Práctica 1: Python Básico. Min Heaps

Pablo Sánchez Redondo. David Kaack Sánchez

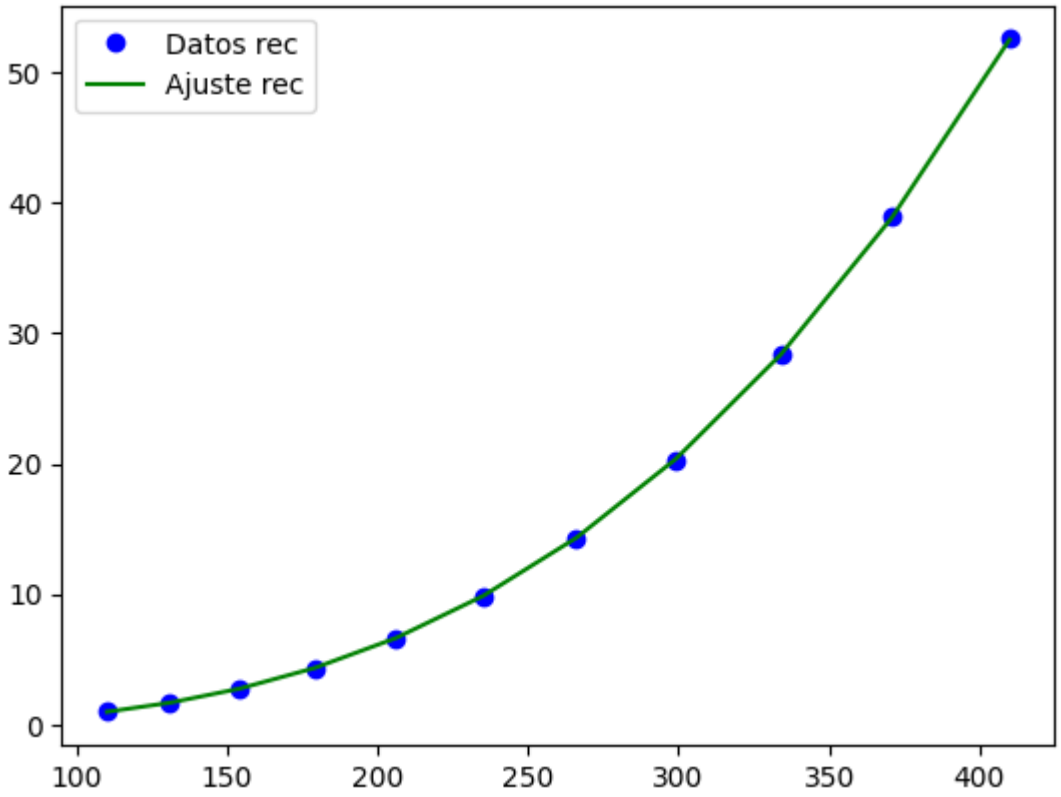
Pareja 03

I. Python Básico

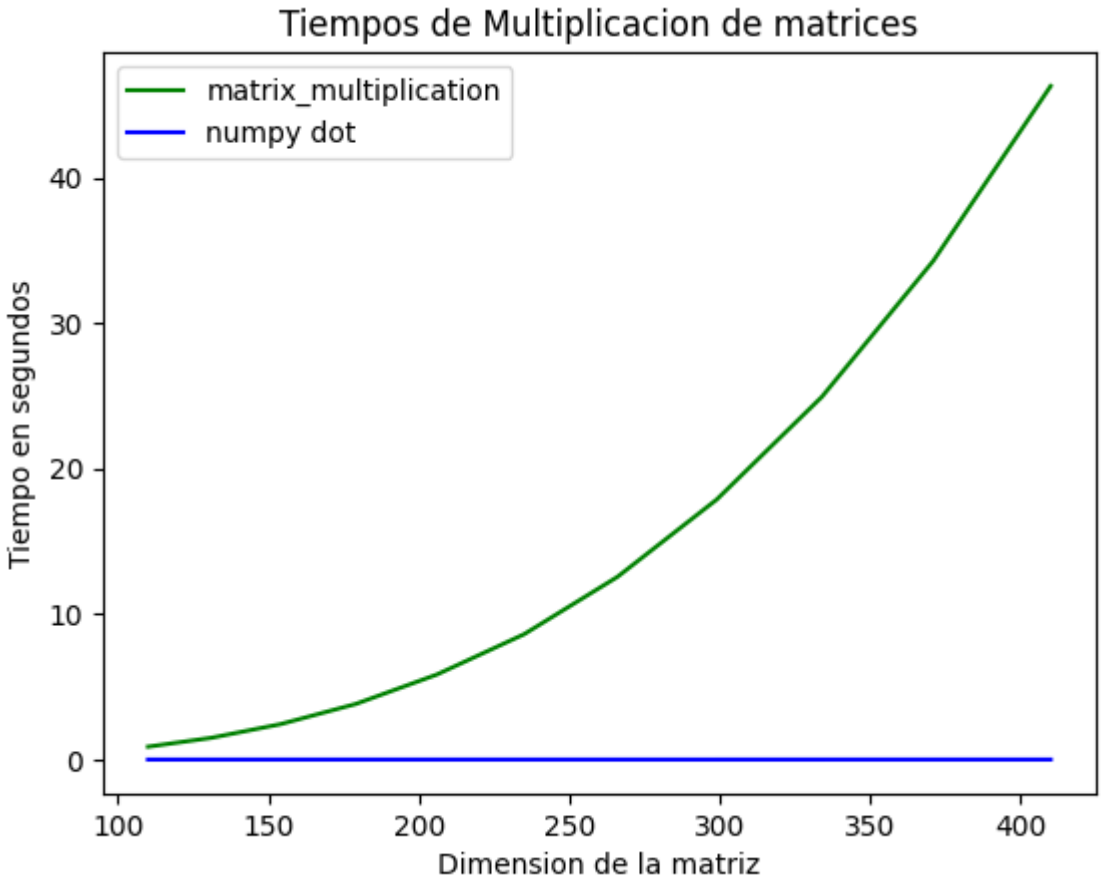
Cuestiones:

1. El algoritmo de multiplicación de matrices que utilizamos tiene un tiempo abstracto de O(n), debido a que realiza una operación dentro de tres bucles *for* anidados.

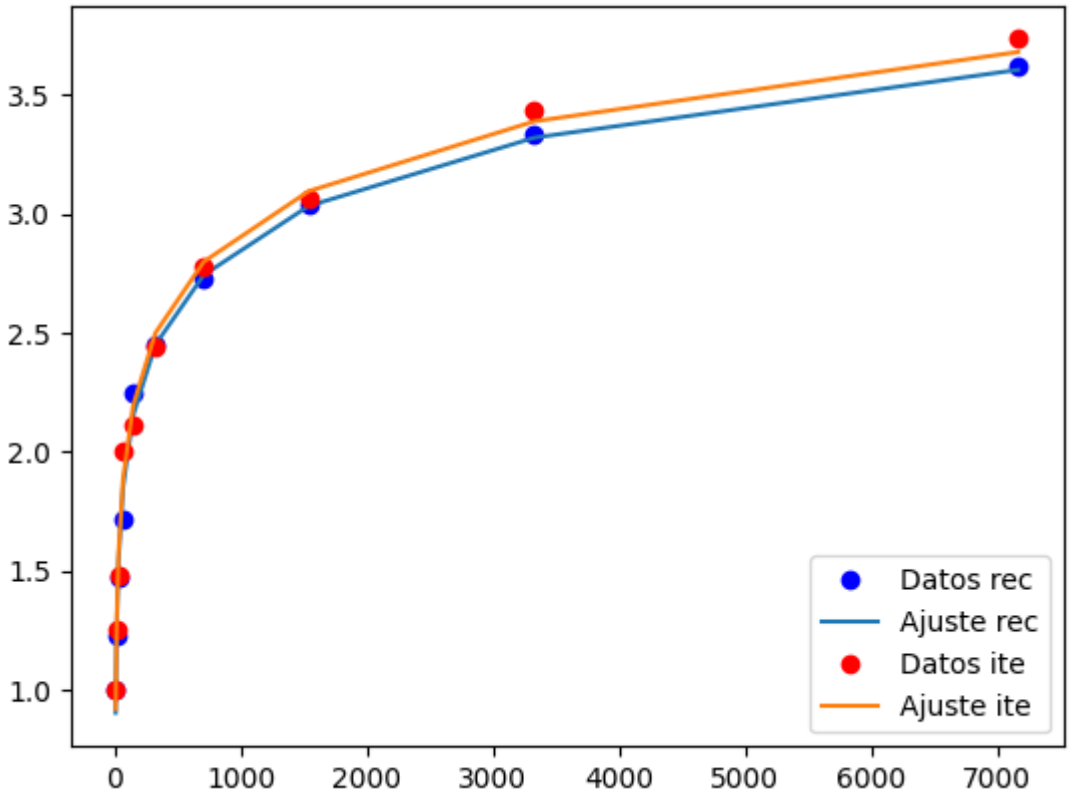
   Para los datos dados, nuestra función ha dado estos resultados. Como se puede observar, los datos son muy similares al ajuste, lo que significa que el tiempo de ejecución abstracto es muy parecido al real.



