



COMPARACIÓN ENTRE LOS ALGORITMOS DE ORDENACIÓN INSERCIÓN BINARIA Y RADIX SORT

**Y SU MEDICIÓN EN TÉRMINOS DE
EFICIENCIA**

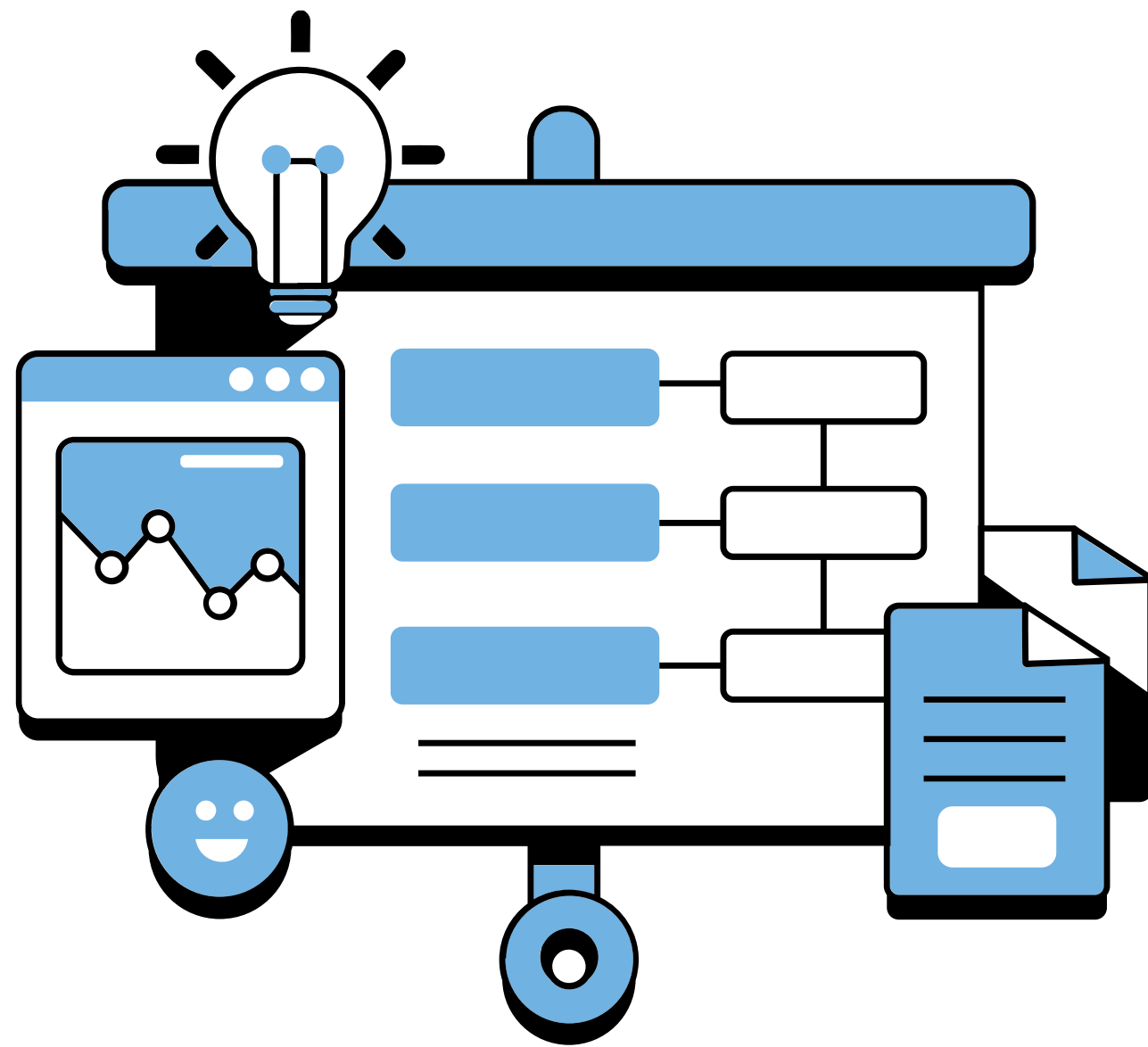
Proyecto Realizado por:

Elvis Juan Yana Cayo

Josué Adrián Sosa Cruz

Diesdanderson Dudu Vela Laurente

INDICE



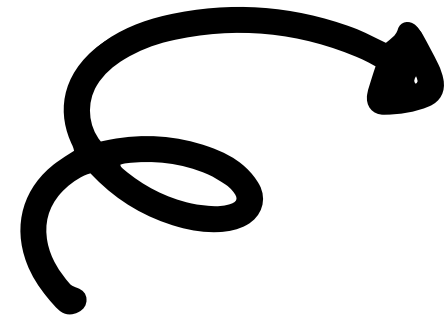
01. INTRODUCCION

02. INSERCIÓN BINARIA

03. RADIX SORT

04. COMPARACIÓN DE ORDENAMIENTO

05. COMPARACIONES E INTERCAMBIOS



INTRODUCCIÓN

Los métodos de ordenación son importantes para la informática puesto que nos ayuda a organizar y optimizar la búsqueda, análisis y la manipulación de información. Entre los diversos algoritmos existentes, el método de ordenación Binaria y el Radix sort tienen dos enfoques distintos, el primero se basa en comparaciones y el segundo en el procesamiento por sus dígitos.

El análisis de algoritmos permite determinar qué tan eficientes son las soluciones al ordenar o procesar datos dentro de un programa. En este trabajo se comparan dos métodos de ordenación con enfoques distintos: Inserción Binaria, que realiza comparaciones usando búsqueda binaria, y Radix Sort, que ordena según los dígitos sin comparar directamente los elementos. El objetivo es medir su rendimiento en tiempo y operaciones, identificando cuál resulta más eficiente según el tamaño y el tipo de los datos.

INSERCIÓN BINARIA

- El algoritmo de inserción binaria constituye una mejora del método de inserción directa (insertion sort). La diferencia principal radica en que, en lugar de recorrer secuencialmente la parte ya ordenada del arreglo para encontrar la posición de inserción del elemento actual, se aplica una búsqueda binaria (binary search) para determinar la posición correcta, reduciendo así el número de comparaciones necesarias

```
Funcion InsercionBin(A, n)
  Para i = 2 Hasta n Hacer
    aux = A[i]
    izq = 1
    der = i - 1
    Mientras izq <= der Hacer
      m = entero((izq + der)/2)
      Si aux < A[m] Entonces
        der = m - 1
      SiNo
        izq = m + 1
      FinSi
    FinMientras
    j = i - 1
    Mientras j >= izq Hacer
      A[j+1] = A[j]
      j = j - 1
    FinMientras
    A[izq] = aux //inserción
  FinPara
FinFuncion
```

