

# Algoritmos de Ordenamiento II

Victor Hugo Flores Márquez  
Universidad de Artes Digitales

Guadalajara, Jalisco

Email: idv16c.vflores@uartesdigitales.edu.mx

Profesor: Efraín Padilla

Mayo 18, 2019

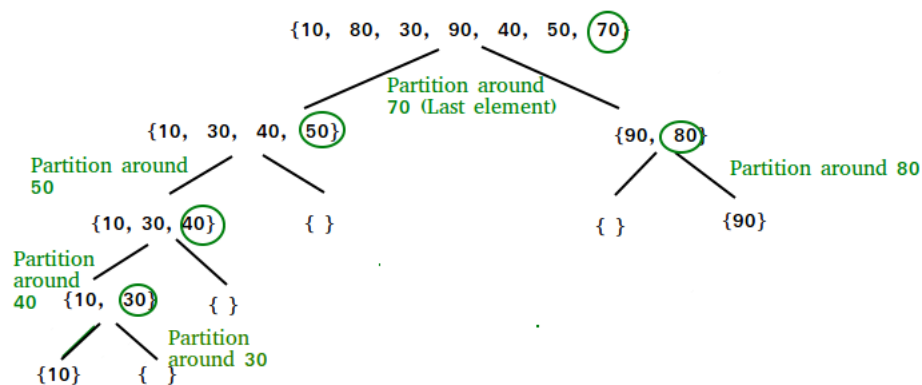
## I. INTRODUCCIÓN

En computación y matemáticas un algoritmo de ordenamiento es un algoritmo que pone elementos de una lista o un vector en una secuencia dada por una relación de orden, es decir, el resultado de salida ha de ser una re-ordenamiento de la entrada que satisfaga la relación de orden dada.

### 1) Quicksort

Quicksort es un algoritmo de ordenación considerado entre los más rápidos y eficientes. El algoritmo usa la técnica divide y vencerás que básicamente se basa en dividir un problema en subproblemas y luego juntar las respuestas de estos subproblemas para obtener la solución al problema central.

#### Ejemplo:



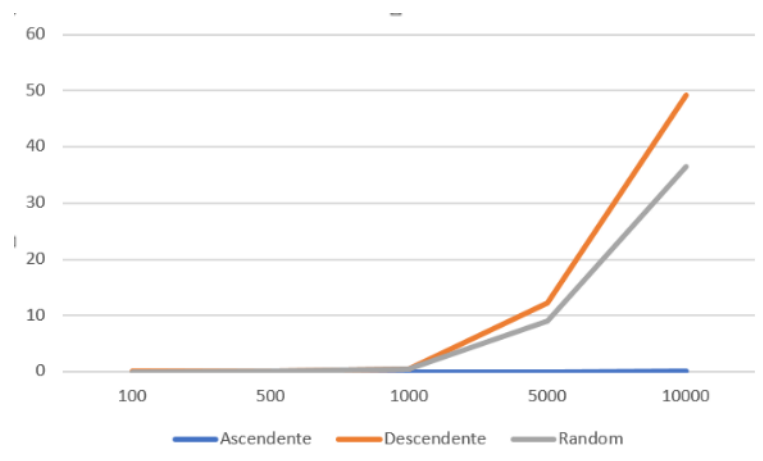
## Codigo C++:

```

1 partition( std::vector<int>& numbers, int start, int end )
2 {
3     unsigned int pivot = numbers[start];
4     unsigned int left = start + 1;
5     unsigned int right = end;
6     unsigned int tmp;
7
8     while ( left != right )
9     {
10        if ( numbers[left] <= pivot ) left++;
11        else
12        {
13            while ( ( left != right ) && ( pivot < numbers[right] ) ) right--;
14            tmp = numbers[right];
15            numbers[right] = numbers[left];
16            numbers[left] = tmp;
17        }
18    }
19
20    if ( numbers[left] > pivot ) left--;
21    numbers[start] = numbers[left];
22    numbers[left] = pivot;
23
24    return ( left );
25 }
26
27
28
29 quickSort( std::vector<int>& numbers, int left, int right )
30 {
31     if ( left < right )
32     {
33         int pi = partition( numbers, left, right );
34
35         quickSort( numbers, left, pi - 1 );
36         quickSort( numbers, pi + 1, right );
37     }
38
39     return numbers;
40 }
41

```

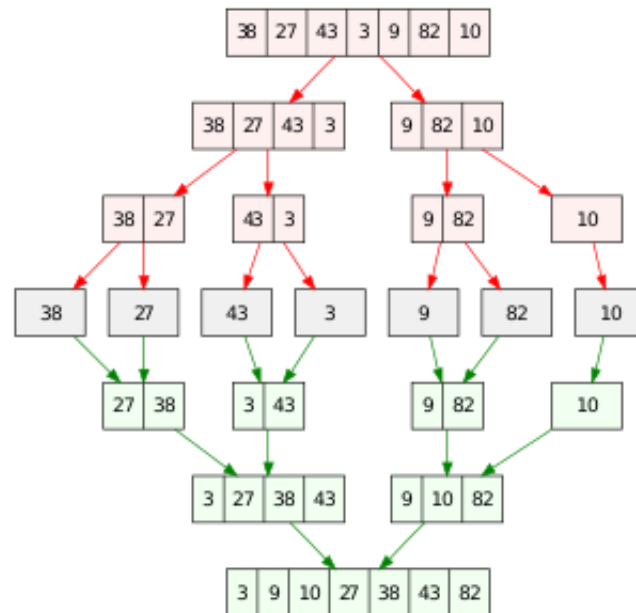
## BenchMark:



## 2) Mergesort

Mergesort usa la técnica divide y vencerás. Divide el vector en dos mitades, se llama a sí mismo para las dos mitades y luego fusiona las dos mitades clasificadas.

**Ejemplo:**



**Codigo C++:**

```

1 mergeSort( std::vector<int>& numbers, int left, int right )
2 {
3     if ( left < right )
4     {
5         int mid = left + ( right - left ) / 2;
6
7         mergeSort( numbers, left, mid );
8         mergeSort( numbers, mid+ 1, right );
9
10        merge( numbers, left, mid, right );
11    }
12    return numbers;
13 }
14
15
16 merge( std::vector<int>& numbers, int left, int mid, int right )
17 {
18     int i, j, k;
19     int n1 = mid - left + 1;
20     int n2 = right - mid;
21
22     std::vector<int> Left, Right;
23
24     for ( i = 0; i < n1; i++ )
25         Left.push_back( numbers[ left + i ] );
26     for ( j = 0; j < n2; j++ )
27         Right.push_back( numbers[ mid + 1 + j ] );
28
29     i = 0;
30     j = 0;
31     k = left;
32
33     while ( i < n1 && j < n2 )
34     {
35         if ( Left[i] <= Right[j] )
36         {
37             numbers[k] = Left[i];
38             i++;
39         }
40         else
41         {
42             numbers[k] = Right[j];
43             j++;
44         }
45         k++;
46     }
47
48     while ( i < n1 )
49     {
50         numbers[k] = Left[i];
51         i++;
52         k++;
53     }
54
55     while ( j < n2 )
56     {
57         numbers[k] = Right[j];
58         j++;
59         k++;
60     }
61 }
62

```

BenchMark:

