Design and Analysis of Algorithms

Lecture - 12

Success is always inevitable with Hard Work and Perseverance

N. Ravitha Rajalakshmi

Learning Objective

- Discuss D&C strategy for classical problems
  - Sorting

Sorting

Input: An array A with n elements.

Output: Permutation of Array A where elements are arranged in non-decreasing order.

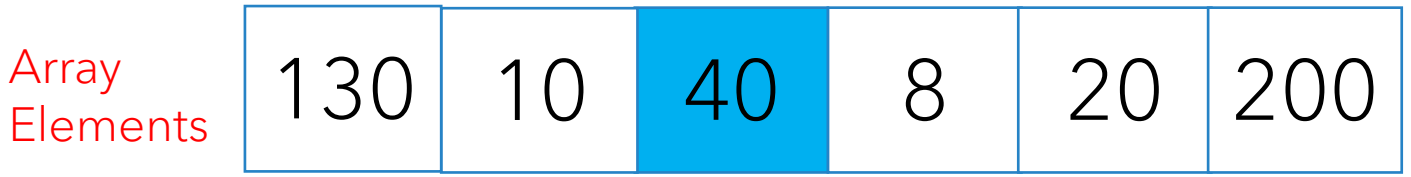
Initial Array

130	10	40	8	20	200

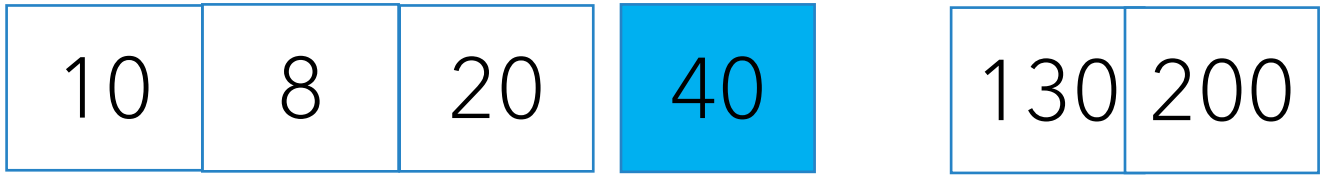
Sorted Array

8	10	20	40	130	200

Quick Sort



- Reorder elements and Split the array into two halves



- Repeat task until the partitions are left with only one element .

Quick Sort - Partition

Pick a pivot element (First element to be the pivot)

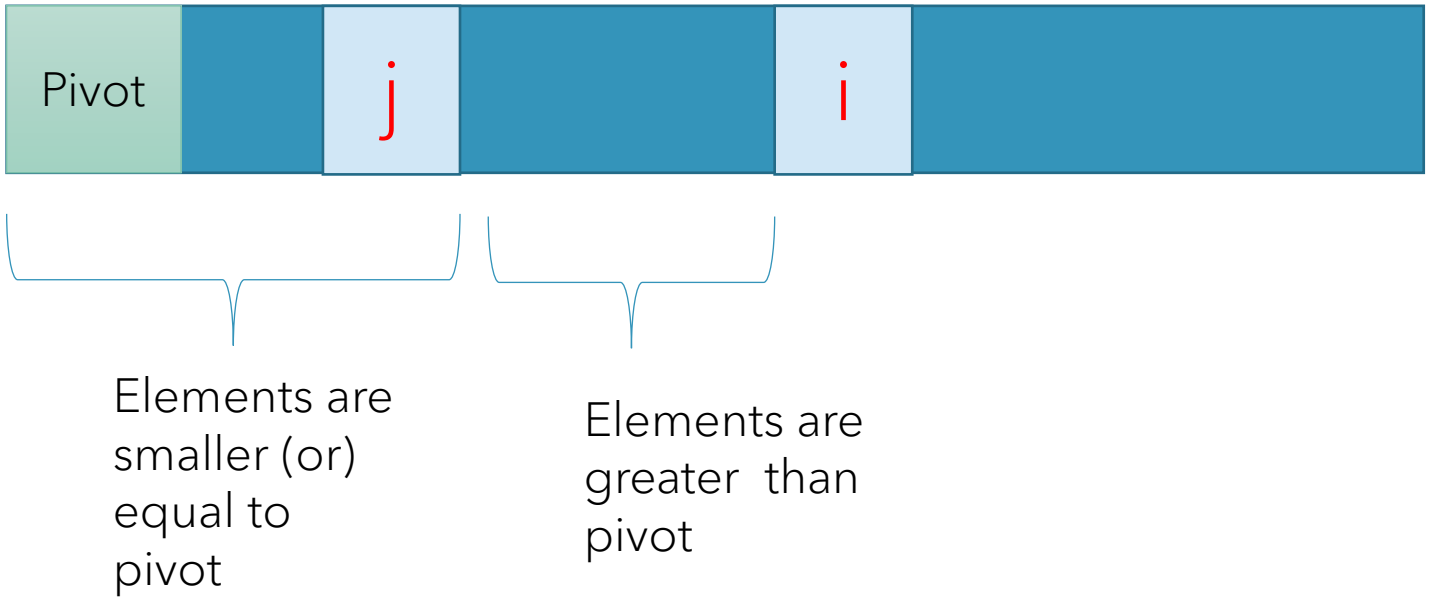
Partition procedure places the pivot element in its correct position and reorder the elements as follows:

