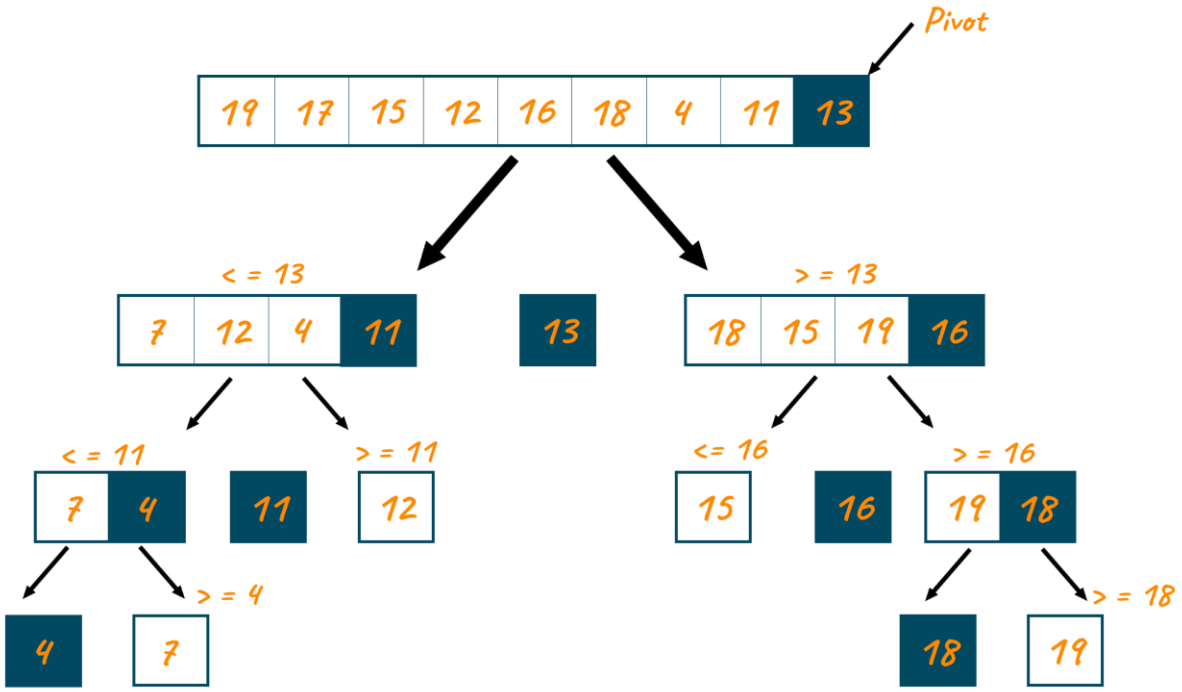
QUICK SORT

This algorithm is quick or fast in terms of speed and this is one of the reasons why it is so famous. This algorithm is based on the divide and conquer approach of programming. The divide and conquer approach involves dividing a task into small atomic sub-tasks and then solving those subtasks. The results of those subtasks are then combined and evaluated to get the final result.

Approach of the quick sort algorithm. In the quick sort algorithm, we select a pivot element and we position the pivot in such a manner that all the elements smaller than the pivot element are to its left and all the elements greater than the pivot are to its right. The elements to the left and right of the pivot form two separate arrays. Now the left and right subarrays are sorted using this same approach. This process continues until each subarray consists of a single element. When this situation occurs, the array is now sorted and we can merge the elements to get the required array.



CODE:

```
import java.util.*;
import java.lang.*;
import java.io.*;

public class Main
{
   public static void main (String[] args)
   {
      int array[] = new int[]{8,4,7,9,3,10,5};

      int n = array.length;
      quickSort(array,0,n-1);

          for(int x: array)
             System.out.print(x+" ");

   }

   static int iPartition(int array[], int l, int h)
   {
```

```
    int pivot=array[h];
    int i=l-1;
    for(int j=l;j<=h-1;j++){
        if(array[j]<pivot){
            i++;
            int temp=array[i];
            array[i]=array[j];
            array[j]=temp;
        }
    }
    int temp=array[i+1];
    array[i+1]=array[h];
    array[h]=temp;
    return i+1;
}

static void quickSort(int array[],int l,int h){
    if(l<h){
        int p=iPartition(array,l,h);
        quickSort(array,l,p-1);
        quickSort(array,p+1,h);
    }
```

```
    }
}
```