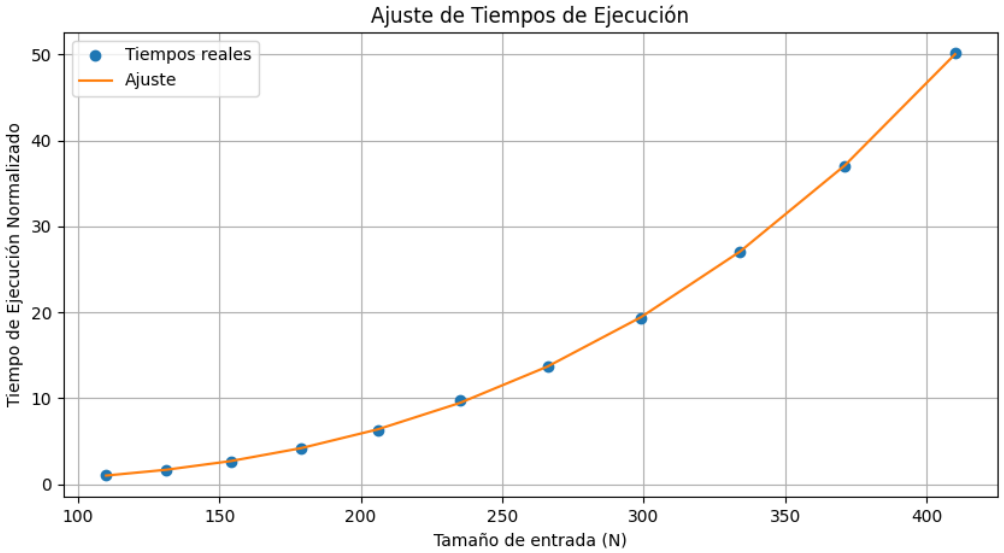
Memoria
Algoritmia y Estructuras de Datos Avanzadas

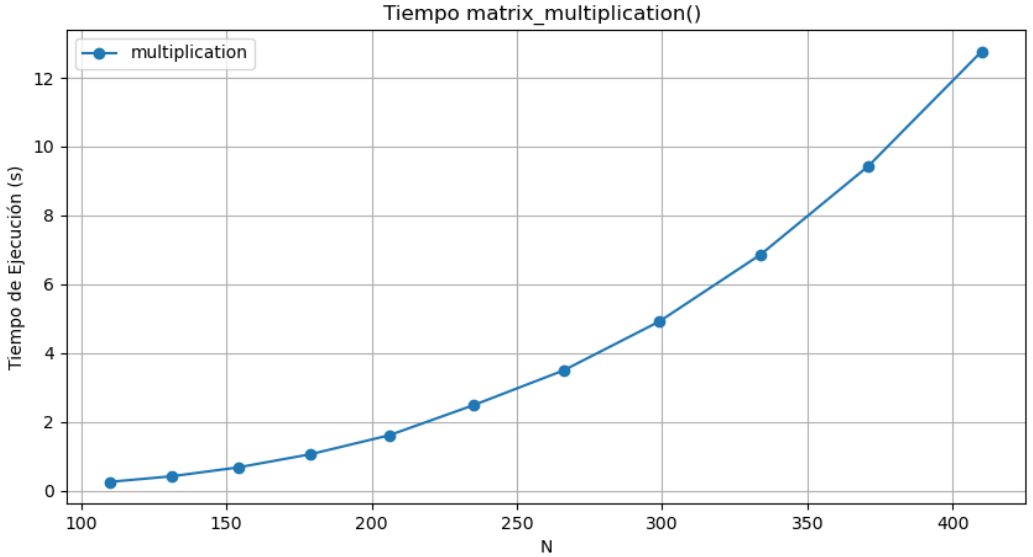
Sergio Hidalgo y Miguel Ibáñez
Grupo: 1262
Pareja: 05

