Quick Sort

Prof. Harish D.G.
Dept. of Computer and IT
College of Engineering,Pune
www.harishgadade.com

Quick Sort

- The quick sort sort algorithm uses the divide and conquer strategy.
- Quick sort on an input array with ‘n’ elements consists of three steps.

1. Divide - Partitions (divide) array into two sublists s1 & s2 with n / 2 approximate elements each.

2. Conquer- Then sort sub list s1 & s2

3. Combine- Merge s1 & s2 into a unique sorted group.

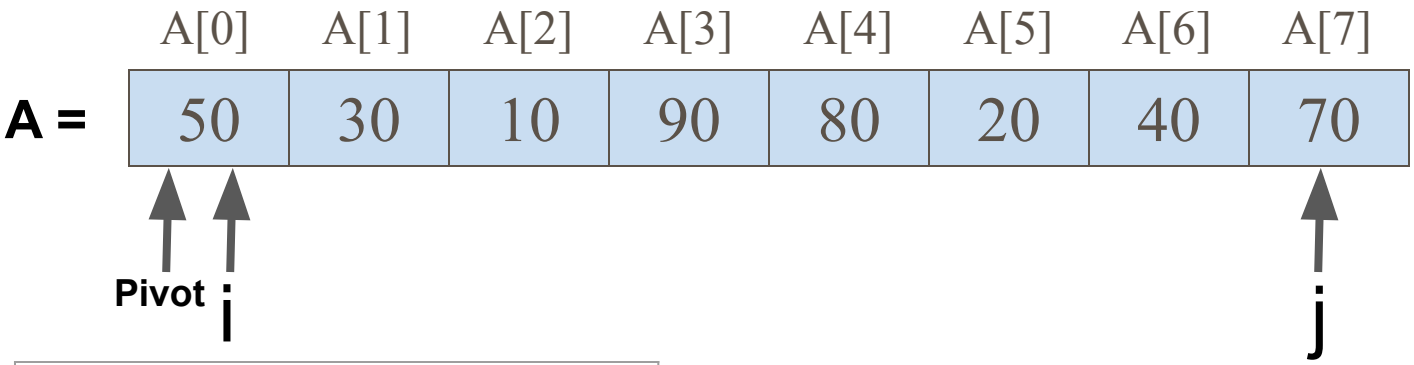
Quick Sort

Quick Sort works in three steps in recursive fashion

- Find pivot element that divides the array into two halves
- Quick sort the left half
- Quick sort the Right half

Average Time Complexity = O(nlogn)
Worst Case Time complexity = $O(n^2)$

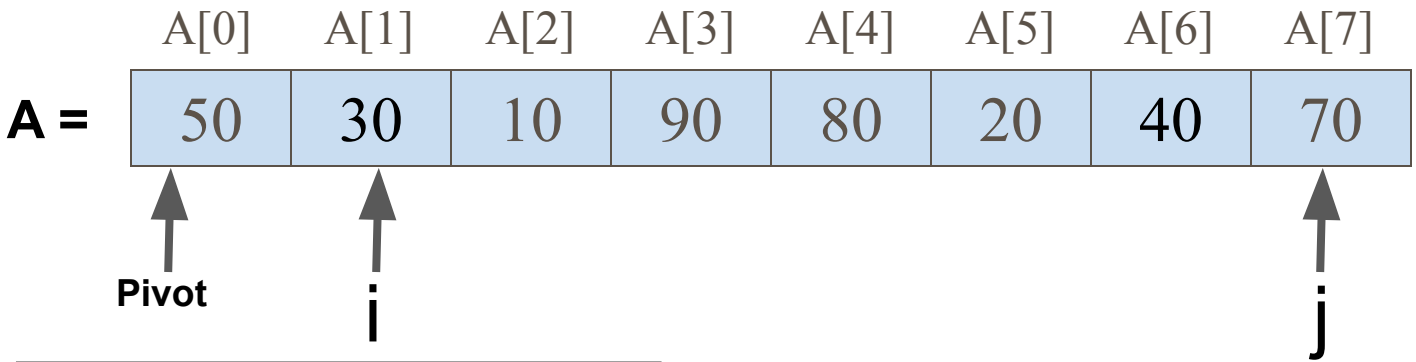
Quick Sort




```
while (a [i] <= pivot)
    i + +;
```

```
Is 50 < = 50 -TRUE
    i++   (i = 1)
```

