**Sorting algorithm In Linear Time:**

**1) Counting Sort**

Counting sort is a stable sorting technique, which is used to sort objects according to the keys that are small numbers. It counts the number of keys whose key values are same. This sorting technique is effective when the difference between different keys are not so big, otherwise, it can increase the space complexity.

**The complexity of counting Sort Technique**

1. Time Complexity: O(n+r)
2. Space Complexity: O(n+r)

**Input and Output**

**Input:**

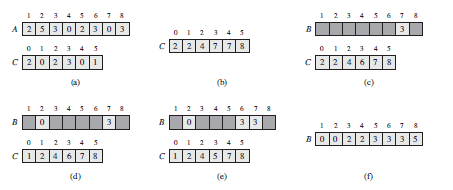
A list of unsorted data: 2 5 6 2 3 10 3 6 7 8

**Output:**

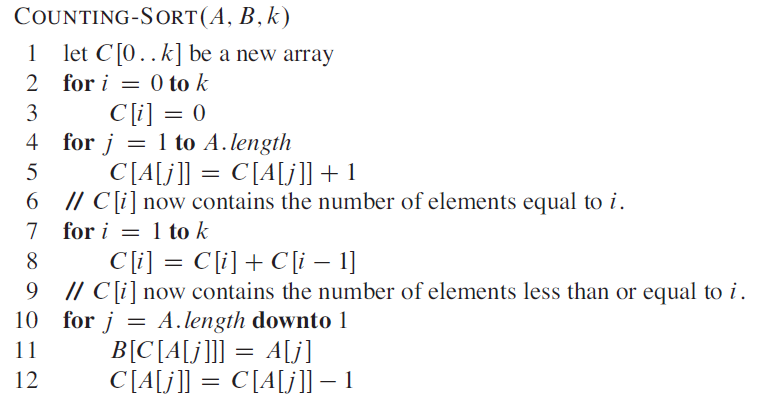
Array before Sorting: 2 5 6 2 3 10 3 6 7 8

Array after Sorting: 2 2 3 3 5 6 6 7 8 10

Example:



**Algorithm**



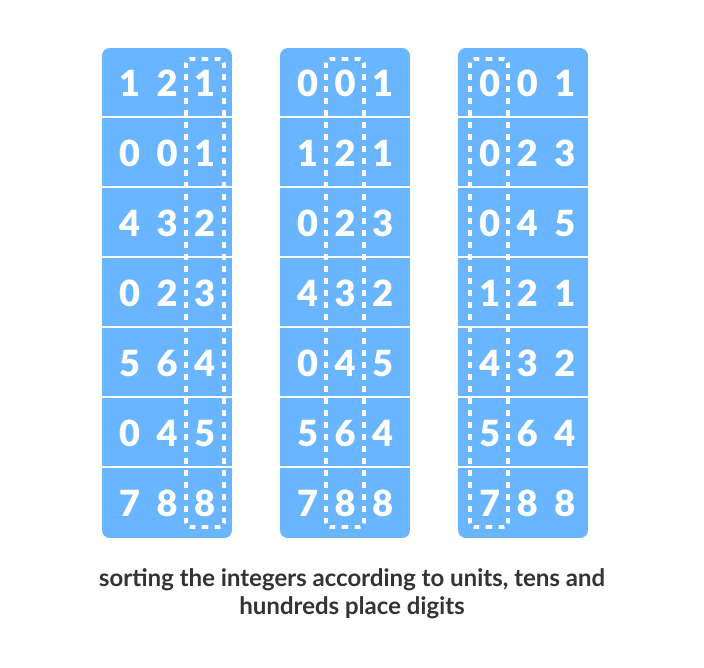
Top of Form

**2) Radix Sort**

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.

Let the initial array be [121, 432, 564, 23, 1, 45, 788]. It is sorted according to radix sort as shown in the figure below.



**How Radix Sort Works?**

1) Find the largest element in the array, i.e. max. Let X be the number of digits in max. X is calculated becausewe have to go through all the significant places of all elements.

In this array [121, 432, 564, 23, 1, 45, 788], we have the largest number 788. It has 3 digits. Therefore, theloop should go up to hundreds place (3 times)

2) Now, go through each significant place one by one.

