Quick Sort

10, 80, 30, 90, 40, 50, 70 start=0, end=6

10, 30, 40, 50, 70, 90, 80

10, 30, 40, 50, 70, 90, 80

10, 30, 40, 50, 70, 90, 80

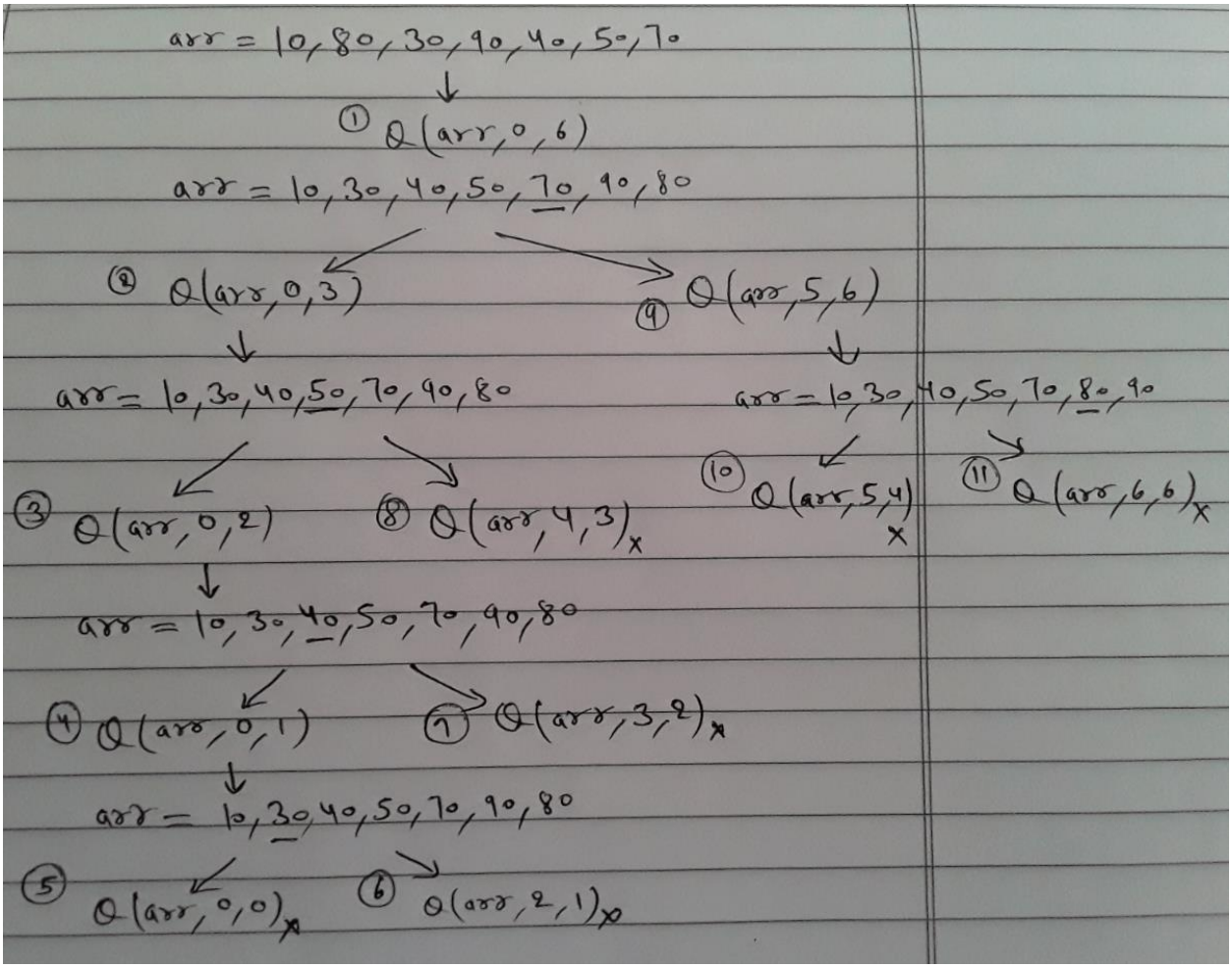
10, 30, 40, 50, 70, 90, 80

10, 30, 40, 50, 70, 90, 80

10, 30, 40, 50, 70, 80, 90

10, 30, 40, 50, 70, 80, 90

```
public static void sort(int arr[], int start, int end)
   {
          if (start < end)
          {
                int pIndex = partition(arr, start, end);
                sort(arr, start, pIndex-1);
                sort(arr, pIndex+1, end);
          }
   }
```



Selecting a Pivot

- *Always select last element as pivot*
- *Always select first element as pivot*
- *Pick middle element as pivot*
- *Pick a random element as pivot*

Example 1 :

Array => [10, 7] start = 0, end =1

quicksort([10, 7], 0, 1) Array => [7, 10] Partition Index = 0

quicksort([7, 10], 0, -1) end is not greater than start

quicksort([7, 10], 1, 1) end is not greater than start

Example 2 :

Array => [7, 10] start=0, end =1

quicksort([7, 10], 0, 1) Array => [7, 10] Partition Index = 1

quicksort([7, 10], 0, 0) end is not greater than start

quicksort([7, 10], 2, 1) end is not greater than start

Left side of Partition Index there are no elements

Right side of Partition Index there are no elements

Left side of Partition Index there is only one element

Right side of Partition Index there is only one element

Partitioning

i is used to keep track of index where pivot element should come

j is used to iterate over the array so that we can compare array elements with pivot

Examples :

Array => [10, 11, 77, 45, 56] start=0, end=4

Array => [10, 11, 45, ***56***, 77] Partition Index =3

```
public static int partition(int arr[], int start, int end)
{
	int pivot = arr[end];
	int i = start ;
	for (int j = start; j < end; j++)
	{
		if (arr[j] <= pivot)
		{
			int temp = arr[i];
			arr[i] = arr[j];
			arr[j] = temp;
			i++;
		}
	}

	int temp = arr[i];
	arr[i] = pivot;
	arr[end] = temp;

	return i;
}
```

Array => [5, 43, 34, 10, 7, 6] start=0, end=5

Array => [5, ***6***, 34, 10, 7, 43] Partition Index = 1

Array => [9, 6, 11, 70, 3] start=0, end =4

Array => [***3***, 6, 11, 70, 9] Partition Index = 0