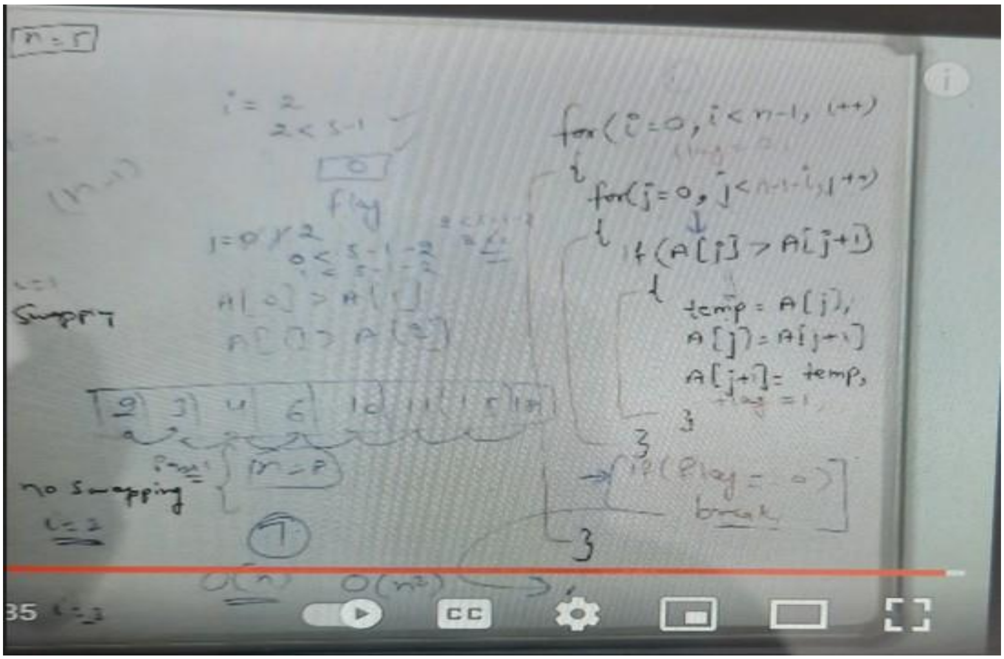
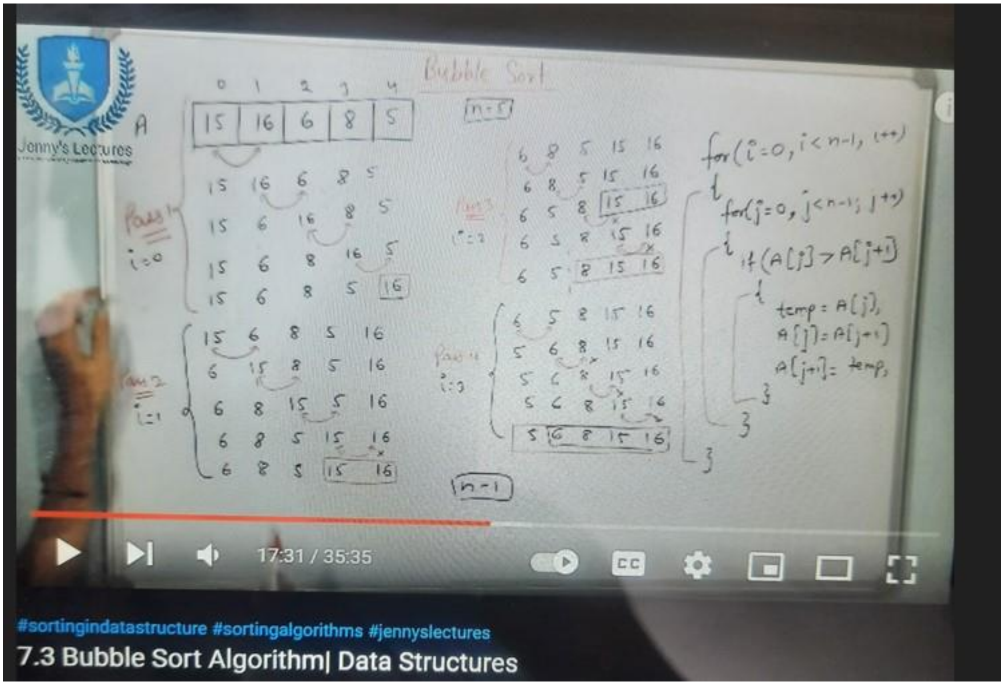
Algorithm Overview:

Bubble sort:

https://www.youtube.com/watch?v=o4bAoo_gFBU&list=PLdo5W4Nhv31bbKJzrsKfMpo_grxuLl8LU&index=97



```
for(i=0, i<n-1, i++)
{
  for(j=0, j<n-1-i, j++)
  {
    if(A[j]>A[j+1])
    {
      temp = A[j],
      A[j]=A[j+1]
      A[j+1]= temp,
      flag =1
    }
  }
  if(flag==0)
    break
}
```



```
for(i=0, i<n-1, i++)
{
  for(j=0, j<n-1, j++)
  {
    if(A[j]>A[j+1])
    {
      temp = A[j],
      A[j]=A[j+1]
      A[j+1]= temp,
    }
  }
}
```

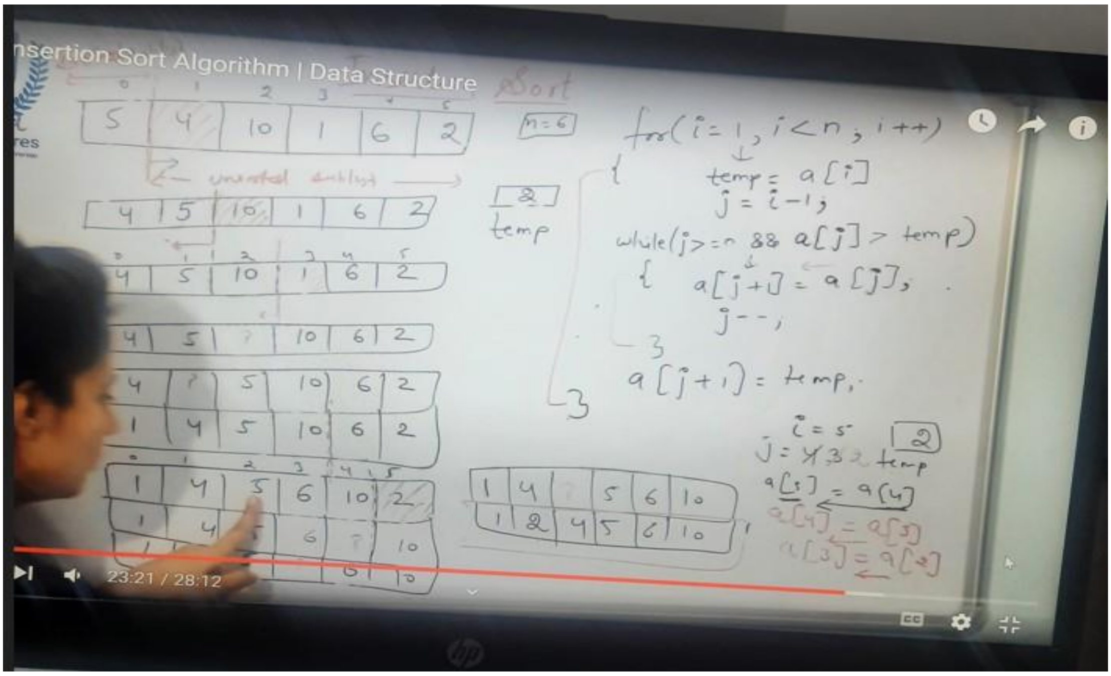
#sortingindatastructure #sortingalgorithms #jennyslectures

