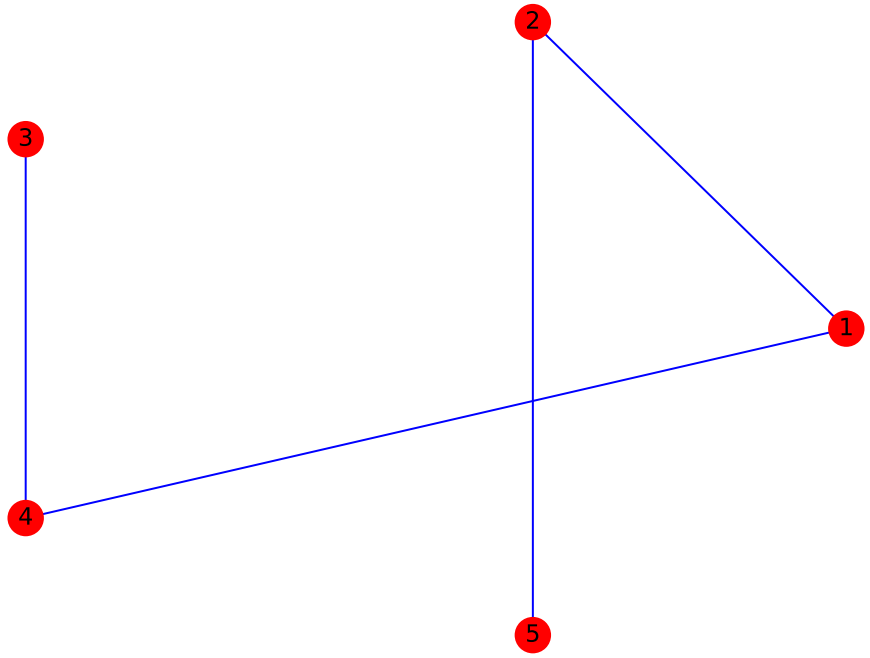
En esta tarea se recopila información de grafos con distintas propiedades. Se utiliza código de Python [1], con la librería NetworkX [3] para la generación de grafos y la librería Matplotlib [4] para guardar el grafo en el formato ".eps". El código empleado se obtuvo consultando la documentación oficial de la librería NetworkX [2] y guías suplementarias [6] [10]. Las imágenes y el código se encuentran disponibles directamente en mi repositorio [5].

1. Grafo simple no dirigido acíclico

Una red de ciudades en un estado puede ser un ejemplo de un grafo simple no dirigido acíclico, ya que podemos representar cada ciudad como un nodo del grafo, y tomar en cuenta que cada arista representan las carreteras que conectan cada ciudad, considerando que sería un grafo no dirigido dado que una carretera puede ir de ida y vuelta entre ciudades, y es acíclico puesto que no se consideran carreteras que van de una ciudad a sí misma [8]. La figura 1 representa un ejemplo de este tipo de grafo.

```
import networkx as nx
import matplotlib.pyplot as plt


#Creamos una instancia de un grafo
G = nx.Graph()

#Agregamos las aristas en el nodo (junto con los nodos)
G.add_edge(1,2)
G.add_edge(1,4)
G.add_edge(3,4)
G.add_edge(2,5)

#Dibujamos el grafo
nx.draw(G, pos=nx.circular_layout(G), node_color='r', edge_color='b
    ', with_labels=True)
#Mostramos el grafo en pantalla
plt.show()
```

