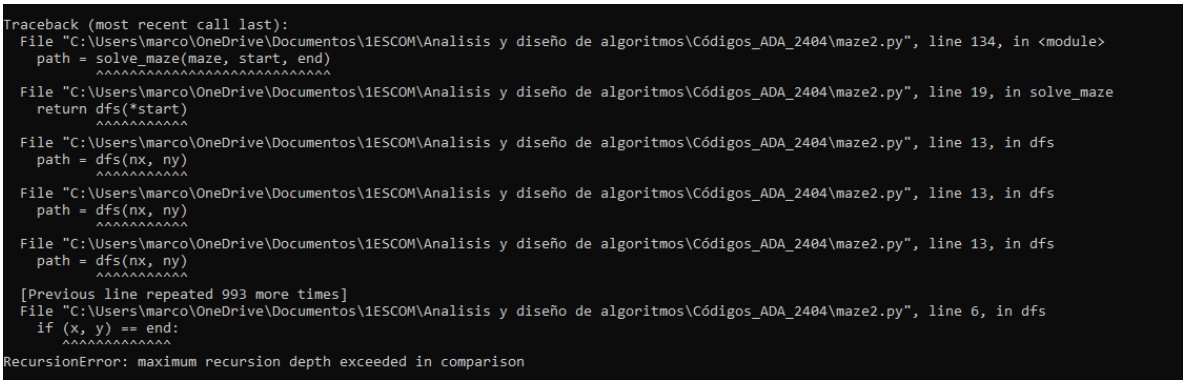
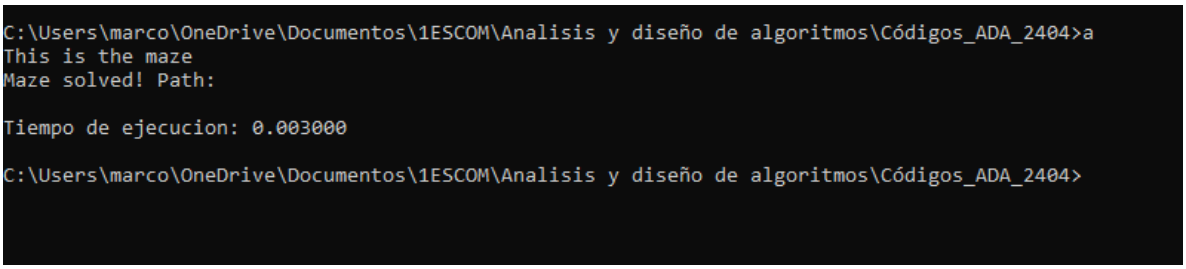
Ejercicio de laboratorio 4

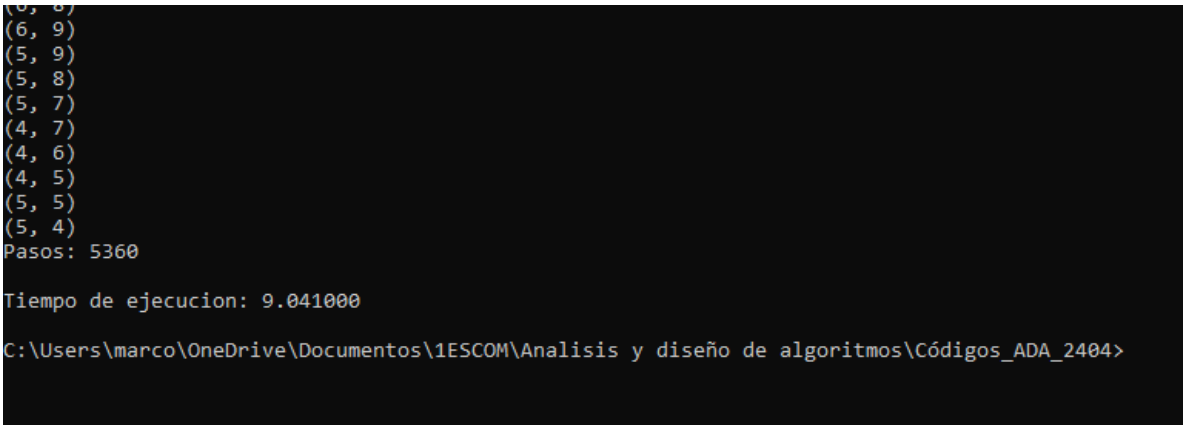
1. El código en Python en mi PC excedió el límite de llamadas recursivas utilizando una matriz de tamaño 100x100



El código en c corre en aproximadamente 3mls sin imprimir el camino ni la matriz



Y el código en c corre en 9 segundos imprimiendo el camino seguido y la matriz



2. Find_min_max: En cada iteración del algoritmo divide el arreglo en dos partes, siendo la primera el primer elemento del arreglo y la segunda el resto, la cual se resuelve recursivamente.

   a. C (Tamaño del arreglo: 10,000)

Éxito #stdin #stdout 0.01s 5392KB comments (0)

stdin copy

```
Standard input is empty
```

stdout copy

```
The minimum number in a given array is : 287073
The maximum number in a given array is : 2147463037
```

Python

Éxito #stdin #stdout 0.05s 10116KB comments (0)

stdin copy

```
Standard input is empty
```

stdout copy

```
The minimum number in a given array is :  352863072
The maximum number in a given array is :  3745948743
```

