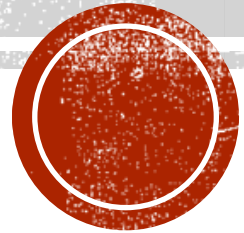
Sort algorithms comparison

Merge Sort and Insertion sort



Joaquín Sanchíz Navarro
Sergio García de la Iglesia

SORT ALGORITHMS

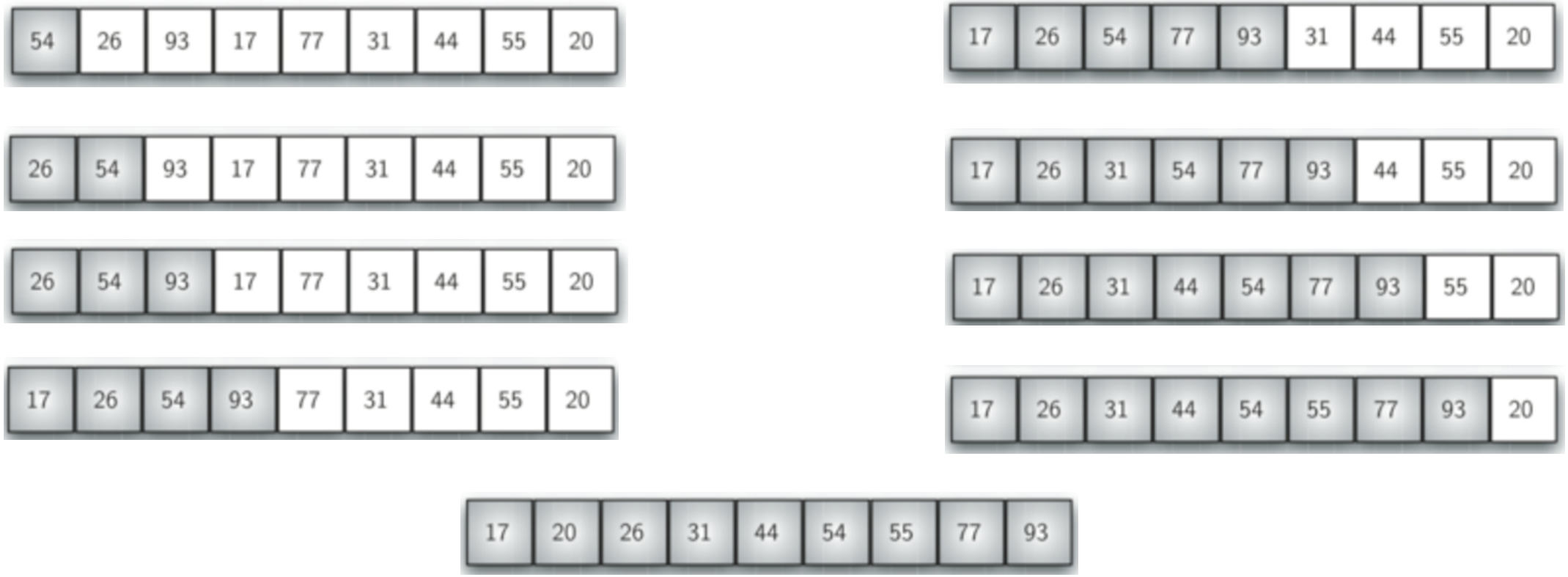
- Description

- Algorithm types
  - *Internal or External.*
  - *Natural or Non-natural.*
  - *Steadiness.*

- Characteristics
  - *Computational complexity.*
  - *Memory's use and other computational resources.*

INSERTION SORT

```
function insertionSort(array A)
    for i from 1 to length[A]-1 do
        value := A[i]
        j := i-1
        while j >= 0 and A[j] > value do
            A[j+1] := A[j]
            j := j-1
        done
        A[j+1] = value
    done
```