7.3 Bubble Sort Algorithm| Data Structures

Insertion sort:

https://www.youtube.com/watch?v=yCxV0kBpA6M&list=PLdo5W4Nhv31bbKJzrsKfMpo_grxuLl8LU&index=99

Given array is divided into two parts. Sorted array and Unsorted array.

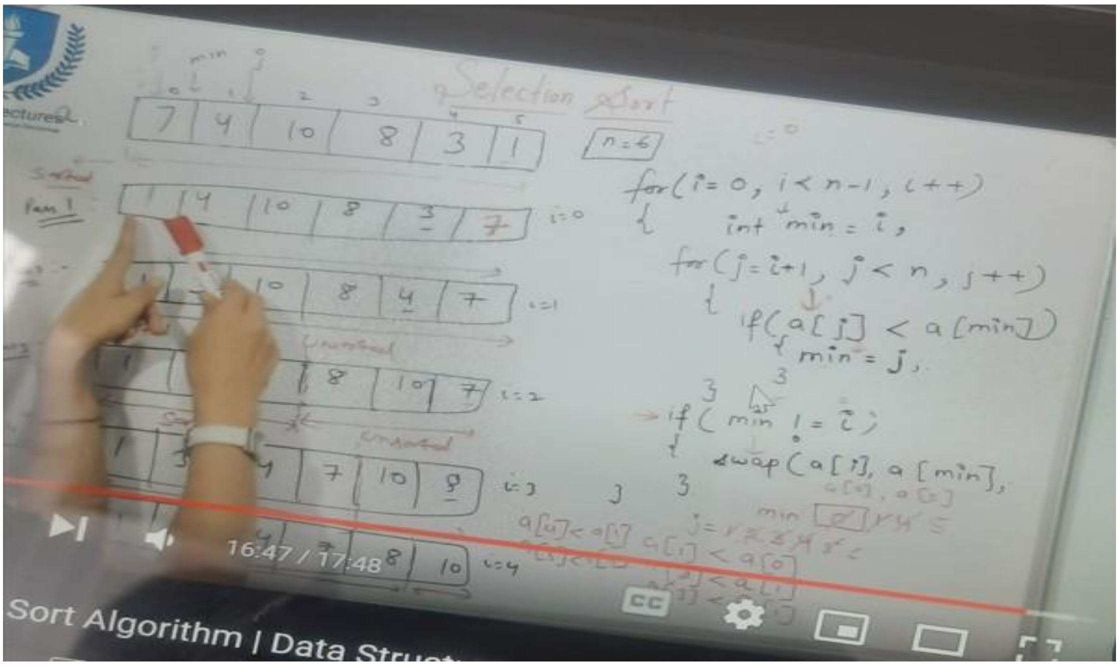


```
for(i=1, i<n ; i++)
{
    temp = a[i]
    j = i-1;
    while(j>=0 && a[j] > temp)
    {   a[j+1] = a[j];
        j--;
    }
    a[j+1] = temp;
}
```

```
i = 5
j = 4 3 2
a[5] = a[4]
a[4] = a[3]
a[3] = a[2]
```

Selection sort:

https://www.youtube.com/watch?v=9oWd4VJOwr0&list=PLdo5W4Nhv31bbKJzrsKfMpo_grxuLl8LU&index=100



```
for(i=0, i<n-1, i++)
{   int min = i,
    for(j=i+1, j<n, j++)
    {
        if(a[j] < a[min])
        {  min = j,
        }
    }
    if( min != i)
    {
        swap(a[i], a[min]),
    }
}
```

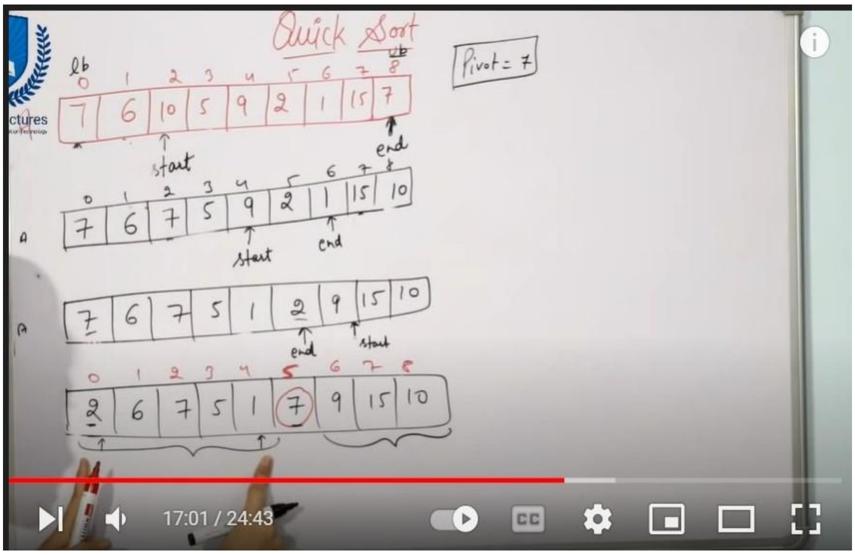
Quicksort:

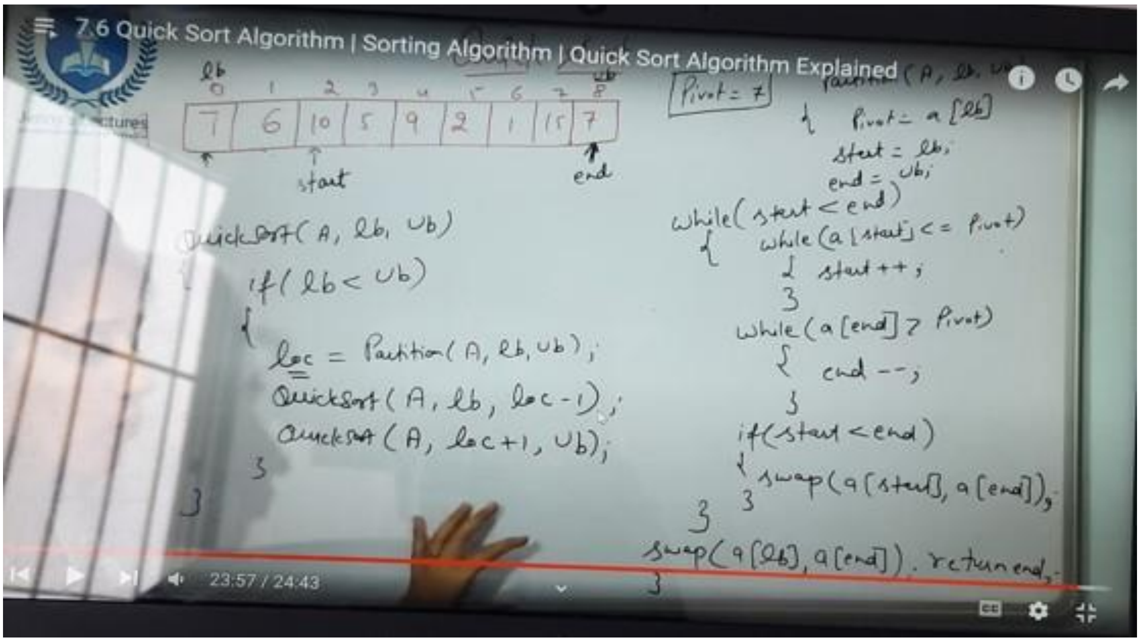
https://www.youtube.com/watch?v=QN9hnmAgmOc&list=PLdo5W4Nhv31bbKJzrsKfMpo_grxuLl8LU&index=100

Quicksort algorithm is based on the divide and conquer approach where an array is divided into subarrays by selecting a pivot element.

While dividing the array, the pivot element should be positioned in such a way that elements less than pivot are kept on the left side and elements greater than pivot are on the right side.

The same process is continued for both left and right subarrays. Finally, sorted elements are combined to form a sorted array.

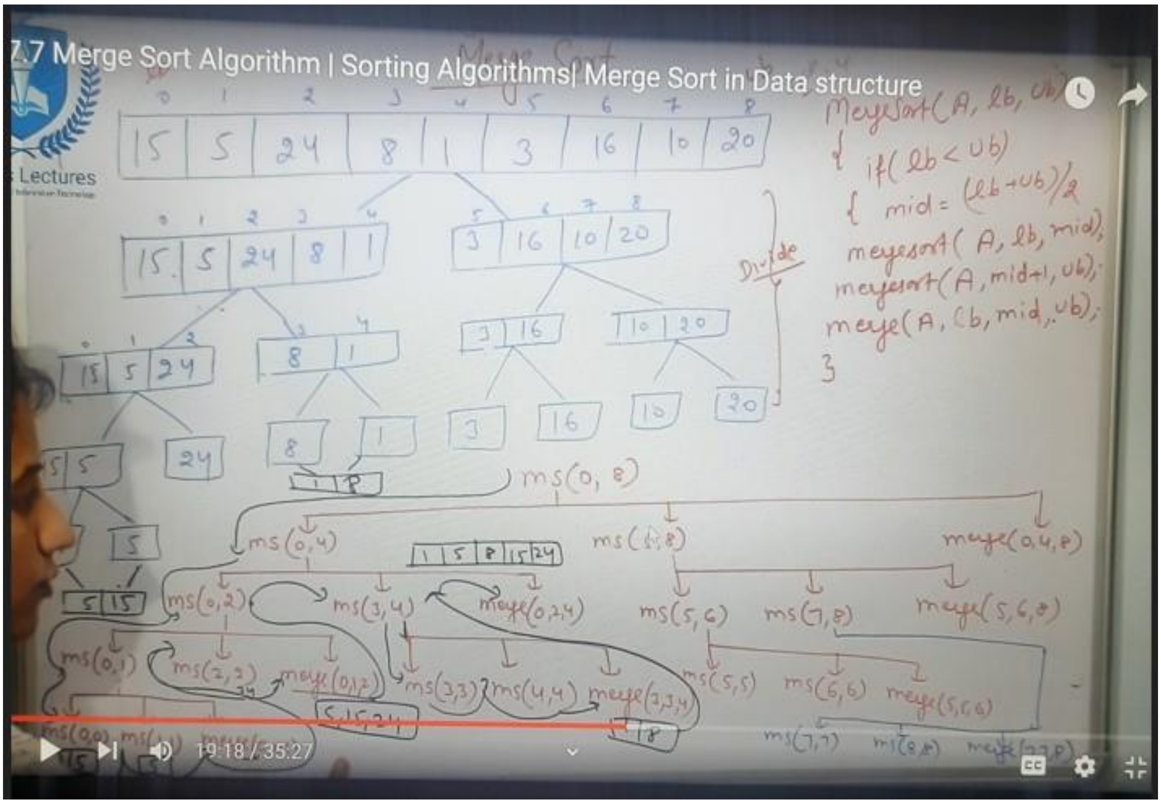



7.6 Quick Sort Algorithm | Sorting Algorithm | Quick Sort Algorithm Explained
lb
ub
0 1 2 3 4 5 6 7 8
7 6 10 5 9 2 1 15 7
start
end
Pivot = 7
23:57 / 24:43

