03/10/2023

Algoritmia y Estructuras de Datos Avanzadas

Práctica - 1

G.1281 - P3

Joaquín Abad Díaz - Carlos García Santa

I-C. Cuestiones

1. ¿A qué función f se deberían ajustar los tiempos de ejecución de la función de multiplicación de matrices?

Los tiempos de ejecución de la función de multiplicación de matrices se deberían ajustar a $f(x) = n^3$. Esto es porque la complejidad del algoritmo de multiplicación de matrices es $O(n^3)$ ya que utiliza 3 bucles los cuales se ejecutan cada uno N veces.

La gráfica obtenida coincide con lo esperado:

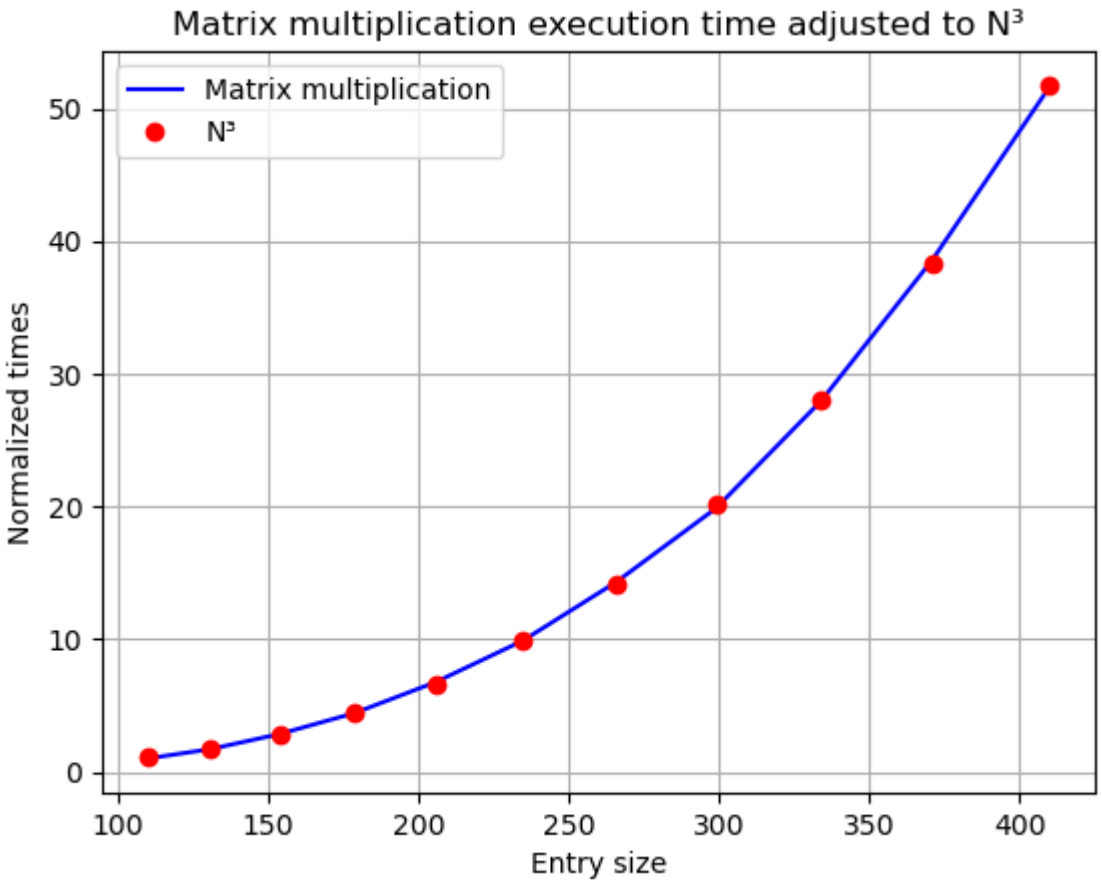

fig. 1

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

Los tiempos son mucho menores ya que np.dot realiza las operaciones aritméticas y el bucle externo en código compilado lo cual es mucho más rápido que el intérprete de Python que es el que usa nuestra función.

