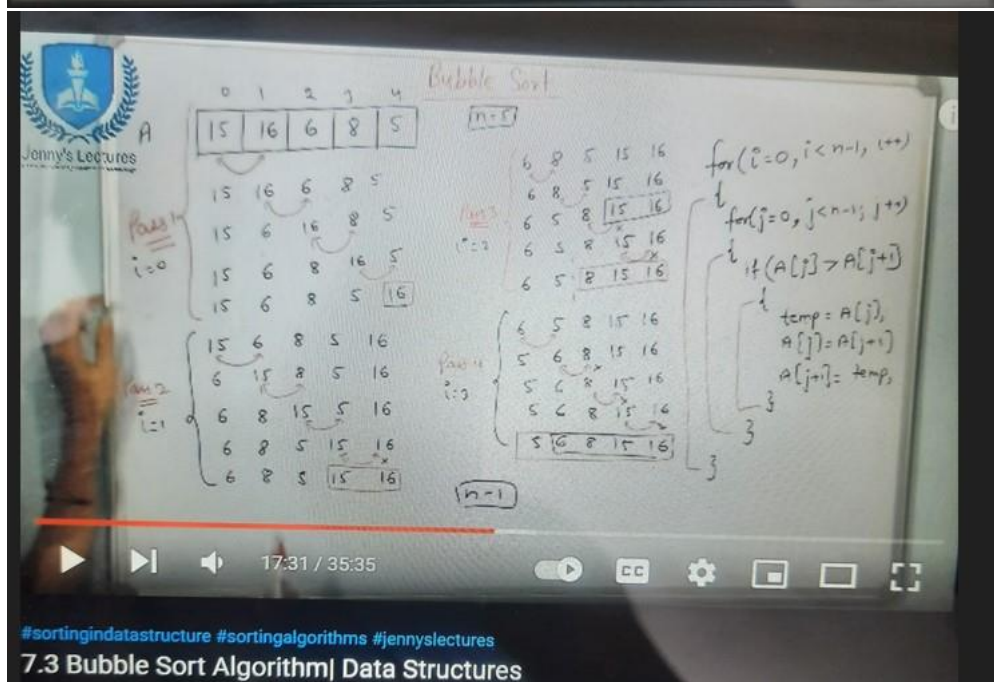
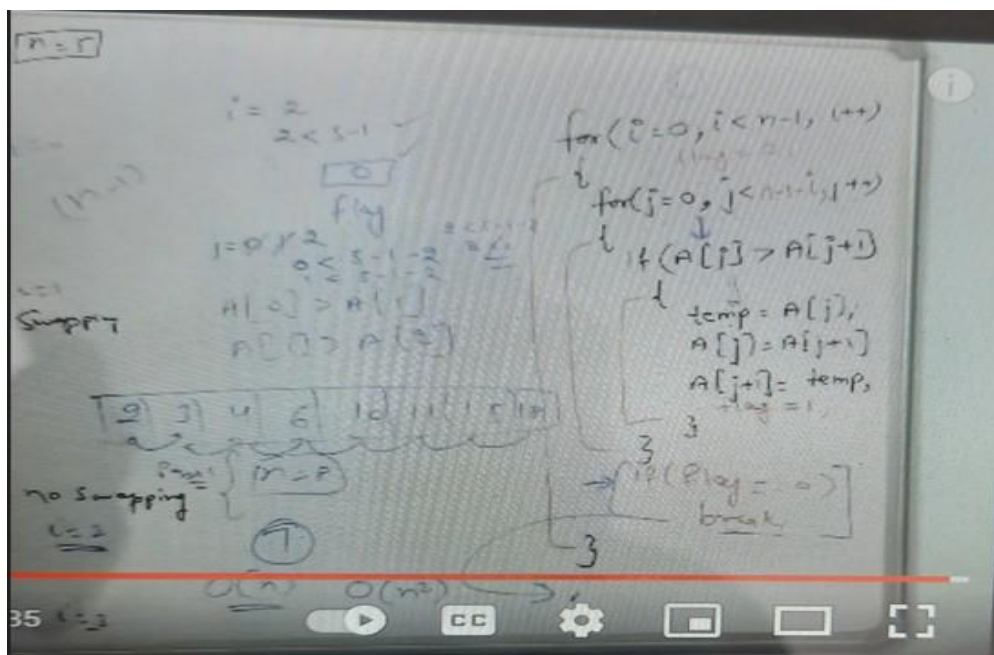