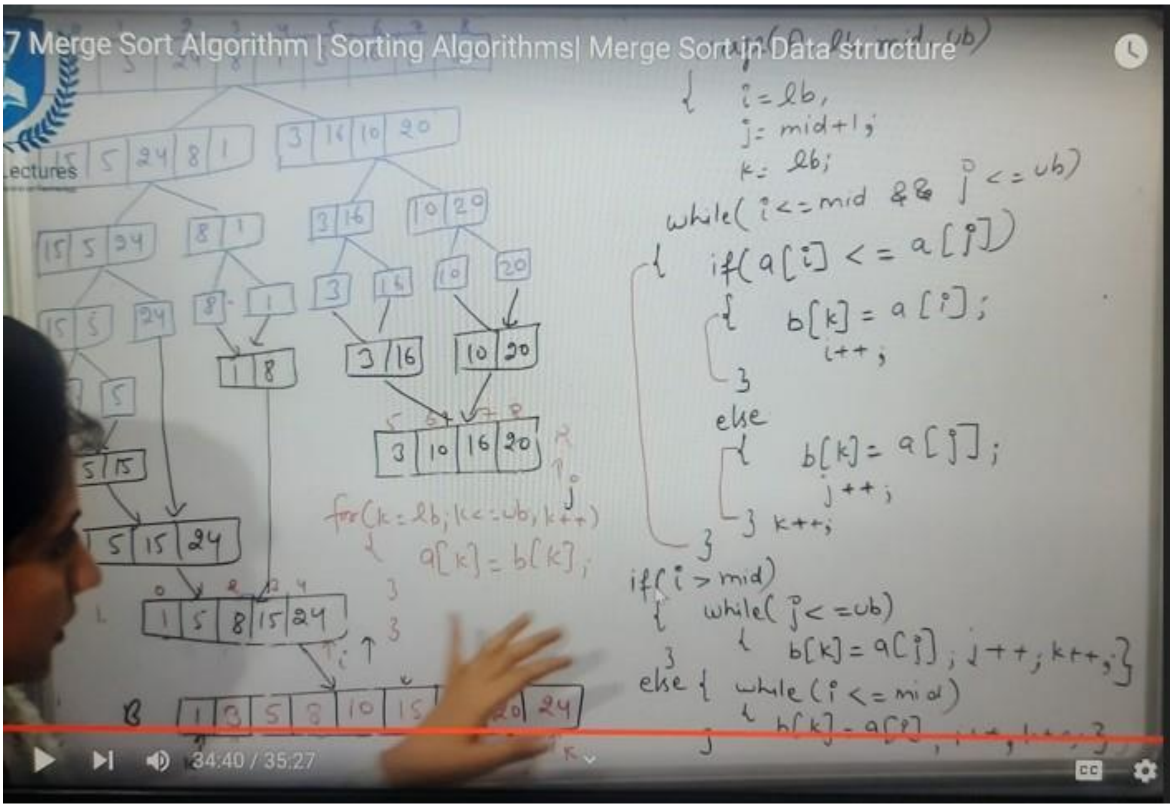
```
QuickSort(A, lb, ub)
{
    if(lb < ub)
    {
        loc = Partition(A, lb, ub);
        QuickSort(A, lb, loc-1);
        QuickSort(A, loc+1, ub);
    }
}

Partition(A, lb, ub)
{
    Pivot = a[lb]
    start = lb;
    end = ub;
    while(start < end)
    {
        while(a[start] <= Pivot)
        {
            start++;
        }
        while(a[end] > Pivot)
        {
            end--;
        }
        if(start < end)
        {
            swap(a[start], a[end]);
        }
    }
    swap(a[lb], a[end]); return end;
}
```

Merge sort:

https://www.youtube.com/watch?v=jlHkDBEumP0&list=PLdo5W4Nhv31bbKJzrsKfMpo_grxuLl8LU&iĀĀĀĀĀĀndex=101



```
MergeSort(A, lb, ub)
{ if(lb<ub)
  { mid= (lb+ub)/2
    mergesort( A, lb, mid);
    mergesort(A, mid+1, ub);
    merge(A, lb, mid, ub);
}
```

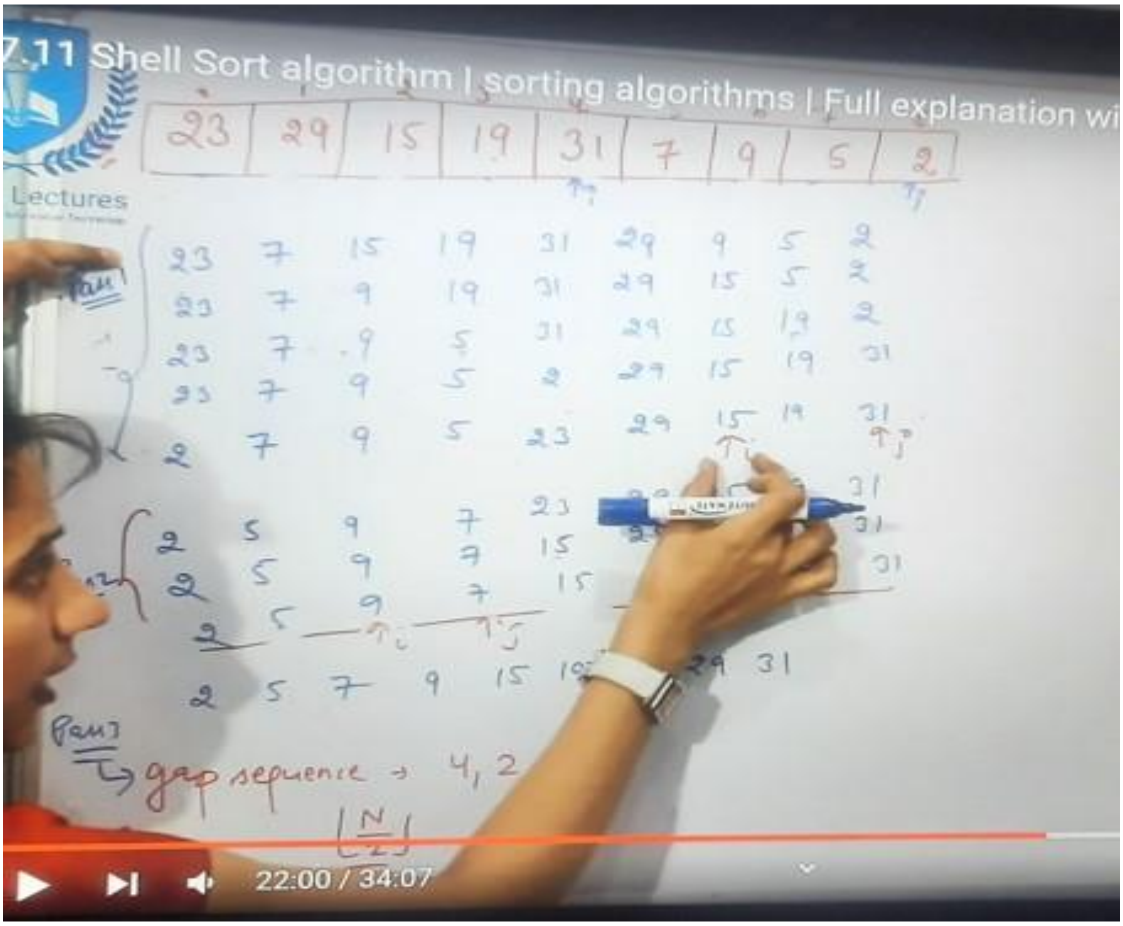

7 Merge Sort Algorithm | Sorting Algorithms| Merge Sort in Data structure
Lectures
3 | 10 | 16 | 20
1 | 5 | 8 | 15 | 24
B
34:40 / 35:27

```
for(k=lb; k<=ub; k++)
{ a[k] = b[k];
}
```

```
{   i=lb,
    j= mid+1;
    k= lb;
while( i<=mid && j<=ub)
{   if(a[i] <= a[j])
    {   b[k] = a[i];
           i++;
    }
    else
    {   b[k]= a[j];
          j++;
    } k++;
}
if(i>mid)
{   while( j<=ub)
      {   b[k]=a[j]; j++; k++; }
}
else { while(i<=mid)
      {   b[k]=a[i]; i++; k++; }
}
```