Comparación de resultados:

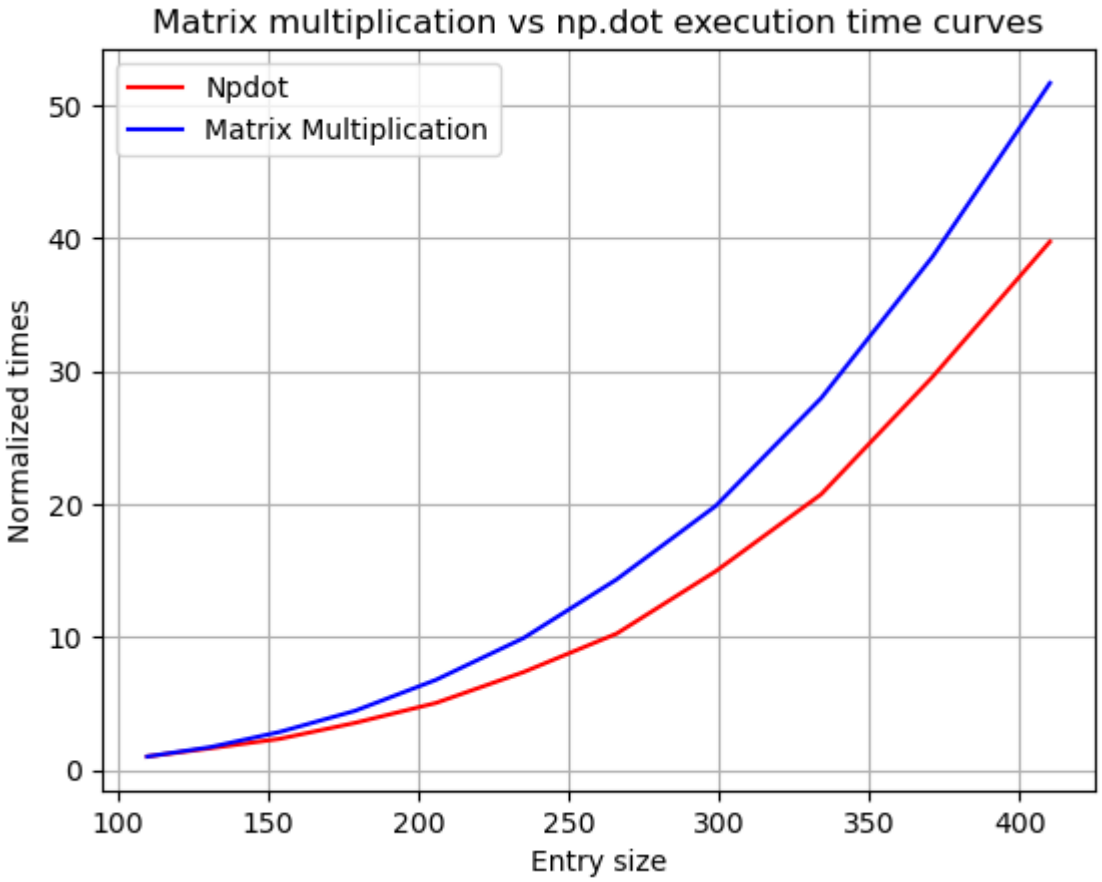

fig. 2

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.

Si asumimos que la clave que buscamos está en la lista, el caso más costoso es aquel en el cual el elemento a buscar está al final de la lista ya que es el momento en el que llevas la búsqueda binaria hasta el punto en que te quedas con un solo elemento en tu sublista.

La relación entre ambos tiempos se representa en la siguiente gráfica:

