**Induccion Estructural RadixSort**

**Paso 1: Caso Base**

El caso más simple ocurre cuando la lista de enteros que queremos ordenar es vacía (Nil) o contiene un solo elemento.

* **Lista vacía**: Si la lista de entrada es vacía, no hay nada que ordenar, por lo que el resultado también será una lista vacía. Esto es consistente con la definición del algoritmo, ya que no se ejecuta ninguna operación sobre una lista vacía.
  + El resultado de radixSort(Nil) es simplemente Nil. Esto es correcto porque una lista vacía ya está ordenada.
* **Lista con un solo elemento**: Si la lista contiene un solo número, no es necesario realizar ningún proceso de ordenación, ya que una lista con un solo elemento ya está ordenada.
  + El resultado de radixSort(List(x)) es simplemente List(x). Esto también es correcto porque una lista con un solo número está ordenada por definición.

Por lo tanto, el algoritmo es **correcto en el caso base**.

**Paso 2: Hipótesis Inductiva**

* **Hipótesis inductiva**: Suponemos que la función radixSort es correcta para todas las listas de enteros de tamaño menor o igual a n, es decir, para listas con n elementos, el algoritmo devuelve una lista ordenada correctamente.

Esta hipótesis significa que, para listas de tamaño n, el algoritmo devuelve una lista ordenada.

**Paso 3: Paso Inductivo**

En este paso, demostramos que si el algoritmo funciona correctamente para listas de tamaño n, también funciona para listas de tamaño n + 1.

* Supongamos que tenemos una lista de n+1 elementos: xs=[x1,x2,...,xn+1]xs = [x\_1, x\_2, ..., x\_{n+1}]xs=[x1​,x2​,...,xn+1​].

El algoritmo **Radix Sort** procede de la siguiente manera:

1. **Proceso por dígitos**: La función radixSort ordena la lista por cada uno de los dígitos, comenzando por el dígito de las unidades, luego las decenas, centenas, etc. En cada fase, utiliza la función countingSortForRadix para ordenar los números según el dígito actual.
2. **Correctitud de cada fase**:
   * Por la hipótesis inductiva, asumimos que countingSortForRadix es correcta para ordenar listas de tamaño n. Dado que countingSortForRadix ordena en función de un solo dígito en cada fase, esto significa que después de cada fase, la lista se ordena correctamente según los dígitos que se han procesado.
   * Después de ordenar por el primer dígito (las unidades), el orden de los elementos por ese dígito es correcto. Luego, en las fases sucesivas, countingSortForRadix reordena los números por los dígitos más significativos (decenas, centenas, etc.), pero preserva el orden relativo de los números ya ordenados en las fases anteriores.
3. **Preservación del orden**: Dado que countingSortForRadix utiliza un algoritmo estable (Counting Sort), el orden de los números que ya estaban correctamente ordenados por los dígitos menos significativos no se deshace al ordenar por dígitos más significativos. Esto asegura que al final del proceso, la lista está completamente ordenada.
4. **Proceso recursivo**: La función sortByDigit es recursiva y va avanzando por cada posición de dígito (unidades, decenas, centenas, etc.) de manera iterativa. Como en cada paso el algoritmo ordena correctamente y preserva el orden de los dígitos previos, podemos concluir que si la lista de tamaño n está ordenada después de d pasos, entonces la lista de tamaño n + 1
5. **Conclusión del Paso Inductivo**

Dado que hemos demostrado que si el algoritmo es correcto para listas de tamaño nnn, también lo es para listas de tamaño n + 1 por el principio de inducción estructural, podemos concluir que **la función radixSort es correcta para cualquier lista de enteros de tamaño arbitrario**.