Shell sort:

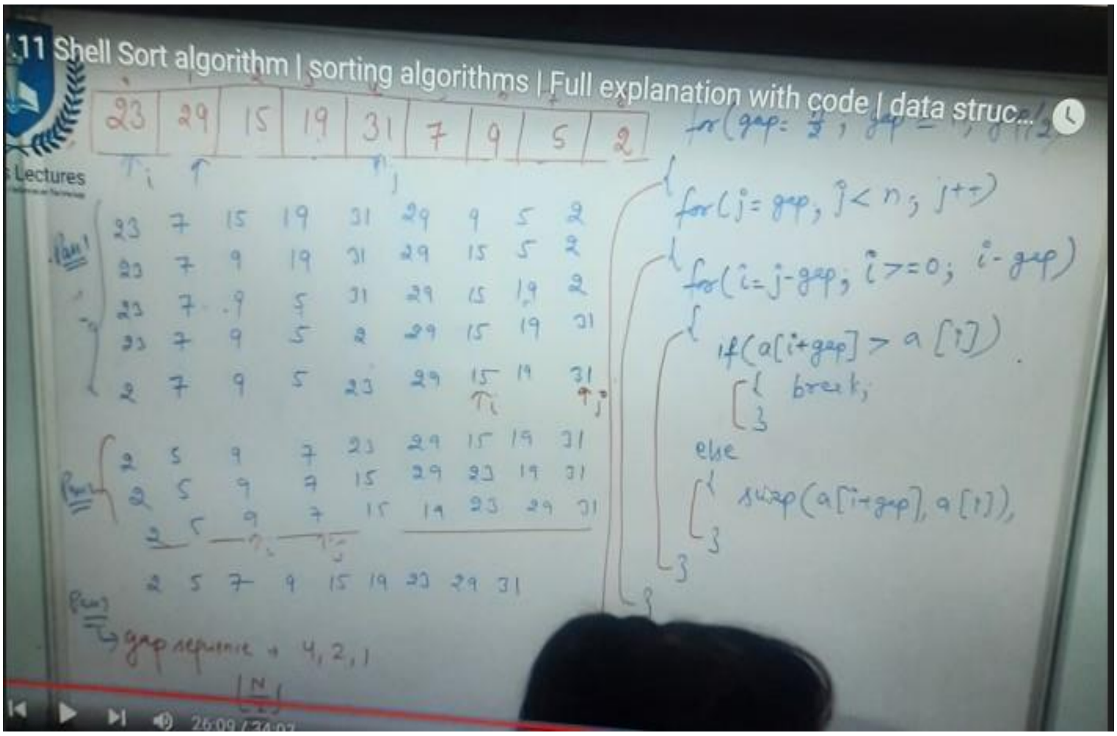
https://www.youtube.com/watch?v=9crZRd8GPWM&list=PLdo5W4Nhv31bbKJzrsKfMpo_grxuLl8LU&index=105



7.11 Shell Sort algorithm | sorting algorithms | Full explanation wi

```
| 23 | 29 | 15 | 19 | 31 | 7 | 9 | 5 | 2 |

Pass1
  23   7   15   19   31   29   9    5    2
  23   7   9    19   31   29   15   5    2
  23   7   9    5    31   29   15   19   2
  23   7   9    5    2    29   15   19   31
  2    7   9    5    23   29   15   19   31

Pass2
  2    5   9    7    23             31
  2    5   9    7    15             31
  2    5   9    7    15             31

  2    5   7    9    15   19   29   31

Pass3
  gap sequence → 4, 2
  ⌊N/2⌋
```

22:00 / 34:07

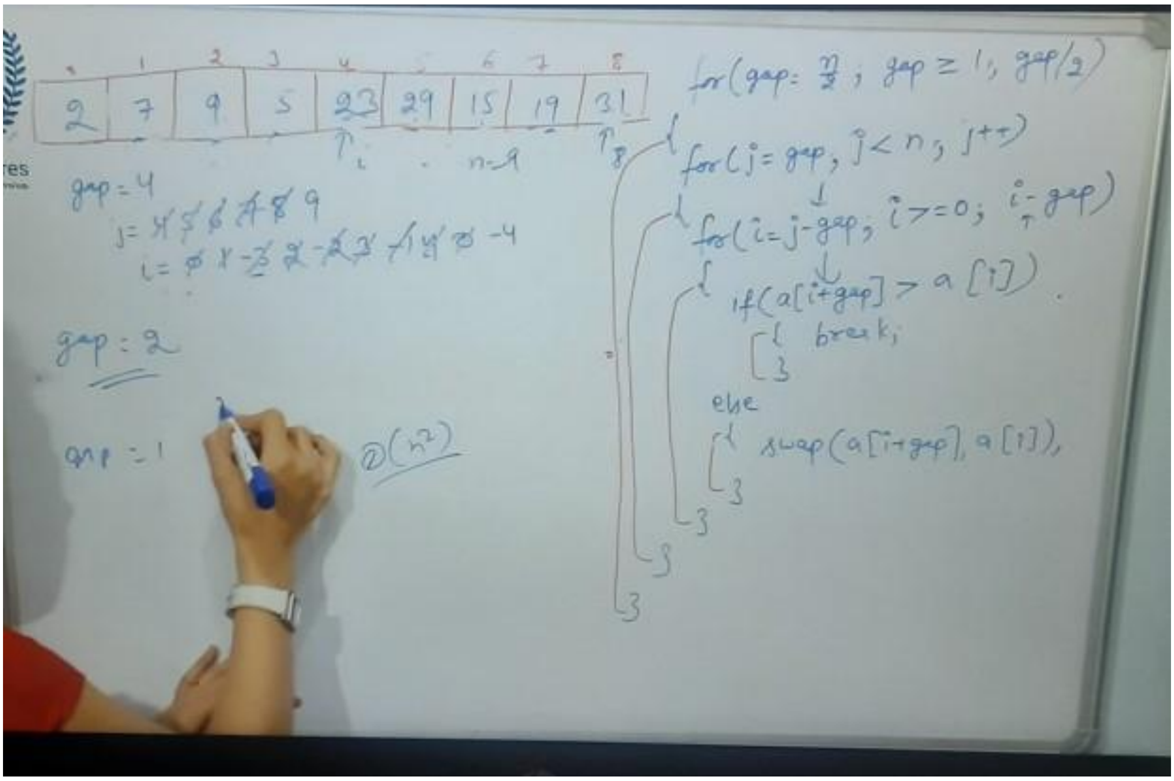


7.11 Shell Sort algorithm | sorting algorithms | Full explanation with code | data struc...

```
| 23 | 29 | 15 | 19 | 31 | 7 | 9 | 5 | 2 |

Pass1
  23   7   15   19   31   29   9    5    2
  23   7   9    19   31   29   15   5    2
  23   7   9    5    31   29   15   19   2
  23   7   9    5    2    29   15   19   31
  2    7   9    5    23   29   15   19   31

Pass2
  2    5   9    7    23   29   15   19   31
  2    5   9    7    15   29   23   19   31
  2    5   9    7    15   19   23   29   31

  2    5   7    9    15   19   23   29   31

Pass3
  gap sequence → 4, 2, 1
  ⌊N/2⌋
```

```
for(gap= N/2 ; gap >= 1; gap/2)
{
  for(j= gap; j< n; j++)
  {
    for(i=j-gap; i>=0; i-gap)
    {
      if(a[i+gap] > a[i])
      {
        break;
      }
      else
      {
        swap(a[i+gap], a[i]);
      }
    }
  }
}
```

