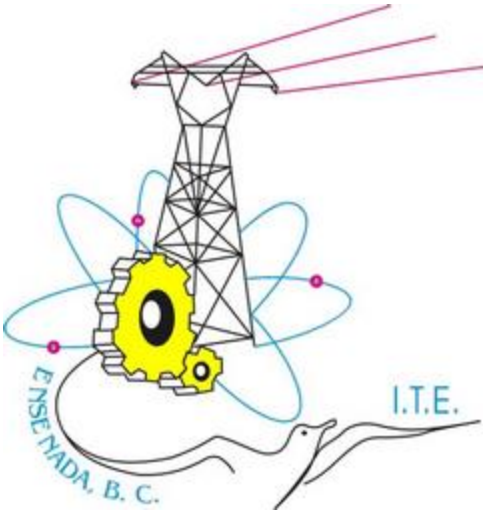
INSTITUTO TECNOLOGICO DE ENSENADA



Algoritmos de ordenamiento

Materia

Inteligencia Artificial

Alumnos

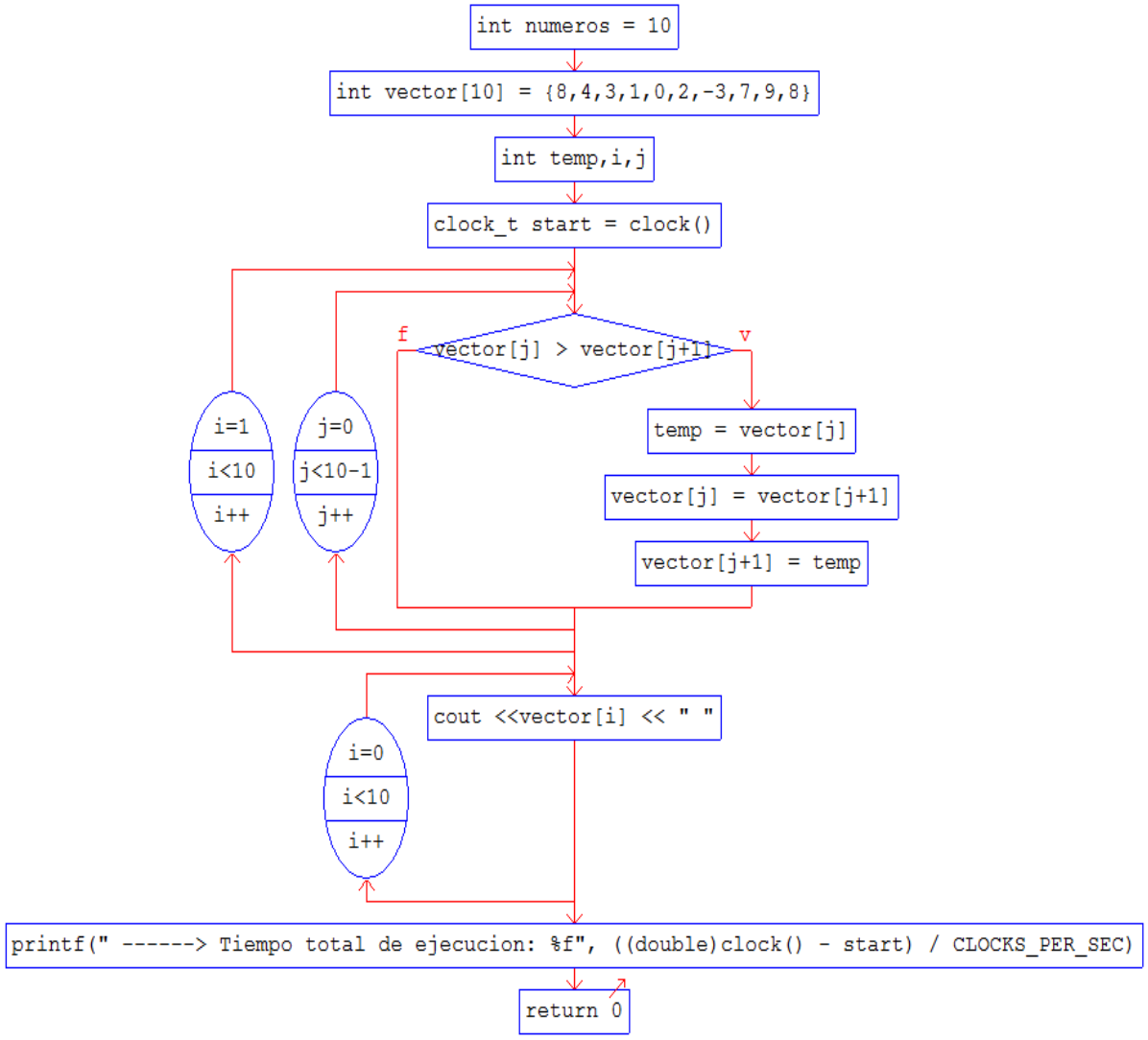
Sergio Antonio Cruz Olivares
Francisco Eduardo Garcia Perea

Algoritmos de Ordenamiento

Ordenamiento por Burbuja.

El método de ordenación por burbuja es el más conocido y popular entre estudiantes y aprendices de programación, por su facilidad de comprensión y programación; por el contrario, es el menos eficiente y por ello, normalmente, se aprende su técnica pero no suele utilizarse.

La técnica utilizada se denomina ordenación por burbuja u ordenación por hundimiento debido a que los valores más pequeños "burbujean" gradualmente (suben) hacia la cima o parte superior del array de modo similar a como suben las burbujas en el agua, mientras que los valores mayores se hunden en la parte inferior del array. La técnica consiste en hacer varias pasadas a través del array. En cada pasada, se comparan parejas sucesivas de elementos. Si una pareja está en orden creciente (o los valores son idénticos), se dejan los valores como están. Si una pareja está en orden decreciente, sus valores se intercambian en el array.



