# 3 - Radix Sort

```
1  Radix-Sort(A, d)
2  //It works same as counting sort for d number of passes.
3  //Each key in A[1..n] is a d-digit integer.
4  //(Digits are numbered 1 to d from right to left.)
5      for j = 1 to d do
6          //A[]-- Initial Array to Sort
7          int count[10] = {0};
8          //Store the count of "keys" in count[]
9          //key- it is number at digit place j
10         for i = 0 to n do
11          count[key of(A[i]) in pass j]++
12
13         for k = 1 to 10 do
14          count[k] = count[k] + count[k-1]
15
16         //Build the resulting array by checking
17         //new position of A[i] from count[k]
18         for i = n-1 downto 0 do
19          result[ count[key of(A[i])] ] = A[j]
20          count[key of(A[i])]--
21
22         //Now main array A[] contains sorted numbers
23         //according to current digit place
24         for i=0 to n do
25           A[i] = result[i]
26
27     end for(j)
28 end func
```

O(d)
number of digits

O(n)
O(k)
O(n)
O(n)

Counting sort - O(n+k)
length of array
length of count array
k=9, because for each place, range 0-9

O(n)+O(n)+O(n)+O(k)

Time complexity : O(d) * O(n+k)
: O(d(n+k))

## Demonstration

A = <154, 56, 77, 134, 186, 56, 94, 24, 13, 83, 95, 143>

sort with ones digit

0:
1:
2:
3: 13, 83, 143
4: 154, 134, 94, 24
5: 95
6: 56, 186, 56
7: 77
8:
9:

Counting sort

new A = <13, 83, 143, 154, 134, 94, 24, 95, 56, 186, 56, 77>

sort with tens digit

0:
1: 13
2: 24
3: 134
4: 143
5: 154, 56, 56
6:
7: 77
8: 83, 186
9: 94, 95

Counting sort

new A = <13, 24, 134, 143, 154, 56, 56, 77, 83, 186, 94, 95>

sort with hundreds digit

0: 13, 24, 56, 56, 77, 83, 94, 95
1: 134, 143, 154, 186
2:
3:
4:
5:
6:
7:
8:
9:

Counting sort

new A = <13, 24, 56, 56, 77, 83, 94, 95, 134, 143, 154, 186>