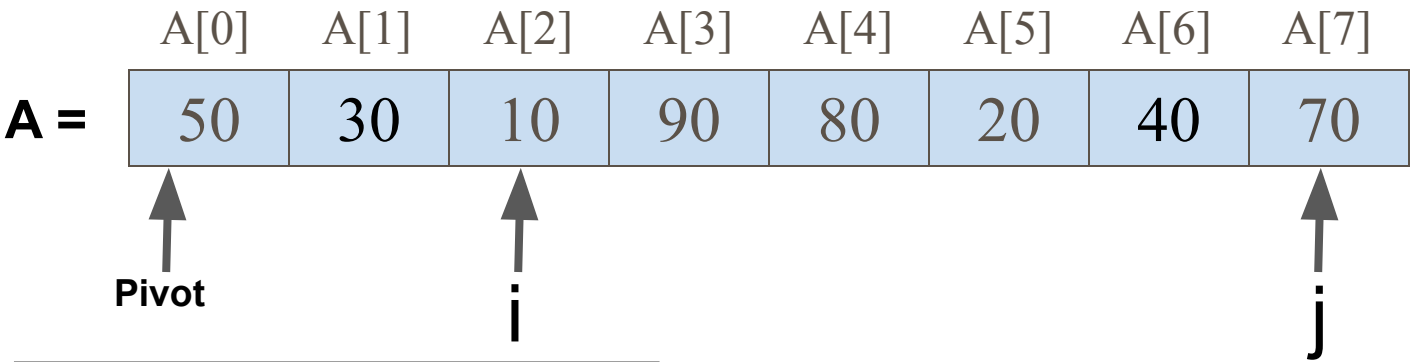
Quick Sort




```
while (a [i] <= pivot)
    i + +;
```

```
Is 30 < = 50 -TRUE
    i++   (i = 2)
```

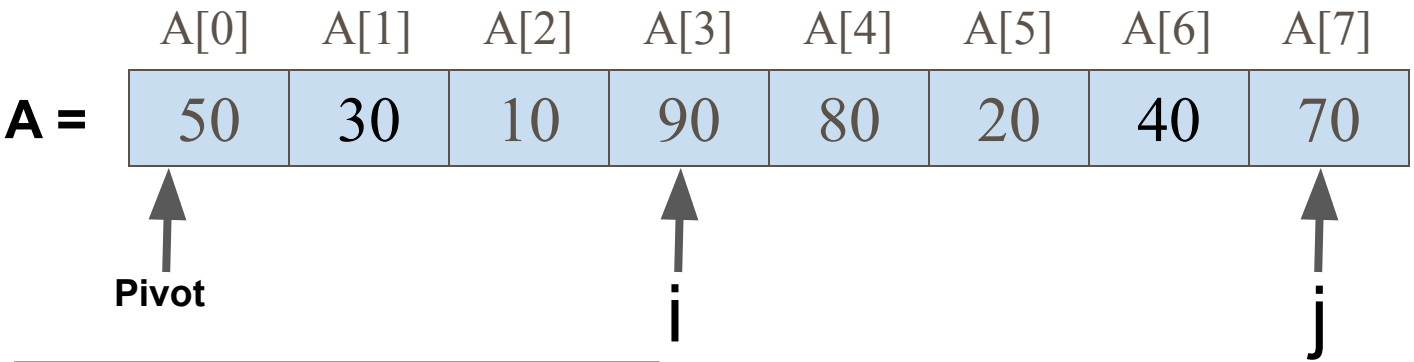
Quick Sort




```
while (a [i] <= pivot)
    i + +;
```

```
Is 10 < = 50 -TRUE
    i++   (i = 3)
```

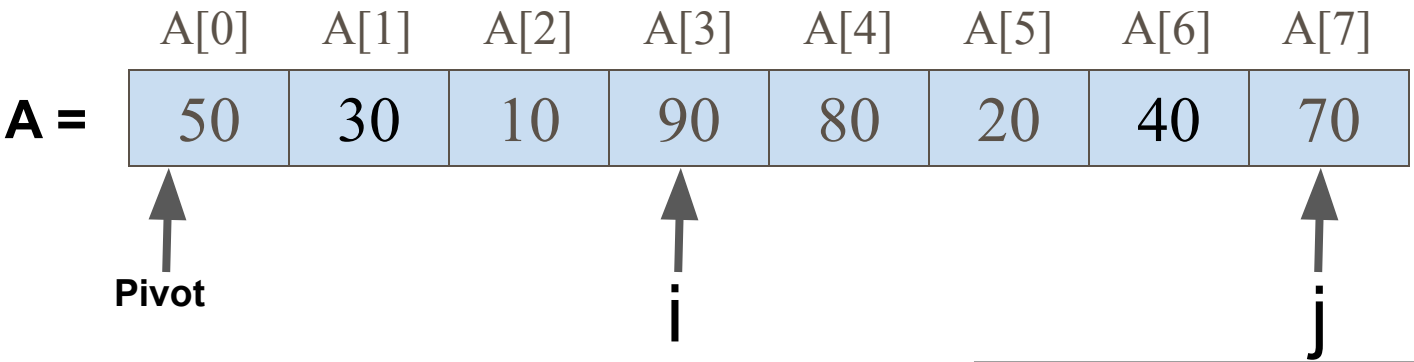
Quick Sort




```
while (a [i] <= pivot)
    i + +;
```

```
Is 90 < = 50 -FALSE
    Stop
```