Resultados

Lenguaje C

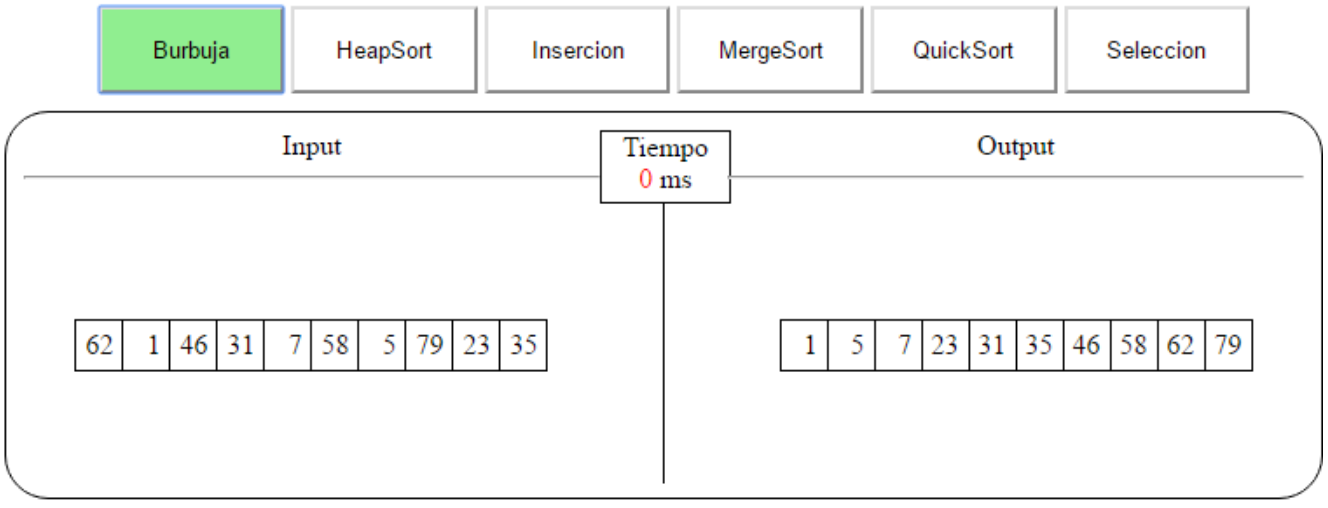
```
/* ITE - IA
Sergio Antonio Cruz Olivares
Francisco Eduardo Garcia Perea */

#include <time.h>
#include <iostream>

using namespace std;
int main()
{
int numeros = 10;   //Longitud del array
    int vector[numeros] = {8,4,3,1,0,2,-3,7,9,8};  //números  contenidos del array
    int temp,i,j;   // variables de comparison y desplazamiento

     clock_t start = clock();   //Comienza cronometro de ejecución

//Inicia ordenamiento por burbuja
        for (i=1; i<numeros; i++){

            for (j=0; j<numeros-1; j++){
                if (vector[j] > vector[j+1]){
                    temp = vector[j];
                    vector[j] = vector[j+1];
                    vector[j+1] = temp;
                }
            }
        }
        for (i=0; i<numeros; i++){
            cout <<vector[i] << " ";
        }

//tiempo de ejecution
        printf("    ------>  Tiempo total de ejecucion: %f", ((double)clock() - start) / CLOCKS_PER_SEC);

        system("pause");
        return 0;

    }
```

Get URL

options | compilation | execution

```
-3 0 1 2 3 4 7 8 8 9     ------>  Tiempo total de ejecucion: 0.000038
```

Lenguaje JavaScript



Ordenamiento por Selección.

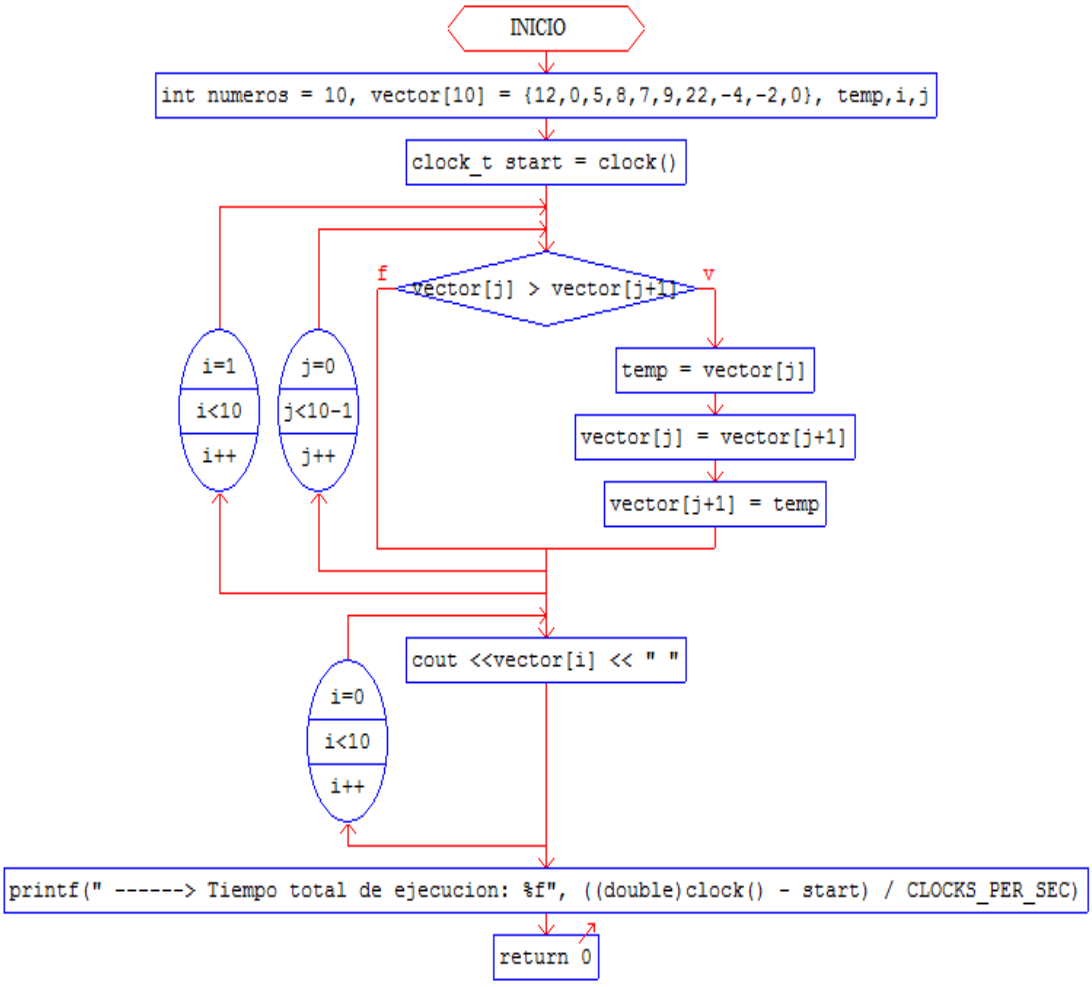
Considérese el algoritmo para ordenar un array A de enteros en orden ascendente, es decir, del número más pequeño al mayor. Es decir, si el array A tiene n elementos, se trata de ordenar los valores del array de modo que el dato contenido en A[0] sea el valor más pequeño, el valor almacenado en A[1] el siguiente más pequeño, y así hasta A[n-1], que ha de contener el elemento de mayor valor. El algoritmo se apoya en sucesivas pasadas que intercambian el elemento más peque- ño sucesivamente con el primer elemento de la lista, A[0] en la primera pasada. En síntesis, se busca el elemento más pequeño de la lista y se intercambia con A[0], primer elemento de la lista.

