- Memoria de la Práctica 1 de DAALG -

Andrés Calvente Rodríguez
Carlos González García

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.

En python, entendemos por serializar un objeto al hecho de convertir objetos a una cadena de bytes. El módulo **pickle**, utilizado en esta práctica, es capaz de transformar un objeto complejo a una cadena de bytes y viceversa. El objetivo obvio de querer convertir objetos a cadenas es poder escribirlas luego en un fichero.

Cuestión 2:
json es otro formato de serialización de objetos. Comentar brevemente posibles diferencias entre pickle y json.

La principal diferencia entre la serialización de objetos python usando pickle y usando json es que json permite interoperabilidad (es un formato de datos ampliamente extendido en el sector, de hecho, es un estándar que puede ser utilizado desde programas escritos en otros lenguajes, etc.), mientras que pickle está sólo recomendado para circunstancias en las que no tenemos ningún tipo de requerimiento de interoperabilidad, por ejemplo, simplemente queremos leer y guardar datos desde un programa python.
Otras diferencias más secundarias es que el formato de datos json es legible por un ser humanos. Por otro lado, según algunos estudios [2], la serialización de objetos usando json parecer ser más rápida que mediante pickle.

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 de las ventajas del formato TGF frente a pickle es que permite a un ser humano hacerse a la idea de la composición y estructura de un grafo, mientras que con las cadenas binarias de pickle, no conocemos a simple vista ninguna información acerca del grafo.

Cuestiones sobre Dijkstra

Cuestión 1:
¿Cuál es el coste teórico del algoritmo de Dijkstra? Justificar brevemente dicho coste.

El coste teórico del algoritmo de Dijkstra utilizando una cola de prioridad es:

$$O(|E|log(|V|))$$

donde *E* representa el número de ramas y *V* representa el número de nodos.
Este coste teórico viene derivado de que usando una cola de prioridad pasamos por todas las ramas del grafo, pero nos evitamos pasar por todos los nodos, ya que no pasamos por los nodos que ya hemos visitado.

Cuestión 2:

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

Analizando el coste teórico descrito en la anterior cuestión, vemos que el factor con más peso dentro del coste es el número de ramas, ya que va multiplicando y no está en el logaritmo. Si incrementamos el *sparse factor*, incrementaremos el número de ramas que hay en nuestro grafo, por lo que tendremos un coste mayor. De esta manera, y tal y como podemos ver en la gráfica mostrada debajo, a mayor sparse factor, mayor será el coste del algoritmo. Cabe destacar que, a la vista de la gráfica, vemos que, a tamaños muy pequeños de grafo, apenas afecta el incremento del *sparse factor*, mientras que, a tamaños notablemente mayores, este incremento se hace más que evidente.

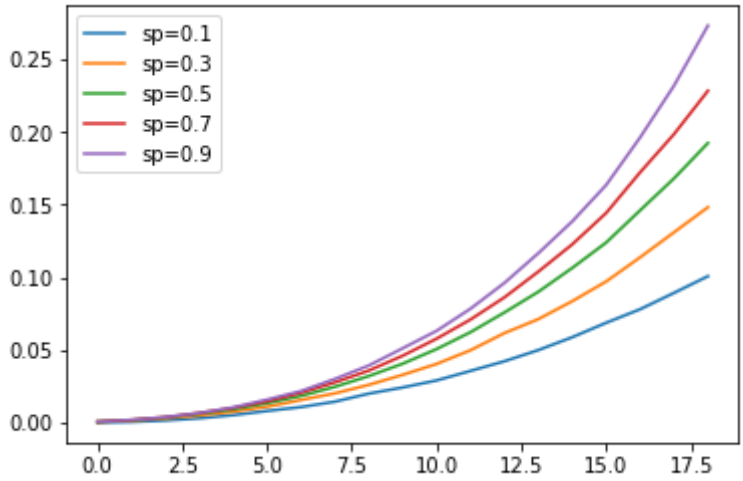


Figura 1.- Gráfica de tiempos de ejecución de Dijkstra para distintos sparse factors.

Cuestión 3:

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

El coste del algoritmo de Dijkstra iterado para encontrar las distancias mínimas entre todos los vértices de un grafo es el mismo que el coste teórico descrito en la cuestión 1, es decir:

$$O(|E|log(|V|))$$

Es el mismo coste debido a que el único cambio que tiene el algoritmo de Dijkstra iterado con respecto al desarrollado en la práctica es la construcción del diccionario de previos, ya que, en lugar de insertar solo el nodo previo en el diccionario, insertamos el previo junto con la lista de nodos previos del nodo previo del nodo actual, de forma que la única diferencia de coste es que ahora hay que insertar una lista en el diccionario, en lugar de

un solo nodo, pero este coste de inserción es despreciable en comparación con el coste teórico del propio algoritmo de Dijkstra.

