Radix Sort

Radix Sort is similar to a register algorithm which we use in day to day work while arranging a list of names, in alphabetical order, in that similar way radix sort arranges the given numbers by arranging them in the order of each digit sequentially starting from one’s place and moving to ten’s or hundred’s place depending upon the given data.

Radix sort is based on the idea that the sorting of the input data is done digit by digit from least significant digit to most significant digit and it uses counting sort as a subroutine to perform sorting. Counting sort is a linear sorting algorithm with overall time complexity $\Theta(N+K)$ in all cases, where N is the number of elements in the unsorted array and K is the range of input data. The idea of radix sort is to extend the counting sort algorithm to get a better time complexity when K goes up.

Consider the array of length 6 given below. Sort the array by using Radix sort.
A = {10, 2, 901, 803, 1024}
Pass 1: (Sort the list according to the digits at 0's place)
10, 901, 2, 803, 1024.
Pass 2: (Sort the list according to the digits at 10's place)
02, 10, 901, 803, 1024
Pass 3: (Sort the list according to the digits at 100's place)
02, 10, 1024, 803, 901.
Pass 4: (Sort the list according to the digits at 1000's place)
02, 10, 803, 901, 1024
Therefore, the list generated in the step 4 is the sorted list, arranged from radix sort.

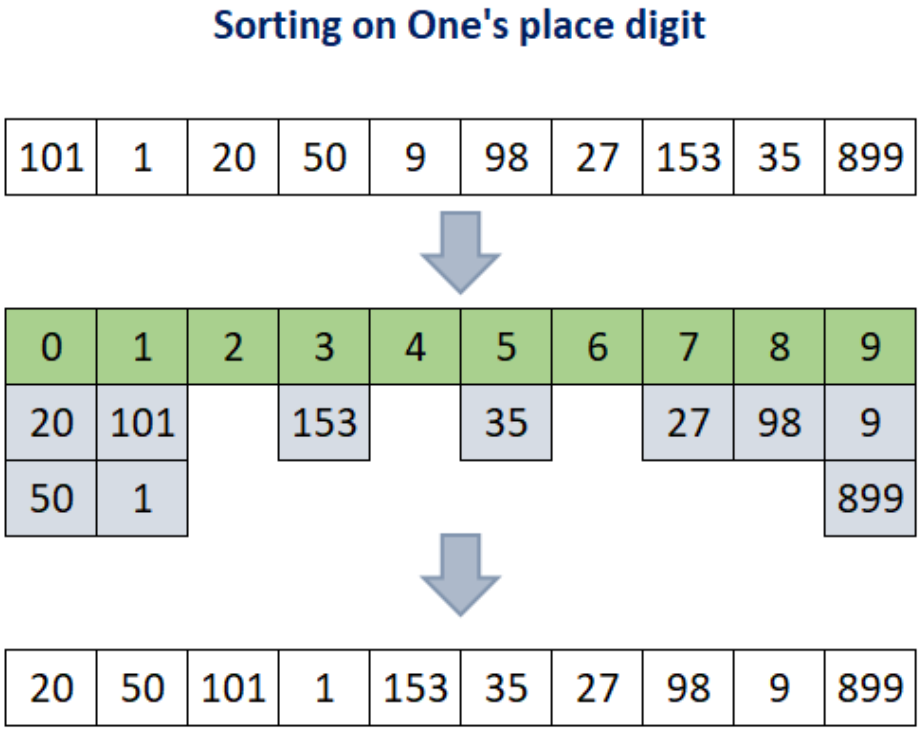
Algorithm
Step 1:Find the largest number in ARR as LARGE

Step 2: [INITIALIZE] SET NOP = Number of digits
in LARGE
Step 3: SET PASS =0
Step 4: Repeat Step 5 while PASS <= NOP-1
Step 5: SET I = 0 and INITIALIZE buckets
Step 6:Repeat Steps 7 to 9 while I<n-1< li=""> </n-1<>
Step 7: SET DIGIT = digit at PASSth place in A[I]

