Out[1]:

Click here to toggle on/off the raw code.

Práctica 1

En esta parte vamos a trabajar con grafos ponderados usando dos EdD:

- Una matriz numpy de adyacencia, donde el elemento i, j indica el peso c_{ij` de la rama (i, j)
- Un diccionario de diccionarios de diccionarios donde las claves del primer diccionario G son índices de nodos, las claves de los diccionarios G[i] son los vértices de su lista de adyacencia y un diccionario G[i] [j] contiene el peso de la rama (i, j).

Por ejemplo, para el gráfico

grafo

el código

l = [[0, 10, 1, np.inf], [np.inf, 0, 1, np.inf], [np.inf, np.inf, 0, 1], [np.inf, 1, np.inf, 0]]

m_g = np.array(l)

generaría su matriz de adyacencia según se describe arriba, mientras que

d_g = { 0: {1: 10, 2:1}, 1: {2: 1}, 2: {3: 1}, 3: {1: 1} }

generaría su descripción como un dict.

```
graph_from_matrix:

( 0 1 ) 10.0
( 0 2 ) 1.0
( 1 2 ) 1.0
( 2 3 ) 1.0
( 3 1 ) 1.0

graph_from_dict:

( 0 1 ) 10
( 0 2 ) 1
( 1 2 ) 1
( 2 3 ) 1
( 3 1 ) 1


[[  0.  46.  inf]
 [ inf   0.  inf]
 [ inf  47.   0.]]
{0: {1: 46.0}, 1: {}, 2: {1: 47.0}}
[[  0.  46.  inf]
 [ inf   0.  inf]
 [ inf  47.   0.]]
```

```
5

true_sparse_factor: 0.750
exp_sparse_factor:  0.711
```

2 Guardando y leyendo grafos

2.1 Guardando y leyendo grafos con pickle

2.2 The Trivial Graph Format

```
graph_from_dict:

( 0 1 ) 10
( 0 2 ) 1
( 1 2 ) 1
( 2 3 ) 1
( 3 1 ) 1

graph_from_dict:

( 0 1 ) 10.0
( 0 2 ) 1.0
( 1 2 ) 1.0
( 2 3 ) 1.0
( 3 1 ) 1.0
```

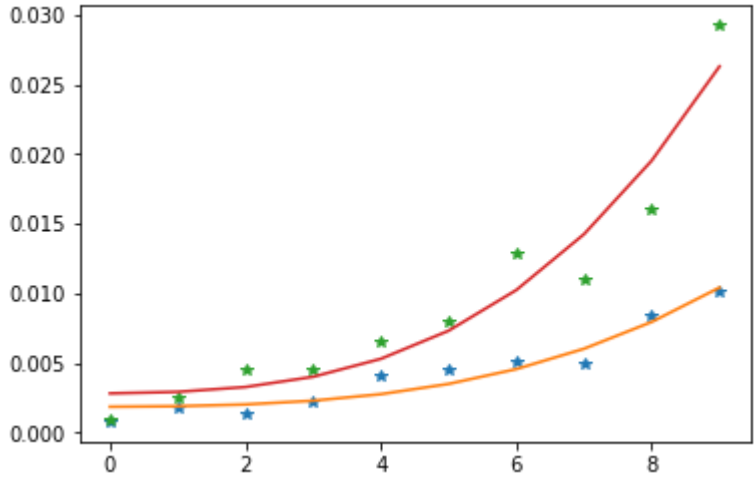
3 Distancias Mínimas en Grafos

3.1 Programming and Timing Dijkstra

```
{0: inf, 1: 1, 2: 2, 3: 0}
 {1: [3, 1], 2: [3, 1, 2], 3: [3]}
{1: 1, 2: 2, 3: 0}
 {1: [3, 1], 2: [3, 1, 2], 3: [3]}
```

3.2 Plotting Dijkstra's Execution Times

Fit below a linear model $An^2logn + B$ to the times in the returned lists and plot the real and fitted times discussing the results.



4 The networkx Library

We are going to use the networkx library to check our Dijkstra results and to get alternative times.

An example of loading a networkx directed graph is to use a list (i, j, w) of (i, j) edges with weights w can be seen in the following cell:

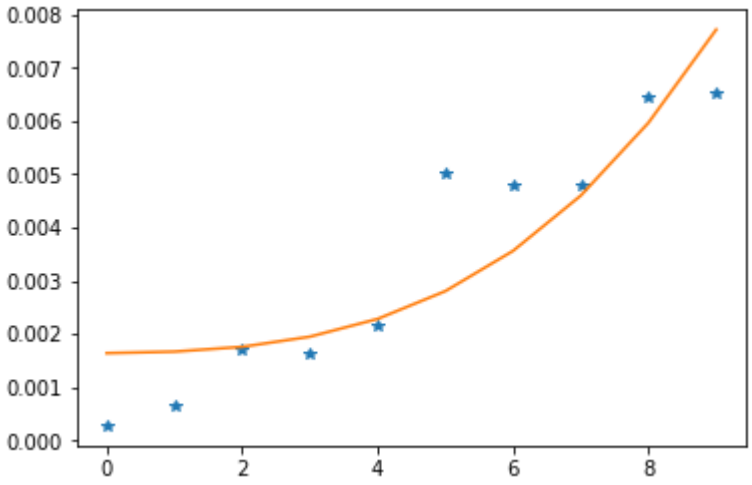
```
( 0 1 ) 10
( 0 2 ) 1
( 1 2 ) 1
( 2 3 ) 1
( 3 1 ) 1
```

```
graph_from_dict:
```

```
( 0 1 ) 10
( 0 2 ) 1
( 1 2 ) 1
( 2 3 ) 1
( 3 1 ) 1
```

Out[15]:

```
{'weight': 10}
```


0.008
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.000
0
2
4
6
8

Cuestiones

Cuestiones sobre guardado de grafos

Añadir código Python y figuras cuando se juzgue necesario.

Cuestión 1:

Describir qué se entiende por serializar un objeto Python.

Serializar un objeto python consiste en guardar dicho objeto en un archivo binario.

Cuestión 2:

json es otro formato de serialiación de objetos. Comentar brevemente posibles diferencias entre pickle y json.

Json se escribe en texto plano y es una representación similar a un diccionario de python con claves y valores. Pickle se escribe en un fichero binario y por tanto, a diferencia de json no es legible para "un humano".

Cuestión 3:

¿Qué ventajas e inconvenientes tendrían las funciones pickle sobre las construidas mediante el formato TGF? Responder algo pertinente y no con lugares comunes.

Una posible ventaja de pickle es que el fichero se guarda en binario y por tanto tanto la lectura como la escritura es más rapida que TGF. El formato TGF nos parece redundante, pues la primera sección del fichero, describe los nodos, mientras que la segunda (a partir de #) vuelve a mostrar los nodos del grafo pero esta vez para describir las ramas. Por otro lado, una ventaja de TGF es que este formato es en texto plano y por tanto legible. Resulta sencillo escribir grafos en este formato para posteriormente leerlos con la función implementada y operar con ellos.

Cuestiones sobre Dijkstra

Cuestión 1:

¿Cuál es el coste teórico del algoritmo de Dijkstra? Justificar brevemente dicho coste.

Sea $G = (V, E)$ un grafo, hemos visto en clase que el coste teórico de Dijkstra es $O(|E| \log_2 |V|)$. El $\log_2 |V|$ se debe al coste de extracción e insercción en la cola de prioridad, ya que esta se construye sobre un heap binario.

Cuestión 2:

Expresar el coste de Dijkstra en función del número de nodos y el sparse factor ρ del grafo en cuestión. Para un número de nodos fijo adecuado, ¿cuál es el creciemiento del coste de Dijkstra en función de ρ?

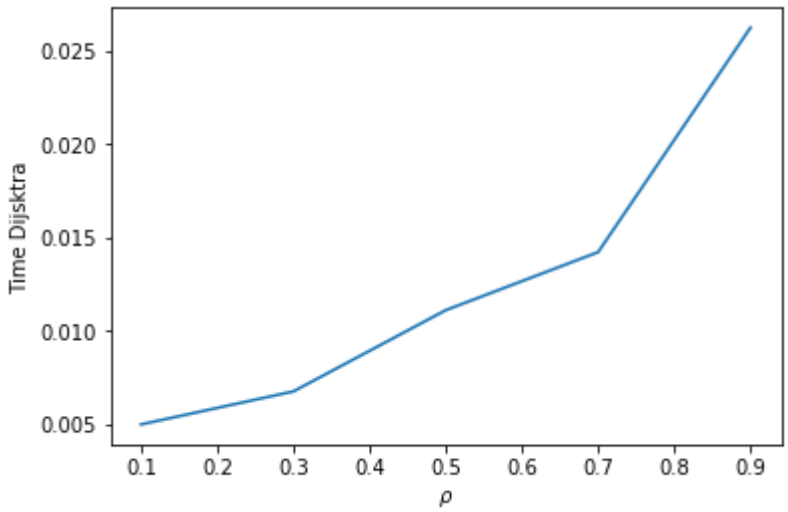
El número de ramas en un grafo dirigido y completo es $|E| = |V|(|V| - 1)$. Si tenemos un sparse factor ρ entonces el número de ramas será $|E| = \rho|V|(|V| - 1)$, es decir $|E| = O(\rho|V|^2)$. Luego el crecimiento de Dijsktra con ρ es lineal. Expresando el coste en función de V, tenemos $O(\rho|V|^2 \log_2 |V|)$.