Figura 1: Grafo simple no dirigido acíclico

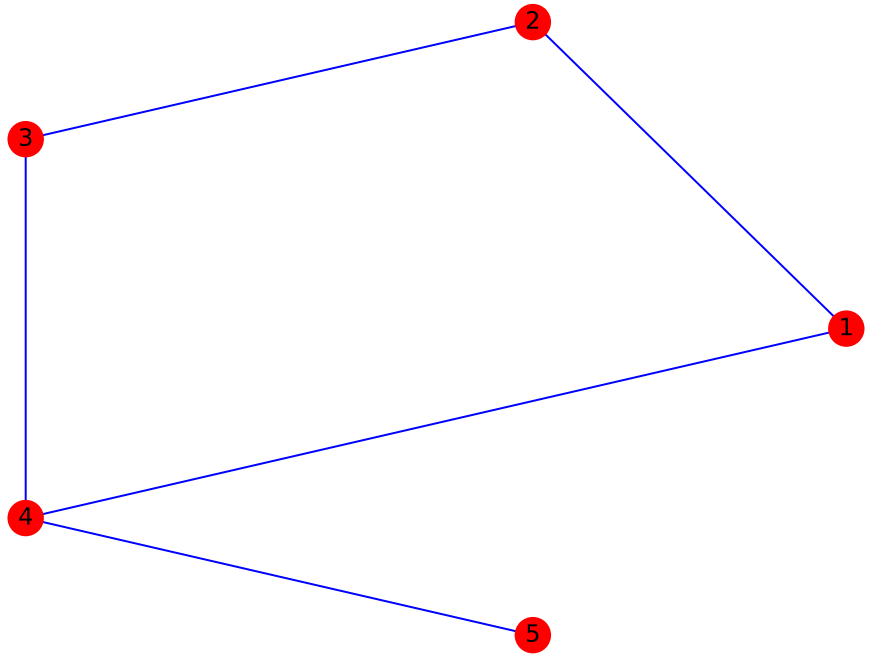


2. Grafo simple no dirigido cíclico

Una ruta de autobús puede ser representada por este tipo de grafos, en donde los autobuses pasan a través de las calles y avenidas (representadas con las aristas), y cada nodo del grafo se puede representar una parada donde se recogen personas. La figura 2 representa un ejemplo de este tipo de grafo.

```
import networkx as nx
import matplotlib.pyplot as plt


#Creamos una instancia de un grafo
G = nx.Graph()

#Agregamos las aristas en el nodo (junto con los nodos)
G.add_edge(1,2)
G.add_edge(2,3)
G.add_edge(3,4)
G.add_edge(4,1)
G.add_edge(4,5)

#Dibujamos el grafo
nx.draw(G, pos=nx.circular_layout(G), node_color='r', edge_color='b
    ', with_labels=True)
```

Figura 2: Grafo simple no dirigido cíclico



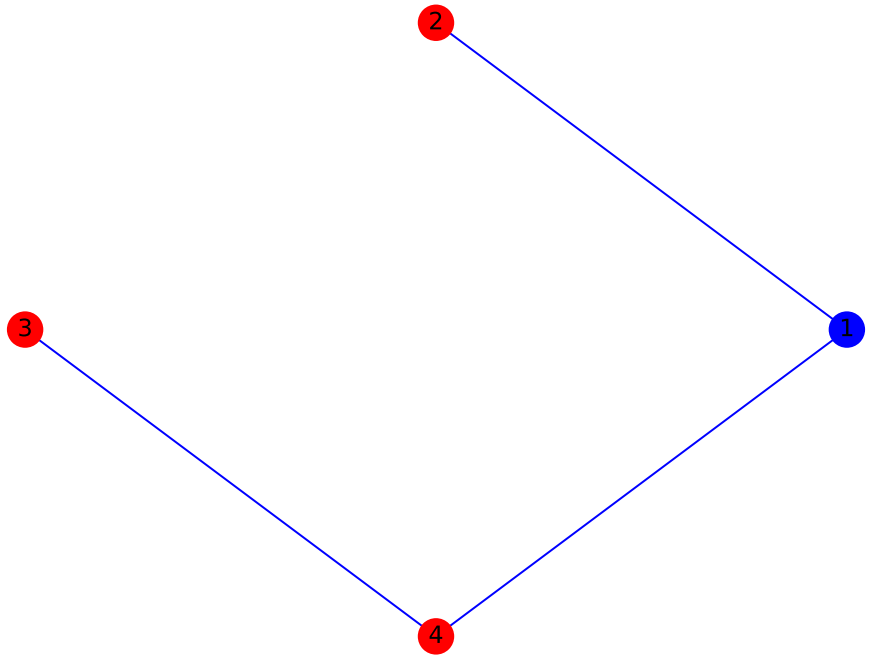

```
#Mostramos el grafo en pantalla
plt.show()
```

3. Grafo simple no dirigido reflexivo

Podemos representar una red de sistemas informáticos por medio de un grafo reflexivo, donde cada computadora es representada por medio de un nodo, y la conexión hacia las demás computadoras son representadas por las aristas. La arista reflexiva representaría la conexión de una computadora consigo misma [7]. La figura 3 representa un ejemplo de este tipo de grafo.

```
import networkx as nx
import matplotlib.pyplot as plt


#Creamos una instancia de un grafo
G = nx.Graph()

#Agregamos las aristas en el nodo (junto con los nodos)
G.add_edge(1,2)
G.add_edge(1,1)
G.add_edge(1,4)
```

Figura 3: Grafo simple no dirigido reflexivo (El nodo 1 tiene una arista reflexiva)