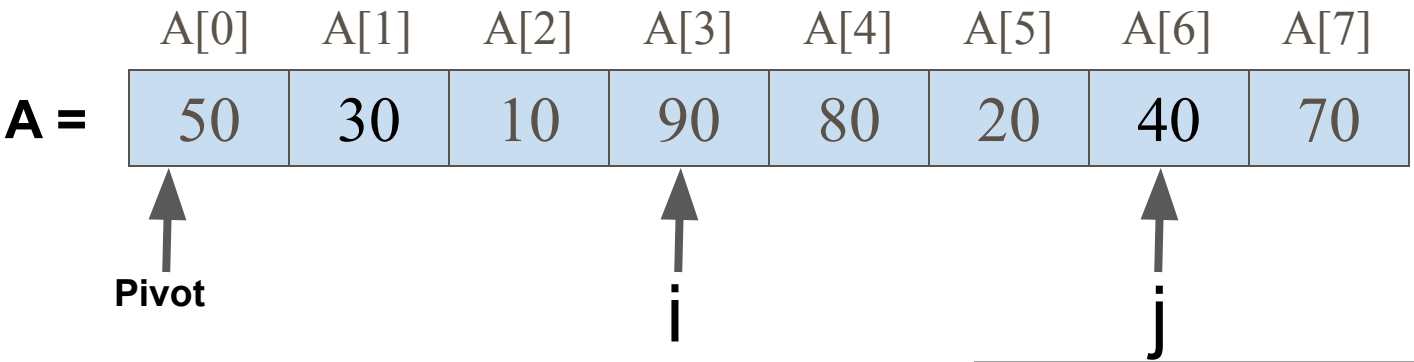
Quick Sort




```
while ( a [j] > pivot)
                j - - ;
```

```
Is 70 > 50 - TRUE
   J-- ( j=6)
```

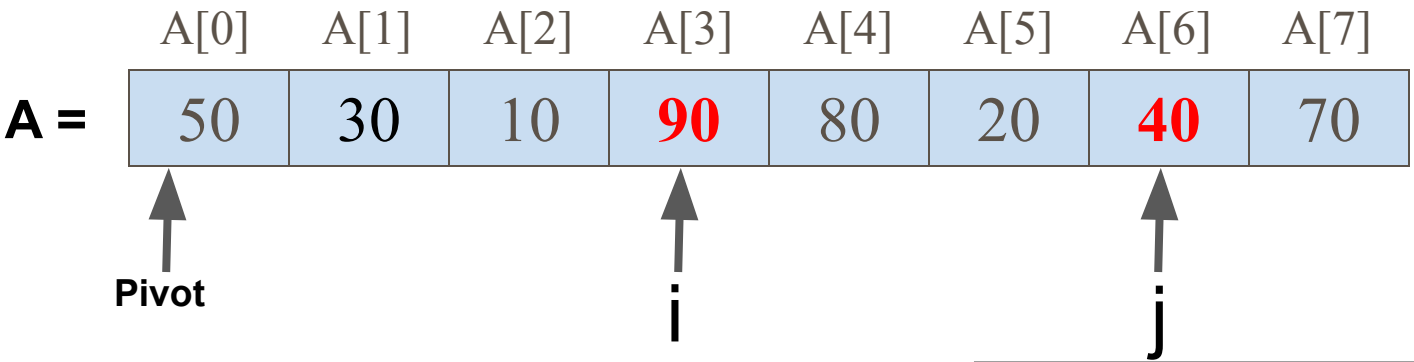
Quick Sort




```
while ( a [j] > pivot)
                    j - - ;
```

```
Is 40 > 50 - FALSE
    Stop
```

