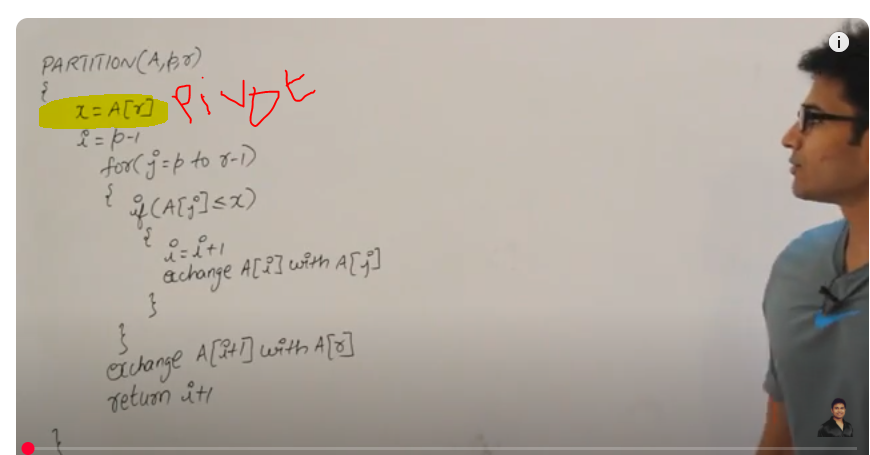
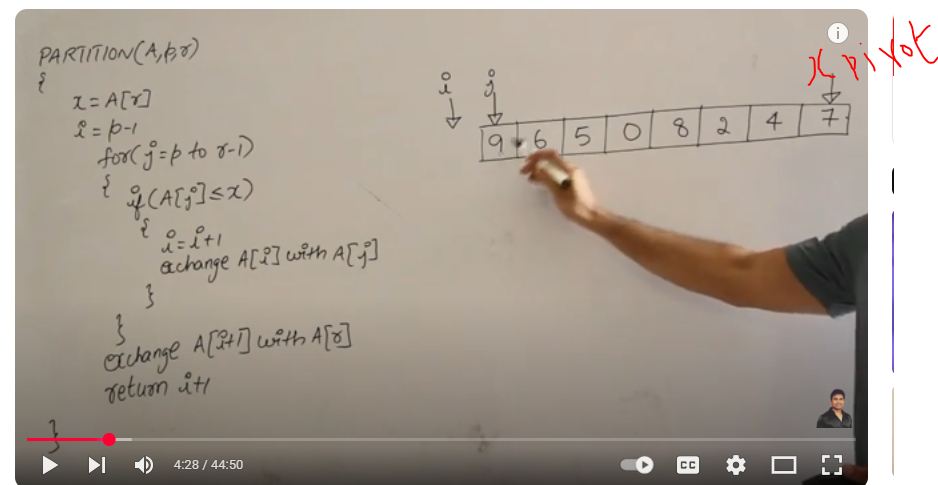
Quick sort will be used for less number of values.





In the above diagram p =lower bound, r=upper bound

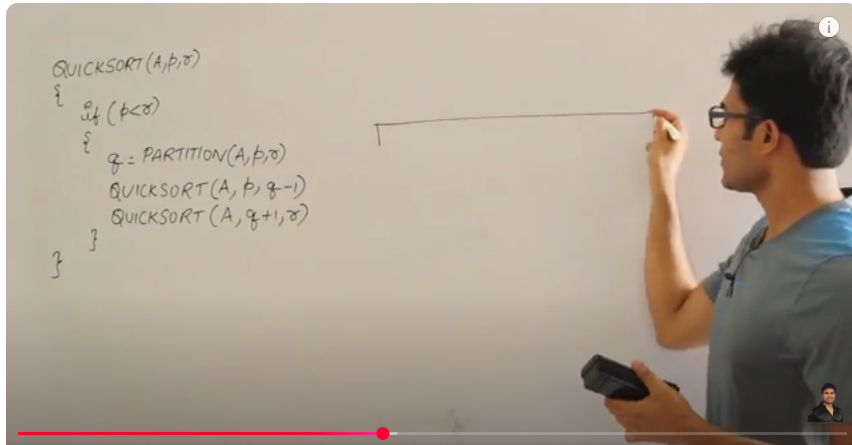
J=0;

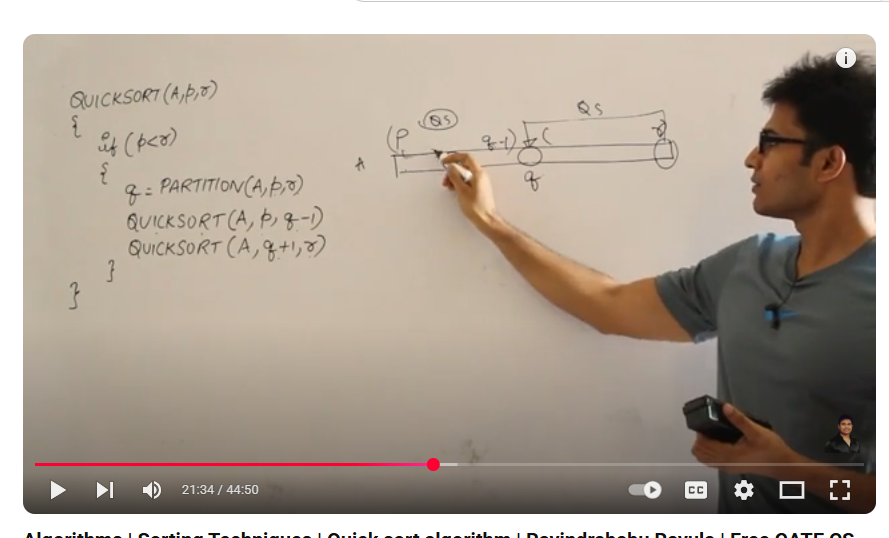
I=j-1;

J travel from 0 to lenth-1

If any element at j position less than pivot value increase I value by one and swap I+1 value with j value.

After the completion of the for loop swap i+1 value with pivot value





public class QuickSort {

public static void quickSort(int[] arr, int low, int high) {

if (low < high) { // to check at least two elements

// Find the partition index

int pi = partition(arr, low, high);

// Recursively sort the subarrays

quickSort(arr, low, pi - 1);

quickSort(arr, pi + 1, high);

}

}

private static int partition(int[] arr, int low, int high) {

int pivot = arr[high]; // Choosing last element as pivot pivot

int i = low - 1; // Index for smaller elements I startsa from -1

for (int j = low; j < high; j++) { j starts from 0

if (arr[j] < pivot) { // If current element is smaller than pivot

i++;

swap(arr, i, j);

}

}

// Place pivot in correct position

swap(arr, i + 1, high); for swapping pivot with i+1 element;

return i + 1;

}

private static void swap(int[] arr, int i, int j) {

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

// Test the quicksort

public static void main(String[] args) {

int[] arr = {10, 7, 8, 9, 1, 5};

quickSort(arr, 0, arr.length - 1);

System.out.println("Sorted array:");

for (int num : arr) {

System.out.print(num + " ");

}

}

}