Quick Sort - Partition

Traversal from low +1 to high

Whenever a smaller element is found , it is moved to left half of array

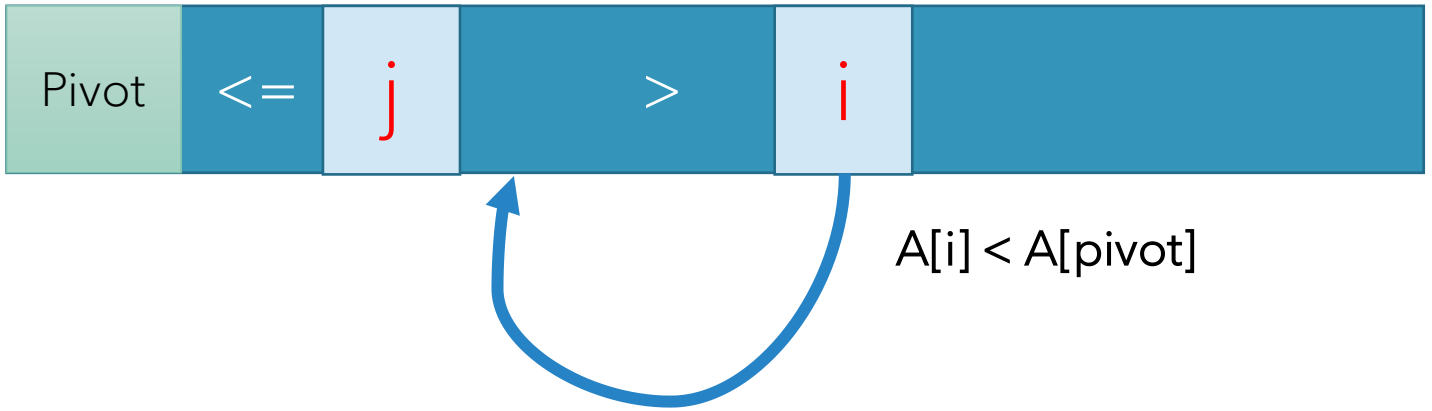
j points to the index (all elements for low+1 to j is smaller than pivot)