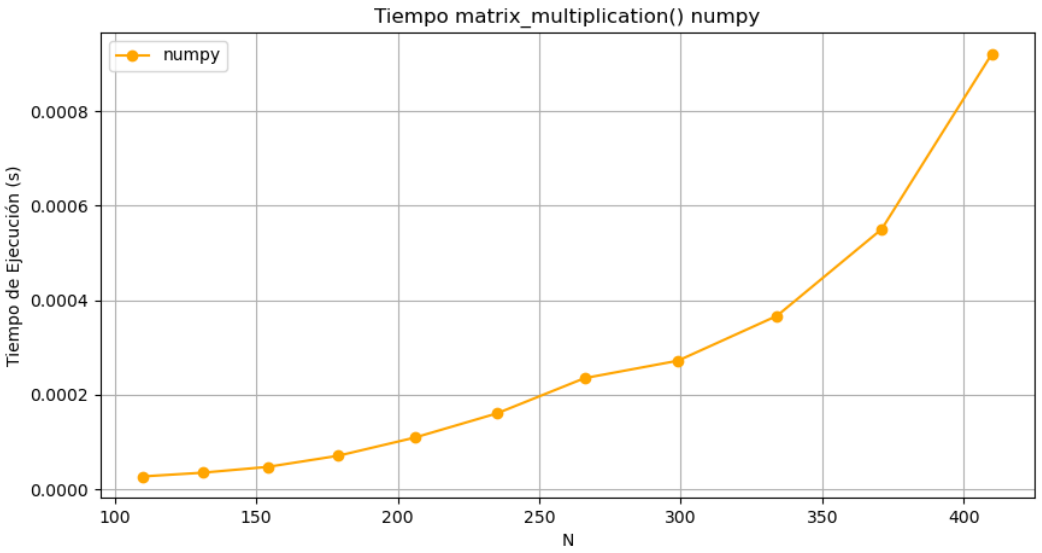
I-C. Cuestiones 1. ¿A qué función f se deberían ajustar los tiempos de ejecución de la función de multiplicación de matrices?
Suponiendo que el tiempo abstracto de ejecución de un algoritmo es f(t), usar el código inferior para ajustar valores de la forma a · f(t) + b a los tiempos reales de ejecución de dicho algoritmo, tiempos que habremos guardado en el array Numpy a_timings , para a continuación dibujar tanto los tiempos como el ajuste calculado por la función, y comentar los resultados.

Se debería ajustar a una función f cúbica (O^3) , ya que se necesitan tres bucles para hacer el recorrido de las matrices.



2. Calcular los tiempos de ejecución que se obtendrían usando la multiplicación de matrices a.dot(b) de Numpy y compararlos con los anteriores.