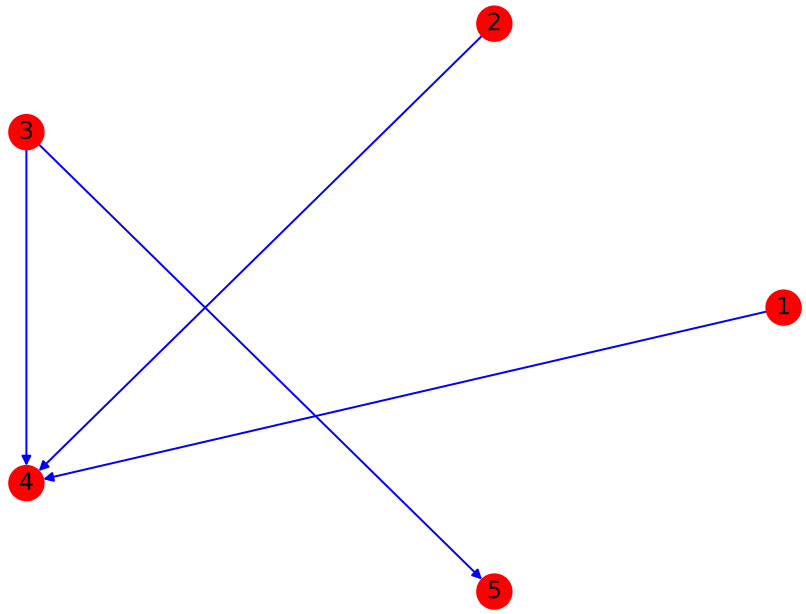

```
G.add_edge(1,5)
G.add_edge(3,4)

#Dibujamos el grafo
nx.draw(G, pos=nx.circular_layout(G), node_color=['b','r','r','r','r'], edge_color='b', with_labels=True)
#Mostramos el grafo en pantalla
plt.show()
```

4. Grafo simple dirigido acíclico

En la Programación Orientada a Objetos es común realizar herencia entre clases, y esta puede ser representada por medio de un grafo dirigido acíclico, teniendo a cada nodo como una clase distinta, y señalando la herencia por medio de una arista dirigida a las clases hijas [9]. La figura 4 representa un ejemplo de este tipo de grafo.

```
import networkx as nx
import matplotlib.pyplot as plt


#Creamos una instancia de un grafo
```

Figura 4: Grafo simple dirigido acíclico

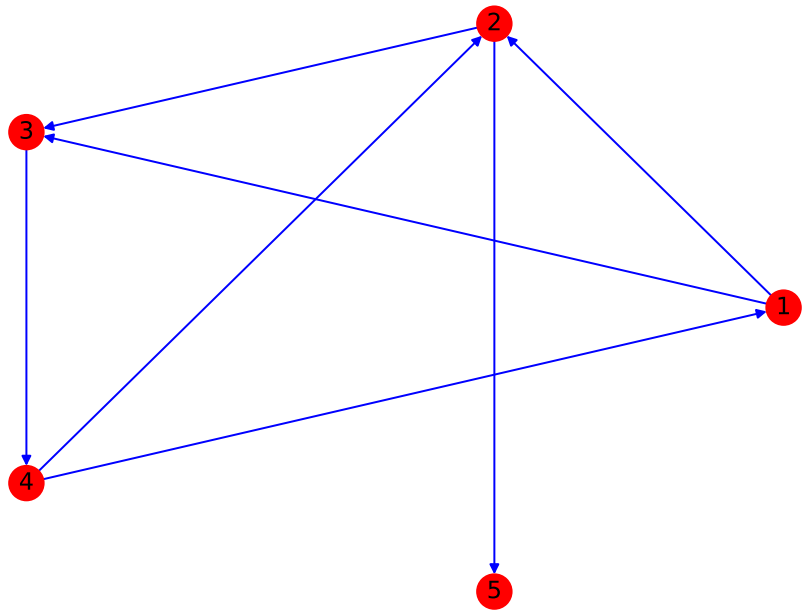



```
G = nx.DiGraph()

#Agregamos las aristas en el nodo (junto con los nodos)
G.add_edge(1,4)
G.add_edge(2,4)
G.add_edge(3,4)
G.add_edge(3,5)

#Dibujamos el grafo
nx.draw(G, pos=nx.circular_layout(G), node_color='r', edge_color='b
    ', with_labels=True)
#Mostramos el grafo en pantalla
plt.show()
```

5. Grafo simple dirigido cíclico

En algunos juegos se tienen distintos estados en los que el jugador se puede encontrar, y esto puede ser representado en un grafo dirigido cíclico, dado que cada estado se puede visualizar por medio de un nodo, y las aristas representarían las maneras en las que un estado puede cambiar a otro [9]. La figura 5 representa un ejemplo de este tipo de grafo.