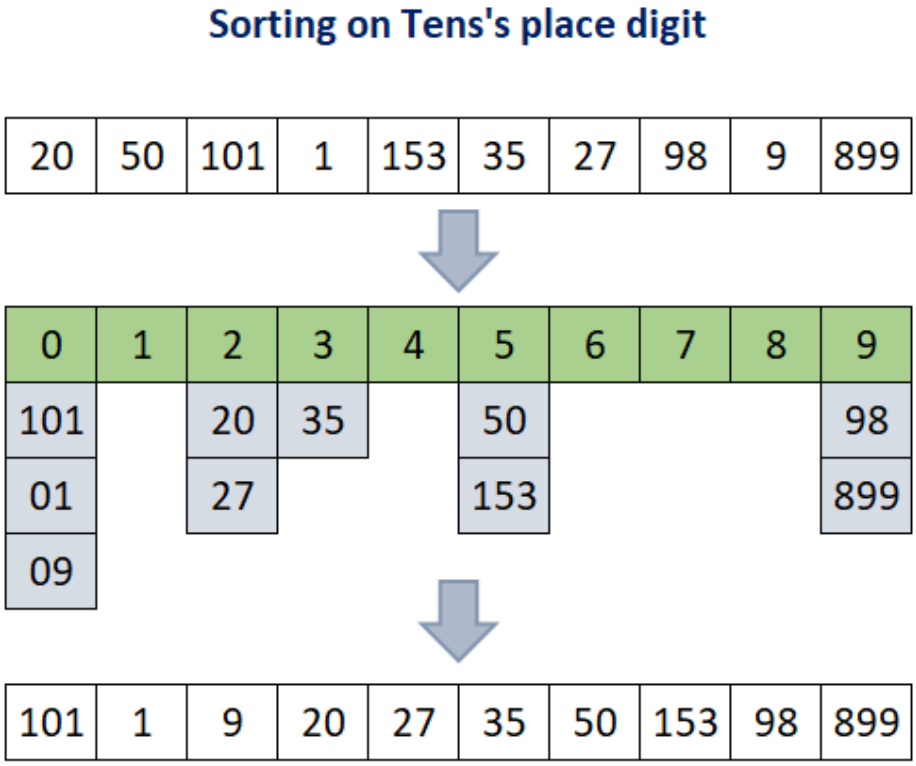
Step 8: Add A[I] to the bucket numbered DIGIT
Step 9: INCREMENT bucket count for bucket numbered DIGIT
[END OF LOOP]
Step 10: Collect the numbers in the bucket
[END OF LOOP]
Step 11: END

A = [101, 1, 20, 50, 9, 98, 27, 153, 35, 899]

Step 1: According to the algorithms, the input data is first sorted based on least significant digit. Therefore, array *A[ ]* is sorted based on one's digit. After sorting it on one's digit it will become *[20, 50, 101, 1, 153, 35, 27, 98, 9, 899]*.