RADIX SORT

El Radix Sort es un algoritmo de ordenamiento no comparativo que organiza los datos según los dígitos individuales de los números. En cada iteración, los elementos se agrupan de acuerdo con el valor de un dígito ya sea de menor a mayor significancia o en sentido inverso, y luego se combinan nuevamente. Este proceso se repite hasta procesar todos los dígitos.

```
void RadixSort(long long A[], long long n, long long &comparaciones, long long &movimientos) {
    long long max = obtenerMax(A, n, comparaciones);

    for (long long peso = 1; max / peso > 0; peso = peso * 10) {
        comparaciones++;

        long long urna[10][n];
        long long cuenta[10] = {0};

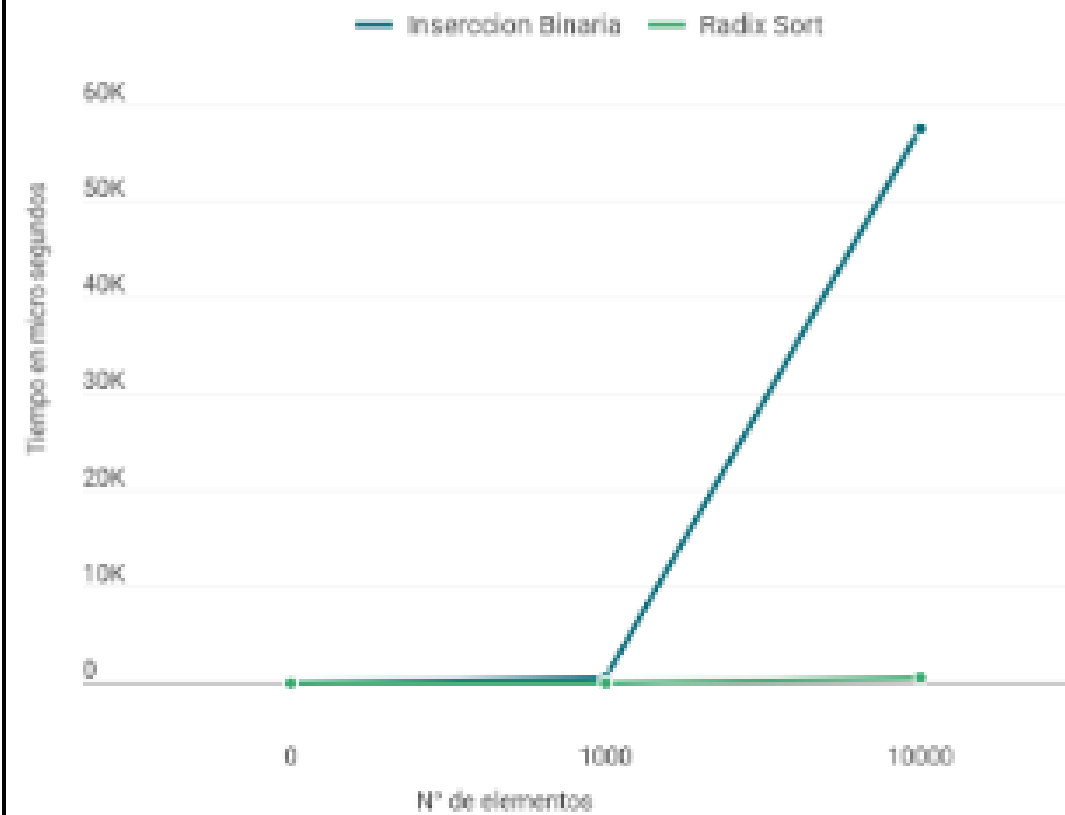
        for (long long i = 0; i < n; i++) {
            long long digito = (A[i] / peso) % 10;
            comparaciones++;
            urna[digito][cuenta[digito]++] = A[i];
            movimientos++;
        }

        long long idx = 0;
        for (long long i = 0; i < 10; i++) {
            for (long long j = 0; j < cuenta[i]; j++) {
                A[idx++] = urna[i][j];
                movimientos++;
            }
        }
    }
}
```