Use any stable sorting technique to sort the digits at each significant place. We have used counting sort forthis.

Sort the elements based on the unit place digits (X=0).

Counting Sort is a linear time sorting algorithm that sort in O(n+k) time when elements are in range from 1 to k.

***What if the elements are in range from 1 to n2?***

We can’t use counting sort because counting sort will take O(n2) which is worse thancomparison-based sorting algorithms. Can we sort such an array in linear time?

Radix sort is the answer. The idea of Radix Sort is to do digit by digit sort starting from least significant digit to most significant digit. Radix sort uses counting sort as a subroutine to sort.

***The Radix Sort Algorithm***

**1)** Do following for each digit i where i varies from least significant digit to the most significant digit.

**a)** Sort input array using counting sort (or any stable sort) according to the i’th digit.

**RELEVANT READING MATERIAL AND REFERENCES:**

**Source Notes:**

1. <https://www.geeksforgeeks.org › counting-sort>
2. <https://www.programiz.com/dsa/radix-sort>
3. <https://www.geeksforgeeks.org/radix-sort/>

**Lecture Video:**

1. <https://youtu.be/7zuGmKfUt7s>
2. <https://youtu.be/nu4gDuFabIM>

**Online Notes:**

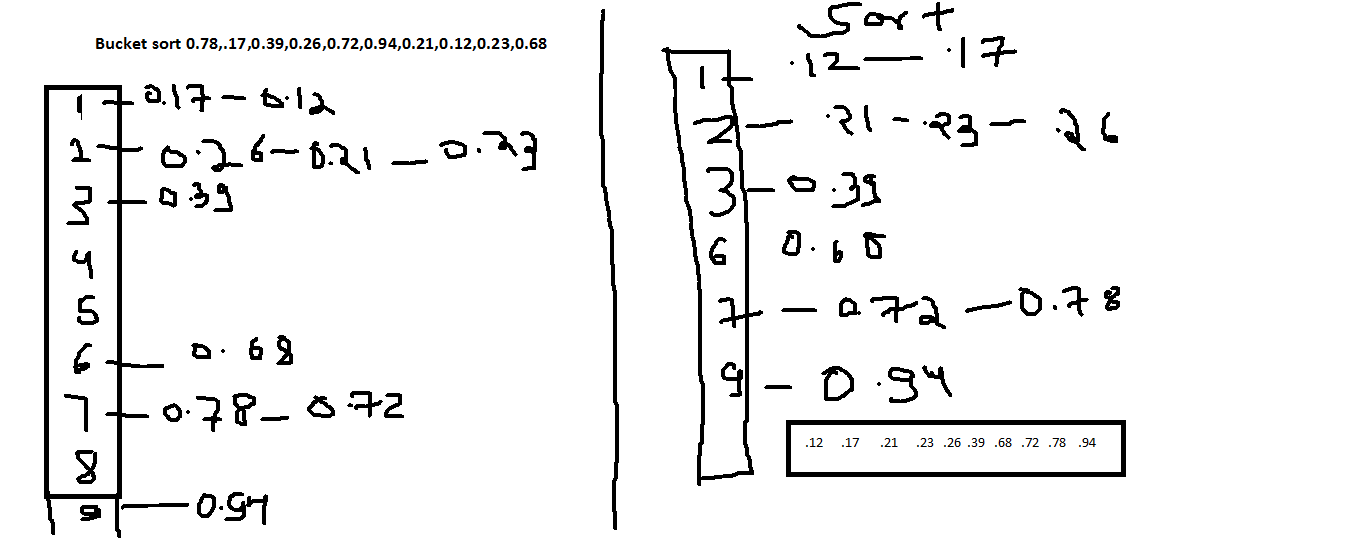
1. <https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-006-introduction-to-algorithms-fall-2011/lecture-videos/MIT6_006F11_lec07.pdf>