A[0] A[1] A[2].... A[n-1]

Después de terminar esta primera pasada, el frente de la lista está ordenado y el resto de la lista A[1], A[2]... A[n-1] permanece desordenado.

La siguiente pasada busca en esta lista desordenada y selecciona el elemento más pequeño, que se almacena entonces en la posición A[1]. De este modo los elementos A[0] y A[1] están ordenados y la sublista A[2], A[3]...A[n-1] desordenada; entonces, se selecciona el elemento más pequeño y se intercambia con A[2].

El proceso continúa n − 1 pasadas y en ese momento la lista desordenada se reduce a un elemento (el mayor de la lista) y el array completo ha quedado ordenado.



Resultados

Lenguaje C

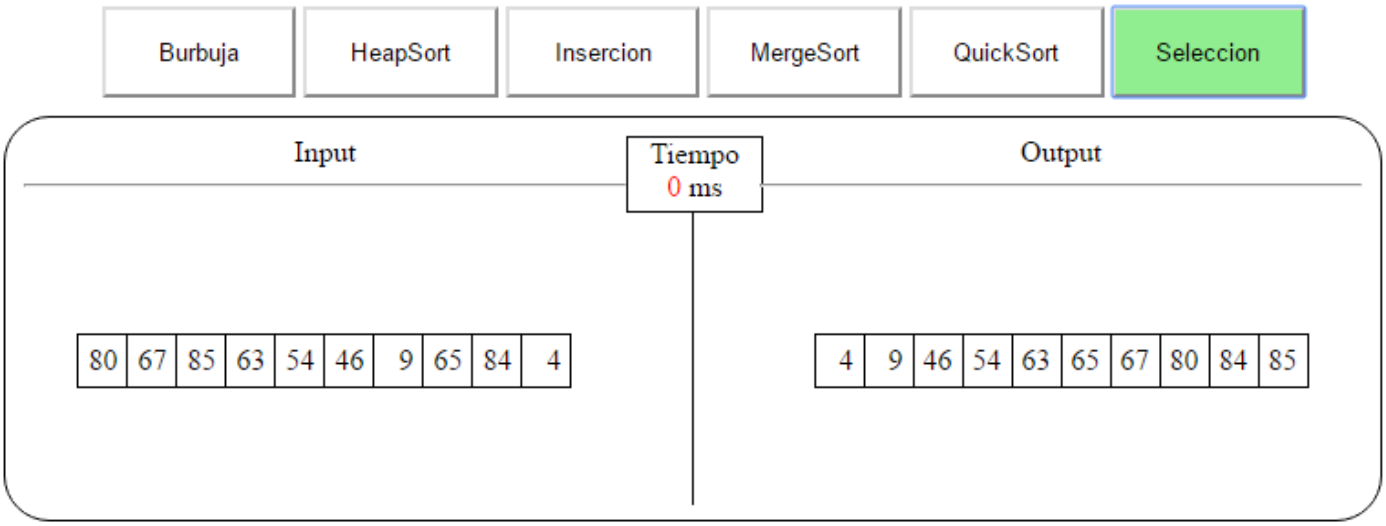
```c
/* ITE - IA
Sergio Antonio Cruz Olivares
Francisco Eduardo Garcia Perea */

#include <time.h>
#include <iostream>

using namespace std;
int main()
{
int numeros = 10;   //Longitud del array
    int vector[numeros] = {8,4,3,1,0,2,-3,7,9,8};  //números  contenidos del array
    int temp,i,j;   // variables de comparison y desplazamiento

     clock_t start = clock();   //Comienza cronometro de ejecución

//Inicia ordenamiento por burbuja
        for (i=1; i<numeros; i++){

            for (j=0; j<numeros-1; j++){
                if (vector[j] > vector[j+1]){
                    temp = vector[j];
                    vector[j] = vector[j+1];
                    vector[j+1] = temp;
                }
            }
        }
        for (i=0; i<numeros; i++){
            cout <<vector[i] << " ";
        }

//tiempo de ejecution
        printf("    ------>  Tiempo total de ejecucion: %f", ((double)clock() - start) / CLOCKS_PER_SEC);

        system("pause");
        return 0;

    }
```

Get URL

options | compilation | execution

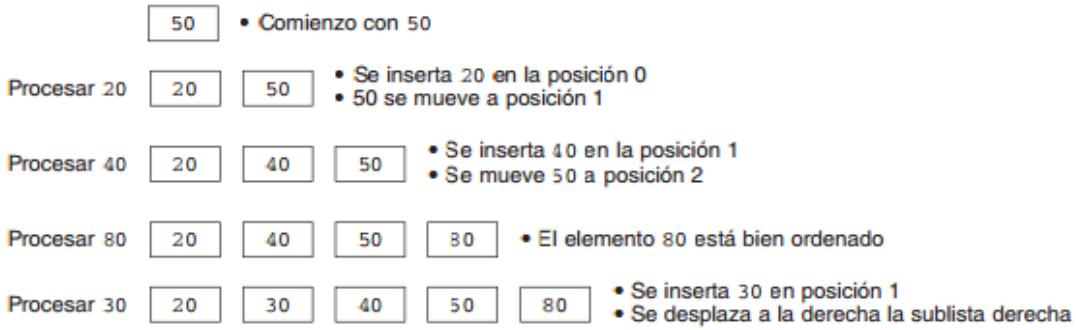
```
-3 0 1 2 3 4 7 8 8 9     ------>  Tiempo total de ejecucion: 0.000038
```