Algorithm Overview:

Bubble sort:

https://www.youtube.com/watch?v=o4bAoo_gFBU&list=PLdo5W4Nhv31bbKJzrSkfMpo_grxUL8LU&index=97

<https://github.com/SushantPoman/DataStructures/blob/master/src/BubbleSort.java>



Insertion sort:

https://www.youtube.com/watch?v=yCxV0kBpA6M&list=PLdo5W4Nhv31bbKJzrsKfMpo_grxulL8LU&index=99

<https://github.com/SushantPoman/DataStructures/blob/master/src/InsertionSort.java>

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

Insertion Sort Algorithm | Data Structure

Sort

$n=6$

Initial Array:

5	4	10	1	6	2
---	---	----	---	---	---

Step 1:

4	5	10	1	6	2
---	---	----	---	---	---

Step 2:

4	5	10	1	6	2
---	---	----	---	---	---

Step 3:

4	5	1	10	6	2
---	---	---	----	---	---

Step 4:

4	1	5	10	6	2
---	---	---	----	---	---

Step 5:

1	4	5	10	6	2
---	---	---	----	---	---

Step 6:

1	4	5	6	10	2
---	---	---	---	----	---

Step 7:

1	2	4	5	6	10
---	---	---	---	---	----

Pseudocode:

```
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;  
}
```

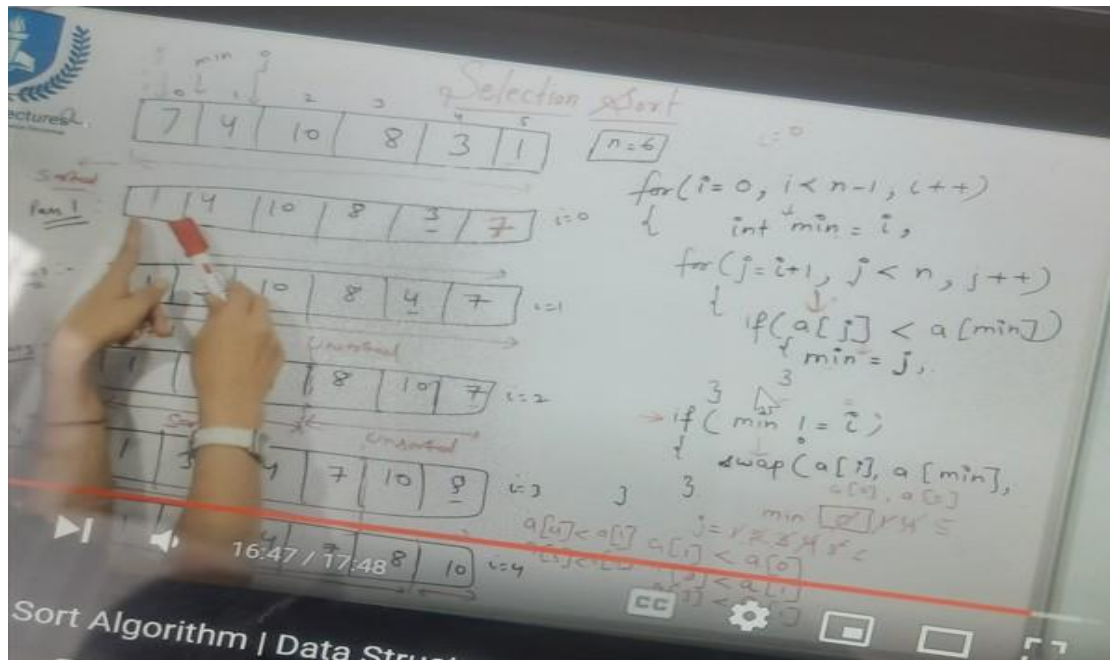
Example:

$i = 5$ (value 2)
 $j = 4$ (value 10)
 $a[5] = a[4]$
 $a[4] = a[3]$
 $a[3] = a[2]$
 $a[2] = a[1]$
 $a[1] = a[0]$

Selection sort:

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

<https://github.com/SushantPoman/DataStructures/blob/master/src/SelectionSort.java>



Quicksort:

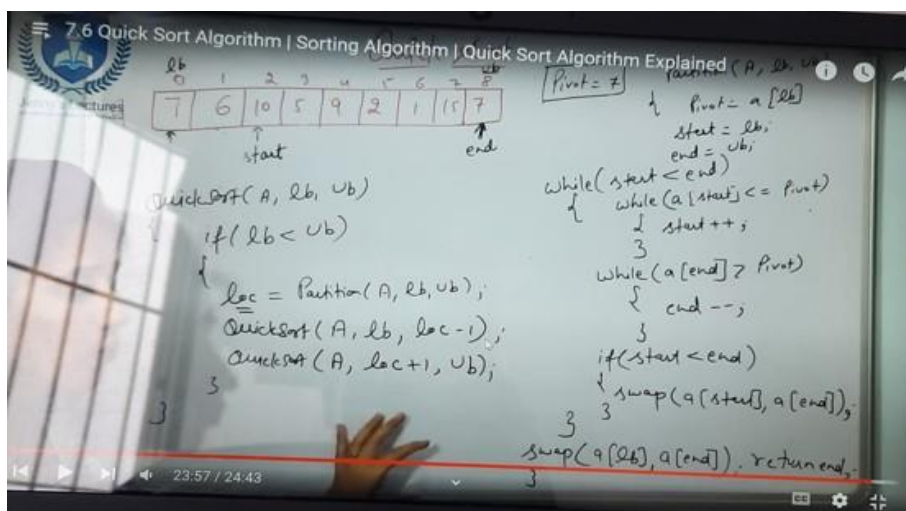
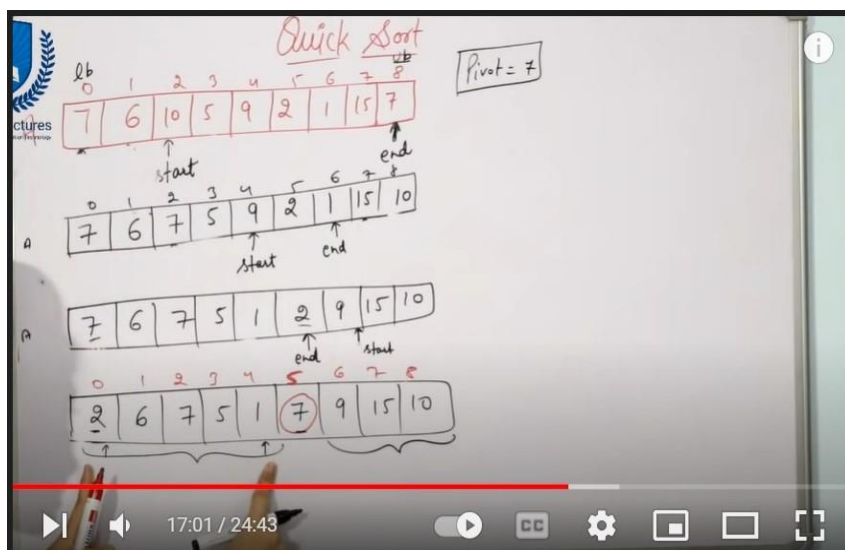
https://www.youtube.com/watch?v=QN9hnmAgmOc&list=PLdo5W4Nhv31bbKJzrsKfMpo_grxuLl8LU&index=100

<https://github.com/SushantPoman/DataStructures/blob/master/src/QuickSort.java>

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.