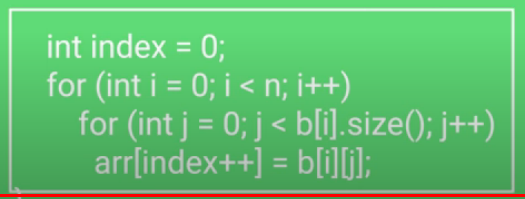
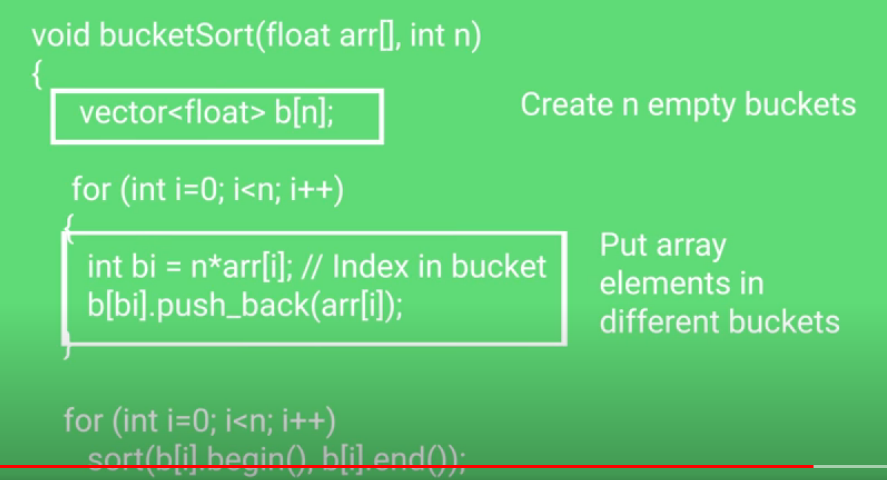
**Text Book Reading:**

1. Cormen, Leiserson, Rivest, Stein, “*Introduction to Algorithms*”, Prentice Hall of India, 3rd edition 2012. problem, Graph coloring.

**In addition: PPT can be also be given.**

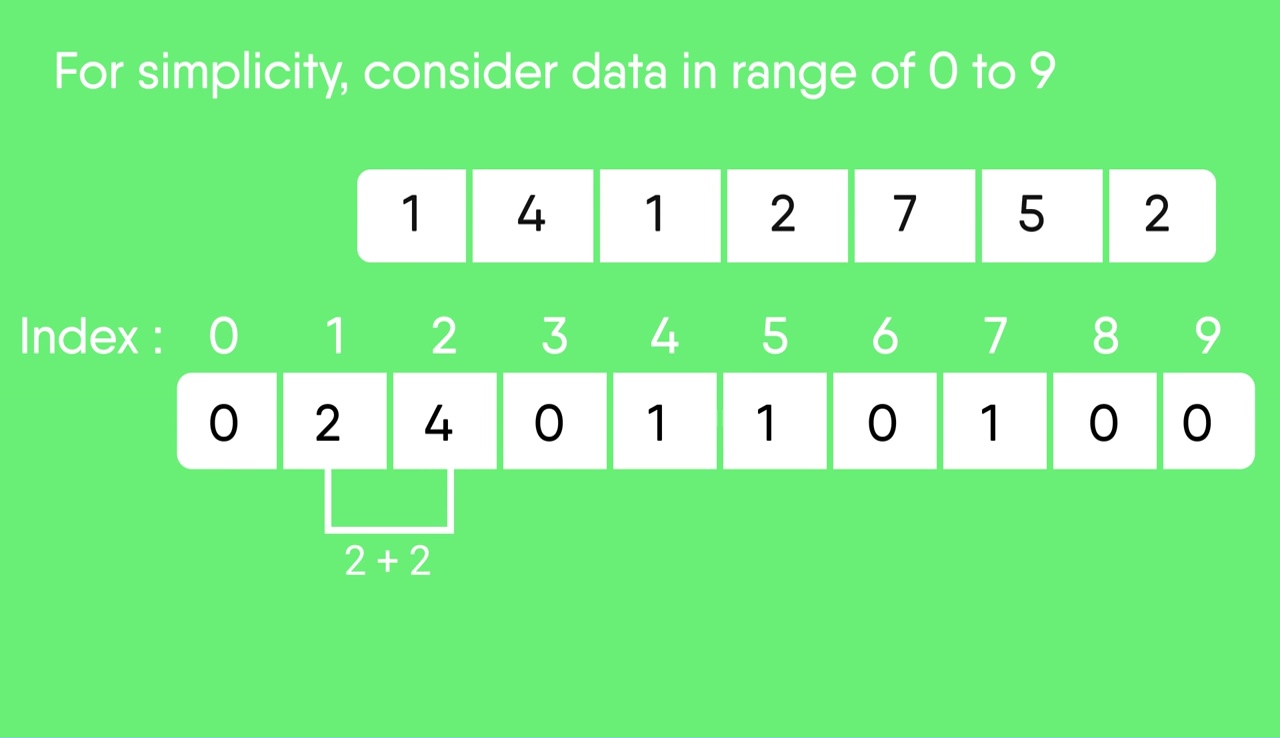
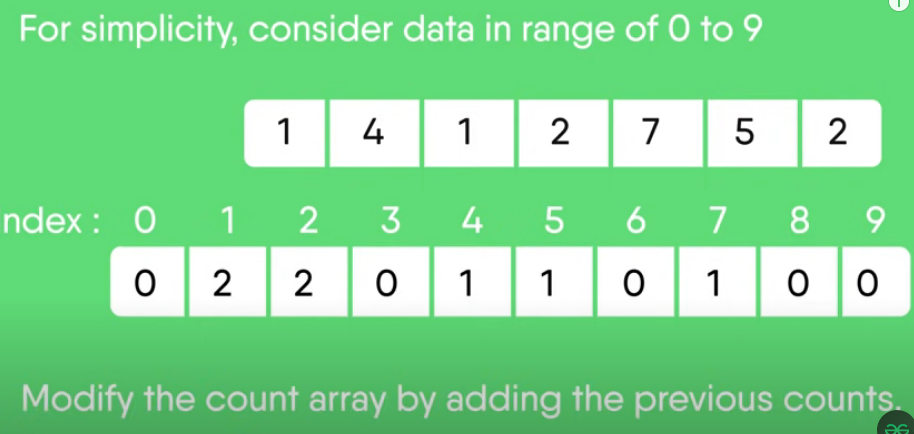
**Bucket sort 0.78,.17,0.39,0.26,0.72,0.94,0.21,0.12,0.23,0.68**