Lenguaje JavaScript



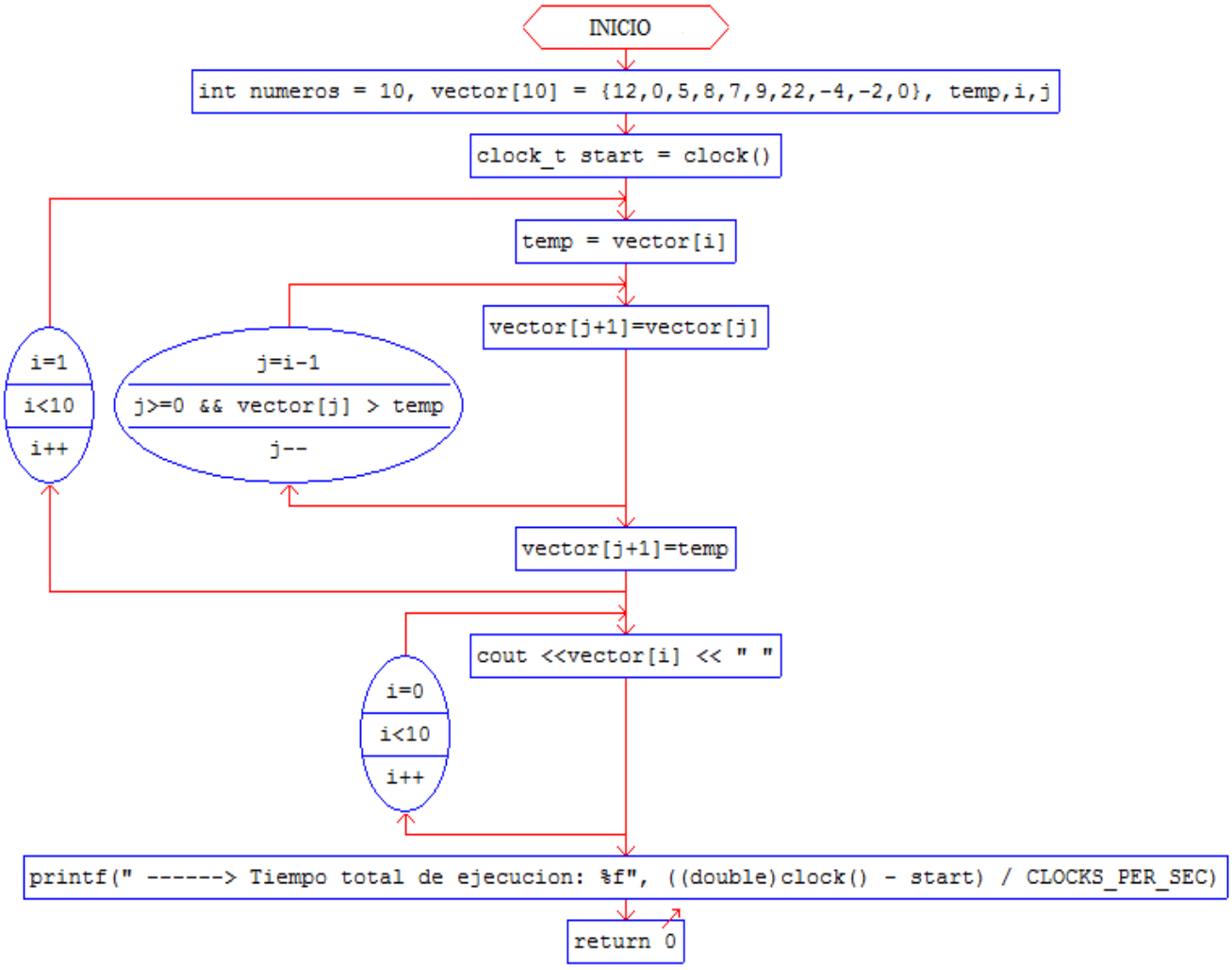
Ordenamiento por Insercion.

El método de ordenación por inserción es similar al proceso típico de ordenar tarjetas de nombres (cartas de una baraja) por orden alfabético, que consiste en insertar un nombre en su posición correcta dentro de una lista o archivo que ya está ordenado. Así el proceso en el caso de la lista de enteros A = 50, 20, 40, 80, 30.



El algoritmo correspondiente a la ordenación por inserción contempla los siguientes pasos:

1. El primer elemento A[0] se considera ordenado; es decir, la lista inicial consta de un elemento.
2. Se inserta A[1] en la posición correcta, delante o detrás de A[0], dependiendo de que sea menor o mayor.
3. Por cada bucle o iteración i (desde i=1 hasta n-1) se explora la sublista A[i-1] . . A[0] buscando la posición correcta de inserción; a la vez se mueve hacia abajo (a la derecha en la sublista) una posición todos los elementos mayores que el elemento a insertar A[i], para dejar vacía esa posición.
4. Insertar el elemento a la posición correcta.