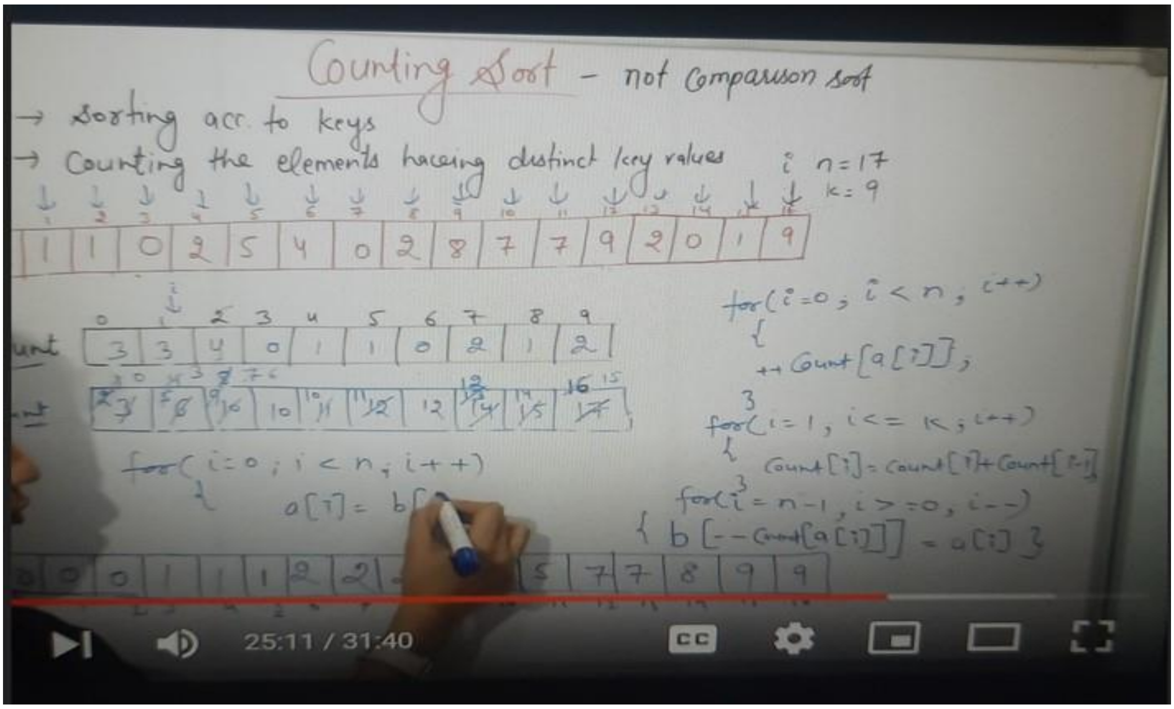
26:09 / 34:07



```
for(gap= n/2 ; gap ≥ 1; gap/2)
{
  for(j= gap, j< n, j++)
  {
    for(i=j-gap, i>=0; i-gap)
    {
      if(a[i+gap] > a [i])
      { break;
      }
      else
      { swap(a[i+gap], a[i]),
      }
    }
  }
}
```

Counting sort:

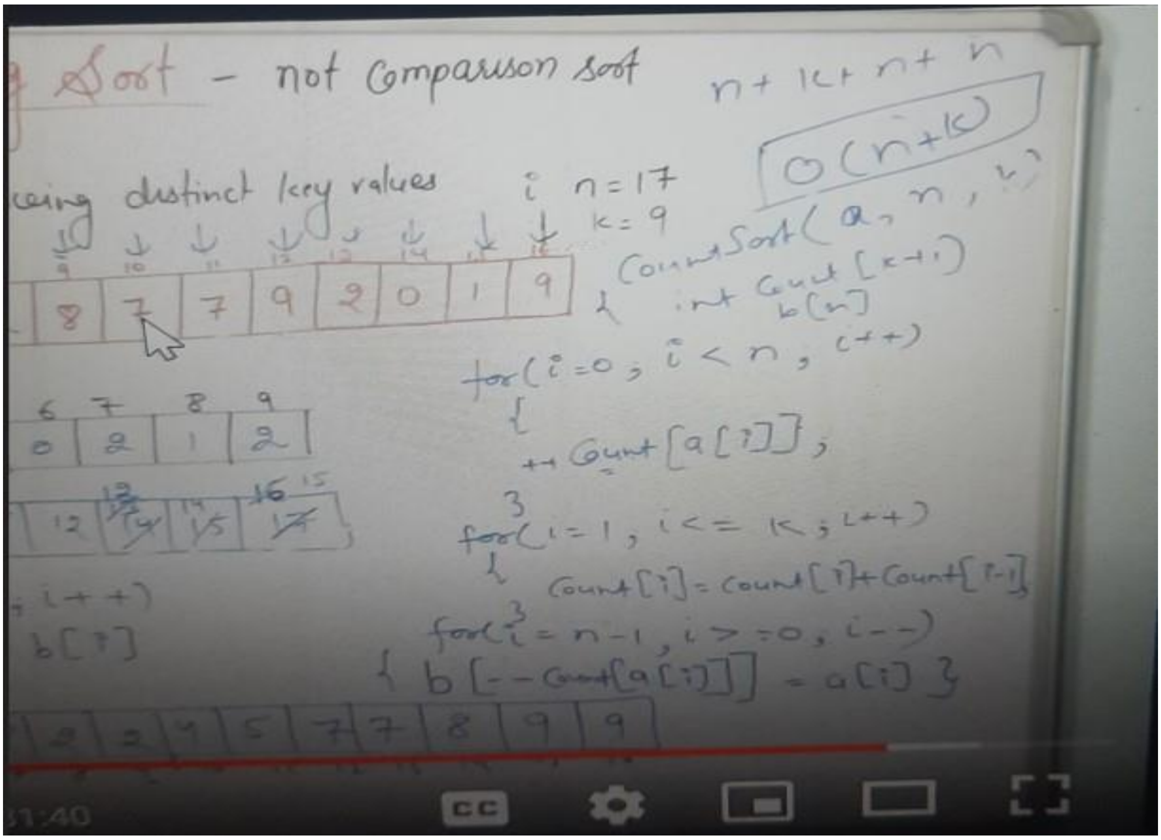
https://www.youtube.com/watch?v=pEJiGC-ObQE&list=PLdo5W4Nhv31bbKJzrsKfMpo_grxuLl8LU&index=106