Ilustrar este crecimiento ejecutando Dijkstra sobre listas de adyacencia de grafos con un número fijo de nodos y sparse factors 0.1, 0.3, 0.5, 0.7, 0.9 y midiendo los correspondientes tiempos de ejecución.

Como se ve en la gráfica de abajo, el coste crece linealmente con ρ.

Out[23]:

```
<matplotlib.text.Text at 0x7ffa08e7a550>
```



Cuestión 3:

¿Cuál es el coste teórico del algoritmo de Dijkstra iterado para encontrar las distancias mínimas entre todos los vértices de un grafo?

Si el coste teórico de Dijsktra para un solo nodo es $O(|E| \log_2 |V|)$, para encontrar las distancias mínimas entre todos los nodos tendremos que aplicar Dijsktra una vez por cada nodo, luego el coste sería $O(|V||E| \log_2 |V|)$.

Cuestiones sobre NetworkX

Responder a las siguientes cuestiones incluyendo gráficas cuando sea necesario.

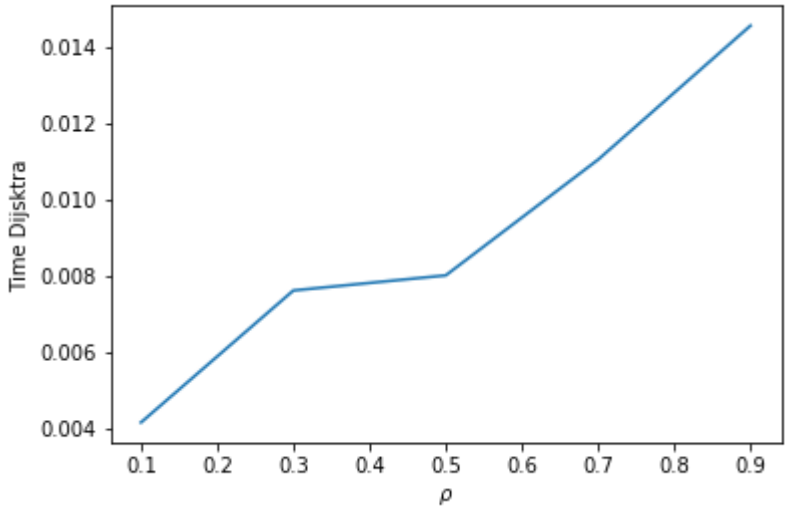
Cuestión 1:

Show graphically the growth of the execution times of Dijkstra's algorthm as a fuction of the number of nodes and the sparsity factor using the NetworkX library.

Work with graphs with 100 nodes and sparse factors 0.1, 0.3, 0.5, 0.7, 0.9.

Out[19]:

```
<matplotlib.text.Text at 0x7ffa08e9ff98>
```



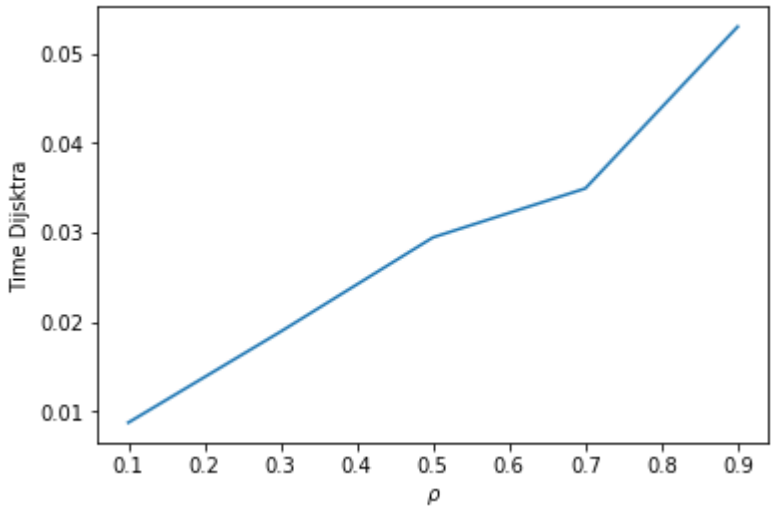
Como ocurre en el caso de matrices y listas de adyacencia con nuestra implementación de Dijsktra, el coste es lineral en sparse factor. La escala en el eje y es mucho mayor que en los casos anteriores porque en este caso hemos usado grafos de 100 nodos y en los anteriores de 10.

Cuestión 2:

Measure and show graphically the execution times of iterated Dijkstra to find the minimum distances between all the vertices of a graph using the NetworkX library and working on graphs with a fixed number of 25 nodes and sparse factors 0.1, 0.3, 0.5, 0.7, 0.9.

Out[21]:

```
<matplotlib.text.Text at 0x7ffa08e6f1d0>
```



De nuevo el crecimiento es lineal. Pese a que usamos grafos de 25 nodos, el cambio de escala se debe a contar el tiempo de aplicar sucesivas veces el algoritmo de Dijsktra, una por cada nodo del grafo.