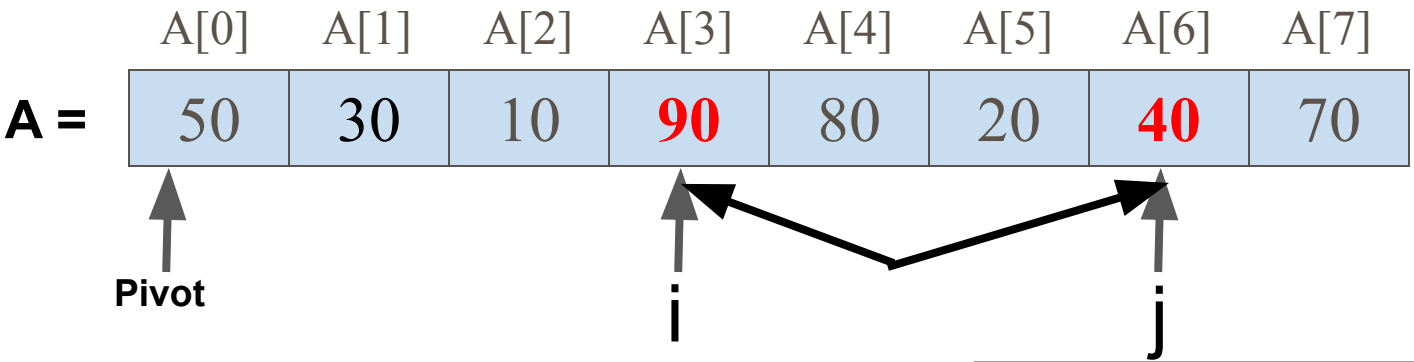
Quick Sort




```
If (i < j) i.e.(3 < 6)
      swap(a[i],a[j])
```

```
Is 3 < 6 - TRUE
   Swap(90,40)
```

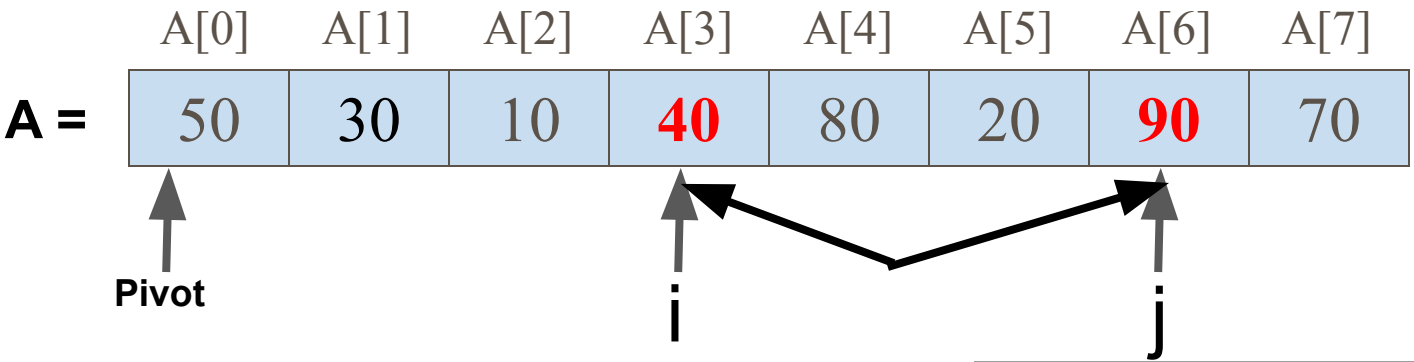
Quick Sort




```
If (i < j) i.e.(3 < 6)
       swap(a[i],a[j])
```

```
Is 3 < 6 - TRUE
   Swap(90,40)
```