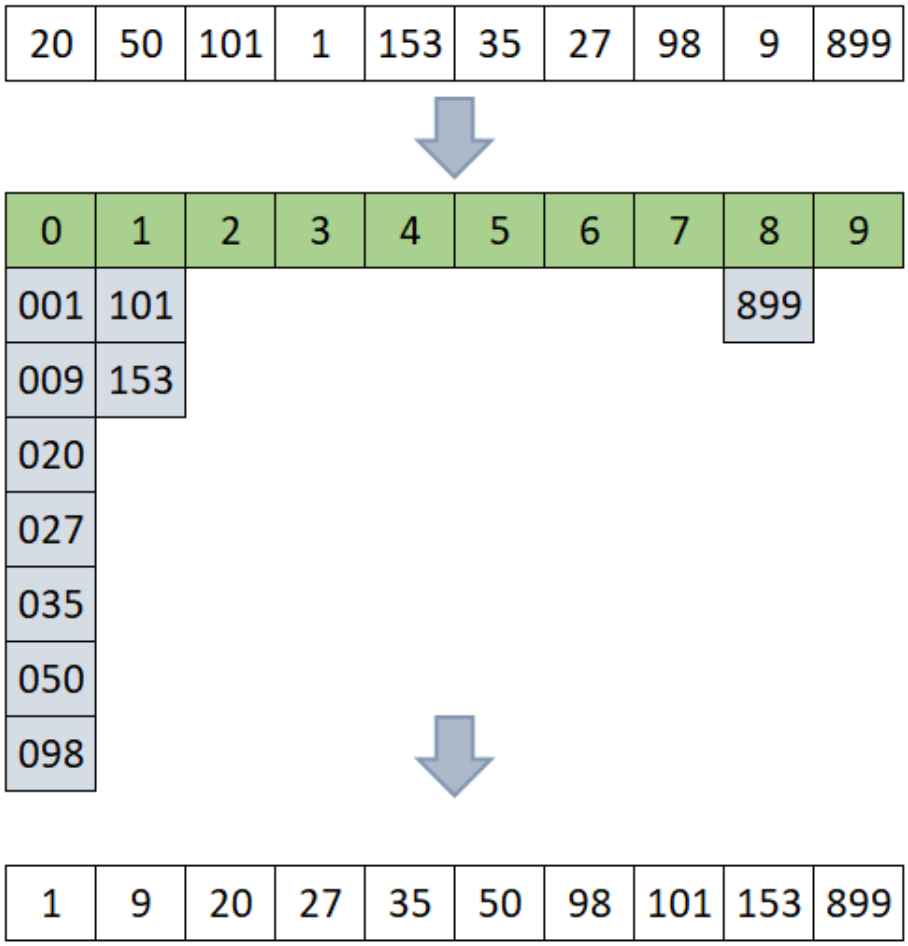


Step 2: In this step, the array *A[ ]* array is sorted based on ten's digit and after this step it will become *[101, 1, 9, 20, 27, 35, 50, 153, 98, 899]*.



Step 3: Finally, the array *A[ ]* is sorted based on hundred's digit (most significant digit) and the array will be sorted after this step and the final array will be *[1, 9, 20, 27, 35, 50, 98, 101, 153, 899]*.

Sorting on Hundred's place digit




```
package Sort;

//Radix sort Java implementation
import java.io.*;
import java.util.*;

public class Radix {

  /* Main Method to check for above function */
  public static void main(String[] args) {
    int a[] = { 17, 45, 75, 90, 82, 24, 12, 60 };
    int len = a.length;
    radixsort(a, len);
    print(a, len);
  }

// A utility function to get maximum value in a[]
  static int getMax(int a[], int len) {
```

```
    int mx = a[0];
    for (int i = 1; i < len; i++)
      if (a[i] > mx)
        mx = a[i];
    return mx;
  }

// A function to do counting sort of arr[] according to
// the digit represented by exp.
  static void countSort(int a[], int len, int exp) {
    int output[] = new int[len]; // output array
    int i;
    int count[] = new int[10];
    Arrays.fill(count, 0);

// Store count of occurrences in count[]
    for (i = 0; i < len; i++)
      count[(a[i] / exp) % 10]++;

// Change count[i] so that count[i] now contains
// actual position of this digit in output[]
    for (i = 1; i < 10; i++)
      count[i] += count[i - 1];

// Build the output array
    for (i = len - 1; i >= 0; i--) {
      output[count[(a[i] / exp) % 10] - 1] = a[i];
      count[(a[i] / exp) % 10]--;
    }

// Copy the output array to arr[], so that arr[] now
// contains sorted numbers according to curent digit
    for (i = 0; i < len; i++)
      a[i] = output[i];
  }

// The main function to that sorts arr[] of size n using
// Radix Sort
  static void radixsort(int a[], int len) {
// Find the maximum number to know number of digits
    int m = getMax(a, len);

// Do counting sort for every digit. Note that instead
// of passing digit number, exp is passed. exp is 10^i
// where i is current digit number
    for (int exp = 1; m / exp > 0; exp *= 10)
      countSort(a, len, exp);
  }

// A utility function to print an array
  static void print(int a[], int len) {
    for (int i = 0; i < len; i++)
      System.out.print(a[i] + " ");
  }
}
```

The time complexity of radix sort is ***Θ((N+b)*logb(max))*** in all cases, where:

- *N* is the number of elements in unsorted array.
- *b* is the base of input array, for example, for decimal system, *b* is 10.
- *max* is the maximum element of the input array.