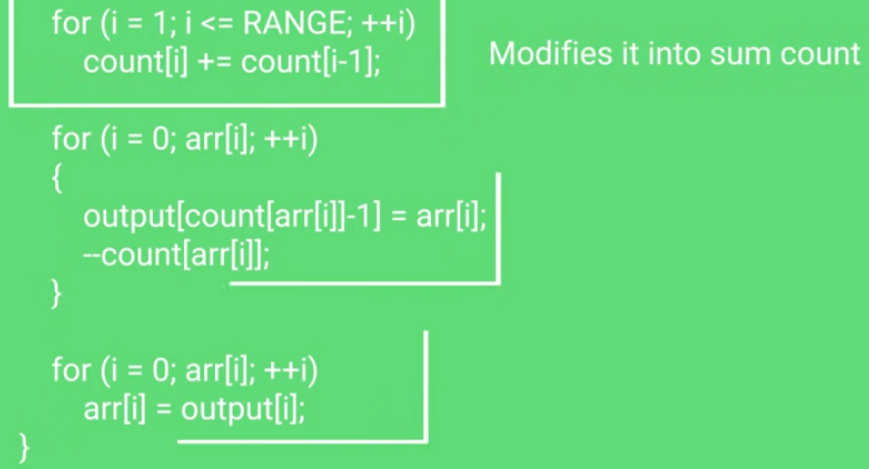
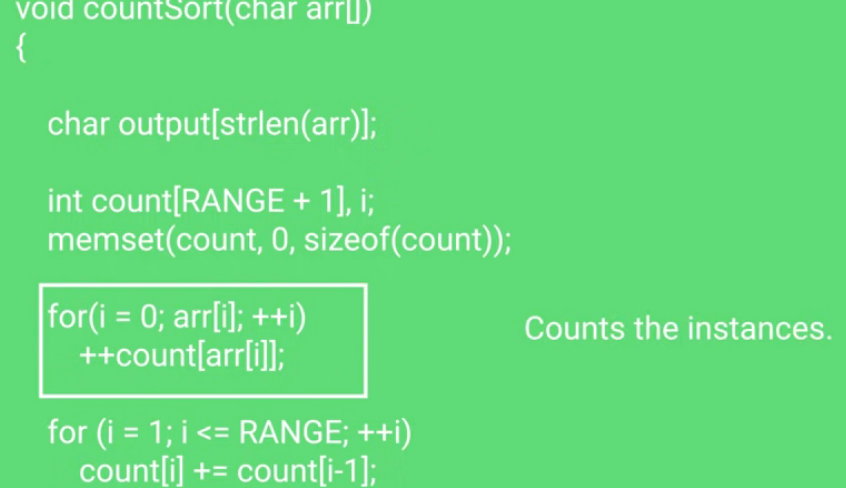


........................................................................................

