Radix Sort

- Radix sort is one of the sorting algorithms used to sort a list of integer numbers in order.
- In radix sort algorithm, a list of integer numbers will be sorted based on the digits of individual numbers. Sorting is performed from **<u>least significant digit to the most significant digit</u>**. Radix sort algorithm requires the number of passes which are equal to the number of digits present in the largest number among the list of numbers.
- For example, if the largest number is a 3 digit number then that list is sorted with 3 passes.
- Radix sort is a sorting technique that sorts the elements by first grouping the individual digits of the same place value.
- Then, sort the elements according to their increasing/decreasing order.
- Suppose, we have an array of 8 elements. First, we will sort elements based on the value of the unit place. Then, we will sort elements based on the value of the tenth place.
- This process goes on until the last significant place.

Step by Step Process

The Radix sort algorithm is performed using the following steps...

- **Step 1 -** Define 10 queues each representing a bucket for each digit from 0 to 9.
- **Step 2 -** Consider the least significant digit of each number in the list which is to be sorted.
- **Step 3 -** Insert each number into their respective queue based on the least significant digit.
- **Step 4 -** Group all the numbers from queue 0 to queue 9 in the order they have inserted into their respective queues.
- **Step 5 -** Repeat from step 3 based on the next least significant digit.
- **Step 6 -** Repeat from step 2 until all the numbers are grouped based on the most significant digit.

Consider the following list of unsorted integer numbers

82, 901, 100, 12, 150, 77, 55 & 23

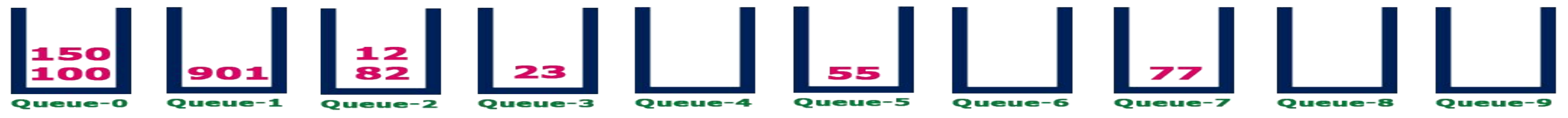
Step 1 - Define 10 queues each represents a bucket for digits from 0 to 9.

Queue-0
Queue-1
Queue-2
Queue-3
Queue-4
Queue-5
Queue-6
Queue-7
Queue-8
Queue-9

Pass - 1

Step 2 - Insert all the numbers of the list into respective queue based on the Least significant digit (once placed digit) of every number.

82, 901, 100, 12, 150, 77, 55 & 23



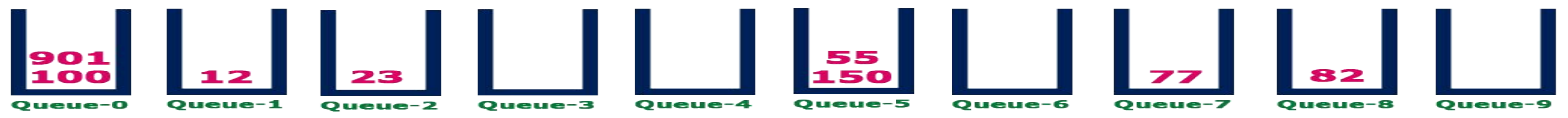
Group all the numbers from queue-0 to queue-9 inthe order they have inserted & consider the list for next step as input list.

100, 150, 901, 82, 12, 23, 55 & 77

Pass - 2

Step 3 - Insert all the numbers of the list into respective queue based on the next Least significant digit (Tens placed digit) of every number.

100, 150, 901, 82, 12, 23, 55 & 77



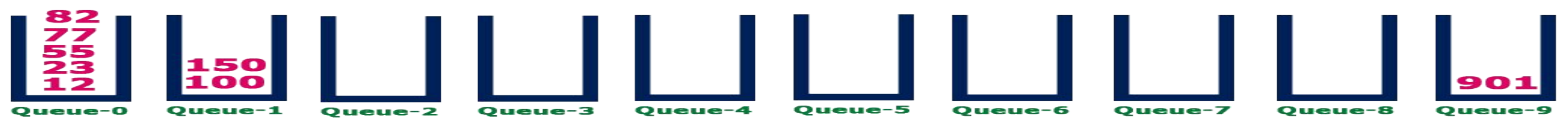
Group all the numbers from queue-0 to queue-9 inthe order they have inserted & consider the list for next step as input list.

100, 901, 12, 23, 150, 55, 77 & 82

Pass - 3

Step 4 - Insert all the numbers of the list into respective queue based on the next Least significant digit (Hundres placed digit) of every number.

100, 901, 12, 23, 150, 55, 77 & 82



Group all the numbers from queue-0 to queue-9 inthe order they have inserted & consider the list for next step as input list.

12, 23, 55, 77, 82, 100, 150, 901

List got sorted in the incresing order.

Following example shows how Radix sort operates on seven 3-digits number

Example

Input	1st Pass	2nd Pass	3rd Pass
329	720	720	329
457	355	329	355
657	436	436	436
839	457	839	457
436	657	355	657
720	329	457	720
355	839	657	839

Complexity of the Radix Sort Algorithm

To sort an unsorted list with **'n'** number of elements, Radix sort algorithm needs the following complexities...

WorstCase:O(n)

BestCase:O(n)

Average Case : O(n)

Advantages :

1. Fast when the keys are short i.e. when the range of the array elements is less.
2. Used in suffix array constuction algorithms like Manber's algorithm and DC3 algorithm.

Disadvantages:

1. Since Radix Sort depends on digits or letters, Radix Sort is much less flexible than other sorts. Hence , for every different type of data it needs to be rewritten.
2. The constant for Radix sort is greater compared to other sorting algorithms.
3. It takes more space compared to Quicksort which is inplace sorting.

Radix sort in Python

```
# Using counting sort to sort the elements in the basis of
significant places
def countingSort(array, place):
   size = len(array)
   output = [0] * size
   count = [0] * 10

 # Calculate count of elements
   for i in range(0, size):
      index = array[i] // place
      count[index % 10] += 1

# Calculate cummulative count
   for i in range(1, 10):
      count[i] += count[i - 1]

 # Place the elements in sorted order
   i = size - 1
   while i >= 0:
      index = array[i] // place
      output[count[index % 10] - 1] = array[i]
      count[index % 10] -= 1
      i -= 1

for i in range(0, size):
      array[i] = output[i]
# Main function to implement radix sort
def radixSort(array):
   # Get maximum element
   max_element = max(array)

   # Apply counting sort to sort elements based on place value.
   place = 1
   while max_element // place > 0:
      countingSort(array, place)
      place *= 10


data = [121, 432, 564, 23, 1, 45, 788]
radixSort(data)
print(data)
```