Problem: Implement radix sort.

Constraints

- **Is the input a list?**
  - **Yes**
- **Can we assume the inputs are valid?**
  - **Check for None in place of an array**
  - **Assume array elements are ints**
- **Do we know the max digits to handle?**
  - **No**
- **Are the digits base 10?**
  - **Yes**
- **Can we assume this fits memory?**
  - **Yes**

Test Cases

- **None -> Exception**
- **[] -> []**
- **[128, 256, 164, 8, 2, 148, 212, 242, 244] -> [2, 8, 128, 148, 164, 212, 242, 244, 256]**

Algorithm

Sample input: [1, 220, 122, 112]

- **We'll evaluate each digit starting with the ones position**
  - **[1, 220, 122, 112]**
    - **Bucket 0: 220**
    - **Bucket 1: 1**
    - **Bucket 2: 122, 112**
    - **Result: [220, 1, 122, 112]**
  - **[220, 1, 122, 112]**
    - **Bucket 0: 1**
    - **Bucket 1: 112**
    - **Bucket 2: 220, 122**
    - **Result: [1, 112, 220, 122]**
  - **[1, 112, 220, 122]**
    - **Bucket 0: 1**
    - **Bucket 1: 112, 122**
    - **Bucket 2: 220**
    - **Result: [1, 112, 122, 220]**

Bucketing example: 123

- **Ones**

    - **123 // 10^0 = 123**
    - **123 % 10 = 3**
- **Tens**
    - **123 // 10^1 = 12**
    - **12 % 10 = 2**
- **Hundreds**
    - **123 // 10^2 = 1**
    - **1 % 10 = 1**

Complexity:

- **Time: O(k*n), where n is the number of items and k is the number of digits in the largest item**
- **Space: O(k+n)**

Misc:

- **Not in-place**
- **Most implementations are stable**

If k (the number of digits) is less than log(n), radix sort can be faster than algorithms such as quicksort.

-