

**INSTITUTO POLITÉCNICO NACIONAL**

**ESCUELA SUPERIOR DE CÓMPUTO**

**ANÁLISIS DE ALGORITMOS**

**PROFESORA: LUZ MARÍA SÁNCHEZ GARCÍA**

**INTEGRANTES:**

**VÁZQUEZ MORENO MARCOS OSWALDO 2016601777**

**DE LOS SANTOS DÍAZ LUIS ALEJANDRO 2017630451**

**PRÁCTICA 4 ANÁLISIS DE ALGORITMOS DE ORDENAMIENTO RECURSIVO**

**3CM2**

**16 DE MARZO DE 2019**

**Introducción**

En esta práctica se estudiará el problema de ordenamiento con base en algoritmos de ordenamiento recursivo.

Optamos por una solución recursiva cuando sabemos cómo resolver de manera directa un problema para un cierto conjunto de datos, y para el resto de los datos somos capaces de resolverlo utilizando la solución al mismo problema con unos datos “más simples”. ¾ Cualquier solución recursiva se basa en un análisis (clasificación) de los datos, x r , para distinguir los casos de solución directa y los casos de solución recursiva: — caso(s) directo(s): x r es tal que el resultado y r puede calcularse directamente de forma sencilla. — caso(s) recursivo(s): sabemos cómo calcular a partir de x r otros datos más pequeños x r ’, y sabemos además cómo calcular el resultado y r para x r suponiendo conocido el resultado y r ’ para x r ’. ¾ Para implementar soluciones recursivas en un lenguaje de programación tenemos que utilizar una acción que se invoque a sí misma −con datos cada vez “más simples”−: funciones o procedimientos. ¾ Para entender la recursividad, a veces resulta útil considerar cómo se ejecutan las acciones recursivas en una computadora, como si se crearan múltiples copias del mismo código, operando sobre datos diferentes (en realidad sólo se copian las variables locales y los parámetros por valor). (Horowitz, 1995)

Estos métodos son aún más complejos, requieren de mayor atención y conocimiento para ser entendidos.

Son rápidos y efectivos, utilizan generalmente la técnica “divide and conquer”, que consiste en dividir un problema grande en varios pequeños para que sea más fácil resolverlos.

Siendo estos óptimos que los iterativos, dentro de los algoritmos recursivos encontramos:

* Ordenamiento de Mezcla (Merge Sort).
* Ordenamiento Rápido (Quick Sort). (García, 2019)

**Planteamiento del problema**

**Diseño de la solución**

A continuación, se muestran los diagramas de flujo de nuestra propuesta de solución para los algoritmos de ordenamiento.

Primeramente, se muestra en el diagrama 1.1 el main.c de nuestra propuesta de solución.

Diagrama 1.1 Main.c

A continuación, se muestra en el diagrama 1.2 la implementación de nuestro archivo Práctica3.c