Figura 5: Grafo simple dirigido cíclico

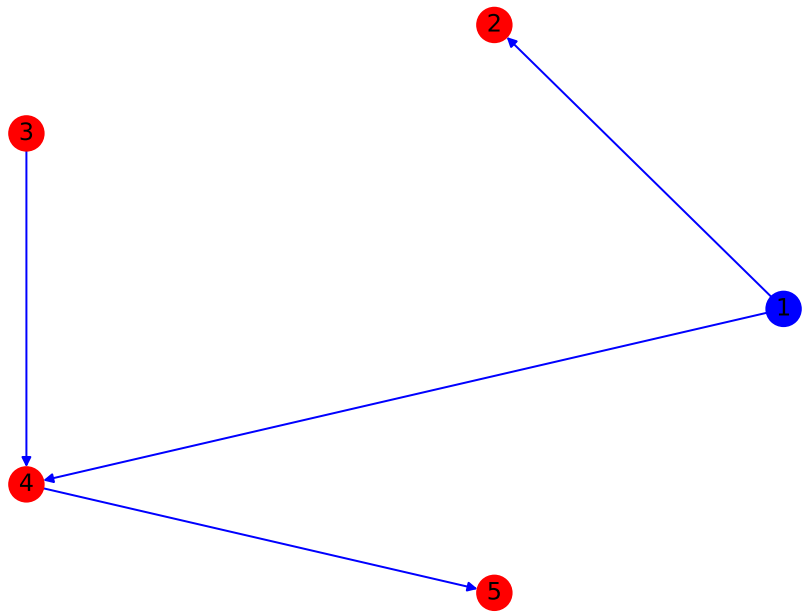



```python
import networkx as nx
import matplotlib.pyplot as plt


#Creamos una instancia de un grafo
G = nx.DiGraph()

#Agregamos las aristas en el nodo (junto con los nodos)
G.add_edge(1,2)
G.add_edge(2,3)
G.add_edge(2,5)
G.add_edge(1,3)
G.add_edge(3,4)
G.add_edge(4,2)
G.add_edge(4,1)

#Dibujamos el grafo
nx.draw(G, pos=nx.circular_layout(G), node_color='r', edge_color='b
    ', with_labels=True)
#Mostramos el grafo en pantalla
plt.show()
```

6. Grafo simple dirigido reflexivo

Se puede realizar un grafo dirigido reflexivo para representar todos los hipervínculos (aristas) que dirigen a una página web (nodos) en los que el usuario puede navegar a través de una web, cuando una página web se redirige a sí misma se utiliza una arista reflexiva [9]. La figura 6 representa un ejemplo de este tipo de grafo.

Figura 6: Grafo simple dirigido reflexivo (El nodo 1 tiene una arista reflexiva)



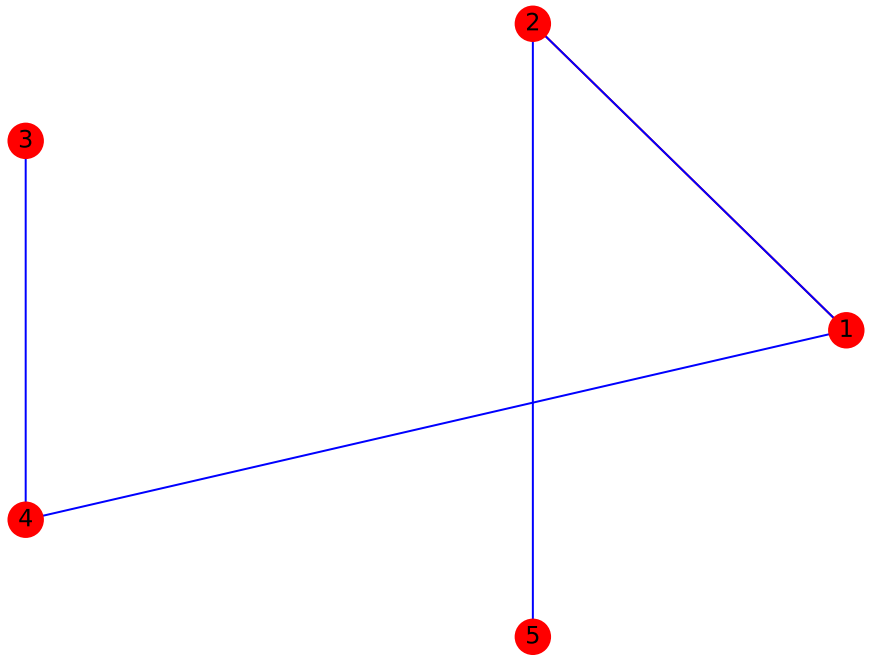

```
import networkx as nx
import matplotlib.pyplot as plt


#Creamos una instancia de un grafo
G = nx.DiGraph()

#Agregamos las aristas en el nodo (junto con los nodos)
G.add_edge(1,2)
G.add_edge(1,1)
G.add_edge(1,4)
G.add_edge(3,4)
G.add_edge(4,5)

#Dibujamos el grafo
```

```
nx.draw(G, pos=nx.circular_layout(G), node_color=['b','r','r','r','r'], edge_color='b', with_labels=True)
#Mostramos el grafo en pantalla
plt.show()
```

