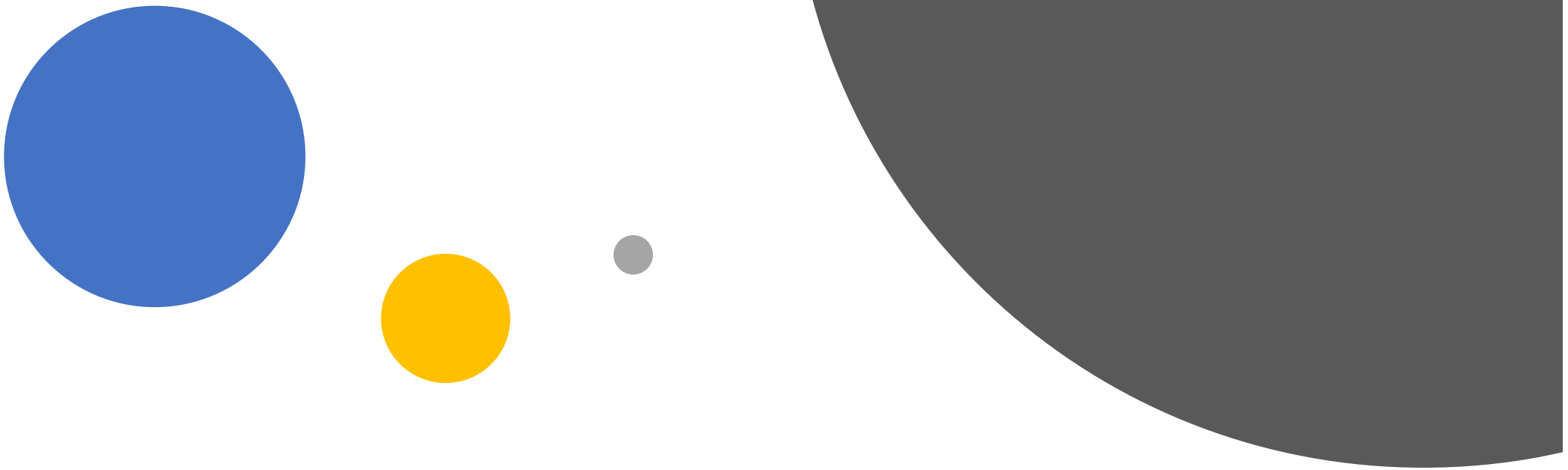


Algoritmos y Programación

Práctica 3.1: Networkx
Creación de un grafo dirigido

NetworkX

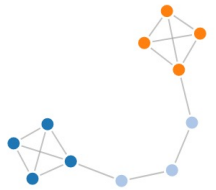
La práctica de esta semana tiene tres ejercicios.

En el primer ejercicio se debe crear un grafo **dirigido con pesos** a partir de un fichero de entrada. Las principales funciones a utilizar son:

- nx.DiGraph()
- G.add_node()
- G.add_edge()

https://networkx.org/documentation/stable/tutorial.html
https://networkx.org/documentation/stable/reference/index.html

NetworkX is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.

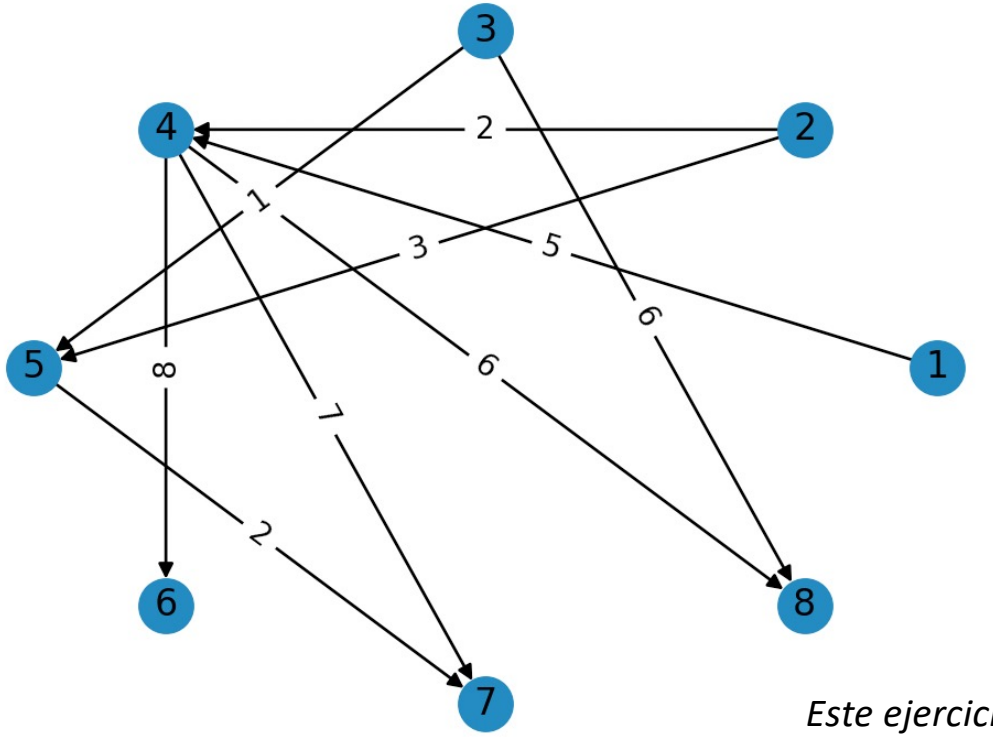


Formato del fichero de entrada

- La primera línea es un descriptor: número de vértices, número de aristas
- El resto de las líneas son las aristas con su peso

Ejemplo:

```
8 9
1 4 5
2 4 2
2 5 3
3 5 1
3 8 6
4 6 8
4 7 7
4 8 6
5 7 2
```



Este ejercicio no puntúa

VPL: Ejercicio 3.1

solve.py

```python
import networkx as nx

def build_digraph_with_weights():
    """
    Read data from the standard input and build the corresponding
    directed graph with weights. Nodes numbering starts with number
    1 (that is, nodes are 1,2,3,...)
    """

    first_line = input().split()
    num_nodes  = int(first_line[0])
    num_edges  = int(first_line[1])

    # Paso 1: Crear grafo direcional con num_nodes


    # Paso 2: Añadir los vértices del grafo


    return graph
```

main.py

```python
import networkx as nx
from solve import *

graph = build_digraph_with_weights()

print("Number of nodes: " + str(graph.number_of_nodes()))
print("Nodes: ", graph.nodes())
print("Number of edges: " + str(graph.number_of_edges()))
print("Edges: ", graph.edges(data=True))

# Paso 3 (Opcional): Utilizando PyCharm añade aqui el código
#         necesario para mostrar gráficamente el grafo.
# ...

```