Count sort 1,4,1,2,7,5,2



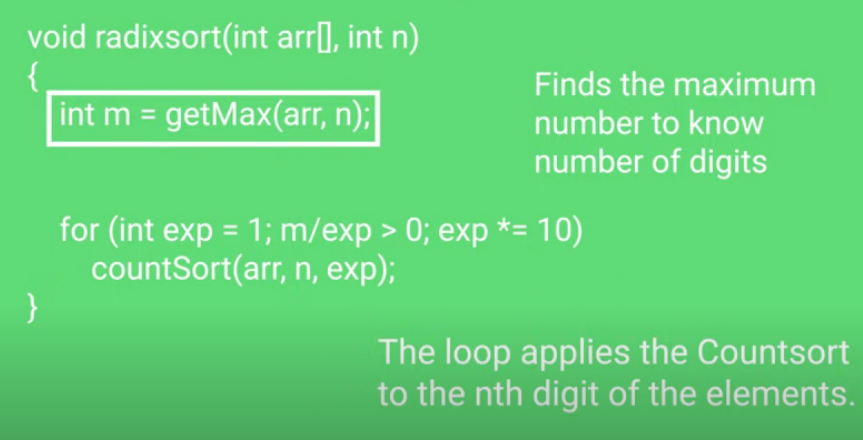
Rest solution will be done as per class lecture .....

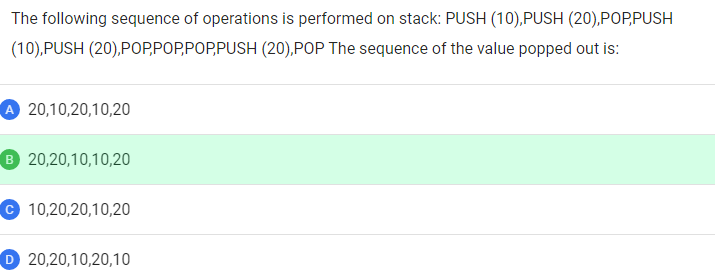
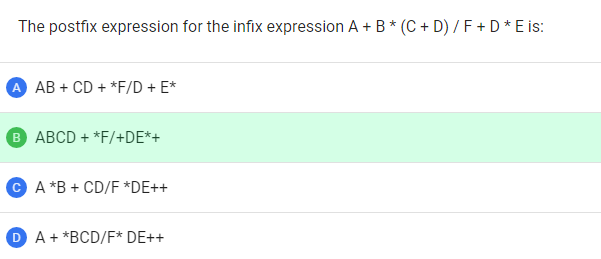


..................................................................................................................

Radix sort 170,45,75,90,802,24,2,66

Solve it as per lecture ...





Top of Form

**Bottom of Form**

**Top of Form**

**Convert following infix to prefix expression  
  
e^d-a\*b^f/g+h\*c/i+j-k**

**Top of Form**

**Bottom of Form**

**Top of Form**

**Precedence rule : ^ > ( / = \* ) > (+ = -)  
  
Step 1 : ^ed - a\* ^bf /g + h\*c / i + j - k  
  
Step 2 : ^ed -/ \*a^bfg + h\*c /i +j -k    (Since \* , / has same precedence we go by the associativity that is left to right .)  
  
Step 3 : ^ed - / \*a ^ bfg + /\* hci + j-k  
  
Step 4 : -^ed  / \*a ^bfg + / \*hci + j-k  
  
Step 5:  + - ^ed /\*a^bfg/\*hci + j-k  
  
Step 6 : ++-^ed/\*a^bfg/\*hcij-k  
  
Step 7 : -++-^ed/\*a^bfg/\*hcijk**

Bottom of Form

Bottom of Form

Bottom of Form