INSTITUTO TECNOLÓGICO DE ENSENADA

**INTELIGENCIA ARTIFICIAL**

Manuel Vega Solorio

José Luis Alárez Orta

**ALGORITMOS DE ORDENAMIENTO**

Fecha de entrega: 21/09/2016

**Índice**

[**Diagramas de flujo** 3](#_Toc462257029)

[**Burbuja** 3](#_Toc462257030)

[**Inserción** 4](#_Toc462257031)

[**Selección** 5](#_Toc462257032)

[**Heapsort** 6](#_Toc462257033)

[**Quicksort** 7](#_Toc462257034)

[**Mergesort** 8](#_Toc462257035)

[**Capturas de pantalla** 9](#_Toc462257036)

[**Burbuja** 9](#_Toc462257037)

[**HeapSort** 9](#_Toc462257038)

[**Insercion** 10](#_Toc462257039)

[**Seleccion** 10](#_Toc462257040)

[**Quicksort** 11](#_Toc462257041)

[**MergeSort** 11](#_Toc462257042)

[**Tablas de tiempos** 12](#_Toc462257043)

[**Pruebas realizadas en Lenguaje C** 12](#_Toc462257044)

[**Pruebas realizadas en Javascript** 12](#_Toc462257045)

[**Explicación de cada algoritmo** 12](#_Toc462257046)

[**Inserción** 12](#_Toc462257047)

[**Selección** 12](#_Toc462257048)

[**Burbuja** 13](#_Toc462257049)

[**Heapsort** 13](#_Toc462257050)

[**Mergesort** 13](#_Toc462257051)

[**Quicksort** 14](#_Toc462257052)

# **Diagramas de flujo**

## **Burbuja**

j=n-1; j>i; j--

Int Vector, i, j, n;

Double tmp;

i=0; i<n-1; i++

V[j]<v[j-1]

tmp=v[j];

v[j]=v[j-1];

v[j-1]=tmp;

Lista Ordenada

## **Inserción**

Lista Ordenada

V[j]=tmp;

V[j]=v[j-1];

J=I;(j>0)&&(tmp<v[j-1]);j--

Tmp=v[i];

i=0; i<n; i++

Int Vector, i, j, n;

Double tmp;

## **Selección**

Lista Ordenada

Tmp=v[i];

V[i]=v[m];

V[m]=tmp;

m=j;

V[j]<v[m]

J=i+1;j<n;j++

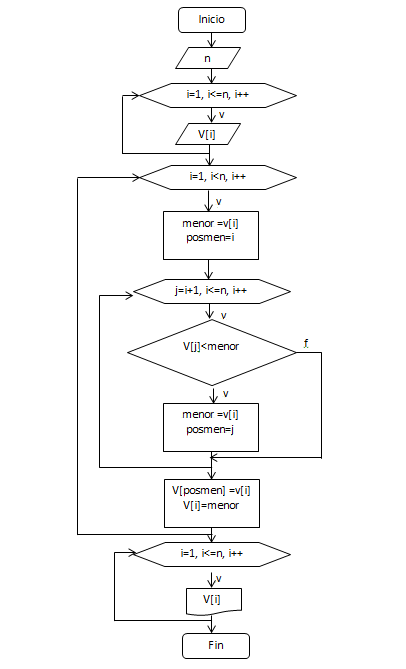
Int Vector, i, j, n,m;

Double tmp;

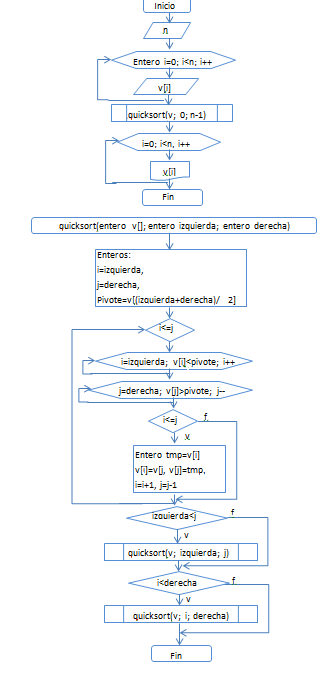
i=0; i<n-1; i++

m=I;