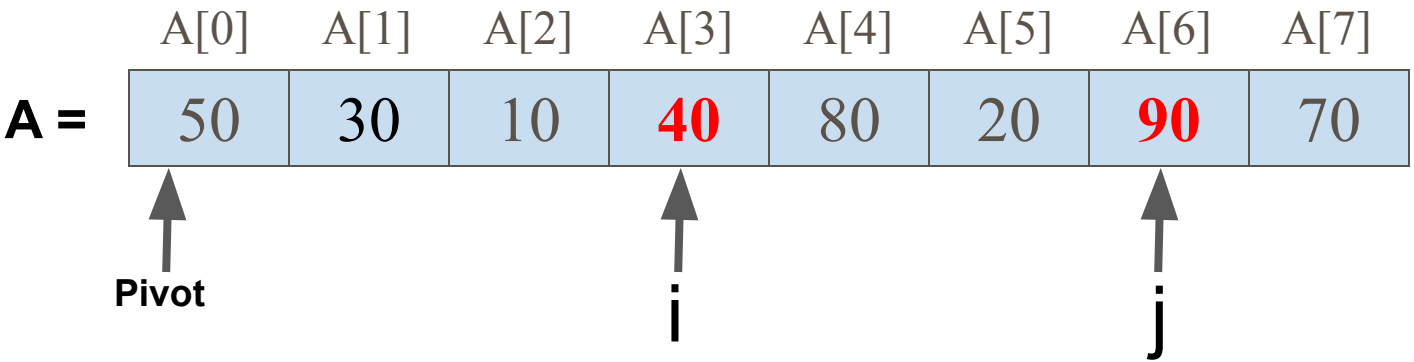
Quick Sort



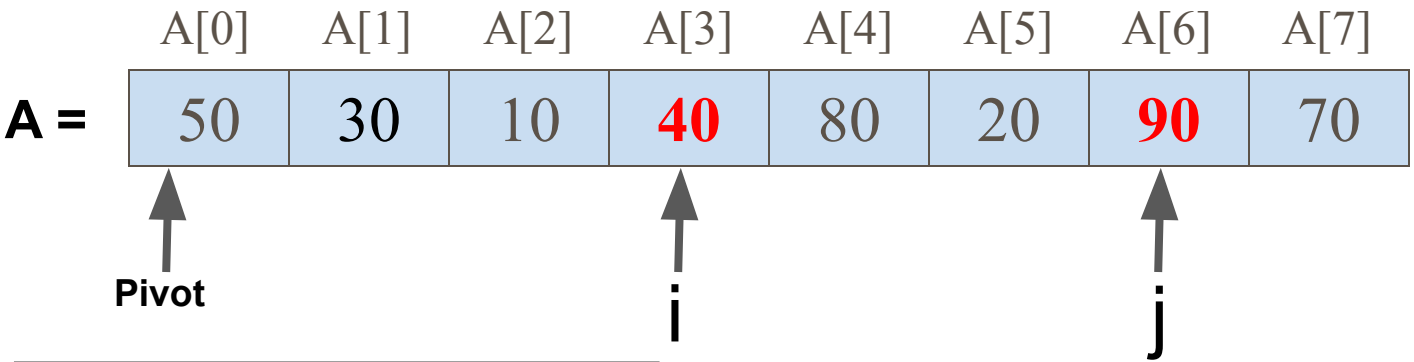

```
If (i < j) i.e.(3 < 6)
       swap(a[i],a[j])
```

```
Is 3 < 6 - TRUE
   Swap(90,40)
```

Quick Sort



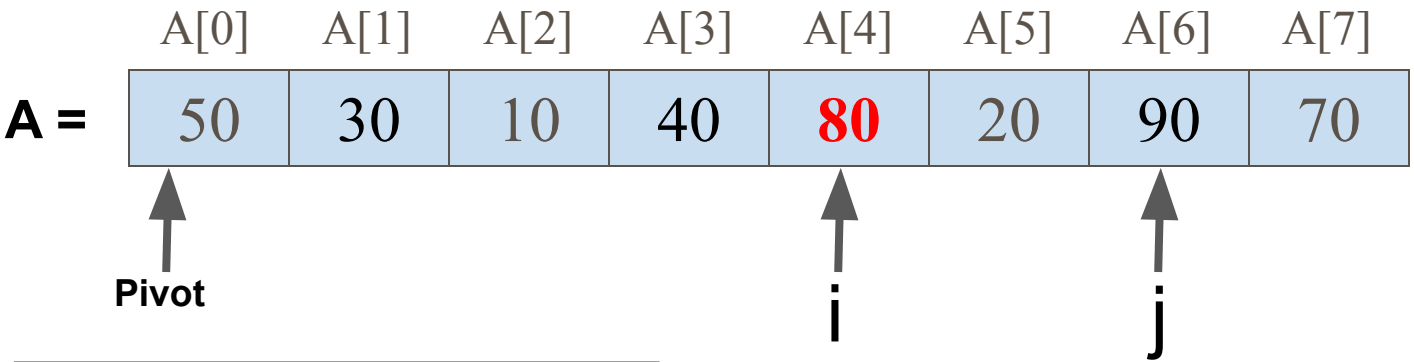
Quick Sort




```
while (a [i] <= pivot)
    i + +;
```

```
Is 40 < = 50 -TRUE
    i++   (i = 4)
```