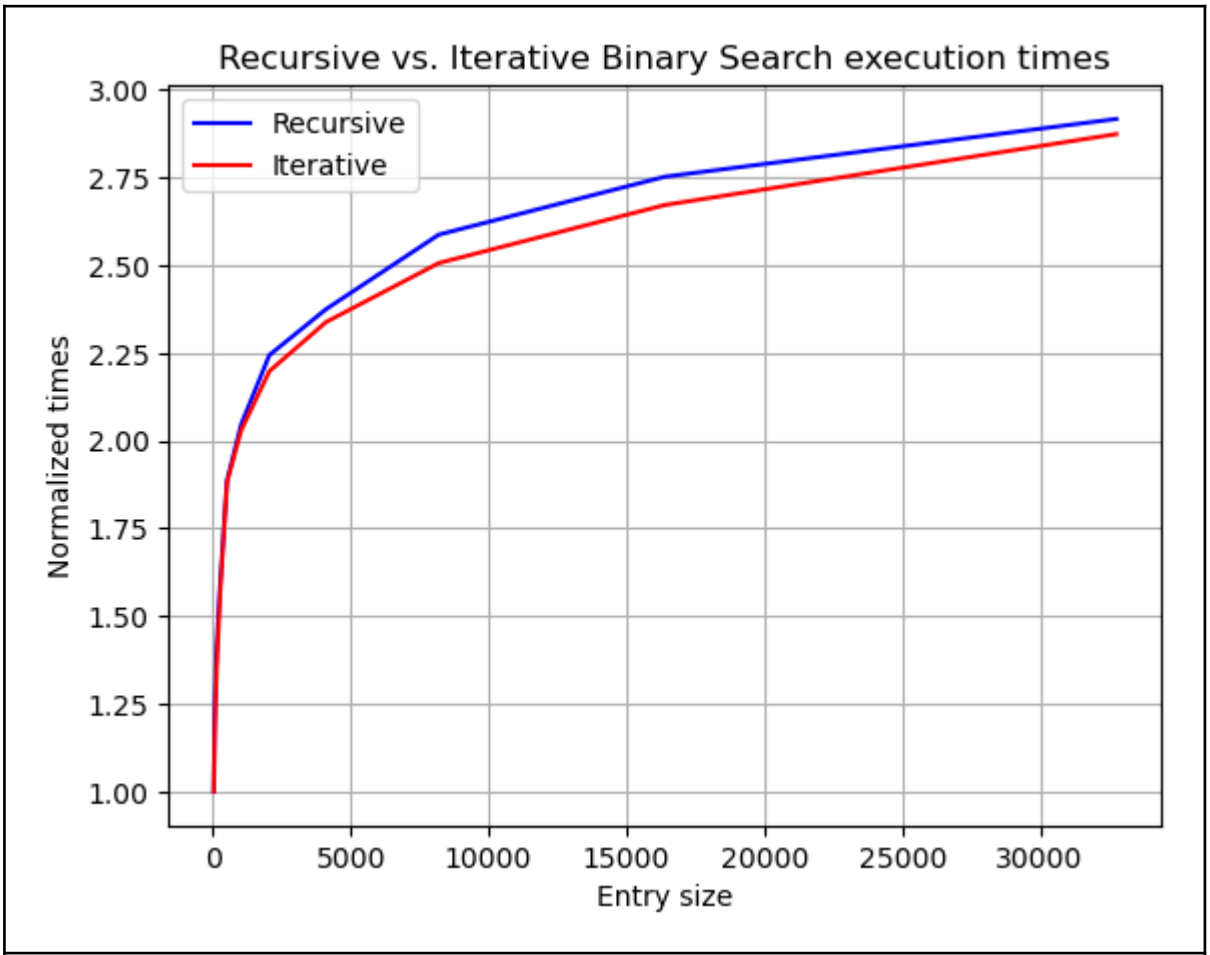

fig. 3

Los tiempos normalizados de ejecución de la versión Iterativa son ligeramente menores a los de la recursiva, debido a que las funciones recursivas requieren del almacenamiento de las variables locales en la pila del sistema cada vez que se hace una llamada recursiva, con lo cual esto aumenta la complejidad espacial haciendo la recursión más lenta que la iteratividad que no utiliza la pila del sistema.

II-C. Cuestiones

1. Analizar visualmente los tiempos de ejecución de nuestra función de creación in place de min heaps. Establecer razonadamente a qué función deberían ajustar dichos tiempos.

Primero tenemos que tener en cuenta la complejidad de min_heapify, en esta en el peor de los casos tendrá que recorrer desde la cima del heap al fondo lo cual teniendo en cuenta que hay n nodos equivale a un tiempo de O(log n).

