PROYECTO TERCER CORTE

RICHARD RAMIREZ PATIÑO

SEBASTIÁN URBANO LUNA

CARLOS ALBERTO LONDOÑO DOCENTE:

ESTRUCTURA DE DATOS 1

CORPORACIÓN DE ESTUDIOS TECNOLÓGICOS

DEL NORTE DEL VALLE

INGENIERÍA SISTEMAS

2017-1

**ERROR DEL SISTEMA**

**Segmentation fault - violación de segmento.**

Es un tipo de error en tiempo de ejecución muy común para los programas de C/C++.

Cuando se ejecuta el programa y el sistema de informes de su sistema lanza una "violación de segmentación", significa que su programa ha intentado acceder a un área de memoria a la cual no le está permitido el acceso. En otras palabras, se trató de acceder a una parte de la memoria que está más allá de los límites que el sistema operativo (Unix GNU/Linux etc) ha asignado para su programa.

**ORDENAMIENTO POR BURBUJA**

Es un sencillo [algoritmo de ordenamiento](https://es.wikipedia.org/wiki/Algoritmo_de_ordenamiento). Funciona revisando cada elemento de la lista que va a ser ordenada con el siguiente, intercambiandolos de posición si están en el orden equivocado. Es necesario revisar varias veces toda la lista hasta que no se necesiten más intercambios, lo cual significa que la lista está ordenada.

|  |  |  |
| --- | --- | --- |
| **Ordenamiento Por Burbuja**  **o Bubblesort**  [**O**](https://es.wikipedia.org/wiki/Cota_superior_asint%C3%B3tica)**(*n*²)** | | |
| Valor | Tiempo Sin Imprimir | Tiempo Impreso |
|  |  |  |
| 100.000 | 00:00:38 | 00:00:54 |
| 250.000 | 00:04:37 | 00:05:15 |
| 500.000 | 00:15:41 | 00:16:57 |
| 1.000.000 | 01:07:31 | 01:10:01 |
| 2.000.000 | 03:45:30 | 03:50:10 |

**ORDENAMIENTO POR INSERCIÓN**

Es una manera muy natural de ordenar para un ser humano, y puede usarse fácilmente para ordenar un mazo de cartas numeradas en forma arbitraria

|  |  |  |
| --- | --- | --- |
| **Ordenamiento Por Inserción**  **o Insertion sort**  [**O**](https://es.wikipedia.org/wiki/Cota_superior_asint%C3%B3tica)**(*n*²)** | | |
| Valor | Sin Imprimir | Impreso |
|  |  |  |
| 100.000 | 00:00:06 | 00:00:21 |
| 250.000 | 00:01:52 | 00:02:30 |
| 500.000 | 00:09:08 | 00:10:23 |
| 1.000.000 | 00:11:33 | 00:14:02 |
| 2.000.000 | 00:46:45 | 00:51:25 |

**ORDENACIÓN POR MONTONES**

Este algoritmo consiste en almacenar todos los elementos del vector a ordenar en un [montículo](https://es.wikipedia.org/wiki/Mont%C3%ADculo_(inform%C3%A1tica)) (*heap*), y luego extraer el nodo que queda como nodo raíz del montículo (cima) en sucesivas iteraciones obteniendo el conjunto ordenado. Basa su funcionamiento en una propiedad de los montículos, por la cual, la cima contiene siempre el menor elemento (o el mayor, según se haya definido el montículo) de todos los almacenados en él. El algoritmo, después de cada extracción, recoloca en el nodo raíz o cima, la última hoja por la derecha del último nivel.

|  |  |  |
| --- | --- | --- |
| **Ordenamiento Por Montones**  **o Heapsort**  [**O**](https://es.wikipedia.org/wiki/Cota_superior_asint%C3%B3tica)**(*n* log *n*)** | | |
| Valor | Sin Imprimir | Impreso |
|  |  |  |
| 100.000 | 00:12:50 | 00:13:05 |
| 250.000 | 00:22:30 | 00:22:30 |
| 500.000 | 01:35:45 | 01:35:45 |
| 1.000.000 | 02:50:53 | 02:53:22 |
| 2.000.000 | 05:05:54 | 05:10:34 |

**ORDENAMIENTO RÁPIDO**

El ordenamiento rápido (quicksort en [inglés](https://es.wikipedia.org/wiki/Idioma_ingl%C3%A9s)) es un [algoritmo](https://es.wikipedia.org/wiki/Algoritmo) creado por el científico británico en computación [C. A. R. Hoare](https://es.wikipedia.org/wiki/C._A._R._Hoare), basado en la técnica de [divide y vencerás](https://es.wikipedia.org/wiki/Divide_y_vencer%C3%A1s), que permite, en promedio, [ordenar](https://es.wikipedia.org/wiki/Algoritmo_de_ordenamiento) *n* elementos en un tiempo proporcional a *n* log *n*.

|  |  |  |
| --- | --- | --- |
| **Ordenamiento Rápido**  **o Quicksort**  **Promedio:** [**O**](https://es.wikipedia.org/wiki/Cota_superior_asint%C3%B3tica)**(*n* log *n*),**  **peor caso: O(*n*²)** | | |
| Valor | Sin Imprimir | Impreso |
|  |  |  |
| 100.000 | 00:00:00 26 | 00:00:16 |
| 250.000 | 00:00:00 40 | 00:00:38 |
| 500.000 | 00:00:01 | 00:01:17 |
| 1.000.000 | 00:00:04 | 00:02:34 |
| 2.000.000 | 00:00:06 | 00:04:46 |