INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

INGENIERÍA EN SISTEMAS COMPUTACIONALES

Algoritmos y Estructura de Datos

Práctica de Primer Parcial

Sebastián Reyes Núñez

Grupo:

2CV3

Fecha de entrega:

04/11/2022

Código

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>


void bubble_sort(int *A,int n);           //Declaración de funciones
void selection_sort (int *B,int n);
void insertion_sort(int *C, int n);
```

```c
int main(int argc)
{
    int n = 2000;                       //Numero de valores
    int A[n];                           //Definicion de arreglo
    int B[n];
    int C[n];
    clock_t t_start, t_finish;          //variables para medir tiempo
    double t_interval;                  //Variable para obtener resultados de tiempo precisos
    int i;


    for (i=0;i<n;i++)                   //Escanea valores del documento seleccionado
    {
        scanf("%d", &A[i]);
        B[i]=A[i];                      //Asigna valores a todas las matrices
        C[i]=A[i];
    }


    t_start=clock();                    //Llama a funcion Clock
    bubble_sort(A,n);                   //Llama a funcion Bubble Sort
    t_finish=clock();
    t_interval = (double)(t_finish - t_start)/CLOCKS_PER_SEC;           //Ecuacion para obtener resultado de tiempo mas preciso
    printf("\n\nTiempo de bubble sort: %.8f seconds.", t_interval);


    t_start=clock();                    //Llama a funcion Clock
    selection_sort (B,n);               //Llama a funcion Selection Sort
    t_finish=clock();
    t_interval = (double)(t_finish - t_start)/CLOCKS_PER_SEC;           //Ecuacion para obtener resultado de tiempo mas preciso
    printf("\n\nTiempo de selection sort: %.8f seconds.", t_interval);


    t_start=clock();                    //Llama a funcion Clock
    insertion_sort(C, n);               //Llama a funcion Insertion Sort
    t_finish=clock();
    t_interval = (double)(t_finish - t_start)/CLOCKS_PER_SEC;           //Ecuacion para obtener resultado de tiempo mas preciso
    printf("\n\nTiempo de insertion sort: %.8f seconds.", t_interval);
}
```

```c
void bubble_sort(int *A,int n)              //Funcion Bubble Sort
{

    int eliminar=0, i=0, aux=0;

    while(n != eliminar)
    {
      for(i=0; i <((n-1)-eliminar); i++)
        {
          if(A[i]>A[i+1])
          {
             aux=A[i];
             A[i]=A[i+1];
             A[i+1]=aux;
          }
        }
       eliminar++;
    }
    for(i=0;i<n;i++)
        printf("\nA[%d]=%d",i,A[i]);
}
```

```c
void selection_sort (int *B,int n)              //Funcion Selection Sort
{
    int aux=0, i=0, j=0, aux2=0;

    for(i=0; i<n; i++)
    {
        aux2=i;
        for(j=i+1; j<n; j++)
        {
            if(B[j]<B[aux2])
            {
                aux2=j;
            }
        aux=B[i];
        B[i]=B[aux2];
        B[aux2]=aux;
        }
    }
    for(i=0;i<n;i++)
        printf("\nB[%d]=%d",i,B[i]);
}
```

```c
void insertion_sort(int *C, int n)                  //Funcion Insertion Sort
{

    for (int i = 1; i < n;i++){
        int key = C[i];
        int j = i-1;

        while (j>=0 && C[j]>key){
            C[j+1]= C[j];
            j = j-1;
        }
        C[j+1]= key;
    }
    for(int i=0;i<n;i++)
        printf("\nC[%d]=%d",i,C[i]);
}
```

Resultados

Bubble sort:

```
A[1972]=71
A[1973]=72
A[1974]=72
A[1975]=72
A[1976]=72
A[1977]=73
A[1978]=73
A[1979]=73
A[1980]=73
A[1981]=74
A[1982]=75
A[1983]=75
A[1984]=75
A[1985]=75
A[1986]=76
A[1987]=76
A[1988]=76
A[1989]=76
A[1990]=77
A[1991]=77
A[1992]=78
A[1993]=79
A[1994]=80
A[1995]=81
A[1996]=88
A[1997]=90
A[1998]=90
A[1999]=90