Ahora sí, como en create_min_heap se llama n veces a min_heapify se podría pensar que la complejidad es de O(n log (n)); sin embargo, como la función trabaja desde los nodos que son hojas hasta el nodo raíz reduciendo el trabajo que tiene que hacer según se acerca a esta, la suma total de ese trabajo resulta en únicamente O(n).

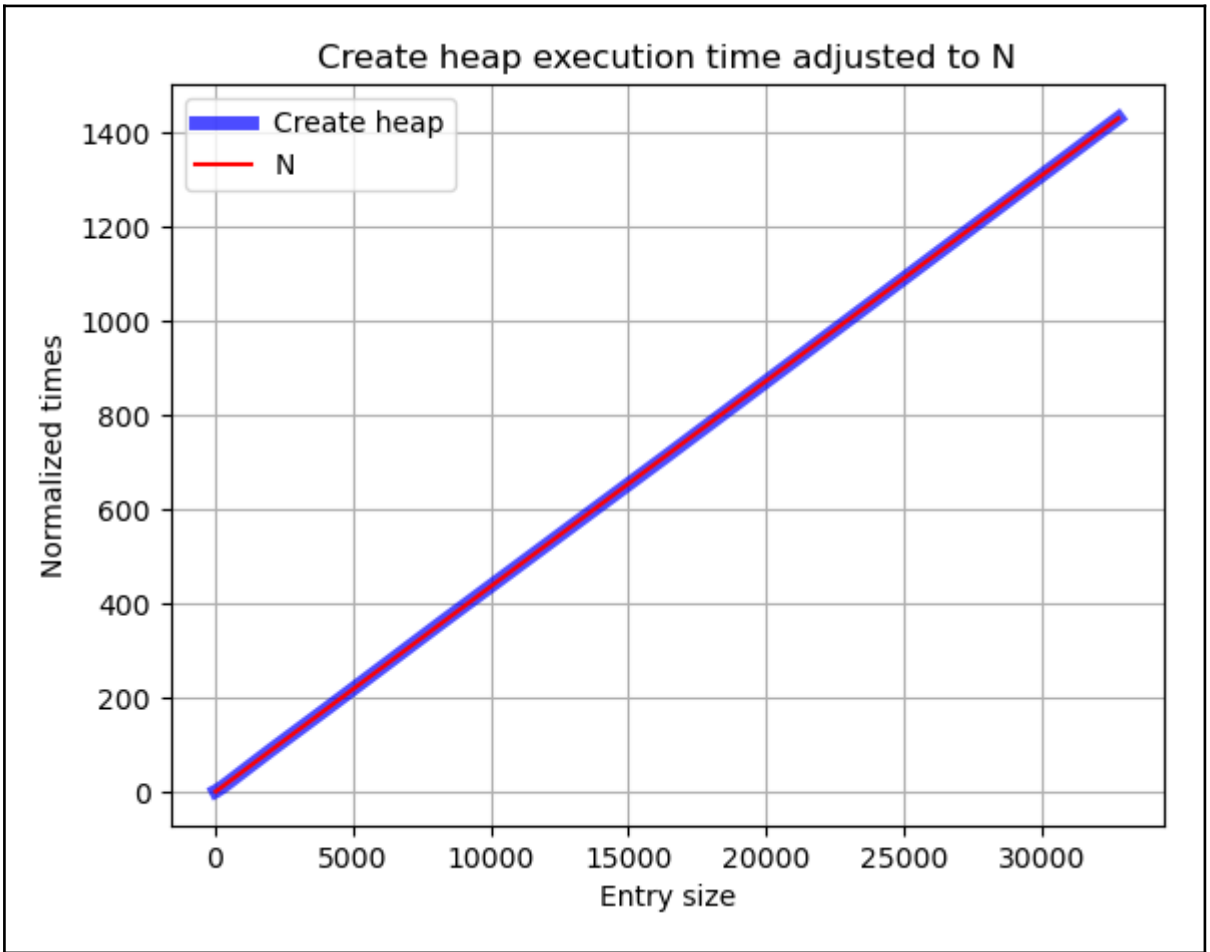

fig. 4

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

Debería ser de O(n log(n)), ya que la operación básica es la llamada a min_heapify la cual tiene una complejidad de O(log(n))que al ejecutarse en el bucle n veces, marca a min_heap_sort con una complejidad de 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 min_heap_sort se ajusta perfectamente a la espera complejidad de n log(n)