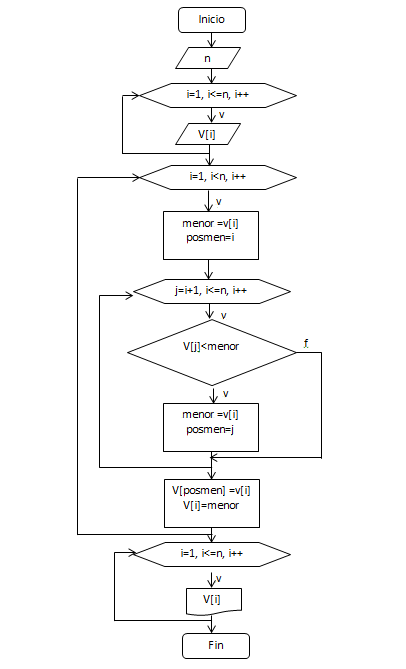
## **Heapsort**



## **Quicksort**

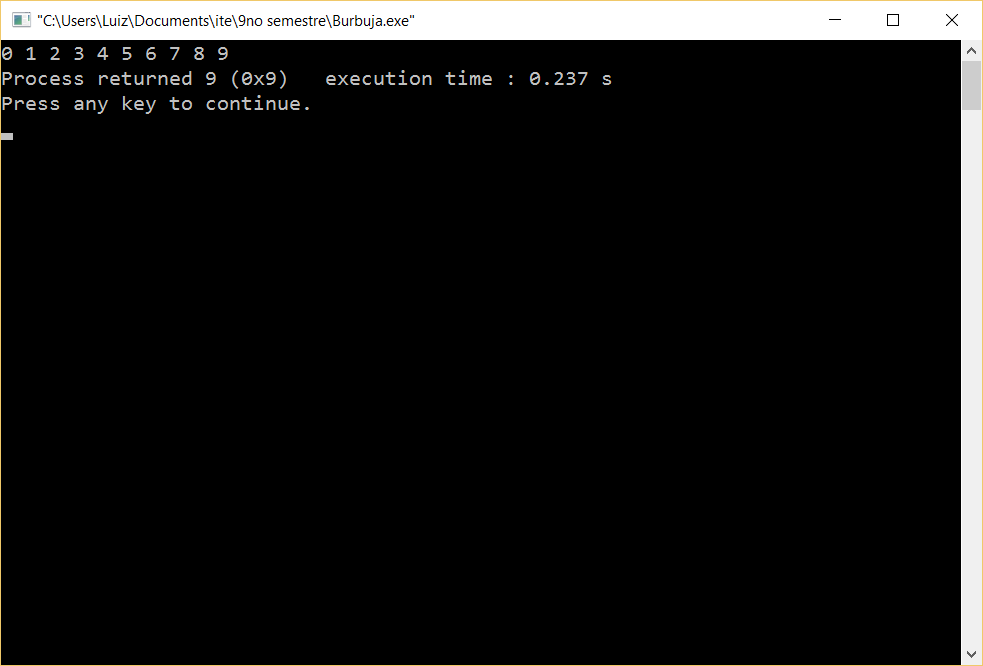


## **Mergesort**

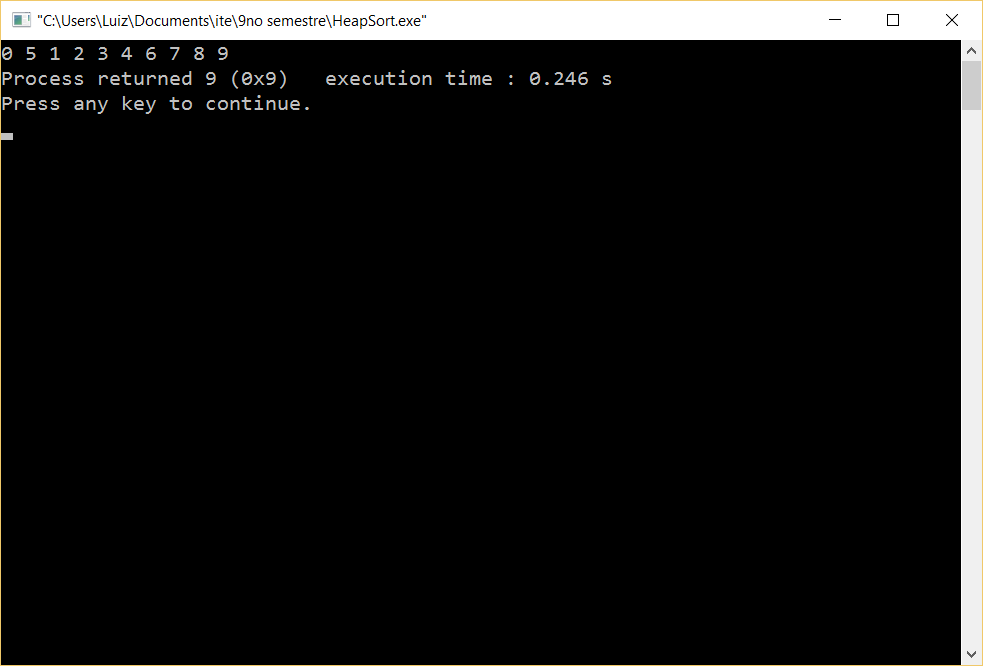


# **Capturas de pantalla**

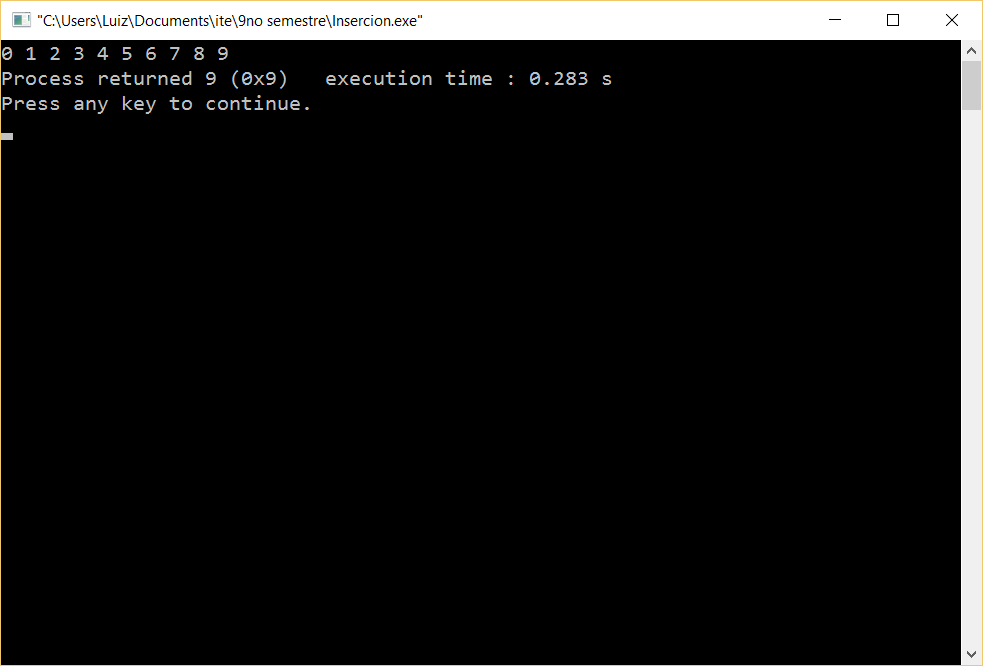
## **Burbuja**



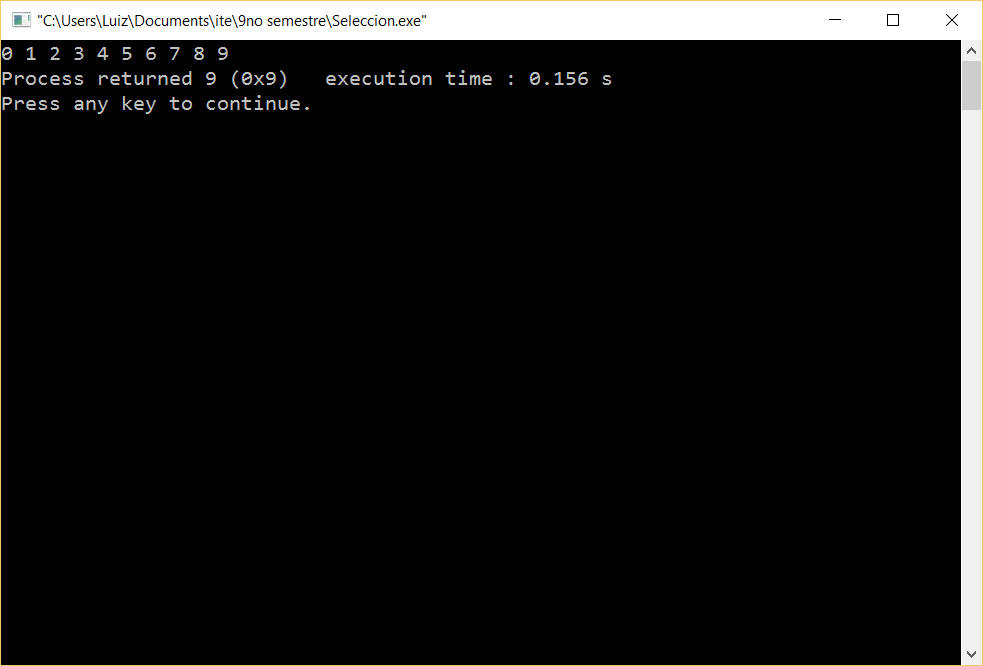
## **HeapSort**



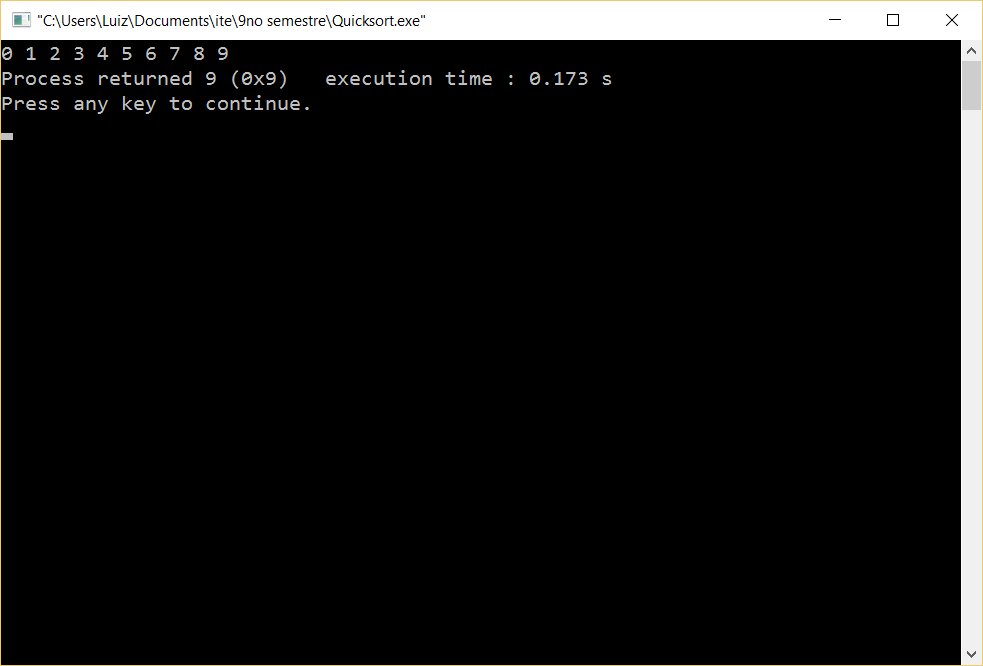
## **Insercion**



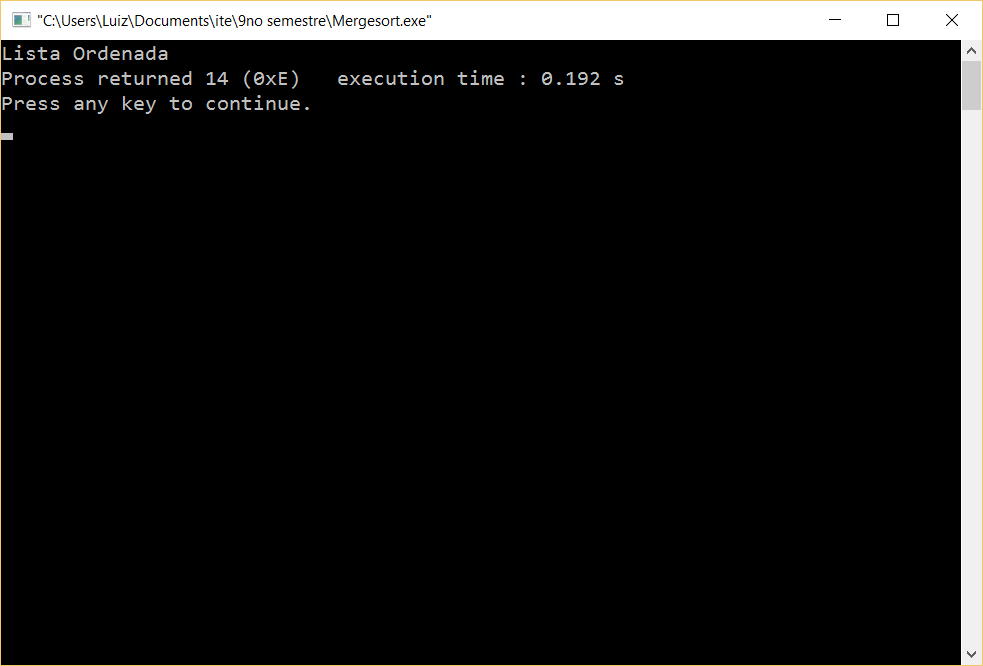
## **Seleccion**



## **Quicksort**



## **MergeSort**



# **Tablas de tiempos**

|  |  |  |
| --- | --- | --- |
| **Pruebas realizadas en Lenguaje C** | | |
| **Algoritmo de Ordenamiento** | **Tiempo** | **Número de Elementos** |
| Burbuja | 0.237 | 10 |
| Heapsort | 0.246 | 10 |
| Insercion | 0.283 | 10 |
| Seleccion | 0.156 | 10 |
| Quicksort | 0.173 | 10 |
| Mergesort | 0.192 | 10 |

|  |  |  |
| --- | --- | --- |
| **Pruebas realizadas en Javascript** | | |
| **Algoritmo de Ordenamiento** | **Tiempo** | **Número de Elementos** |
| Burbuja | 0.237 | 10 |
| Heapsort | 0.246 | 10 |
| Insercion | 0.283 | 10 |
| Seleccion | 0.156 | 10 |
| Quicksort | 0.173 | 10 |
| Mergesort | 0.192 | 10 |

