**1. Complejidad del Algoritmo Radix Sort**

El algoritmo Radix Sort ordena los números dígito por dígito, desde el dígito menos significativo (unidades) hasta el más significativo (decenas, centenas, etc.). Para ordenar en cada fase (por dígito), Radix Sort suele usar un algoritmo eficiente como Counting Sort. Por lo tanto, la complejidad del algoritmo dependerá de:

* n: El número de elementos en la lista.
* d: El número de dígitos en el número más grande de la lista.

La complejidad total de Radix Sort será el número de dígitos que necesitamos procesar (dígitos d) multiplicado por el tiempo que nos lleva ordenar en cada fase usando Counting Sort, que tiene una complejidad de O(n + k), donde k es el rango de los dígitos (en este caso, 10, porque trabajamos con dígitos del 0 al 9).

Entonces, la complejidad general de Radix Sort es:

T(n) = O(d(n + k))

Dado que k (el rango de los dígitos) es una constante y es siempre 10, podemos simplificarlo a:

T(n) = O (d\*n)

**2. Caso de Complejidad**

**Mejores Casos y Peores Casos**

* **Mejor Caso**: Radix Sort no tiene un mejor caso especial, ya que en cada fase procesa todos los elementos y sigue ordenando los dígitos. En cualquier escenario, Radix Sort tendrá la misma complejidad asintótica.
* **Peor Caso**: De manera similar, el peor caso no cambia la complejidad de Radix Sort, ya que en cada fase se ordena por dígitos sin importar el orden inicial de los números.

Por lo tanto, en todos los casos (mejor y peor), la complejidad de Radix Sort es **O(d \cdot n)**.

**3. Ecuación de Recurrencia (Counting Sort)**

El Counting Sort utilizado en cada fase del Radix Sort tiene la siguiente ecuación de recurrencia:

T(n) = O(n + k)

Esto se debe a que Counting Sort realiza dos fases principales:

1. Contar las ocurrencias de cada dígito.
2. Colocar cada número en su posición correcta en función del conteo de los dígitos.

**Condiciones iniciales**

* **Cuando n = 1 la lista ya está ordenada, por lo que el tiempo es constante, es decir, T(1) = O(1).**

**4. Forma General de la Solución**

En Radix Sort, el Counting Sort se ejecuta **d** veces (una vez por cada dígito). Como ya hemos visto, la complejidad de Counting Sort es **O(n + k)** por cada fase. Dado que **k** es una constante pequeña (10), la complejidad del algoritmo es **O(n)** por cada fase de Counting Sort.

Así que, ejecutando Counting Sort **d** veces, la complejidad total es **O(d \cdot n)**.

**Conclusión Final**

La complejidad asintótica de Radix Sort, sin importar el caso, es:

T(n) = O (d \* n)

Donde:

* **n** es el número de elementos de la lista.
* **d** es el número de dígitos del número más grande.

Si **d** es constante (por ejemplo, si estamos trabajando con números que tienen un máximo de 10 dígitos), la complejidad de Radix Sort puede considerarse lineal, es decir, **O(n)**.