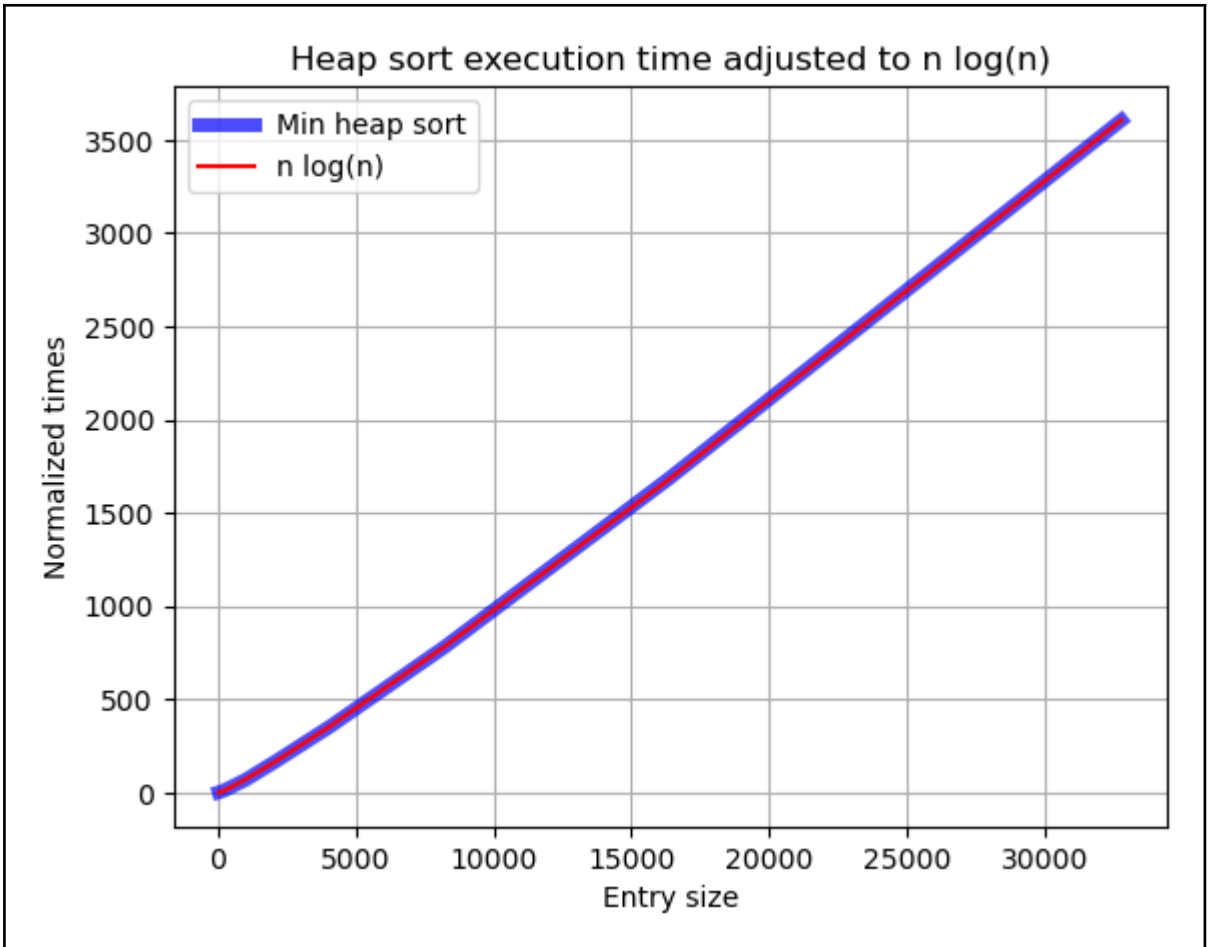

fig. 5