Cuestión 4:

¿Cómo se podrían recuperar los caminos mínimos si se utiliza Dijkstra iterado?

Tal y como hemos descrito en el apartado anterior, en el diccionario de previos del algoritmo de Dijkstra iterado vamos almacenando la lista de previos del nodo previo al nodo actual, de forma que, al acabar la ejecución del algoritmo, tendremos un diccionario con todos los caminos mínimos ya construidos, y no será necesario llamar a la función *min_paths*, que también hemos desarrollado en esta práctica.

Cuestiones sobre NetworkX

Responder a las siguientes cuestiones incluyendo gráficas cuando sea necesario.

Cuestión 1:

Show graphically the growth of the execution time of Dijkstra's algorithm as a function 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.

Como vemos en la gráfica inferior, la evolución del tiempo de ejecución en función del número de nodos y del *sparse factor* para el caso del algoritmo de Dijkstra de la librería NetworkX es algo diferente al que hemos desarrollado nosotros en la práctica. Sin embargo, podemos ver que se siguen cumpliendo las mismas afirmaciones: a mayor número de nodos y a mayor *sparse factor*, tenemos un tiempo de ejecución mucho mayor. También podemos ver que el tiempo de ejecución tiene un crecimiento logarítmico, aunque menos marcado que en la gráfica anterior (haciendo más pruebas con mayor tamaño de nodos, si se llega a ver un crecimiento logarítmico más marcado).

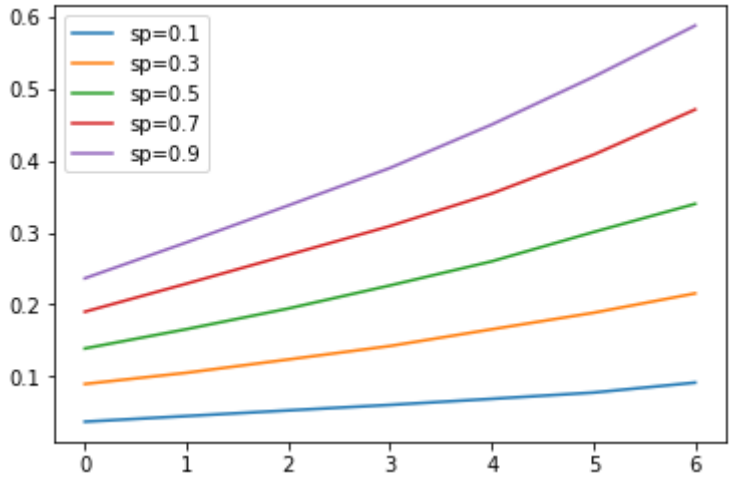
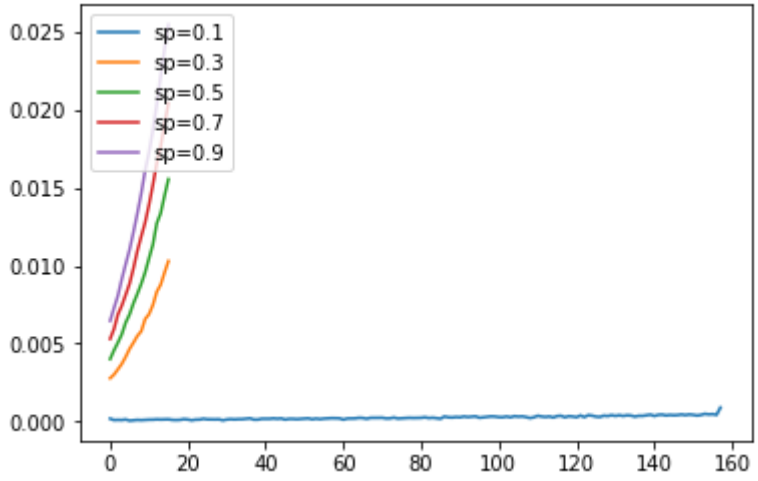


Figura 2.- Gráfica de tiempos de ejecución del algoritmo de Dijkstra de la librería NetworkX.

<u>Cuestión 2:</u>

Measure and show graphically the execution times and 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.

A la vista de la gráfica, vemos que, para el tamaño fijo de 25 nodos, el factor de dispersión de 0.1 da prácticamente un coste lineal, mientras que para los otros factores de dispersión se obtienen costes logarítmicos:



Referencias de consulta

En este apartado se muestran las referencias consultadas para la elaboración de esta memoria.

[1] Página de Moodle de la asignatura.

[2] Pickle vs JSON - Which is faster?
https://konstantin.blog/2010/pickle-vs-json-which-is-faster/

[3] NetworkX manual:
https://networkx.github.io/documentation/latest/_downloads/networkx_reference.pdf