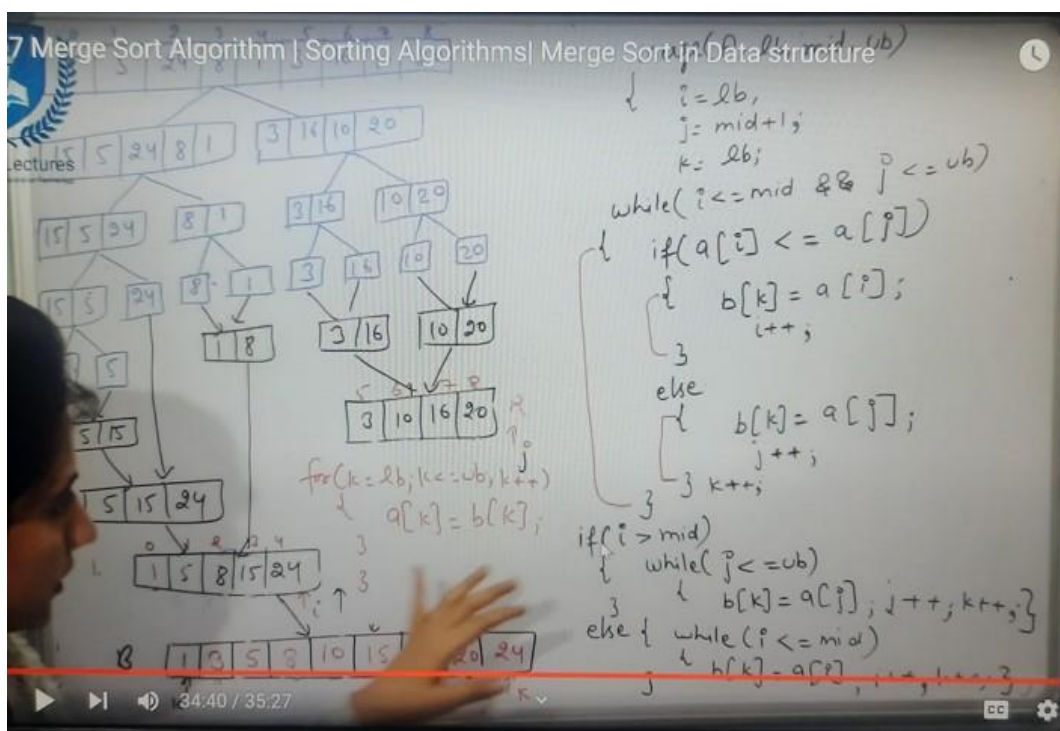
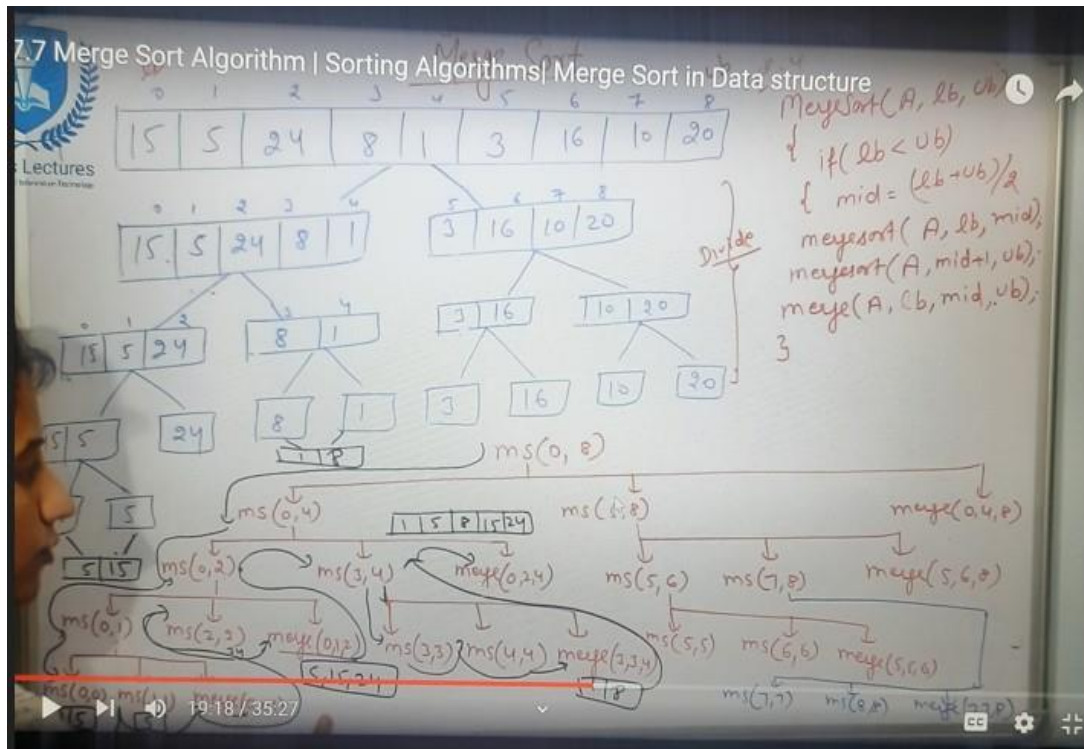
The same process is continued for both left and right subarrays. Finally, sorted elements are combined to form a sorted array.