# **Explicación de cada algoritmo**

## **Inserción**

La idea de este algoritmo de ordenación consiste en ir insertando un elemento de la lista ó un arreglo en la parte ordenada de la misma, asumiendo que el primer elemento es la parte ordenada, el algoritmo ira comparando un elemento de la parte desordenada de la lista con los elementos de la parte ordenada, insertando el elemento en la posición correcta dentro de la parte ordenada, y así sucesivamente hasta obtener la lista ordenada.

## **Selección**

Se basa en realizar varias pasadas, intentando encontrar en cada una de ellas el elemento que según el criterio de ordenación es mínimo y colocándolo posteriormente en su sitio.

* Buscar el mínimo o máximo elemento de la lista
* Intercambiarlo con el primero
* Buscar el mínimo o el máximo en el resto de la lista
* Intercambia con el segundo.

## **Burbuja**

Este método consiste en acomodar el vector moviendo el mayor hasta la última casilla comenzando desde la casilla cero del vector hasta haber acomodado el número más grande en la última posición, una vez acomodado el más grande, prosigue a encontrar y acomodar el siguiente más grande comparando de nuevo los números desde el inicio del vector, y así sigue hasta ordenar todo los elementos el arreglo.

## **Heapsort**

Este algoritmo consiste en almacenar todos los elementos del vector a ordenar en un montículo (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. Lo cual destruye la propiedad heap del árbol. Pero, a continuación realiza un proceso de "descenso" del número insertado de forma que se elige a cada movimiento el mayor de sus dos hijos, con el que se intercambia. Este intercambio, realizado sucesivamente "hunde" el nodo en el árbol restaurando la propiedad montículo del arbol y dejándo paso a la siguiente extracción del nodo raíz.

## **Mergesort**

Conceptualmente, el ordenamiento por mezcla funciona de la siguiente manera:

1. Si la longitud de la lista es 0 ó 1, entonces ya está ordenada. En otro caso:
2. Dividir la lista desordenada en dos sublistas de aproximadamente la mitad del tamaño.
3. Ordenar cada sublista recursivamente aplicando el ordenamiento por mezcla.
4. Mezclar las dos sublistas en una sola lista ordenada.

El ordenamiento por mezcla incorpora dos ideas principales para mejorar su tiempo de ejecución:

1. Una lista pequeña necesitará menos pasos para ordenarse que una lista grande.
2. Se necesitan menos pasos para construir una lista ordenada a partir de dos listas también ordenadas, que a partir de dos listas desordenadas. Por ejemplo, sólo será necesario entrelazar cada lista una vez que están ordenadas.

## **Quicksort**

El algoritmo trabaja de la siguiente forma:

* Elegir un elemento de la lista de elementos a ordenar, al que llamaremos **pivote**.
* Resituar los demás elementos de la lista a cada lado del pivote, de manera que a un lado queden todos los menores que él, y al otro los mayores. Los elementos iguales al pivote pueden ser colocados tanto a su derecha como a su izquierda, dependiendo de la implementación deseada. En este momento, el pivote ocupa exactamente el lugar que le corresponderá en la lista ordenada.
* La lista queda separada en dos sublistas, una formada por los elementos a la izquierda del pivote, y otra por los elementos a su derecha.
* Repetir este proceso de forma recursiva para cada sublista mientras éstas contengan más de un elemento. Una vez terminado este proceso todos los elementos estarán ordenados.