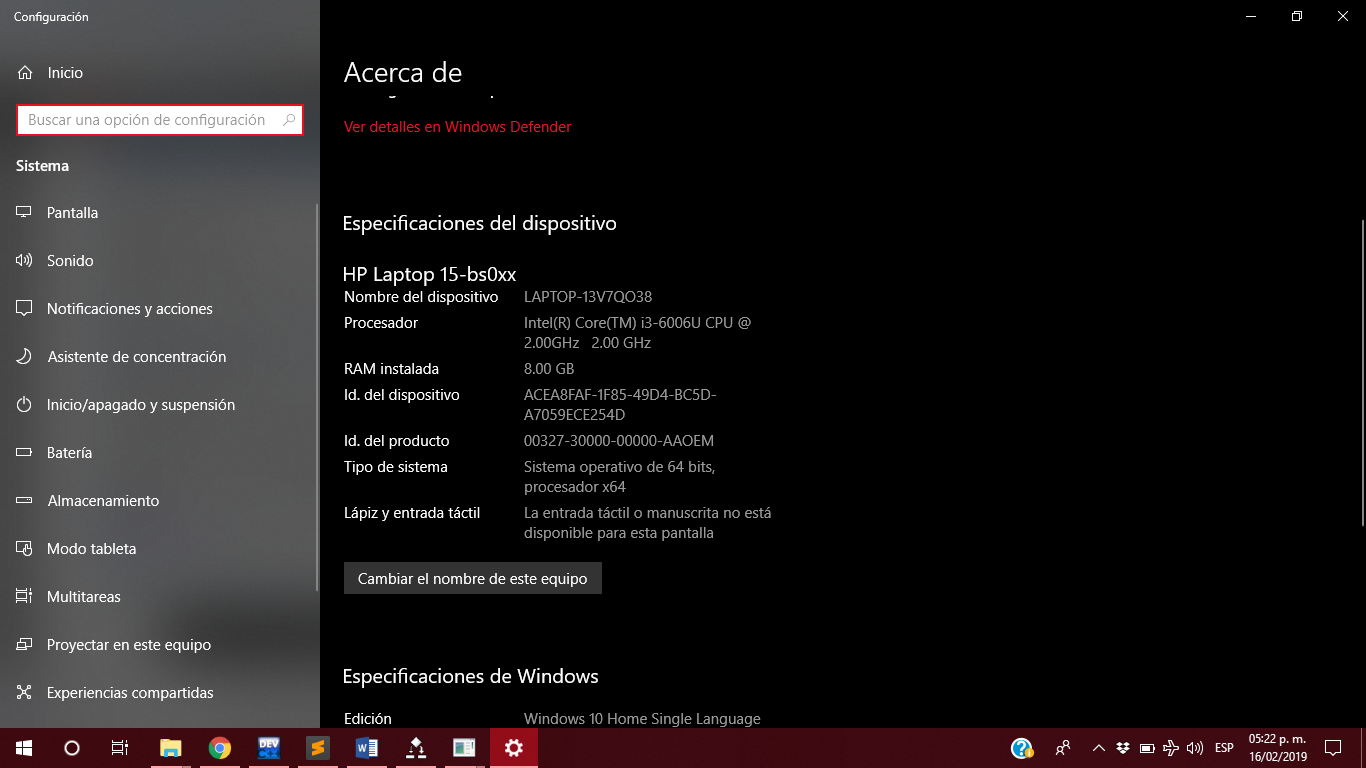
Diagrama 1.2 Práctica3.c

**Implementación de la solución**

**Funcionamiento**

**Plataforma experimental**

La ejecución de los algoritmos anteriores se llevó a cabo en una computadora personal que se describe en la siguiente imagen.

  
Imagen 3.1 Plataforma Experimental

El compilador utilizado fue gcc integrado dentro del IDE DevC en un sistema operativo de 64 bits Windows 10.

**Gráficas de funciones**

**Graficas de tiempos con muestras**

**Conclusiones**

**Bibliografía**

# Bibliografía

Díaz, I. N. (2006). *FIET UNICAUCA*. Obtenido de http://artemisa.unicauca.edu.co/~nediaz/EDDI/cap02.htm

g, S. (s.f.).

García, L. M. (15 de 03 de 2019). *Classrom.* Obtenido de Práctica 4 Ánalisis de Algoritmos: https://classroom.google.com/c/NzEzNTUxMDU1NVpa/a/NzgxNTE2NTA3MVpa/details

Gimeno, J., & González, J. (s.f.). *ocw.udl.cat.* Recuperado el 2017, de http://ocw.udl.cat/enginyeria-i-arquitectura/programacio-2/continguts-1/2-recursividad.pdf

Gurin, S. (Septiembre de 2004). *Algoritmos de ordenación*. Obtenido de Algoritmos de ordenación: http://es.tldp.org/Tutoriales/doc-programacion-algoritmos-ordenacion/alg\_orden.pdf

Horowitz, E. (1995). Fundamentals of Data Structures in C++. En D. Mehta, *Fundamentals of Data Structures in C++* (págs. 5-7). c.