Merge sort:

https://www.youtube.com/watch?v=jlHkDBEumPO&list=PLdo5W4Nhv31bbKJzrsKfMpo_grxuLl8LU&i&index=101

<https://github.com/SushantPoman/DataStructures/blob/master/src/MergeSort.java>



Shell sort:

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

<https://github.com/SushantPoman/DataStructures/blob/master/src/ShellSort.java>

7.11 Shell Sort algorithm | sorting algorithms | Full explanation with

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

Lectures

Part 1

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

Part 2

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

Part 3

gap sequence $\rightarrow 4, 2$

$\lfloor \frac{N}{2} \rfloor$

22:00 / 34:07

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

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

Lectures

Part 1

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

Part 2

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

Part 3

gap sequence $\rightarrow 4, 2, 1$

$\lfloor \frac{N}{2} \rfloor$

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

25:09 / 34:07

	1	2	3	4	5	6	7	8
2	7	9	5	23	29	15	19	31

Counting sort:

[https://www.youtube.com/watch?v=pEJiGC-](https://www.youtube.com/watch?v=pEJiGC-ObQE&list=PLdo5W4Nhv31bbKJzrsKfMpo_grxuLI8LU&index=106)

[ObQE&list=PLdo5W4Nhv31bbKJzrsKfMpo_grxuLI8LU&index=106](https://www.youtube.com/watch?v=pEJiGC-ObQE&list=PLdo5W4Nhv31bbKJzrsKfMpo_grxuLI8LU&index=106)

<https://github.com/SushantPoman/DataStructures/blob/master/src/CountSort.java>

Counting Sort - not Comparison sort

→ Sorting acc. to keys
→ Counting the elements having distinct key values

$n = 17$
 $k = 9$

1	1	0	2	5	4	0	2	8	7	7	9	2	0	1	9
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

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

0	1	2	3	4	5	6	7	8	9
2	3	4	5	6	7	8	9	10	11
12	13	14	15	16	17				

```
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];  
}
```

