ALGORITMO BUSQUEDA LOCAL(GREEDY)

Explicación de como resuelve Funciona el algoritmo:

1. Ordena la lista de tuplas en orden descendente por el número de vecinos.
2. Selecciona los k vértices con más vecinos. Si hay empate, selecciona los vértices con más vecinos en orden de aparición en la lista.
3. Cuenta el número de aristas entre los vértices seleccionados.

TIPO DE ALGORITMO

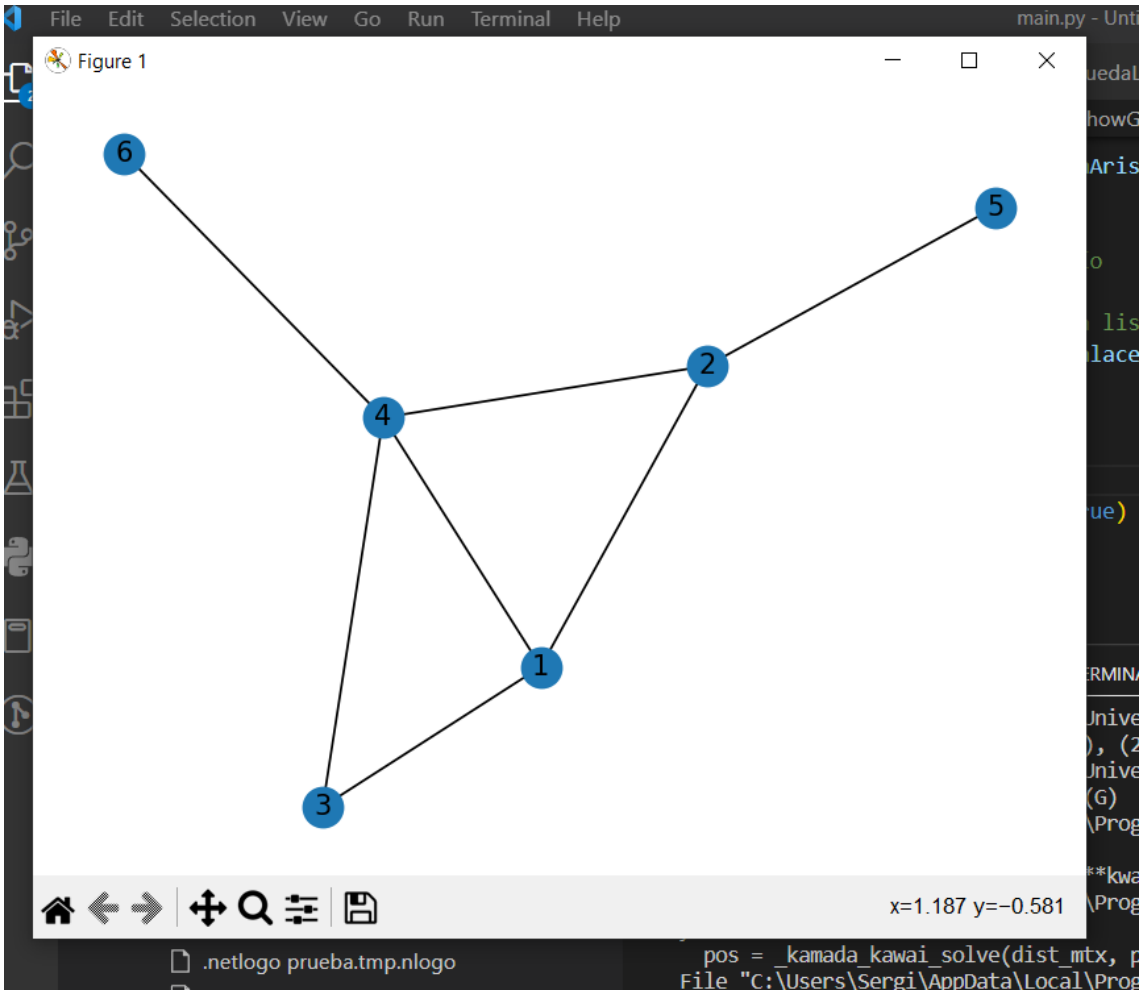
Este algoritmo es una solución heurística para el problema de encontrar un subconjunto de vértices en un grafo que maximice el número de aristas entre ellos. Una heurística es una técnica que utiliza un enfoque aproximado para resolver un problema de manera más rápida que un algoritmo exhaustivo, que examinaría todas las posibles soluciones.
Este algoritmo utiliza un enfoque greedy, es decir, va tomando decisiones locales óptimas en cada paso con la esperanza de llegar a una solución global óptima.
En este caso, la decisión local óptima **es seleccionar en cada paso el vértice con más vecinos**, ya que tiene más posibilidades de tener más aristas con otros vértices.

Pruebas

Probaremos con los datos del ejemplo del .pdf.

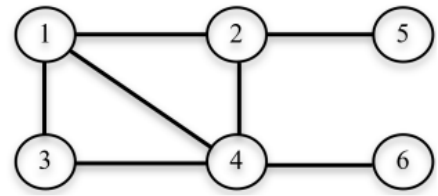
- La solución mas optima debería ser:
  - S2 = {1,2,3,4} -> Number of Edges = 5
- Podemos ver que el grafo del ejemplo es el mismo que el que estamos utilizando mediante la siguiente función:

```python
def showGraph(listaEnlaces):
        # Crea un grafo vacío
    G = nx.Graph()
    # Añade los enlaces a la lista de enlaces del grafo
    G.add_edges_from(listaEnlaces)
    # Dibuja el grafo
    nx.draw(G, with_labels=True)
    # Muestra el gráfico
    plt.show()
```



Example:
Consider a graph with 6 vertices and 7 arcs, represented below (***test.txt*** in Moodle). The image on the right illustrates the graph. The goal is to find a solution of the problem for k=4.

```
k 4
p edge 6 7
e 1 2
e 1 3
e 1 4
e 2 4
e 2 5
e 3 4
e 4 6
```



Consider the three solutions (there will be others):
S1 = {1, 2, 5, 6} - number of edges = 2
S2 = {1, 2, 3, 4} - number of edges = 5
S3 = {2, 3, 5, 6} - number of edges = 1

```
def AlgoritmoGreedy(kValue,vertices,nAristas,listaEnlaces):
```

Le añadimos los valores del ejemplo:

```
AlgoritmoGreedy(4,6,7,[(1, 2), (1, 3), (1, 4), (2, 4), (2, 5), (3, 4), (4, 6)])
```

Y el resultado es:

```
i/Desktop/code/Universidad/Erasmus/IA/p2/Entregable/main.py
[4, 3, 2, 1] -> Number of Edges:  5
PS C:\Users\Sergi\Desktop\code\Universidad\Erasmus\IA>
```

- (Este resultado puede variar dependiendo la iteración del algoritmo):