Resultados

Lenguaje C

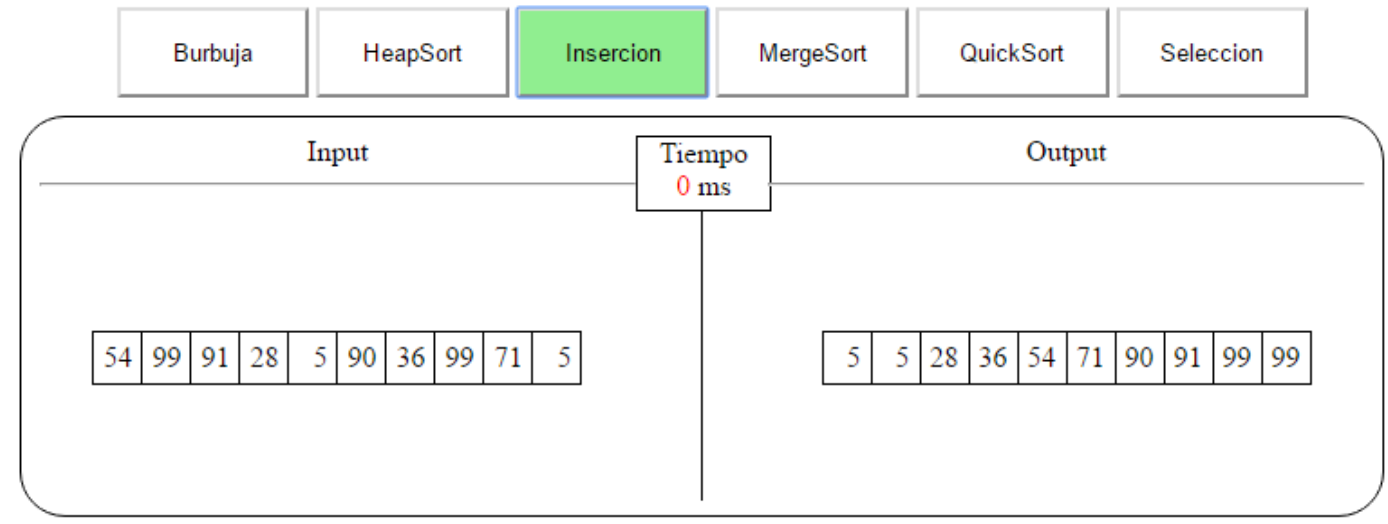
```c
/* ITE - IA
Sergio Antonio Cruz Olivares
Francisco Eduardo Garcia Perea */


#include <stdio.h>
#include <iostream>
#include <time.h>

using namespace std;

int main()
{
        //Declracion de variables y asignacion de valores
        int numeros = 10, vector[numeros] = {12,0,5,8,7,9,22,-4,-2,0}, temp,i,j;

        clock_t start = clock(); //Inicio del cronometro

        //Inicia ciclo de ordenamiento
        for (i=1; i<numeros; i++){
             temp = vector[i];
             for (j=i-1; j>=0 && vector[j] > temp; j--){
                vector[j+1]=vector[j];
             }
             vector[j+1]=temp;
             }

        for (i=0; i<numeros; i++){
            cout <<vector[i] << " ";
        }

        //Finaliza el Conometro

    printf("    ------>  Tiempo total de ejecucion: %f", ((double)clock() - start) / CLOCKS_PER_SEC);

        system("pause");
        return 0;
    }
```

Get URL

options | compilation | execution

```
-4 -2 0 0 5 7 8 9 12 22     ------>  Tiempo total de ejecucion: 0.000040
```

Lenguaje JavaScript

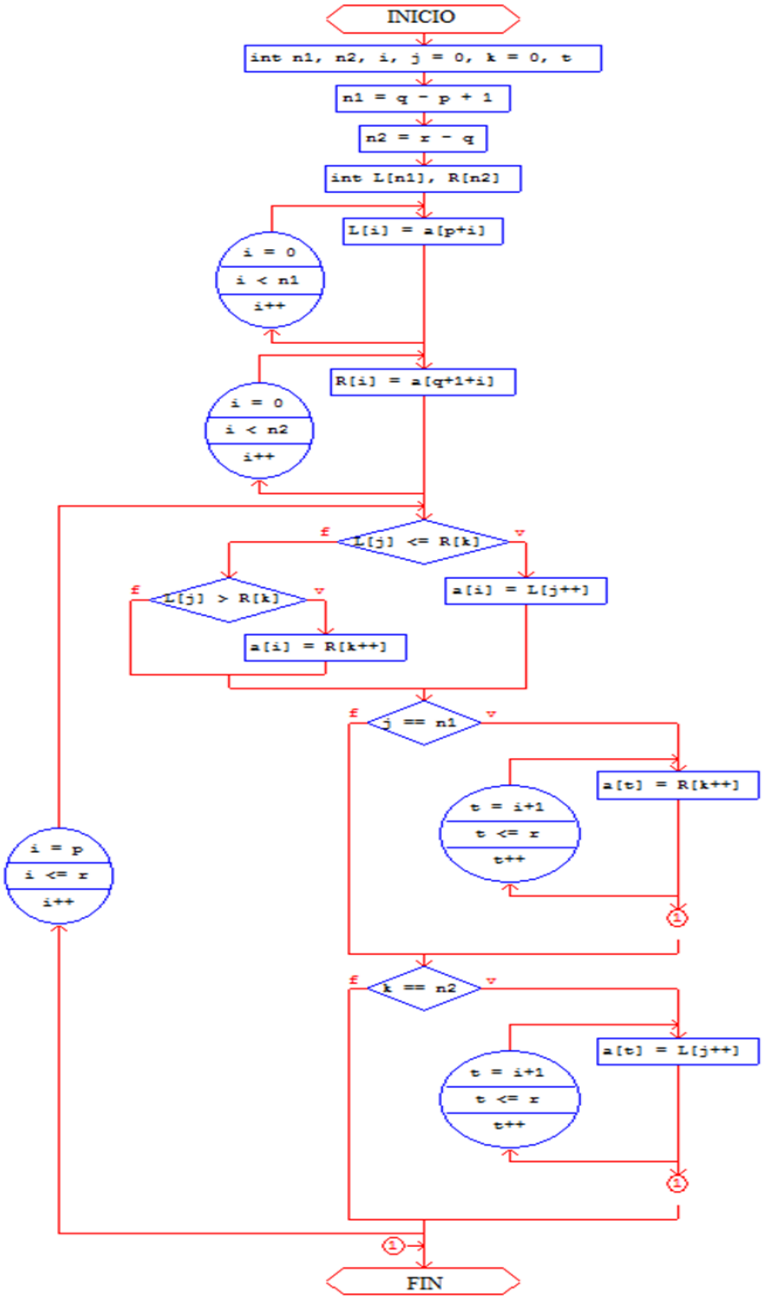


Ordenamiento por Mergesort.