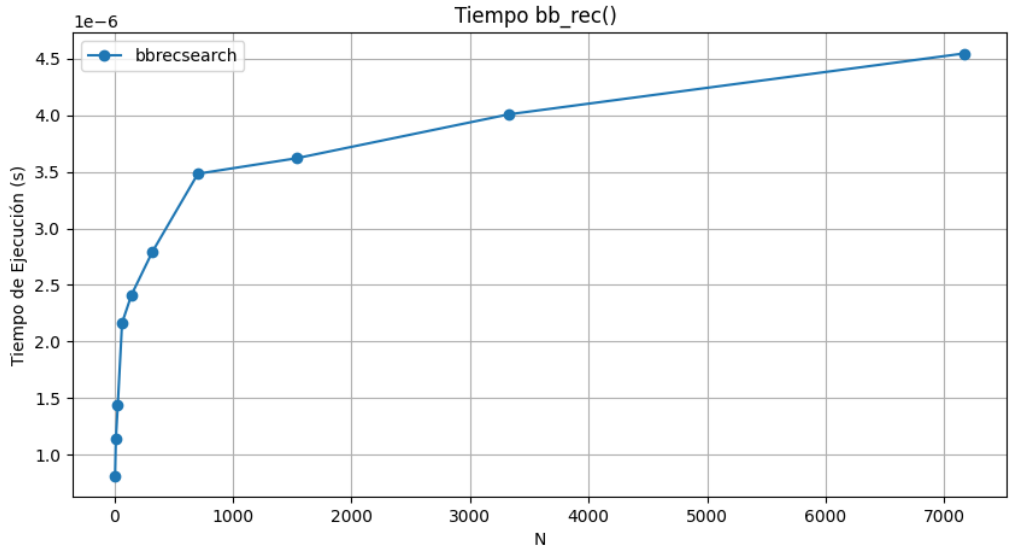


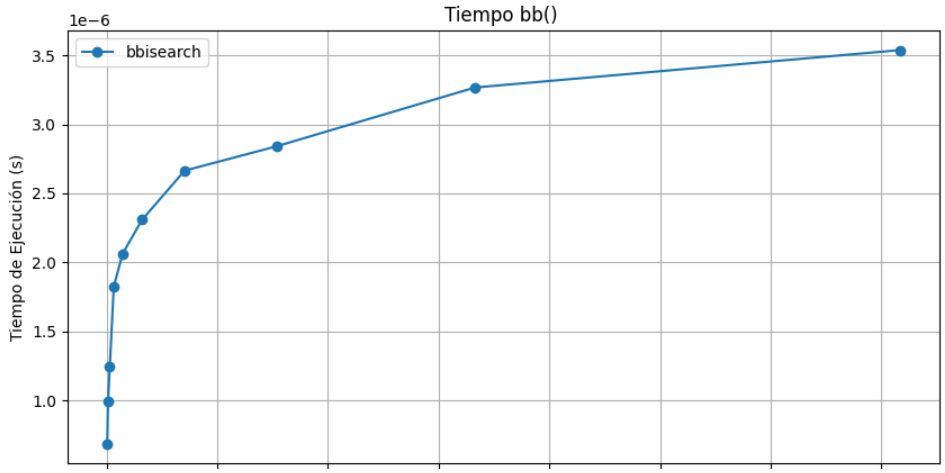


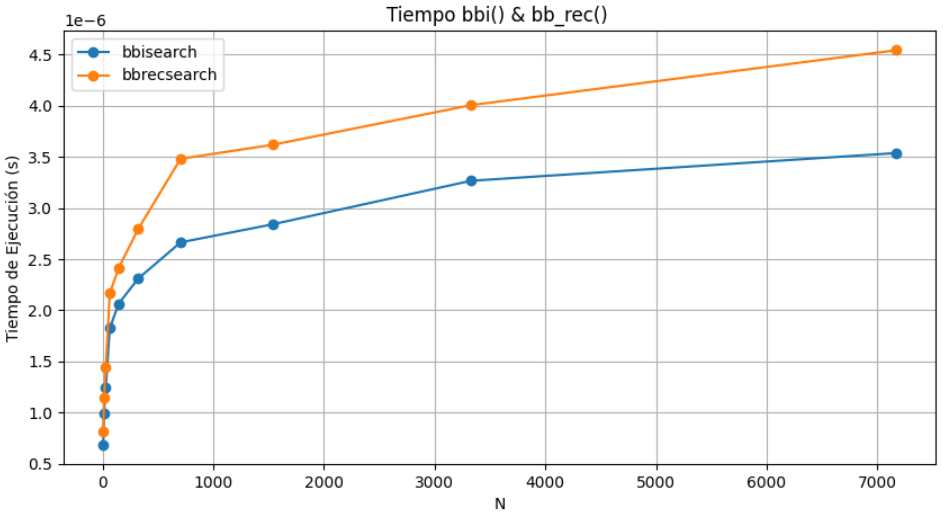
Con estas dos gráficas podemos comprobar que el método de multiplicación de matrices de numpy es mucho más eficaz que la función implementada por nosotros en python (multiplication_matrix()). Esto es así porque numpy.dot realiza la función en código compilado, al contrario que la función que se ha implementado que está interpretada.