7. Multigrafo no dirigido acíclico

De manera similar al ejemplo de un grafo simple no dirigido acíclico, una red de ciudades puede ser representado por un multigrafo, el cual puede proporcionar más información que un grafo simple al añadir diversos caminos por el que se puede trasladar de un nodo a otro. La figura 7 representa un ejemplo de este tipo de grafo.

Figura 7: Multigrafo no dirigido acíclico (Los nodos 1 y 2 tienen múltiples aristas)



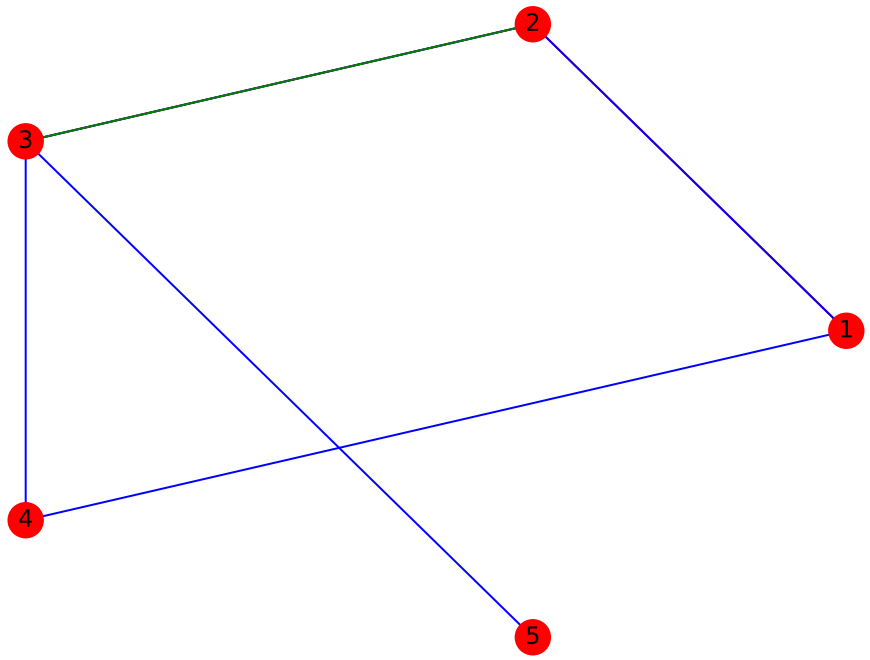

```
import networkx as nx
import matplotlib.pyplot as plt


#Creamos una instancia de un multigrafo
G = nx.MultiGraph()

```

```
#Agregamos las aristas en el nodo (junto con los nodos)
G.add_edge(1,2)
G.add_edge(1,2)
G.add_edge(1,4)
G.add_edge(3,4)
G.add_edge(2,5)
#Dibujamos el grafo
nx.draw(G, pos=nx.circular_layout(G), node_color=['r','r','r','r','r'], edge_color=['r','b','b','b','b'], with_labels=True)
#Mostramos el grafo en pantalla
plt.show()
```

8. Multigrafo no dirigido cíclico

Considerando el ejemplo de la ruta de autobuses, podemos tener entre dos paradas (nodos) múltiples rutas para llegar a la parada siguiente. La figura 8 representa un ejemplo de este tipo de grafo.