Este método se basa en la siguiente idea:

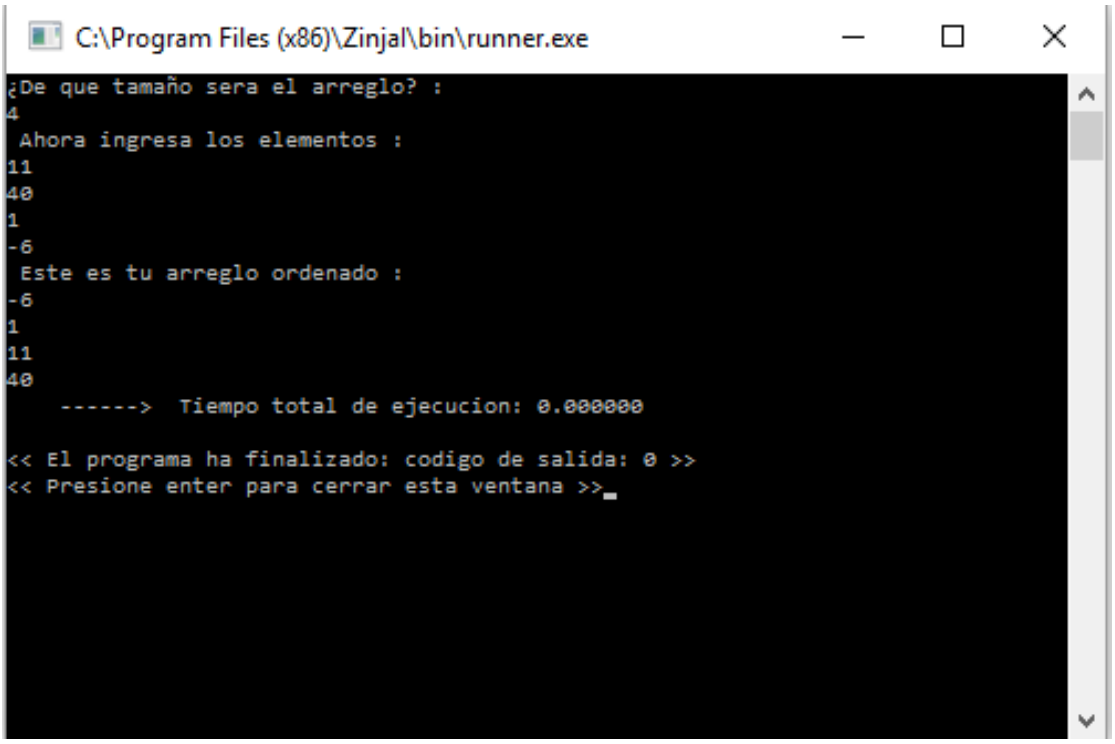
1. Si la lista es pequeña (vacía o de tamaño 1) ya está ordenada y no hay nada que hacer. De lo contrario hacer lo siguiente:
2. Dividir la lista al medio, formando dos sublistas de (aproximadamente) el mismo tamaño cada una.
3. Ordenar cada una de esas dos sublistas (usando este mismo método).
4. Una vez que se ordenaron ambas sublistas, intercalarlas de manera ordenada.

Por ejemplo, si la lista original es [6, 7, -1, 0, 5, 2, 3, 8] deberemos ordenar recursivamente [6, 7, -1, 0] y [5, 2, 3, 8] con lo cual obtendremos [-1, 0, 6, 7] y [2, 3, 5, 8]. Si intercalamos ordenadamente las dos listas ordenadas obtenemos la solución buscada: [-1, 0, 2, 3, 5, 6, 7, 8].

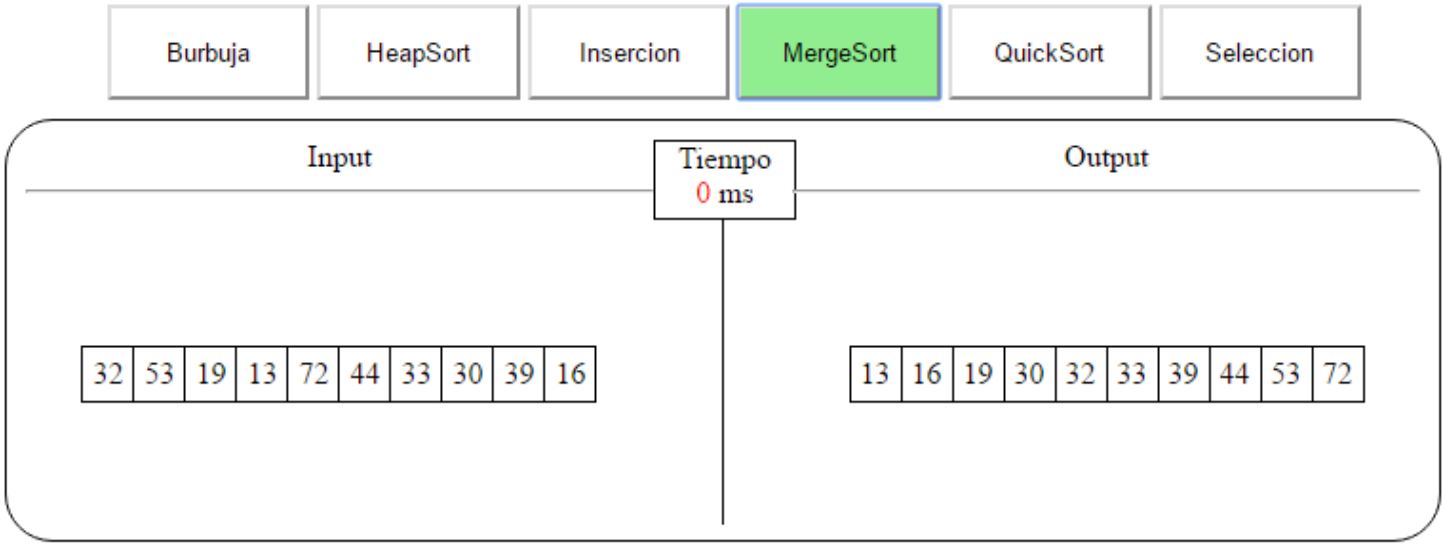


Resultados

Lenguaje C



Lenguaje JavaScript



Ordenamiento por Quicksort.

