Análisis de complejidad de Radix Sort

Radix Sort realiza d pasadas sobre el arreglo de n elementos, donde en cada pasada aplica Counting Sort, el cual tiene una complejidad de $\mathcal{O}(n+k)$, siendo k la base del sistema (por ejemplo, $k = 10$ en decimal).

$$T(n) = t_0 + \sum_{i=1}^{d} \left(t_1 + \sum_{j=0}^{k-1} t_2 + \sum_{l=0}^{n-1} t_3 \right)$$

$$T(n) = t_0 + d \cdot t_1 + d \cdot \sum_{j=0}^{k-1} t_2 + d \cdot \sum_{l=0}^{n-1} t_3$$

$$T(n) = t_0 + d \cdot t_1 + d \cdot (k \cdot t_2) + d \cdot (n \cdot t_3)$$

$$T(n) = t_0 + d(t_1 + k \cdot t_2 + n \cdot t_3)$$

Definimos constantes:

- $c_1 = t_3$
- $c_2 = t_2$
- $c_3 = t_1$
- $c_4 = t_0$

Reescribiendo la fórmula:

$$T(n) = d \cdot n \cdot c_1 + d \cdot k \cdot c_2 + d \cdot c_3 + c_4$$

Agrupando términos:

$$T(n) = c_1' \cdot n + c_2' \quad \text{donde} \quad c_1' = d \cdot c_1, \quad c_2' = d \cdot k \cdot c_2 + d \cdot c_3 + c_4$$

Conclusión: La complejidad total del algoritmo Radix Sort es:

$$T(n) = \mathcal{O}(d \cdot (n+k))$$

Esta complejidad se mantiene en el mejor, promedio y peor caso, ya que el número de pasadas (d) es fijo para cada elemento y el algoritmo no depende del orden inicial de los datos. Si d y k son constantes (por ejemplo, enteros pequeños de base 10), entonces la complejidad se vuelve lineal:

$$T(n) = \mathcal{O}(n)$$