Análisis y Algoritmos

Roberto Charreton Kaplun
Universidad de Artes Digitales

Guadalajara, Jalisco

Email: idv17c.rcharreton@uartesdigitales.edu.mx

Profesor: Efraín Padilla

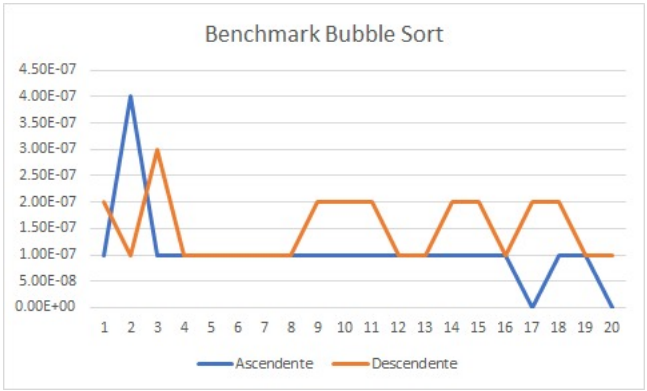
Mayo 5, 2019

1) Benchmarking Bubble Sort
Bubble Sort Funciona revisando cada elemento de la lista que va a ser ordenada con el siguiente, intercambiándolos 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

Desarrolle el codigo leyendo el vector con los multiples casos Ascendente, Descente o Random, de estos dos obtuve el mejor y el peor caso en su benchmarking.

Codigo

```
vector<int> CManager::BubbleSort(vector<int> Vector)
{
        for (size_t i = 1; i < Vector.size(); i++)
        {
                for (size_t j = 0; j < Vector.size() - 1; j++)
                {
                        if (Vector[j] > Vector[j+1])
                        {
                                int temp = Vector[j];
                                Vector[j] = Vector[j + 1];
                                Vector[j + 1] = temp;
                        }
                }
        }
        return Vector;
}
```



2) Benchmarking Bubble Sort

Bubble Sort 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. Requiere O(n^2)$operacionesparaordenarunalistadenelementos$.

Desarrolle el codigo leyendo el vector con los multiples casos Ascendente, Descente o Random, de estos dos obtuve el mejor y el peor caso en su benchmarking.

Codigo

```
vector<int> CManager::InsertionSort(vector<int> Vector)
{// Complejidad N, Mejor caso Ordenado N, Peor caso Desordenado N^2,
        if (Vector.size() > 1)
        {
                for (int i = 1; i < Vector.size(); ++i) {
                        int j = i - 1;
                        int temp = Vector[i];
                        while (j >= 0 && Vector[j] > temp) {
                                Vector[j + 1] = Vector[j];
                                --j;
                        }
                        Vector[j + 1] = temp;
                }
        }
        return Vector;
}
```