3. ¿Qué clave resultaría en el caso más costoso de la búsqueda binaria? Comparar los tiempos de ejecución de las versiones recursiva e iterativa de la búsqueda binaria en su caso más costoso y dibujarlos para unos tamaños de tabla adecuados. ¿Qué relación encuentras entre ambos tiempos? Argumentar gráficamente dicha relación.

El caso más costoso es el primer elemento de la lista o el último, pues el algoritmo tendrá que iterar hasta el final para encontrar dicho elemento.

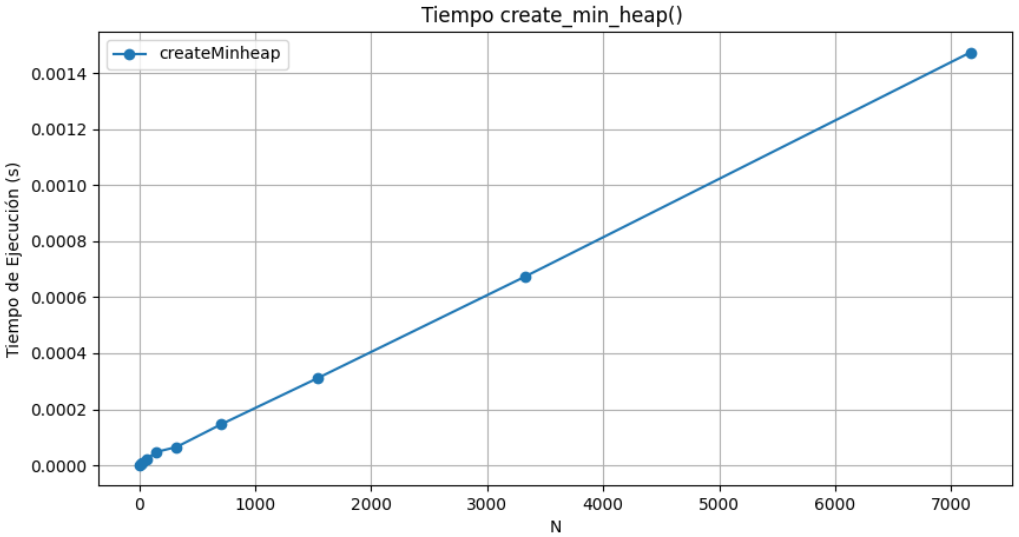






II-D. Cuestiones 1. Analizar visualmente los tiempos de ejecución de nuestra función de creación in place de min heaps. Establecer razonadamente a qué función se deberıan ajustar dichos tiempos.

Se puede observar en las gráficas que se corresponde con una función lineal.



2. Dar razonadamente cuál debería ser el coste de nuestra función de ordenación mediante Min Heaps en función del tamaño del array.

> En la función create_min_heap tomamos un array no ordenado y lo convertimos en un Min Heap a través de la función min_heapify. Dado que min_heapify tiene un tiempo de ejecución de O(log N) en el peor de los casos, el costo de crear el Min Heap sería O(N log N).
>
> Tras esto se itera en un bucle desde N hasta 0, por lo que tendrá complejidad O(N) así que la función completa seguirá teniendo complejidad O(N log N). Durante este último bucle se realizarán llamadas a min_heap_extract, que tiene otro bucle de complejidad O(N) y al final de dicha función, se vuelve llamar a create min heap por lo que seguirá teniendo complejidad O(N log N).

3. Analizar visualmente los tiempos de ejecución de nuestra función de ordenación mediante Min Heaps y comprobar que los mismos se ajustan a la función de coste del punto anterior.

Como se puede ver como las dos funciones realizan de manera aproximada el mismo trazo