Counting Sort – not Comparison sort

→ sorting acc. to keys

→ Counting the elements having distinct key values

```
for(i=0; i<n; i++)
{
  ++ Count[a[i]];
}
for(i=1, i<= k; i++)
{
  Count[i]= Count[i]+Count[i-1]
}
for(i= n-1, i>=0, i--)
{ b[--Count[a[i]]] = a[i] }

for(i=0; i<n; i++)
{
  a[i]= b[
```



Sort - not Comparison sort

n+ k+ n+ n

O(n+k)

ceiing distinct key values

n=17
k=9

8	7	7	9	2	0	1	9

6	7	8	9
0	2	1	2

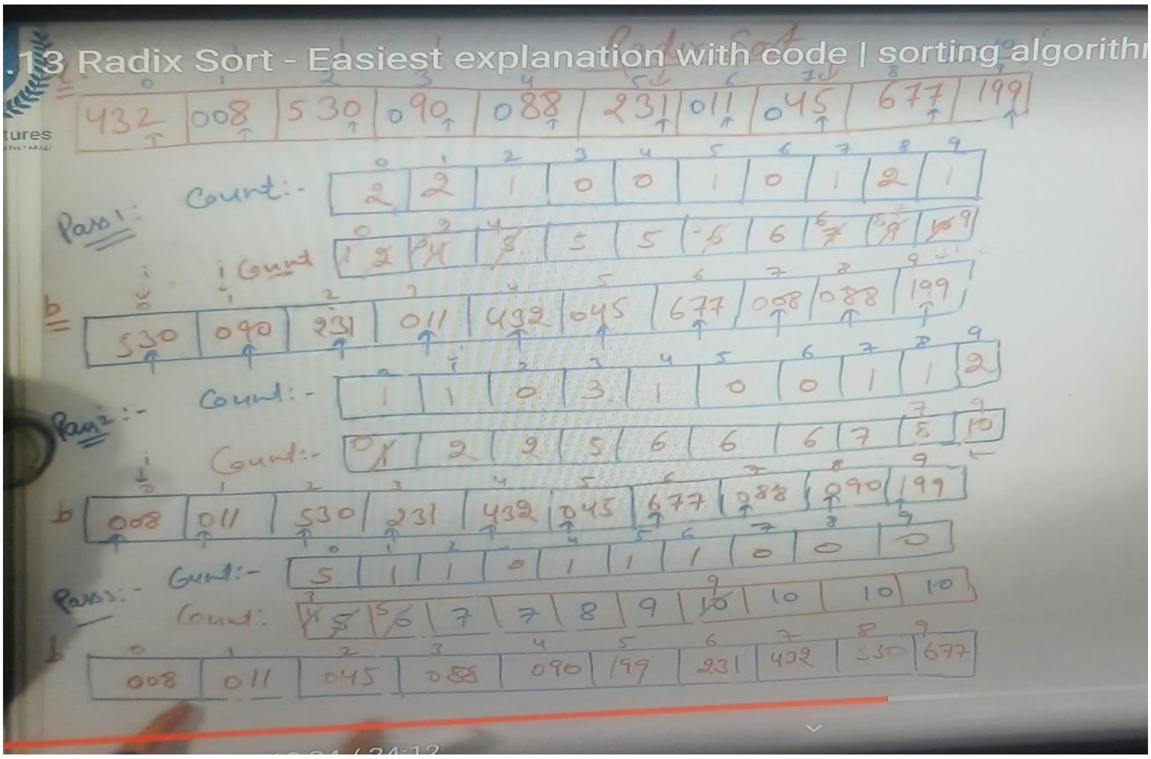
12	~~14~~ 13	~~15~~ 14	~~17~~ 16 15

```
CountSort(a, n, k)
{   int Count[k+1]
        b[n]
    for(i=0; i<n, i++)
    {
        ++Count[a[i]];
    }
    for(i=1, i<=k; i++)
    {
        Count[i]= Count[i]+Count[i-1]
    }
    for(i=n-1, i>=0, i--)
    { b[--Count[a[i]]] = a[i] }
```

i++)
b[i]

2	2	4	5	7	7	8	9	9

Radix/Bucket sort:
https://www.youtube.com/watch?v=Il45xNUHGp0&list=PLdo5W4Nhv31bbKJzrsKfMpo_grxuLl8LU&index=107



.13 Radix Sort - Easiest explanation with code | sorting algorith

Radix Sort

0	1	2	3	4	5	6	7	8	9
432	008	530	090	088	231	011	045	677	199

Pass1:

	0	1	2	3	4	5	6	7	8	9
Count:-	2	2	1	0	0	1	0	1	2	1
Count	~~1~~ 2	~~2~~ 4	~~4~~ 5	5	5	~~6~~ 6	6	~~7~~ 6	~~9~~ 8	~~10~~ 9