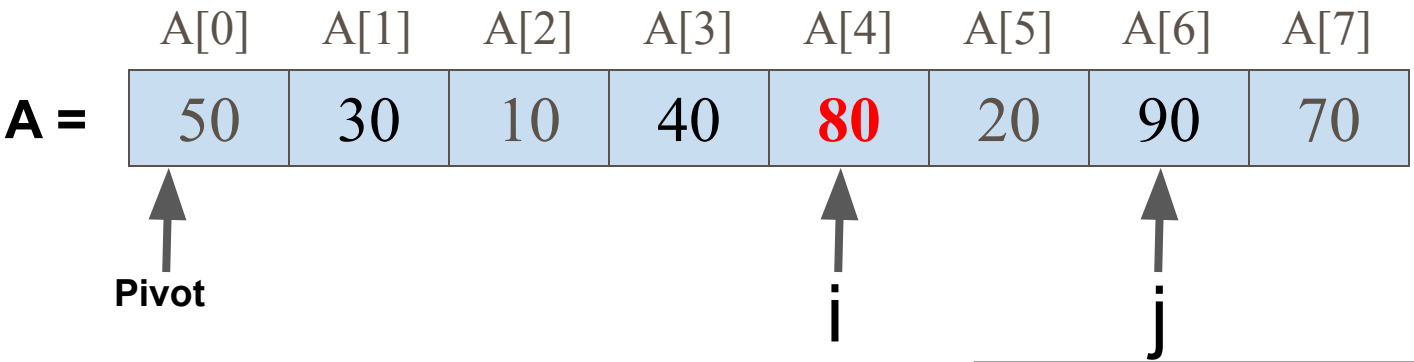
Quick Sort




```
while (a [i] <= pivot)
    i + +;
```

```
Is 80 < = 50 -FALSE
    Stop
```

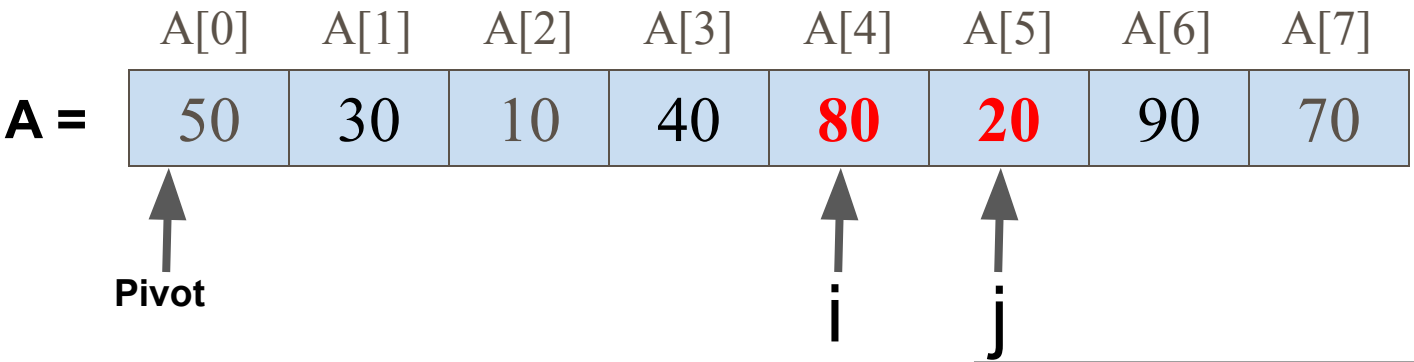
Quick Sort




```
while ( a [j] > pivot)
                    j - - ;
```

```
Is 90 > 50 - TRUE
   J-- ( j=5)
```

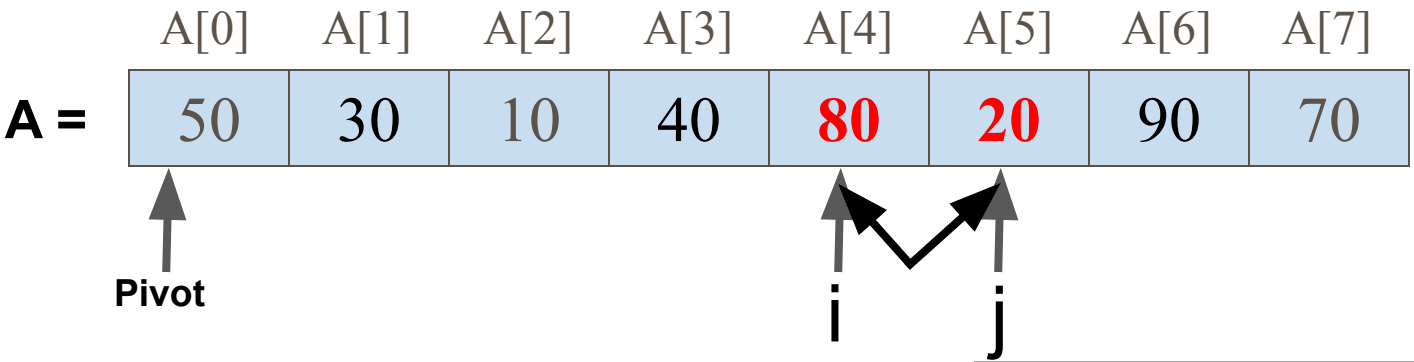
Quick Sort




```
while ( a [j] > pivot)
                j - - ;
```

```
Is 20 > 50 - FALSE
   Stop
```