Figura 8: Multigrafo no dirigido cíclico (Los nodos 1, 2 y 3 tienen múltiples aristas)




```
import networkx as nx
import matplotlib.pyplot as plt

```

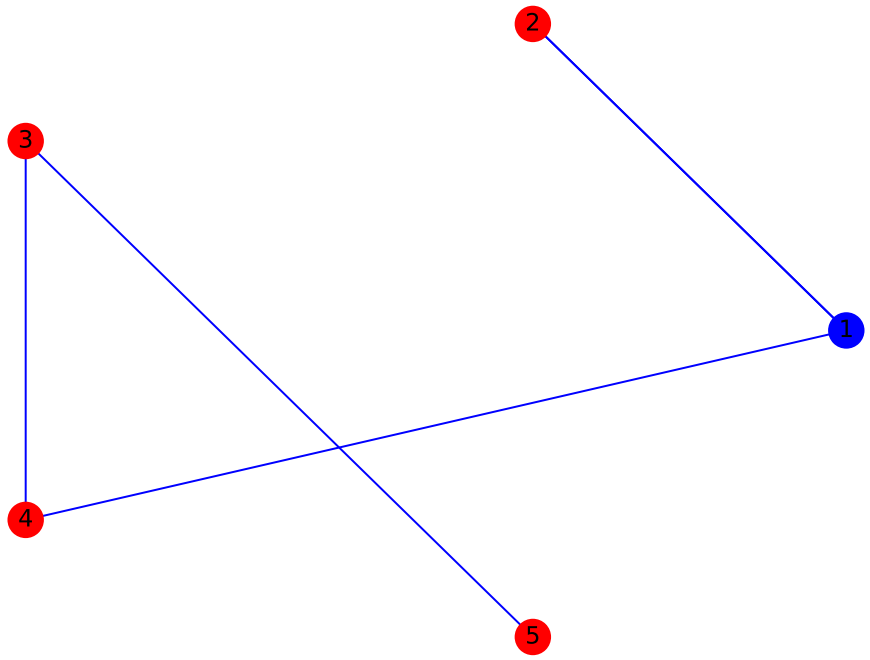
```
4
5 #Creamos una instancia de un multigrafo
6 G = nx.MultiGraph()
7
8 #Agregamos las aristas en el nodo (junto con los nodos)
9 G.add_edge(1,2)
10 G.add_edge(1,2)
11 G.add_edge(2,3)
12 G.add_edge(2,3)
13 G.add_edge(2,3)
14 G.add_edge(3,4)
15 G.add_edge(3,5)
16 G.add_edge(4,1)
17
18 #Dibujamos el grafo
19 nx.draw(G, pos=nx.circular_layout(G), nodecolor='r', edge_color=['r
       ','b','b','r','b','g','b','b'], with_labels=True)
20 #Mostramos el grafo en pantalla
21 plt.show()
```

9. Multigrafo no dirigido reflexivo

En las redes sociales, se puede utilizar un multigrafo reflexivo para representar las menciones que se hacen entre usuarios por medio de publicaciones, donde cada nodo representa un usuario y cada publicación representa la arista con la que conecta el usuario que realizo la publicación con el que es mencionado. La figura 9 representa un ejemplo de este tipo de grafo.

```
1 import networkx as nx
2 import matplotlib.pyplot as plt
3
4
5 #Creamos una instancia de un grafo
6 G = nx.MultiGraph()
7
8 #Agregamos las aristas en el nodo (junto con los nodos)
9 G.add_edge(1,2)
10 G.add_edge(1,2)
11 G.add_edge(1,1)
12 G.add_edge(1,1)
13 G.add_edge(1,4)
14 G.add_edge(3,4)
15 G.add_edge(3,5)
16
17 #Dibujamos el grafo
18 nx.draw(G, pos=nx.circular_layout(G), node_color=['b','r','r','r','
       r'], edge_color=['r','b','b','b','b','b','b'], with_labels=True
       )
19 #Mostramos el grafo en pantalla
20 plt.show()
```

Figura 9: Multigrafo no dirigido reflexivo (Los nodos 1 y 2 tienen múltiples aristas, el nodo 1 tiene una arista reflexiva)

