

Algoritmos de Ordenamiento

Tarea 2: Bubble sort e Insertion sort

José Alberto Villa García

8vo cuatrimestre

18 de mayo de 2019

1. Teoría

Teniendo un arreglo de datos (en este caso, de números se va a querer ordenar al arreglo de alguna manera (ascendente o descendente, por ejemplo).

2. Planteamiento del problema

Se tiene un arreglo de n números enteros que puede o no estar desordenado y se busca ordenarlo de la manera más eficiente posible

3. Solución

Existen varios algoritmos que toman como entrada un arreglo de números y lo ordenan de alguna manera.

3.1. Bubble sort

Bubble sort es el algoritmo más simple, pues sólo intercambia los elementos adyacentes si están en el orden equivocado.

Ejemplo:

Primer recorrido:

(5 1 4 2 8) \rightarrow (1 5 4 2 8), Aquí, el algoritmo compara los primeros dos elementos e intercambia $5 > 1$
(1 5 4 2 8) \rightarrow (1 4 5 2 8), Intercambiar $5 > 4$
(1 4 5 2 8) \rightarrow (1 4 2 5 8), Intercambiar $5 > 2$
(1 4 2 5 8) \rightarrow (1 4 2 5 8), Ahora, como los elementos $8 > 5$ ya están en orden, se quedan igual

Segundo recorrido:

(1 4 2 5 8) \rightarrow (1 4 2 5 8)
(1 4 2 5 8) \rightarrow (1 2 4 5 8), Intercambiar $4 > 2$
(1 2 4 5 8) \rightarrow (1 2 4 5 8)
(1 2 4 5 8) \rightarrow (1 2 4 5 8)

En este punto el arreglo ya está ordenado, pero como el algoritmo no lo sabe todavía tiene que dar otro pase sin intercambiar nada para salir.

Tercer recorrido:

(1 2 4 5 8) \rightarrow (1 2 4 5 8)
(1 2 4 5 8) \rightarrow (1 2 4 5 8)
(1 2 4 5 8) \rightarrow (1 2 4 5 8)
(1 2 4 5 8) \rightarrow (1 2 4 5 8)

3.2. Insertion sort

Insertion sort es un algoritmo de ordenamiento que funciona de la misma manera en que usualmente ordenamos cartas en nuestra mano, seleccionando una carta de valor mayor y desplazándola lo más que se puede a la izquierda.

4. Code

4.1. Bubble sort

4.1.1. include

```
#pragma once

#include "Vector.h"

/**
 * Description:
 *
 * Sample usage:
 *
 */
namespace Bubble {
    /**
     * @brief
     * @param
     */
    void
        sort(Vector& v);
}
```

4.1.2. source

```
#include "Bubble.h"

void
Bubble::sort(Vector& v) {
    int temp = 0;
    for(size_t i = 0; i < v.m_vector.size() - 1; ++i) {
        for(size_t j = 0; j < v.m_vector.size() - i - 1; ++j) {
            if(v.m_vector[j] > v.m_vector[j + 1]) {
                temp = v.m_vector[j];
                v.m_vector[j] = v.m_vector[j + 1];
                v.m_vector[j + 1] = temp;
            }
        }
    }
}
```

4.2. Insertion sort

4.2.1. include

```
#pragma once

#include "Vector.h"
```

```

/**
 *   Description:
 *
 *   Sample usage:
 *
 */
namespace Insert {
    /**
     *   @brief
     *   @param
     */
    void
        sort(Vector& v);
}

```

4.2.2. source

```

#include "Insert.h"

void
Insert::sort(Vector& v) {
    int key = 0, j = 0;
    for(size_t i = 1; i < v.m_vector.size(); ++i)
    {
        key = v.m_vector[i];
        j = i - 1;
        while(j >= 0 && v.m_vector[j] > key)
        {
            v.m_vector[j + 1] = v.m_vector[j];
            j = j - 1;
        }
        v.m_vector[j + 1] = key;
    }
}

```

5. Benchmark