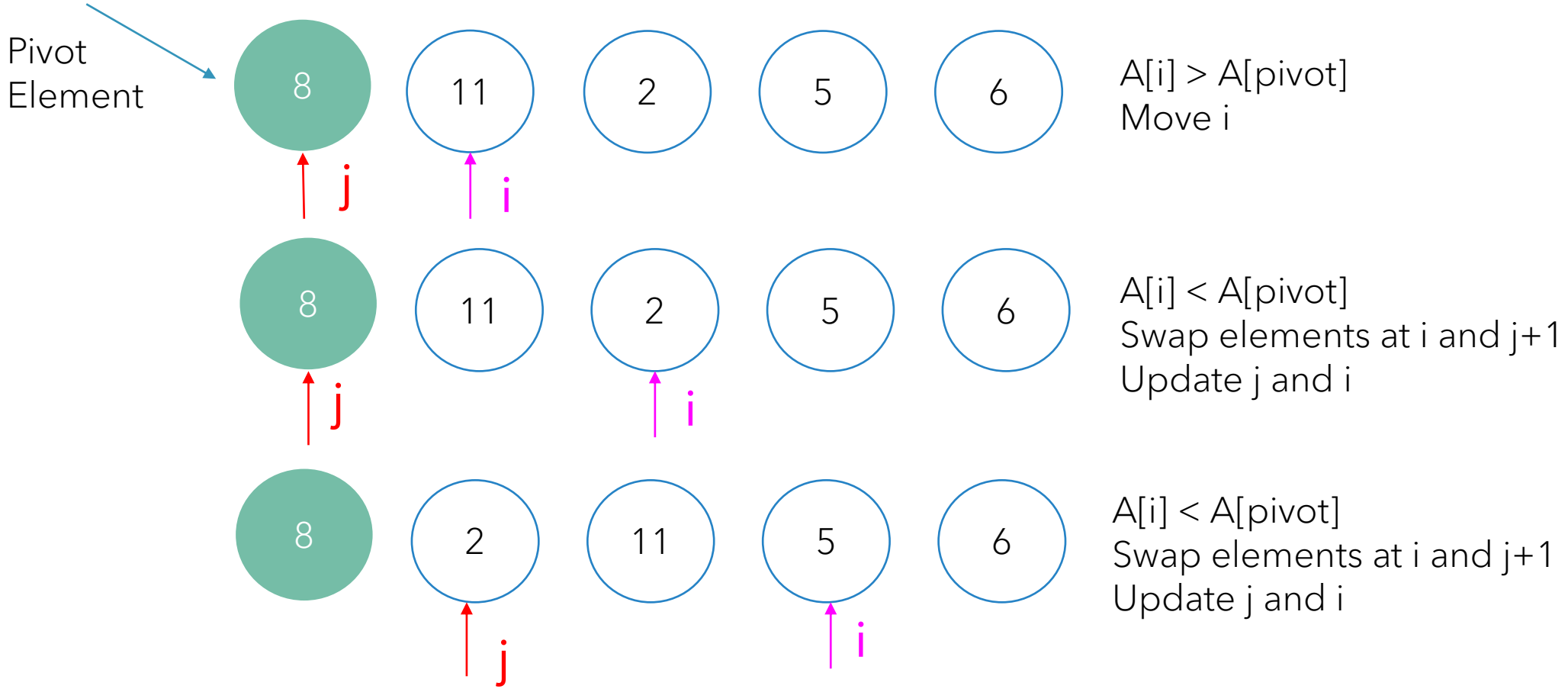
Pause & Think

During traversal, if a smaller element is found(i), then it should be swapped with some larger element between j+1 to i-1