0	0	1	1	1	2	2	5	7	7	8	9	9
---	---	---	---	---	---	---	---	---	---	---	---	---

Counting Sort - not Comparison sort

Counting the elements having distinct key values

$n = 17$
 $k = 9$

8	7	7	9	2	0	1	9
---	---	---	---	---	---	---	---

6	7	8	9
0	2	1	2

12	13	14	15	16	17
----	----	----	----	----	----

$O(n+k)$

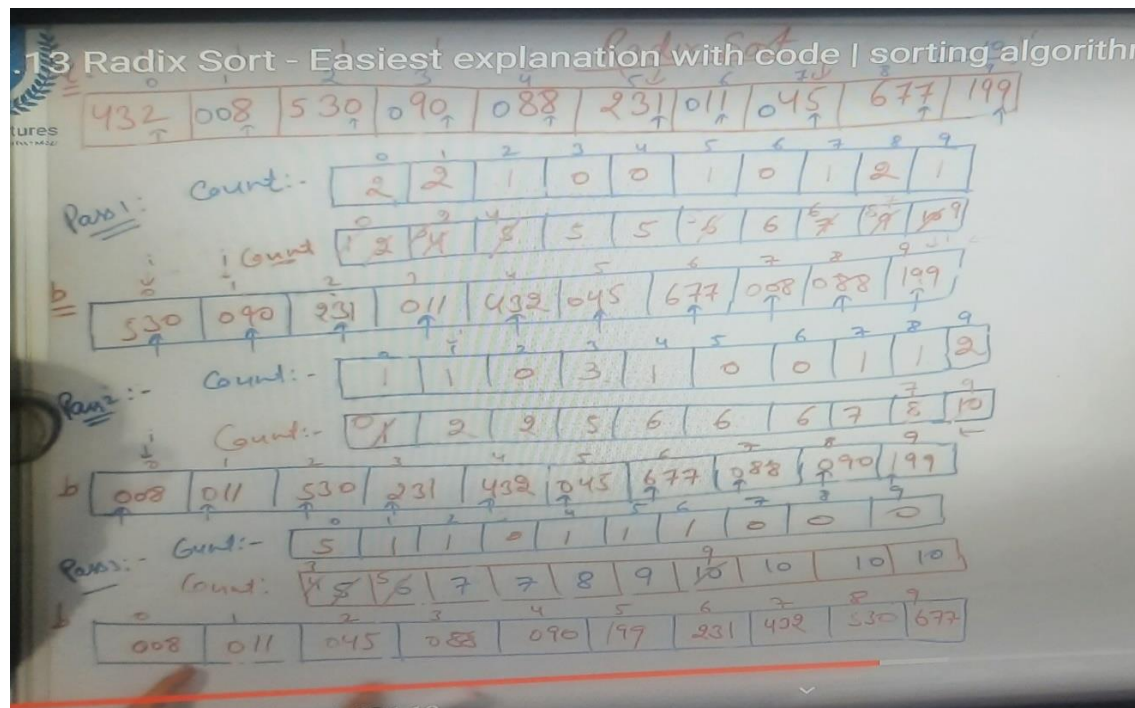
```
CountSort(a, n, k)  
{  
    int Count[k+1];  
    for(i=0; i<n; i++)  
    {  
        ++Count[a[i]];  
    }  
    for(i=1; i<=k; i++)  
    {  
        Count[i] = Count[i] + Count[i-1];  
    }  
    for(i=n-1; i>=0; i--)  
    {  
        b[--Count[a[i]]] = a[i];  
    }  
}
```

0	0	1	1	1	2	2	5	7	7	8	9	9
---	---	---	---	---	---	---	---	---	---	---	---	---

Radix/Bucket sort:

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

<https://github.com/SushantPoman/DataStructures/blob/master/src/RadixSort.java>



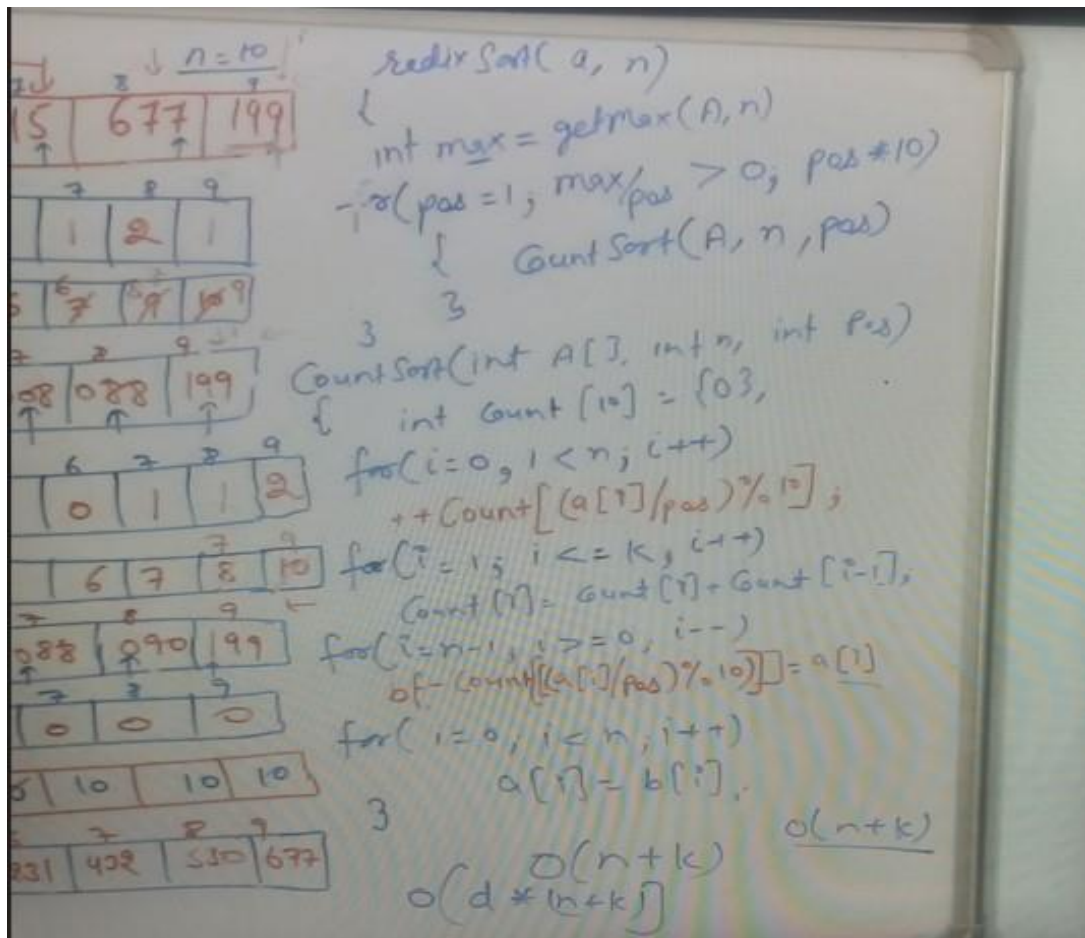
$n = 10$

Radix Sort(a, n)

```
int max = getMax(A, n)
for (pos = 1; max / pos > 0; pos *= 10)
    CountSort(A, n, pos)
```

CountSort($\text{int } A[], \text{int } n, \text{int } pos$)

```
int Count[10] = {0};
for (i = 0; i < n; i++)
    ++Count[(A[i] / pos) % 10];
for (i = 1; i <= 9; i++)
    Count[i] = Count[i] + Count[i - 1];
for (i = n - 1; i >= 0; i--)
    a[i] = Count[(A[i] / pos) % 10];
```



2D Array:

https://www.youtube.com/watch?v=KDQXUysHLL8&list=PLdo5W4Nhv31bbKJzrskfMpo_grxul8LU&index=5