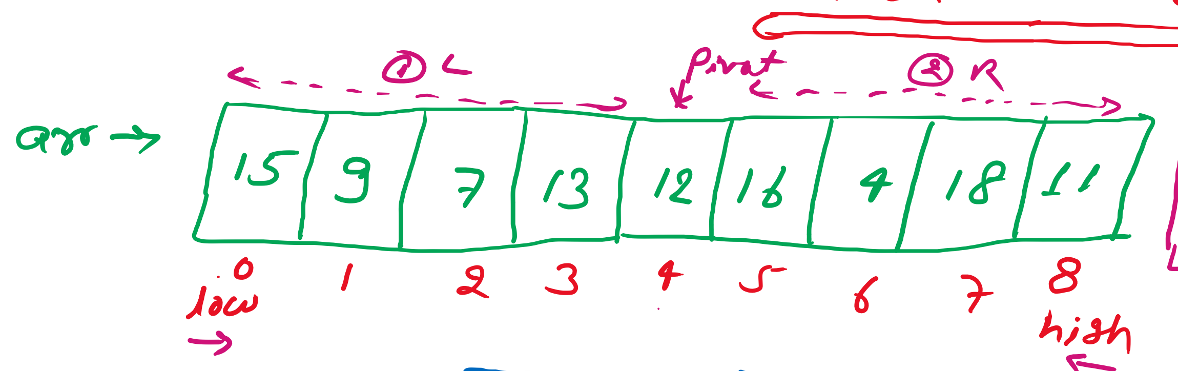
Quick Sort Algorithm



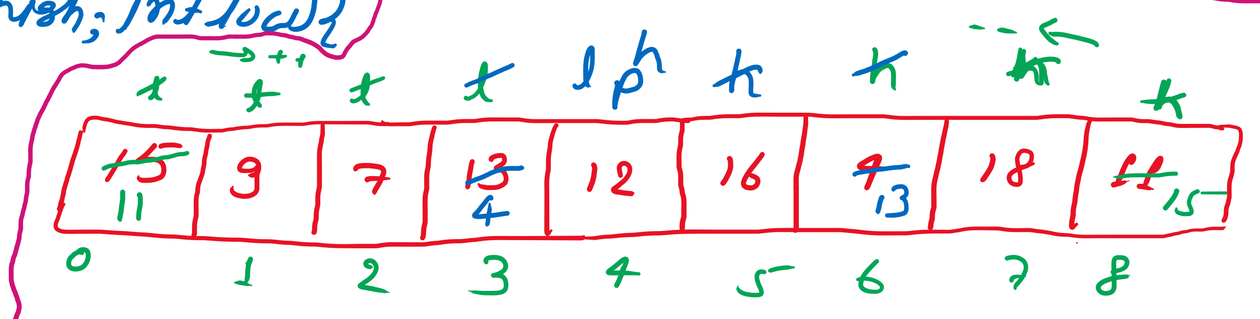
length = 9 , low = 0 , high = length-1 = 8
pivot = arr[(low+high)/2] ; (0+8)/2 ⇒ 4

index → low = 0 (l)
index → high = 8 (h)
~~Nodde~~ Pivot = 4 (P)

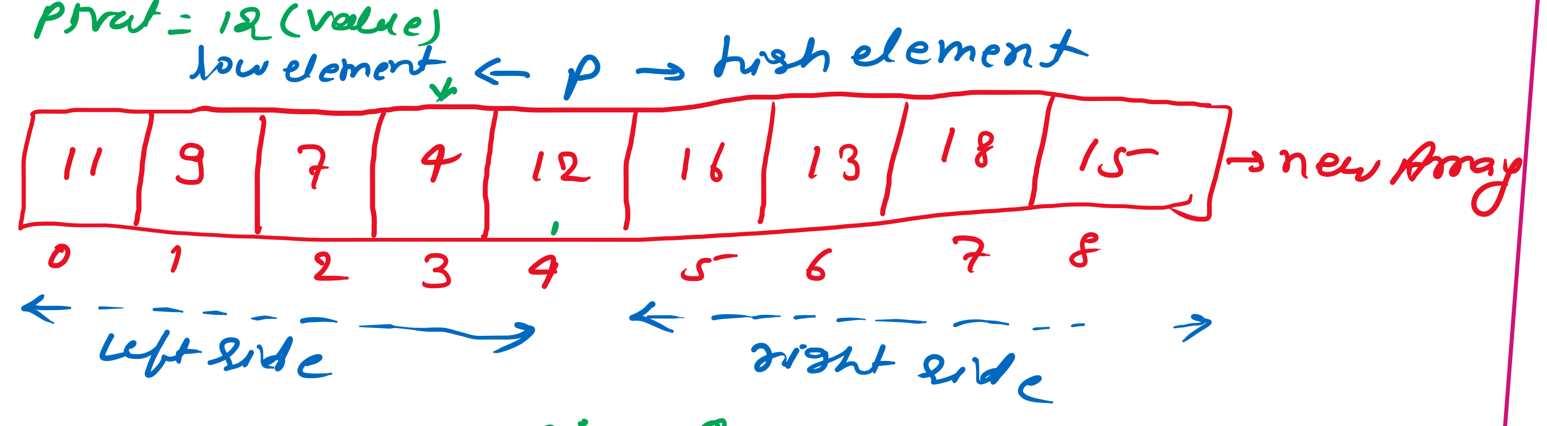
① Arrange left side which is less than pivot.
② Arrange right sid which is greater than pivot

element →	15	9	7	13	12	16	4	18	11
index →	0	1	2	3	4	5	6	7	8
	l				p				h

```
public static int partition(int arr[], int high, int low){
    while( low <= high){
        while (arr[low] < pivot){
                low++;
        }
        while(arr[high] > pivot){
                high--;
        }
        if( low <= high){
            int temp = arr[low];
            arr[low] = arr[high];
            arr[high] = temp;
              low++;
              high--;
        }
    return low;
}
```



low = 0(index) → ~~1~~ ~~2~~ ~~3~~ ~~4~~ 5
high = 8 (index) → ~~7~~ ~~6~~ ~~5~~ ~~4~~ 3
pivot = 12 (value)



if(low < pi-1)   pi = 5

```
public static void quickSort
    (int arr[], int low, int high){
  int partitionIndex = partition(arr, low, high);
    if( low < partitionIndex-1){
        quickSort(arr, low, partitionIndex-1);
    }
    if(partitionIndex < high){
        quickSort(arr, partitionIndex, high);
    }
}
```

```
public static void main(String args[]){
  int arr[] = {15, 9, 7, 13, 12, 16, 4, 18, 11};
     quickSort(arr, 0, arr.length-1);
   printArray(arr);
}
```

```
public static void printArray( int arr[]){
    for( int i : arr){ // foreach
    sop( i+" ");
    }
}
```

Covered topic (02-10-2023)
① QuickSort Algorithm Ascending order

tomorrow topic (03-10-2023)
① merge Sort Algorithm
② QuickSort Algorithm Descending order.