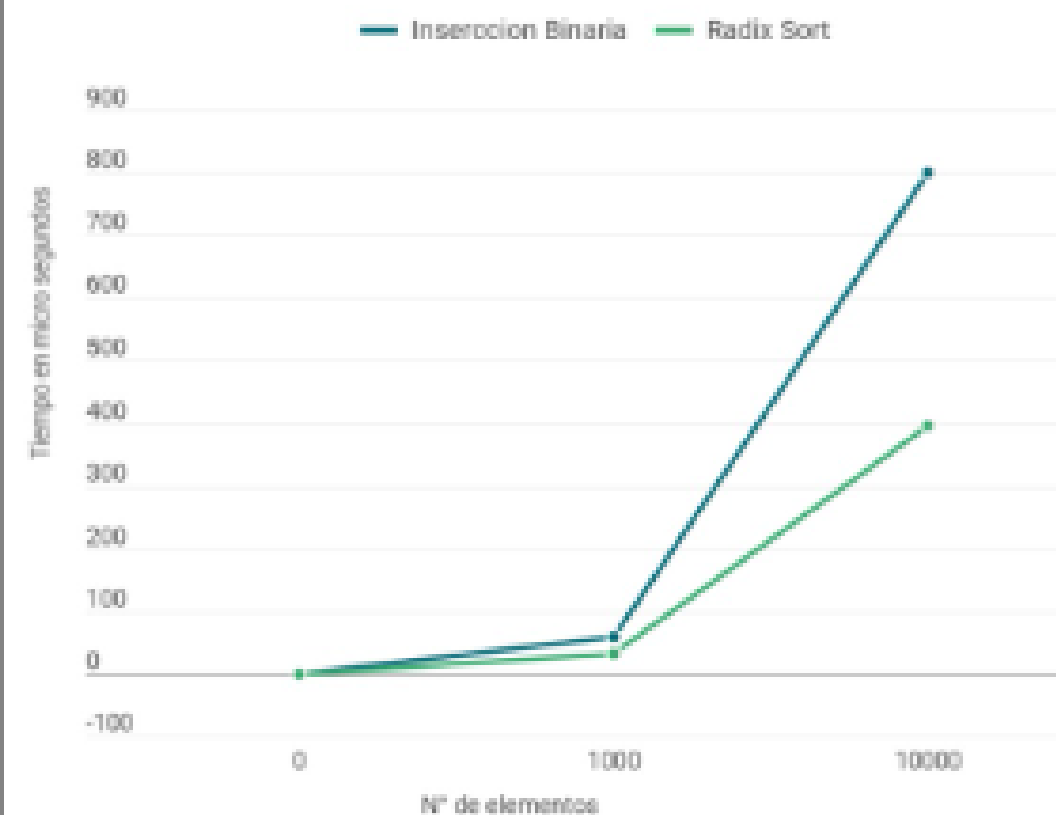


COMPARACION DE ORDENAMIENTOS

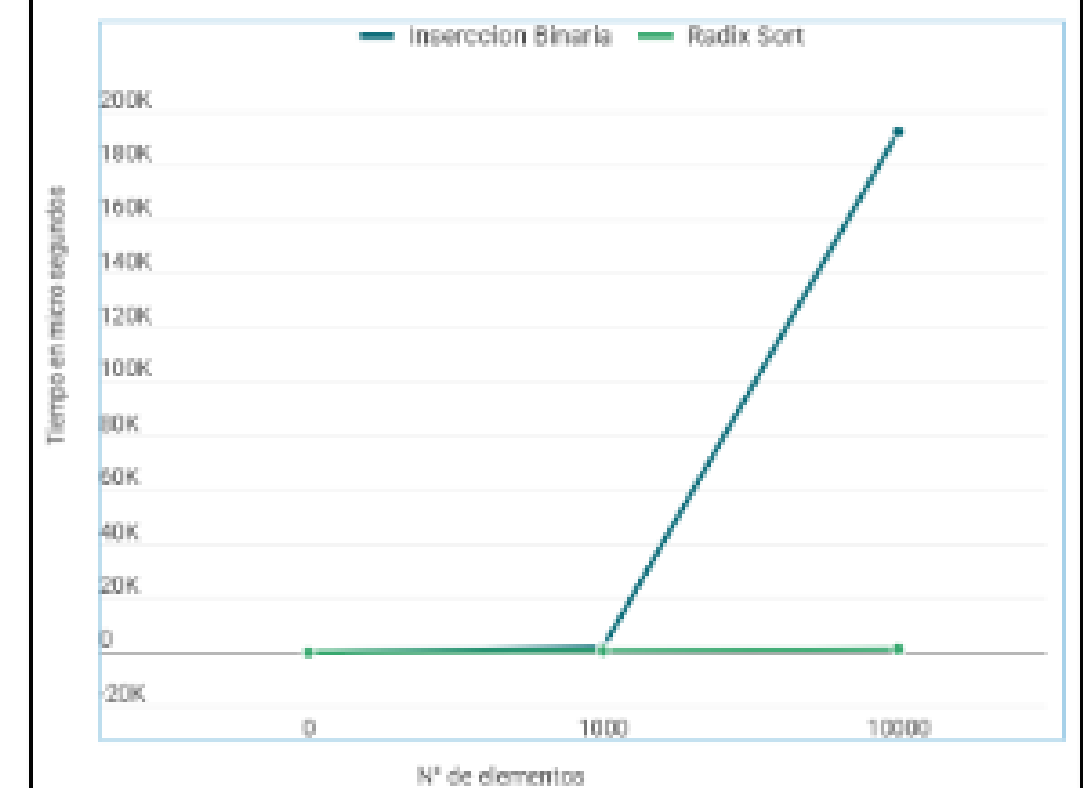
Aleatoriamente



Orden Ascendente



Orden Descendente



COMPARACIONES E INTERCAMBIOS


Algoritmo	Tamaño	Aleatorio	Ascendente	Descendente
Inserción Binaria	1000	8,571	8,977	7,987
Inserción Binaria	10000	118,938	123,617	113,631
Radix Sort	1000	5,003	5,003	5,003
Radix Sort	10000	60,004	60,004	60,004



COMPARACIONES

INTERCAMBIOS →

Algoritmo	Tamaño	Aleatorio	Ascendente	Descendente
Inserción Binaria	1000	251,381	999	500,499
Inserción Binaria	10000	25,058,436	9,999	50,004,999
Radix Sort	1000	8,000	8,000	8,000
Radix Sort	10000	100,000	100,000	100,000



MUCHAS

GRACIAS

www.unsitiogenial.es