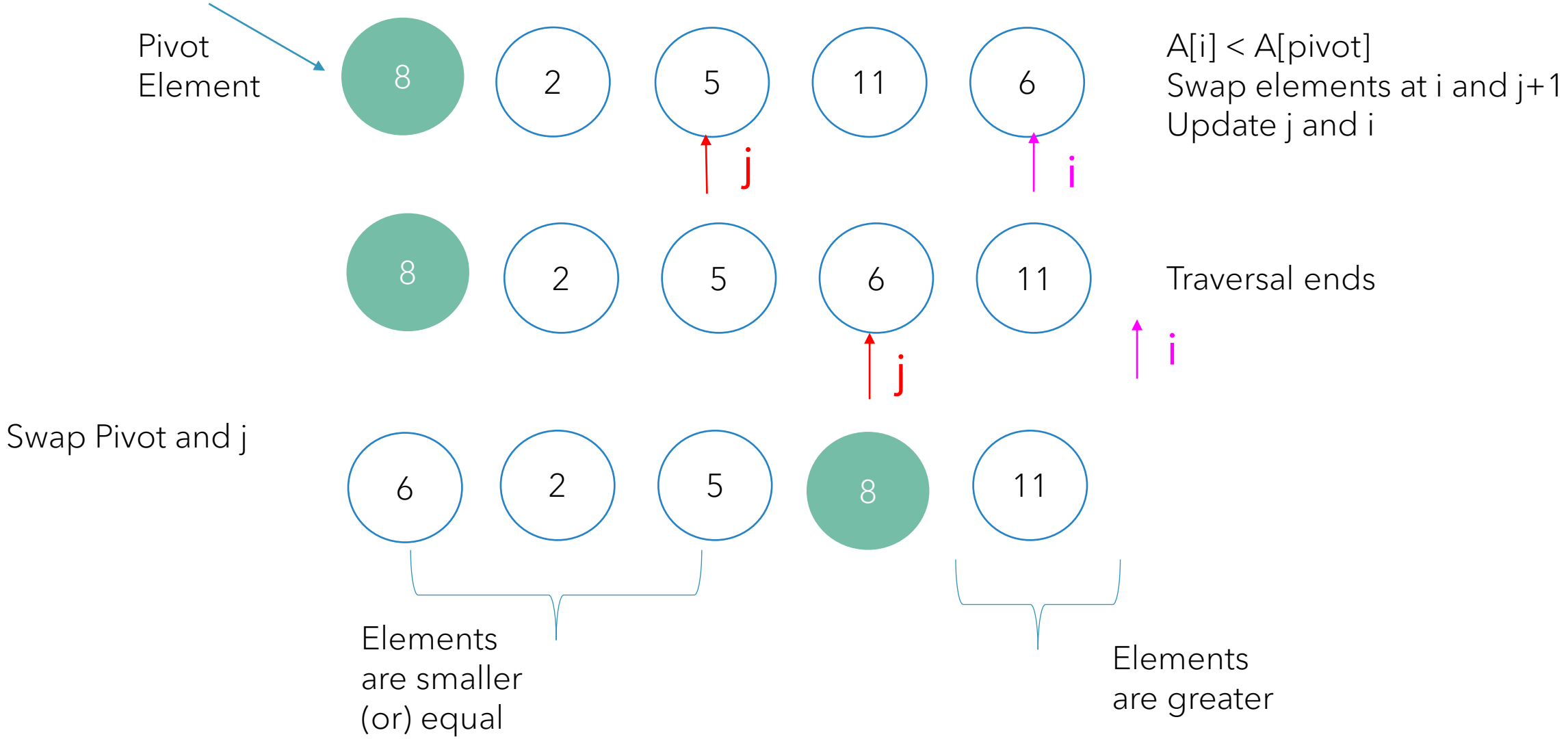
Swap (j+1 , i) [So that condition is satisfied]



Example - Partition



Example - Partition



Quick Sort - Partition

Function Partition(A, low, high)

```
Pivot <- low
j <- low
i <- low+1
while(i<= high){
        if(A[i]<=A[pivot]){
                j = j+1
                swap (A[i], A[j])
        }
}
swap(A[j], A[pivot])
return j
```

Function QuickSort(A, low, high)

```
# If the array contains more than one element
if (low <high) {

        # Divide the array into two subarrays
        m = Partition (A, low, high)
        QuickSort(A, low, m-1)
        QuickSort(A, m+1, high)

}
```

Summary

- How D&C is applied for Quick Sort

Thank You

Happy Learning

Success is always inevitable with Hard Work and Perseverance