Tiempo de bubble sort: 0.52700000 seconds.
```

Selection Sort:

```
B[1972]=71
B[1973]=72
B[1974]=72
B[1975]=72
B[1976]=72
B[1977]=73
B[1978]=73
B[1979]=73
B[1980]=73
B[1981]=74
B[1982]=75
B[1983]=75
B[1984]=75
B[1985]=75
B[1986]=76
B[1987]=76
B[1988]=76
B[1989]=76
B[1990]=77
B[1991]=77
B[1992]=78
B[1993]=79
B[1994]=80
B[1995]=81
B[1996]=88
B[1997]=90
B[1998]=90
B[1999]=90

Tiempo de selection sort: 0.59200000 seconds.
```

Insertion Sort:

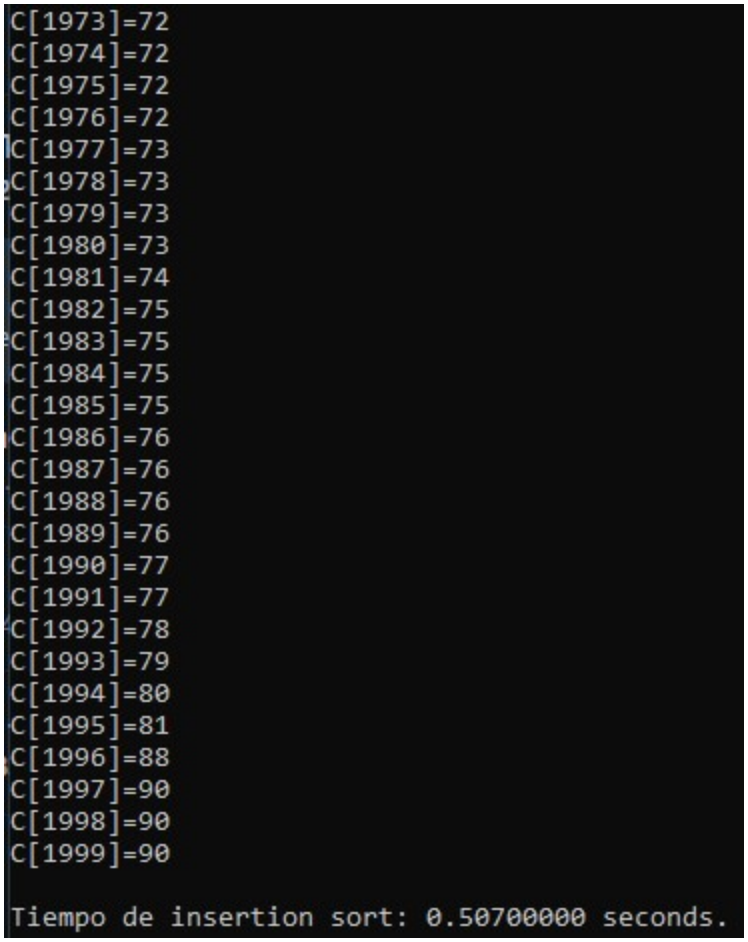

Tabla Comparativa

Función	Tiempo
Bubble Sort	**0.52700000 seg**
Selection Sort	**0.59200000 seg**
Insertion Sort	**0.50700000 seg**

Analisis

En este caso el código con menor tiempo de ejecución fue el de Insertion Sort. Esto se debe a que la forma en que el algoritmo trabaja, mueve a varios valores a la vez asegurándose que el valor que fue colocado en la posición izquierda es el menor. También con este proceso, se recorren los valores mayores poco a poco hacia la derecha del arreglo y con esto recortando el número de pasadas necesarias para poder ordenar la información. De esta forma, ahorrando tiempo en el proceso de ordenamiento. La diferencia se podría apreciar más si se estuvieran arreglando una cantidad mucho más grande de valores, por ejemplo alrededor de los 100,000 se empezaría a ver una gran diferencia entre los diferentes algoritmos.