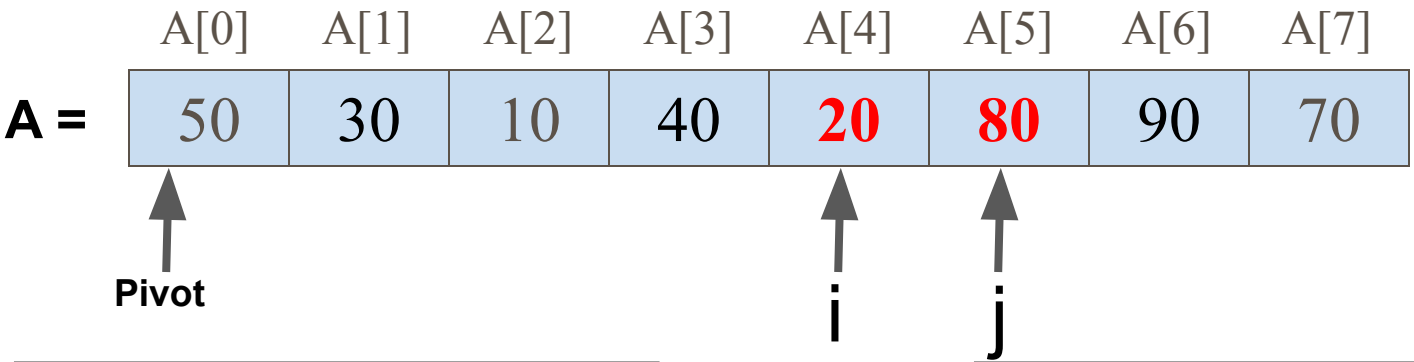
Quick Sort




```
If (i < j) i.e.(4 < 5)
      swap(a[i],a[j])
```

```
Is 3 < 6 - TRUE
   Swap(80,20)
```

Quick Sort



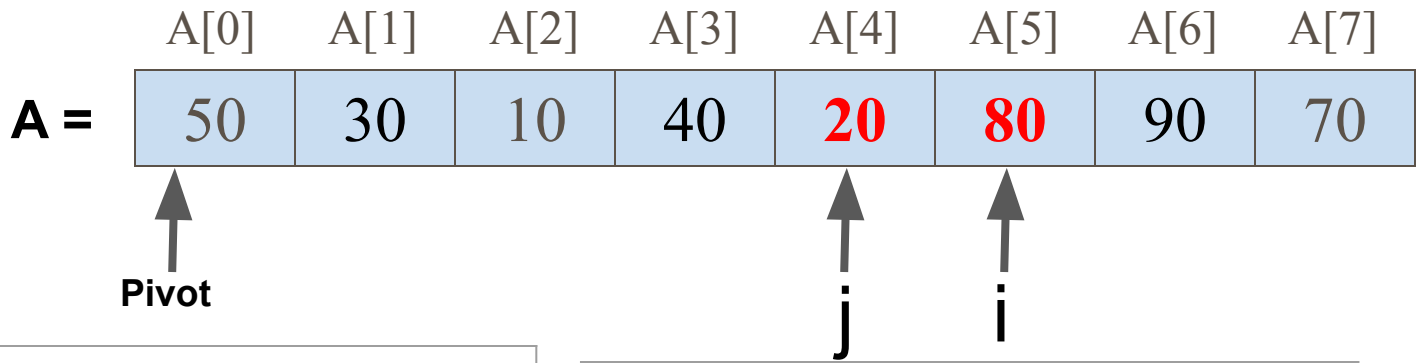

```
while (a [i] <= pivot)
    i + +;
```

```
while ( a [j] > pivot)
                j - - ;
```

```
Is 20 < = 50 -TRUE
    i++   (i = 5)
```

```
Is 80 > 50 - TRUE
   J--  ( j=4)
```

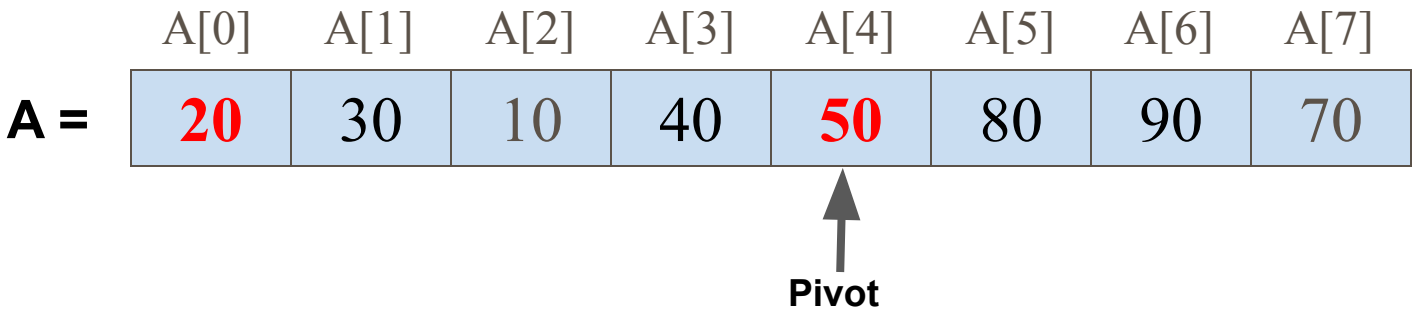
Quick Sort



- Here i value becomes greater than j means i and j value gets crossed
- In this case, swap pivot and j value i.e. 50 and 20