	0	1	2	3	4	5	6	7	8	9
b	530	090	231	011	432	045	677	008	088	199

Pass2:-

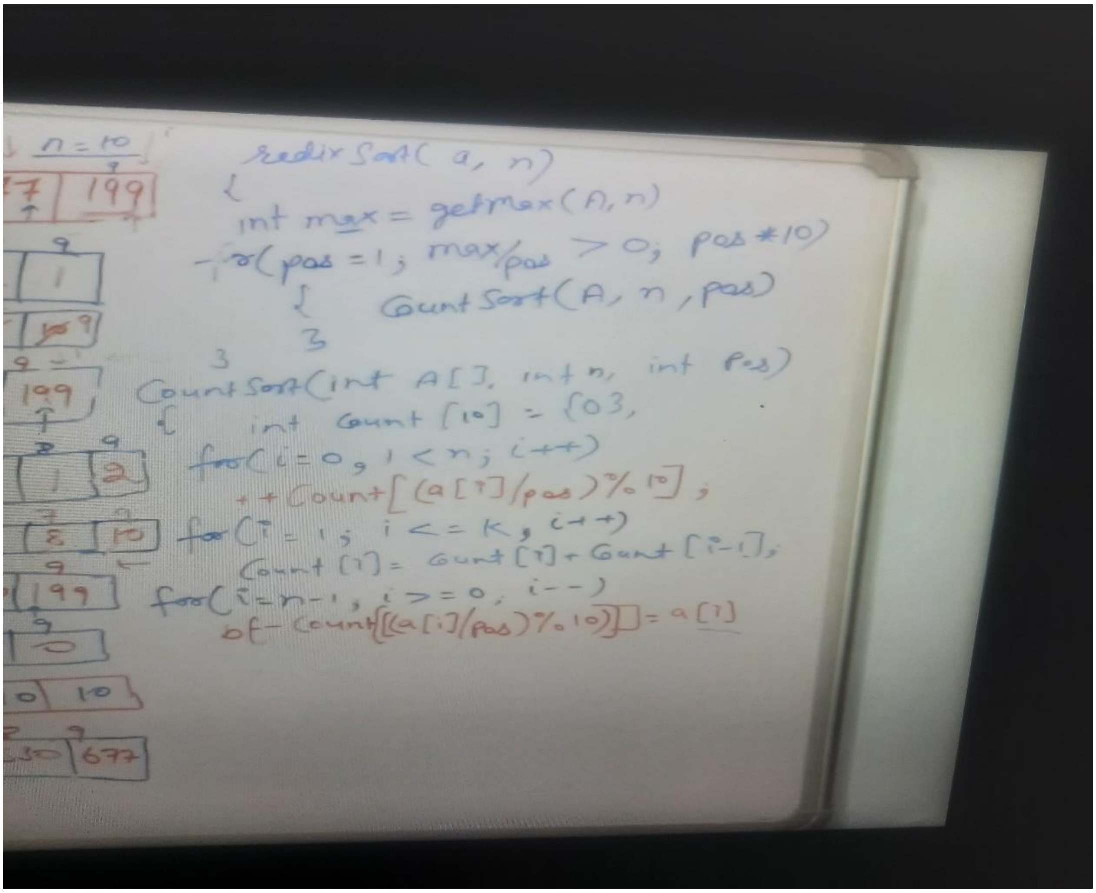
	0	1	2	3	4	5	6	7	8	9
Count:-	1	1	0	3	1	0	0	1	1	2
Count:-	~~1~~ 0	2	2	5	6	6	6	7	~~8~~ 7	~~10~~ 9

	0	1	2	3	4	5	6	7	8	9
b	008	011	530	231	432	045	677	088	090	199

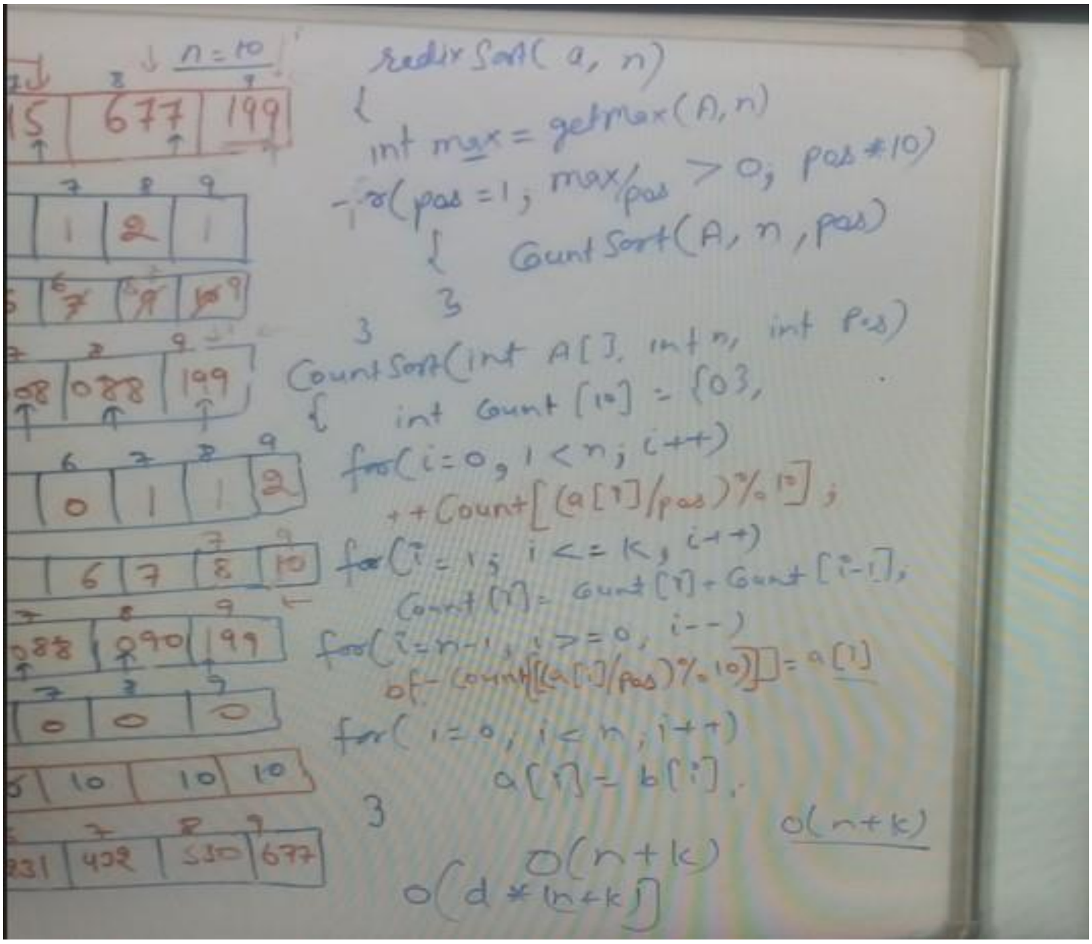
Pass3:-

	0	1	2	3	4	5	6	7	8	9
Count:-	5	1	1	0	1	1	1	0	0	0
Count:	~~4~~ 5	~~5~~ 6	7	7	8	9	~~10~~ 9	10	10	10

	0	1	2	3	4	5	6	7	8	9
b	008	011	045	088	090	199	231	432	530	677



```
redix SortA( a, n)
{
   int max = getmax(A, n)
   for(pos = 1; max/pos > 0; pos *10)
      {   Count Sort(A, n, pos)
      }
}
CountSort(int A[], int n, int Pos)
{    int Count [10] = {0},
   for(i=0, i<n; i++)
      ++Count[(a[i]/pos)%10];
   for(i = 1; i <= k, i++)
      Count [i] = Count [i] + Count [i-1];
   for(i=n-1, i>=0, i--)
      of- Count[[(a[i]/pos)%10)]] = a [i]
```



```
redix SortA( a, n)
{
   int max = getmax(A, n)
   for(pos = 1; max/pos > 0; pos *10)
      {   Count Sort(A, n, pos)
      }
}
CountSort(int A[], int n, int Pos)
{    int Count [10] = {0},
   for(i=0, i<n; i++)
      ++Count[(a[i]/pos)%10];
   for(i = 1; i <= k, i++)
      Count [i] = Count [i] + Count [i-1];
   for(i=n-1, i>=0, i--)
      of- Count[[(a[i]/pos)%10)]] = a [i]
   for( i=0, i<n, i++)
      a[i] = b[i].
}
```

$O(n+k)$

$O(n+k)$

$O(d * (n+k))$