3. Quicksort es un algoritmo basado en la técnica Divide and Conquer, en cada iteración del algoritmo divide el arreglo en 2 partes (sin contar el pivote) la primera de los números menores o iguales al pivote y la segunda con los mayores, y resuelve combinando ambas soluciones. También nos damos cuenta de que el funcionamiento y por tal la complejidad del algoritmo es dependiente del pivote utilizado, por lo que es de importancia elegirlo inteligentemente para no tener una complejidad muy elevada.

   a. Tiempo de ejecución con 10,000 elementos en Python (Error en tiempo de ejecución al imprimir los 10,000 elementos del arreglo) :

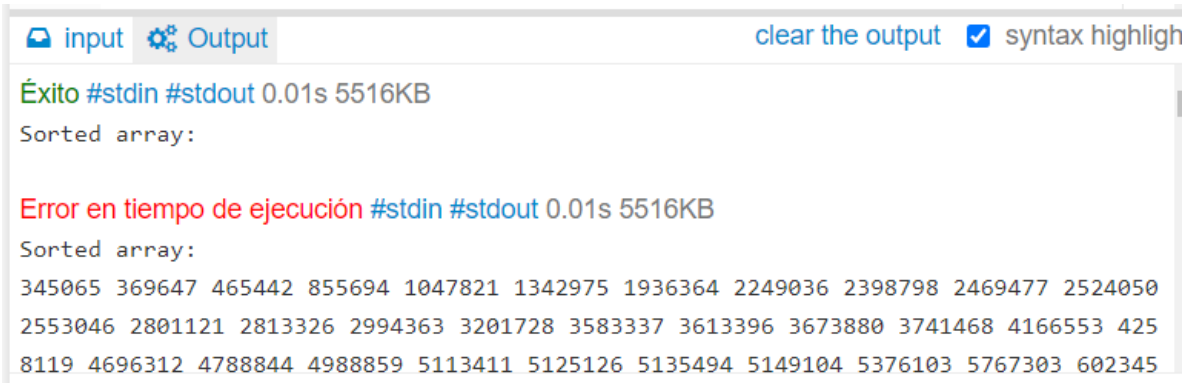
Éxito #stdin #stdout 0.09s 9988KB

```
Sorted array:
```

Error en tiempo de ejecución #stdin #stdout 0.08s 10132KB

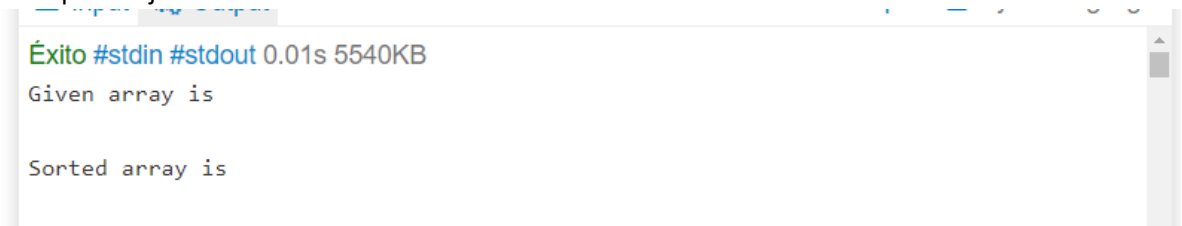
```
Sorted array:
337544 441581 538360 820414 938295 1179500 1452264 1869746 1896025 3259298 3810920
4492566 4503144 5021215 5208362 5271848 5695141 5913920 7285710 7461455 8212394 836
2395 8567389 9430084 9484320 9680992 11382975 11680100 12517857 12685456 12924998 1
```

b. Tiempo de ejecución con 10,000 elementos en C:

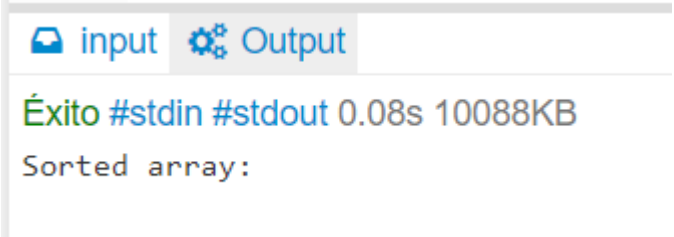


4. Merge-sort:
Este algoritmo de ordenamiento también utiliza una técnica DaC. En cada iteración se divide el arreglo en dos partes y se resuelve recursivamente cada una de ellas, luego se llena el arreglo original tomando el menor de los elementos restantes de cada arreglo hasta terminar con los subarreglos. Como el algoritmo es recursivo, siempre se llega a un caso base que es "fácil" de resolver y se puede combinar para obtener la solución del arreglo original. A diferencia del algoritmo Quicksort, este método no depende de un pivote por lo que la complejidad no varia de manera significativa respecto dos entradas diferentes.

a. Tiempo de ejecución en C



b. Tiempo de ejecución en Python



En los puntos anteriores los algoritmos utilizados en los códigos de ambos lenguajes son los mismos, solo cambia el lenguaje empleado para ejecutarlos. Los códigos en Python parecen mas simples y tienen menos líneas de código, pero se observa que los códigos toman mayor tiempo de ejecución.