2. Los tiempos de ejecución de *np.dot* son notablemente más bajos, comparados con los nuestros tiempos son instantáneos. Esto es porque la función *np.dot* usa otro algoritmo que es mucho más eficiente.



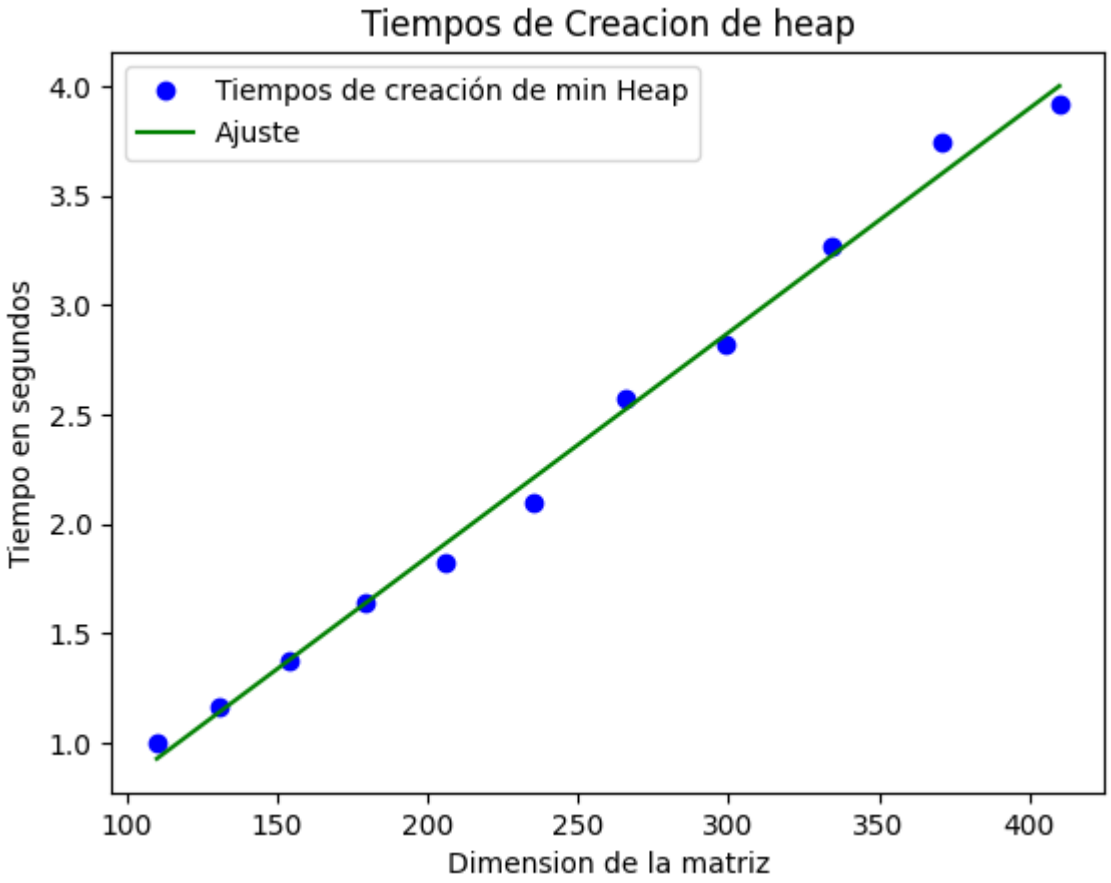
3. En la búsqueda binaria el peor caso es que el elemento que se busque esté al principio o al final de la lista, en el que hace *n log(n)* operaciones hasta encontrarlo.



Dado que son el mismo algoritmo, tienen tiempos de ejecución parecidos, aunque la recursiva sea ligeramente inferior.

II. Min Heaps

1. La creación de un *Min Heap* tiene un coste de tiempo abstracto de O(n). Al recorrer la lista mientras ajustas de abajo a arriba los elementos de la lista, sólo hace falta hacer una pasada por la lista, haciendo un coste lineal, como podemos observar en nuestros resultados.



2. La fase de construcción del heap inicial lleva un tiempo de ejecución de O(n) y la restructuración del mismo por cada elemento es de O(log n). Con lo que lleva a un tiempo de ejecución de O(n log n).
3. Visualizando el tiempo de ordenación por *Heap.*