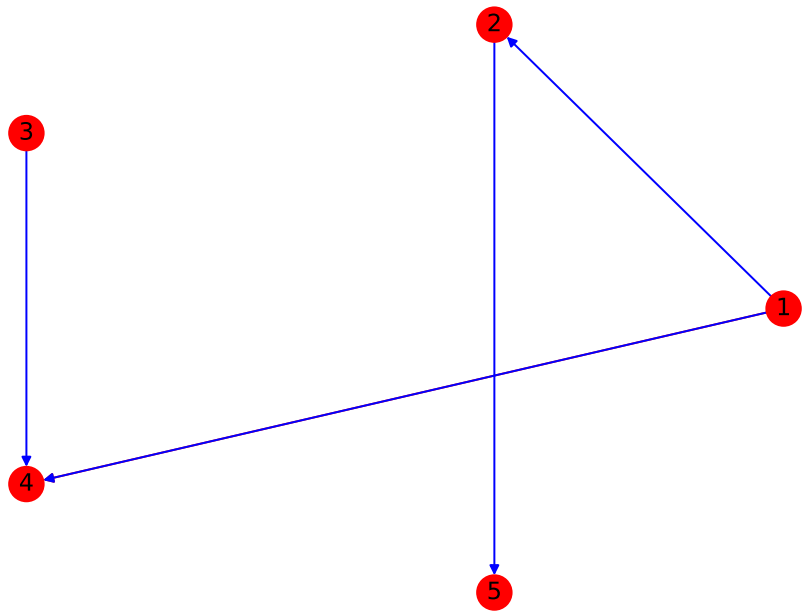


10. Multigrafo dirigido acíclico

Con un multigrafo dirigido acíclico es posible representar el flujo de dinero entre cuentas bancarias, tomando las cuentas bancarias como los nodos del grafo y las aristas el método de traspaso de una cuenta a otra. La figura 10 representa un ejemplo de este tipo de grafo.

```
import networkx as nx
import matplotlib.pyplot as plt


#Creamos una instancia de un multigrafo dirigido
G = nx.MultiDiGraph()

#Agregamos las aristas en el nodo (junto con los nodos)
G.add_edge(1,2)
G.add_edge(1,4)
G.add_edge(1,4)
G.add_edge(3,4)
G.add_edge(2,5)

#Dibujamos el grafo
```

Figura 10: Multigrafo dirigido acíclico (Los nodos 1 y 4 tienen múltiples aristas)



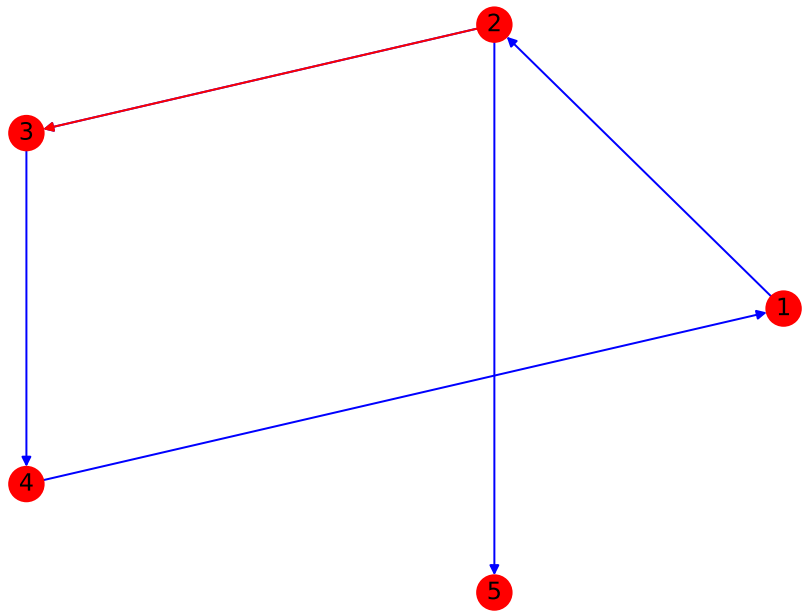

```
nx.draw(G, pos=nx.circular_layout(G), nodecolor='r', edge_color=['b
    ','r','b','b','b'], with_labels=True)
#Mostramos el grafo en pantalla
plt.show()
```

11. Multigrafo dirigido cíclico

Tomando el ejemplo utilizado en los grafos simples dirigidos cíclicos, es posible que haya diferentes maneras de pasar de un estado a otro, lo cual es representado por aristas múltiples entre los nodos. La figura 11 representa un ejemplo de este tipo de grafo.

```
import networkx as nx
import matplotlib.pyplot as plt


#Creamos una instancia de un multigrafo
G = nx.MultiDiGraph()

#Agregamos las aristas en el nodo (junto con los nodos)
G.add_edge(1,2)
G.add_edge(2,3)
```