INSERTION SORT PROCEDURE

ANALYSIS OF INSERTION SORT

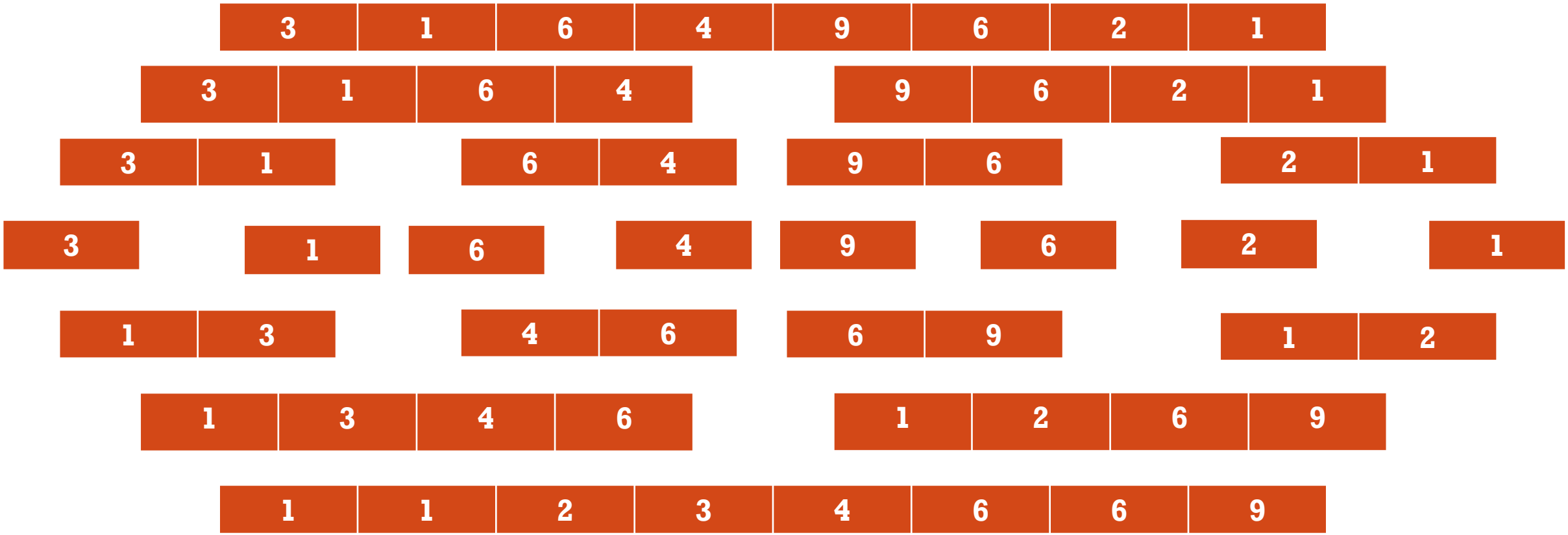
$$c \cdot 1 + c \cdot 2 + c \cdot 3 + \cdots c \cdot (n-1) = c \cdot (1 + 2 + 3 + \cdots + (n-1))$$

$$c \cdot (n - 1 + 1)((n-1)/2) = cn^2/2 - cn/2 \,.$$

$$\Theta(n^2)$$

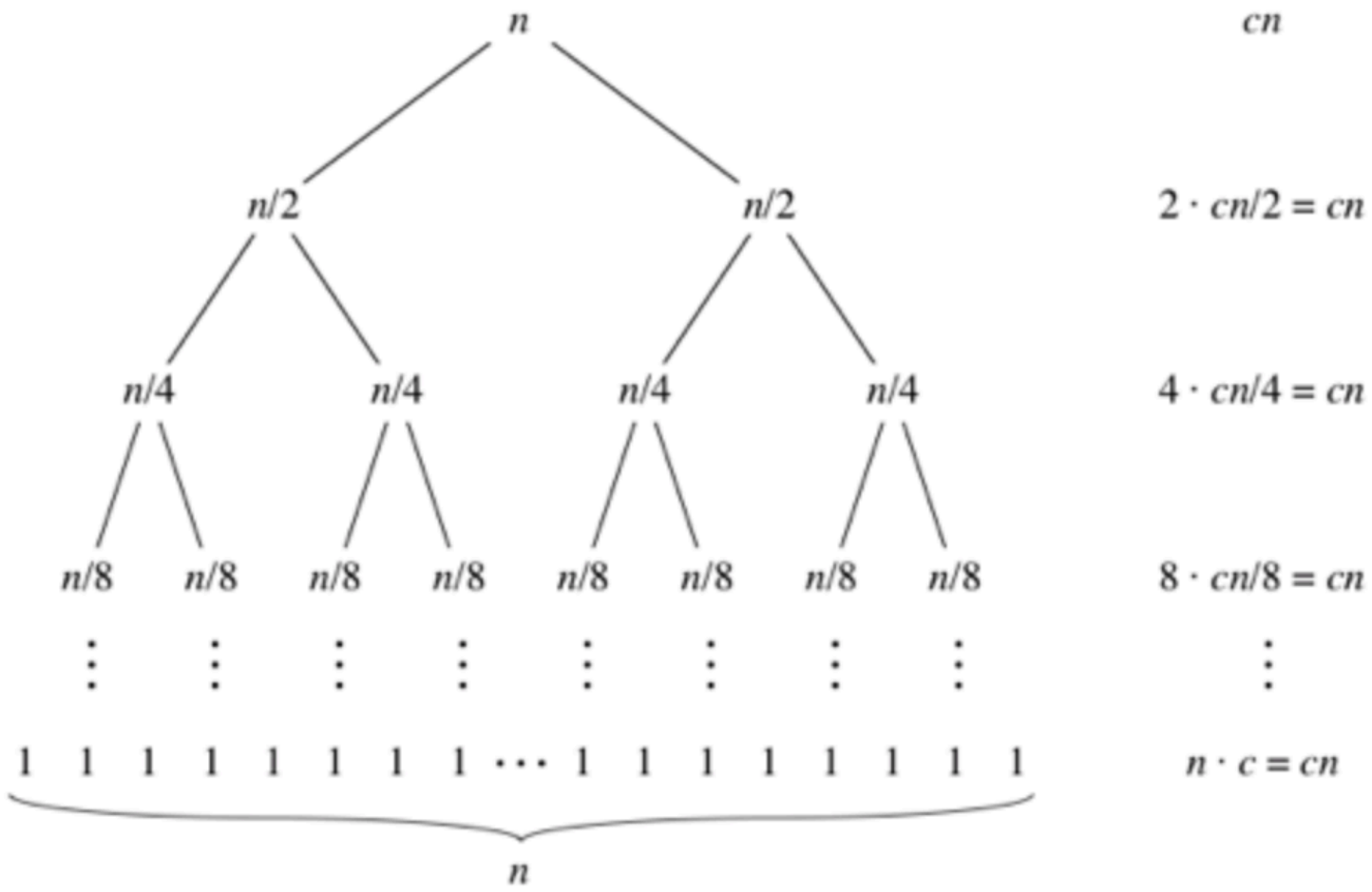
MERGE SORT

```java
public static void mergesort(Integer[] numeros, int izq, int der){
  if (izq < der){
    int m = (izq + der) / 2;
    mergesort(numeros, izq, m);
    mergesort(numeros, m + 1, der);
    merge(numeros, izq, m, der);
  }
}

public static void merge(Integer[] numeros, int izq, int m, int der){
  int i, j, k;

  Integer[] auxiliar = new Integer[numeros.length]; //array auxiliar

  for (i = izq; i <= der; i++) //copia ambas mitades en el array auxiliar
    auxiliar[i] = numeros[i];

  i = izq; j = m + 1; k = izq;

  while (i <= m && j <= der) //copia el siguiente elemento más grande
    if (auxiliar[i] <= auxiliar[j])
      numeros[k++] = auxiliar[i++];

    else
      numeros[k++] = auxiliar[j++];

  while (i <= m) //copia los elementos que quedan de la
    numeros[k++] = auxiliar[i++]; //primera mitad (si los hay)
}
```