es uno de los algoritmos más eficientes para ordenar. Este método se basa en la siguiente idea:

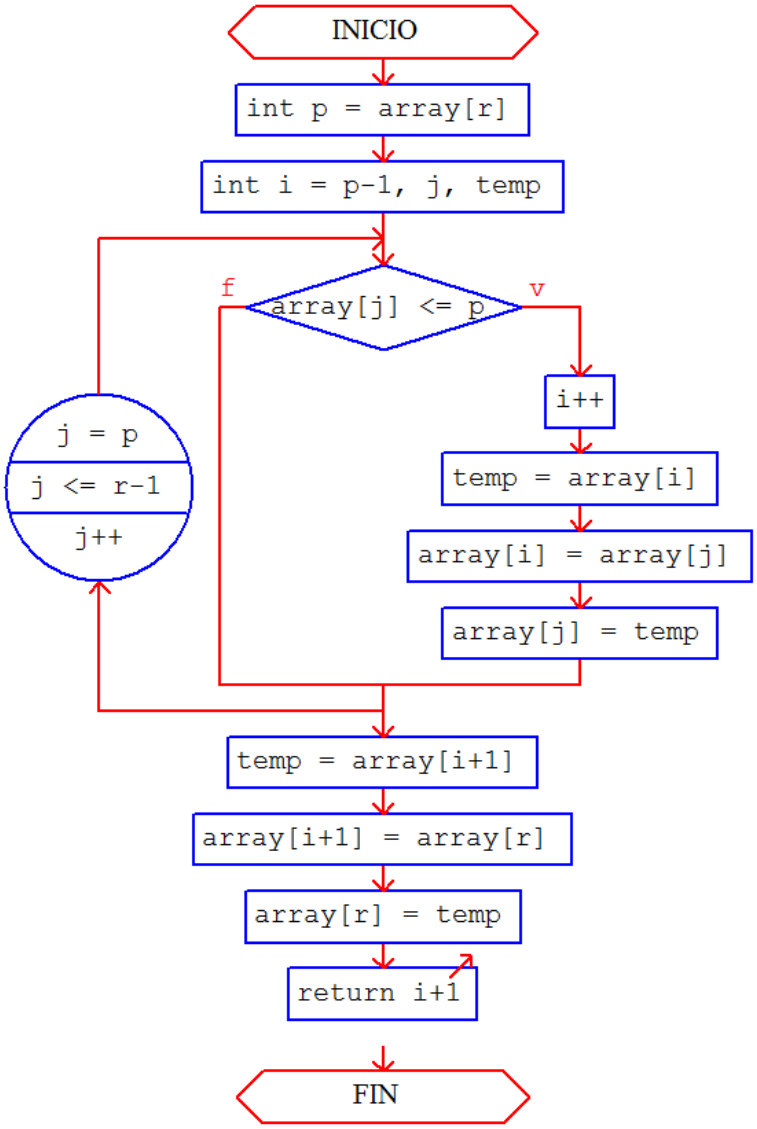
Si la lista es pequeña (vacía o de tamaño 1) ya está ordenada y no hay nada que hacer. De lo contrario hacer lo siguiente:

Tomar un elemento de la lista (por ejemplo el primero) al que llamaremos pivote y armar a partir de esa lista tres sublistas: la de todos los elementos de la lista menores al pivote, la formada sólo por el pivote, y la de los elementos mayores o iguales al pivote, pero sin contarlo al pivote.

Ordenar cada una de esas tres sublistas (usando este mismo método).

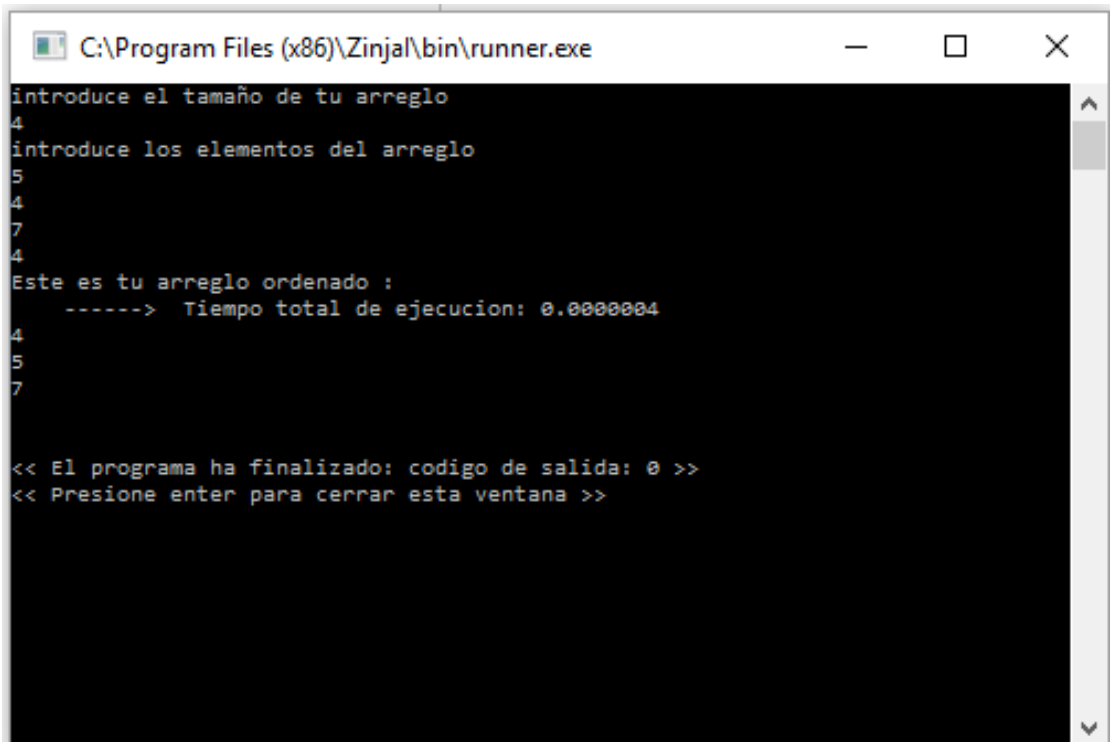
Concatenar las tres sublistas ya ordenadas.

Por ejemplo, si la lista original es [6, 7, -1, 0, 5, 2, 3, 8] consideramos que el pivote es el primer elemento (el 6) y armamos las sublistas [-1, 0, 5, 2, 3], [6] y [7,8]. Se ordenan recursivamente [-1, 0, 5, 2, 3] (obtenemos [-1, 0, 2, 3, 5]) y [7, 8] (obtenemos la misma) y concatenamos en el orden adecuado, y así obtenemos [-1, 0, 2, 3, 5, 6, 7, 8].

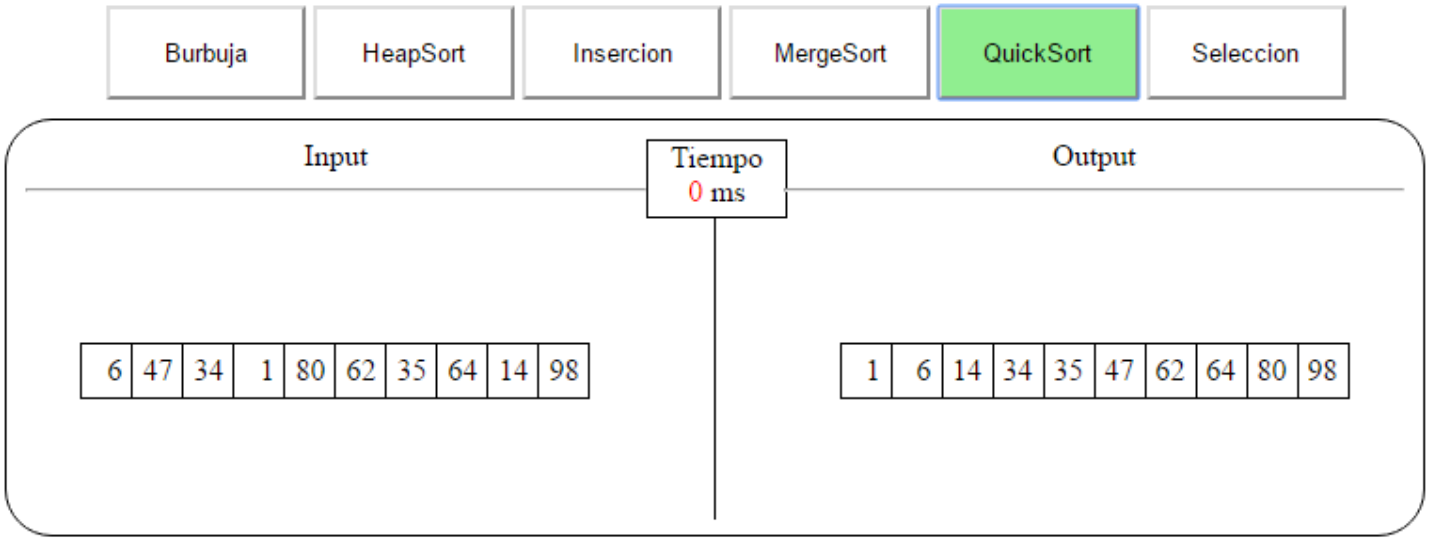


Resultados

Lenguaje C

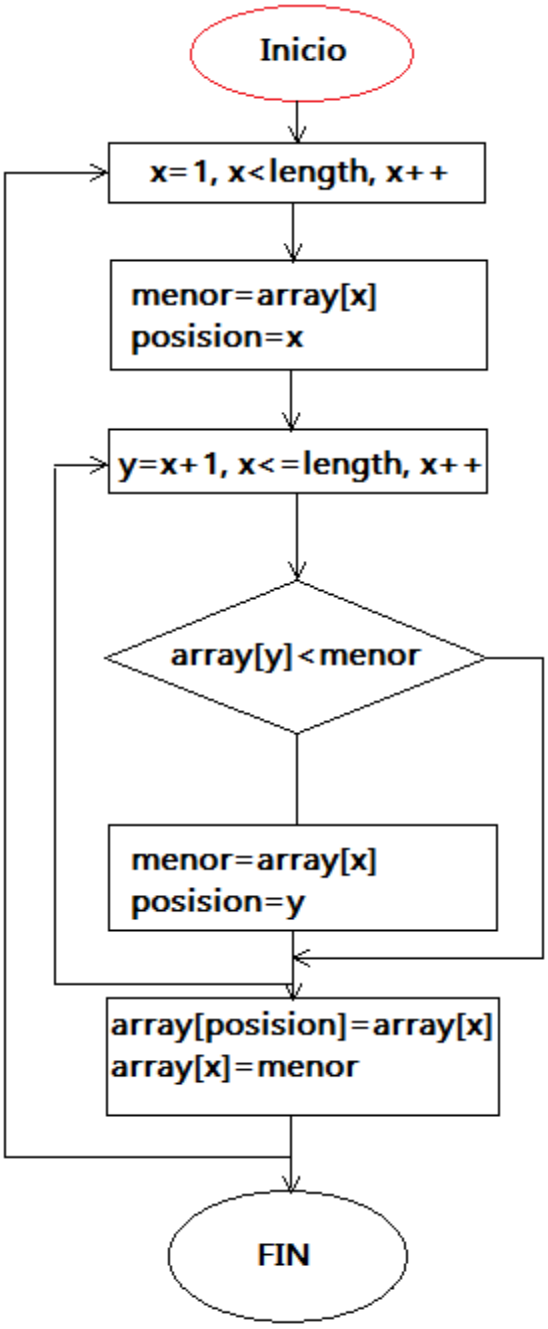


Lenguaje JavaScript



Ordenamiento por 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.



Resultados

Lenguaje C

options | compilation | execution

```
10
-5
3
9
8
8
10
5
3
7
2
Lista: 10       8       9       7       8       3       5       -5      3       2
 Lista ordenada :       -5      2       3       3       5       7       8       8       9       10
Tiempo transcurrido: 0.000017
Exit code: 0 (normal program termination)
```

Lenguaje JavaScript