Figura 11: Multigrafo dirigido cíclico (Los nodos 2 y 3 tienen múltiples aristas)



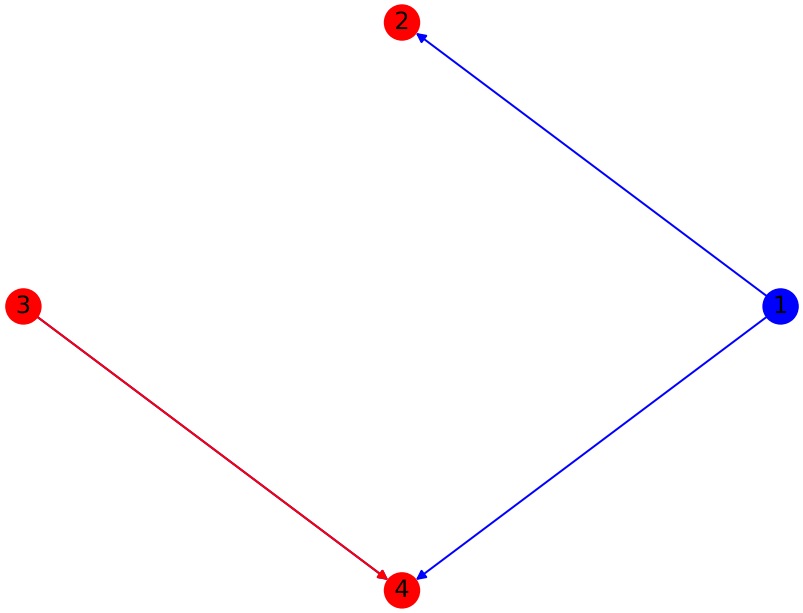

```
G.add_edge(2,5)
G.add_edge(2,3)
G.add_edge(3,4)
G.add_edge(4,1)

#Dibujamos el grafo
nx.draw(G, pos=nx.circular_layout(G), nodecolor='r', edge_color=['b
    ','b','r','b','b','b'], with_labels=True)
#Mostramos el grafo en pantalla
plt.show()
```

12. Multigrafo dirigido reflexivo

Como en el grafo simple dirigido reflexivo, en una página web se pueden tener múltiples hipervínculos que te redirigen a una misma página web. La figura 12 representa un ejemplo de este tipo de grafo.

```
import networkx as nx
import matplotlib.pyplot as plt


#Creamos una instancia de un multigrafo
```

Figura 12: Multigrafo dirigido reflexivo (Los nodos 3 y 4 tienen múltiples aristas, el nodo 1 tiene una arista reflexiva)




```
G = nx.MultiDiGraph()

#Agregamos las aristas en el nodo (junto con los nodos)
G.add_edge(1,2)
G.add_edge(1,1)
G.add_edge(1,4)
G.add_edge(3,4)
G.add_edge(3,4)

#Dibujamos el grafo
nx.draw(G, pos=nx.circular_layout(G), node_color=['b','r','r','r'],
     edge_color=['b','b','b','b','r'], with_labels=True)
#Mostramos el grafo en pantalla
plt.show()
```

Referencias

[1] Python Software Foundation Versión 3.7.2. https://www.python.org/. Copyright ©2001-2019.

[2] NetworkX developers con última actualización el 19 de Septiembre 2018. https://networkx.github.io/documentation/stable/index.html. Copyright ©2004-2018.

[3] NetworkX developers Versión 2.0. https://networkx.github.io/. Copyright ©2004-2018.

[4] The Matplotlib development team Versión 3.0.2. https://matplotlib.org/. Copyright @2002-2012.

[5] Repositorio Optimización Flujo en Redes de Jesús Angel Patlán Castillo. https://github.com/JAPatlanC/Flujo-Redes.

[6] Plotting NetworkX graph in Python Pregunta en Stackoverflow. https://stackoverflow.com/questions/44692644/plotting-networkx-graph-in-python. Copyright ©2019.

[7] What is the application of reflexive graph Pregunta en Quora. https://www.quora.com/What-is-the-application-of-reflexive-graph.

[8] E. Novo and A. Méndez Alonso. Aplicaciones de la teoría de grafos a algunos juegos de estrategia. *Suma*, 46:31–35, 2004.

[9] Example of Digraphs Applications Oxford Math Center. http://www.oxfordmathcenter.com/drupal7/node/678.

[10] How to set colors for nodes in NetworkX Pregunta en Stackoverflow. https://stackoverflow.com/questions/27030473/how-to-set-colors-for-nodes-in-networkx-python.