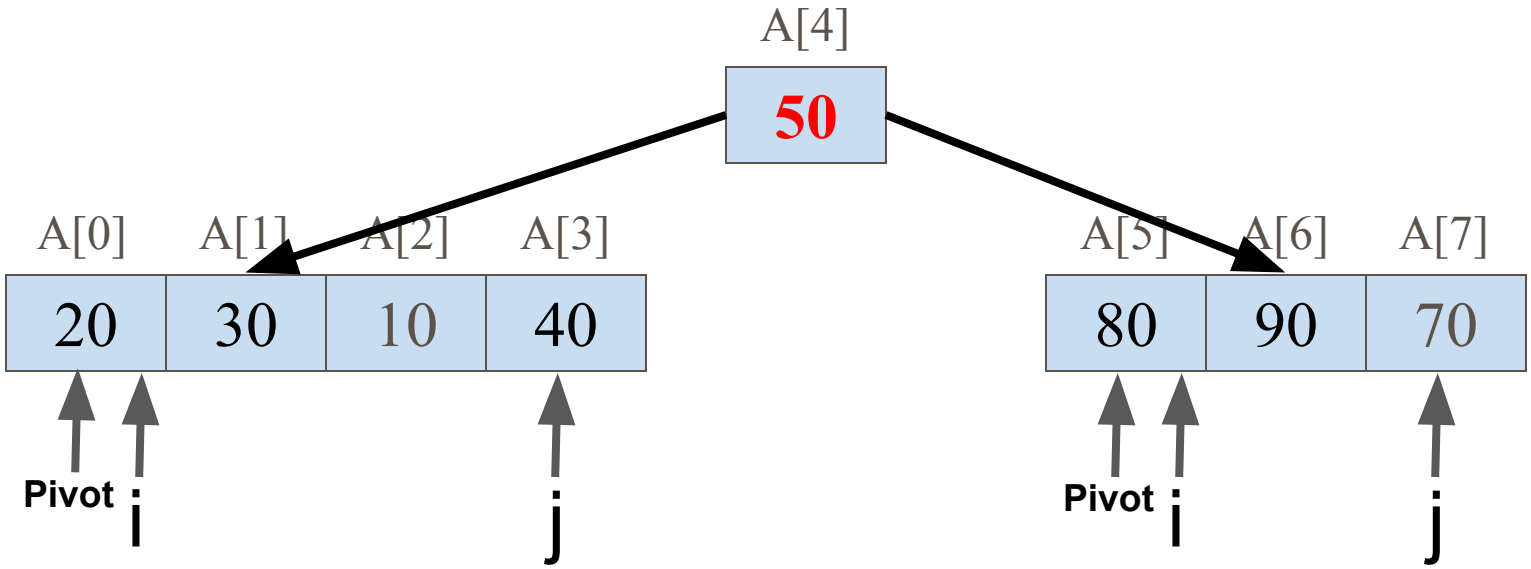
```
swap ( a [pivot], a [j] )
```

Quick Sort



- We can observe here, all the values left of pivot are smaller and all the values right of pivot are greater.
- Therefore, array is partitioned into two sub arrays
- COntinue same process for next partitions too

Quick Sort



A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]
10	20	30	40	50	70	80	90

Sorted Array

Quick Sort

```
void Quickesort (int a [max], int low, int high)
{
    int m, i;
    if (low < high)
    {
        m = partition ( a, low, high ); /* Setting Pivot element */
        Quicksort ( a, low, m-1);       /* Splitting list */
        Quicksort ( a, m + 1, high);    /* Splitting list */
    }
}
```

Quick Sort

```
int partition (int a [max], int low,  int high )
{
    int pivot = a [low], i = low, j = high;
    while (( i <= j)
    {       while (a [i] <= pivot)
                i + +;
            while ( a [j] > pivot)
                j - - ;
            if ( i < j)
                swap( a [i], a [j] );
    }
    swap ( a [low], a [j] );
    return ( j );
}
```

Swapping

```
temp = a[low];
a[low] = a[j];
a[j] = temp;
```