Radix Sort:

O **Radix Sort** é um algoritmo de ordenação não comparativo que ordena números inteiros processando seus dígitos de maneira sistemática, começando pelo dígito menos significativo (LSD - **Least Significant Digit**) até o mais significativo (MSD - **Most Significant Digit**), ou vice-versa. A ideia principal é ordenar os números com base em cada dígito individualmente, utilizando um algoritmo de ordenação estável, como o **Counting Sort**.

Como o Radix Sort funciona:

1. **Ordenação dos Dígitos**: O algoritmo começa pela posição menos significativa (unidade) e vai avançando para a posição mais significativa (milhares, milhões, etc.).
2. **Stable Sort**: A cada dígito, utiliza-se um algoritmo de ordenação estável para classificar os números com base nesse dígito. O algoritmo mais comum para isso é o **Counting Sort**.
3. **Repetição**: O processo é repetido para cada dígito, do menos significativo ao mais significativo.
4. **Uso de Bases Diferentes**: O algoritmo pode ser adaptado para diferentes bases (como base 10 ou base 2). No caso da base 10, ele usa dígitos de 0 a 9. Se fosse utilizado para binários, usaria apenas os dígitos 0 e 1.

Como o Radix Sort lida com bases diferentes:

O **Radix Sort** pode ser adaptado para bases diferentes, como base 2 (binário) ou base 10 (decimal). Para a base 10, cada dígito do número é considerado de 0 a 9, e para a base 2, cada dígito seria considerado de 0 ou 1. O processo seria similar:

- **Base 10**: A ordenação é feita com base nos dígitos de 0 a 9.
- **Base 2**: A ordenação seria feita com base nos dígitos 0 ou 1.

Se quisermos usar uma base diferente, como **base 2**, o código precisaria ser ajustado para lidar com os bits, e em vez de contar os dígitos de 0 a 9, contaríamos os bits de 0 ou 1.

Exemplos de Entrada e Saída:

1. **Lista com 2 dígitos**:
    - Entrada: `[34, 23, 12, 45, 98]`
    - Saída: `[12, 23, 34, 45, 98]`
2. **Lista com 5 dígitos**:
    - Entrada: `[52347, 12345, 98765, 23456, 54321]`
    - Saída: `[12345, 23456, 52347, 54321, 98765]`
3. **Lista com 10 dígitos**:

    - Entrada: `[9876543210, 1234567890, 2345678901, 3456789012, 4567890123]`
    - Saída: `[1234567890, 2345678901, 3456789012, 4567890123, 9876543210]`

Complexidade e Considerações:

- **Tempo de Execução**: O tempo de execução do **Radix Sort** é O(d · (n+b))O(d \cdot (n + b))O(d · (n+b)), onde:
    - ddd é o número de dígitos do maior número.
    - nnn é o número de elementos na lista.
    - bbb é a base de ordenação (no caso da base 10, b=10b = 10b=10).
- Para uma lista com números de kkk dígitos, o algoritmo é muito eficiente em comparação com algoritmos de comparação como o **Quick Sort** ou **Merge Sort**.
- **Espaço**: O **Radix Sort** requer espaço adicional para armazenar a contagem e o resultado de cada iteração, o que pode ser um fator importante para listas muito grandes.

Conclusão:

O **Radix Sort** é um algoritmo de ordenação eficiente para listas com números inteiros, especialmente quando a quantidade de dígitos não é muito grande. Ele pode ser adaptado para diferentes bases, dependendo do contexto de aplicação.