MERGE SORT PROCEDURE

RECURRENCY TREE



- ¿How many levels are?

  $\frac{n}{2^i} = 1$

  $n = 2^i$

  log(n) = i (last level)

  log(n) + 1 levels

- ¿T(n)?

  (log (n) + 1) * cn

  cn * log(n) + cn

  O (n * log(n))

MASTER METHOD

Método maestro

$$T(n) = aT(n/b) + \epsilon(n^d)$$

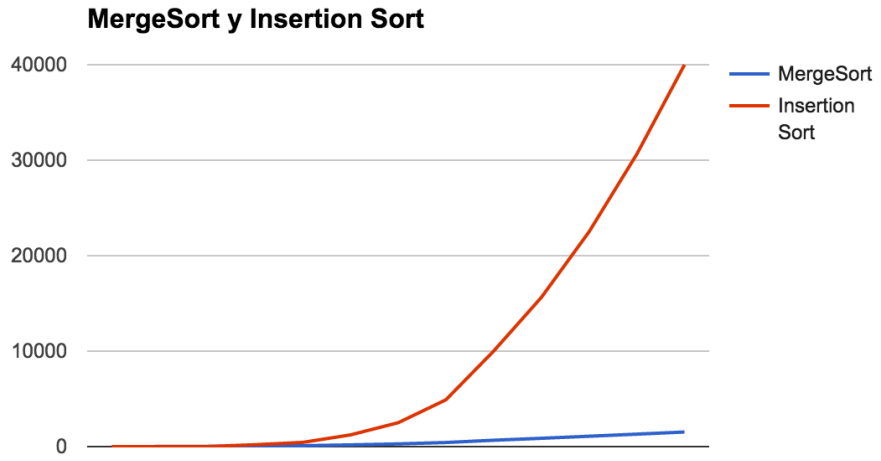
$$T(n) = 2T(n/2) + \epsilon(n)$$

$$a == b^d \rightarrow \begin{cases} a = 2 \\ b = 2 \\ d = 1 \end{cases} \qquad 2 = 2^1 \checkmark \qquad (\text{caso } 1)$$

$$\boxed{O(n \log(n))}$$

MERGE SORT VS INSERTION SORT

- Why Merge Sort is usually better than Insertion Sort
- Case where Insertion is better
- Optimization of Merge Sort: including Insertion Sort to sort.



THANKS YOU!

- Bibliography
  - http://rosettacode.org/wiki/Sorting_algorithms/Merge_sort
  - http://rosettacode.org/wiki/Sorting_algorithms/Insertion_sort
  - https://es.wikipedia.org/wiki/Ordenamiento_por_mezcla
  - https://es.wikipedia.org/wiki/Ordenamiento_por_inserción
  - Apuntes de otros años.