Design and Analysis of Algorithms

Lecture - 13

Success is always inevitable with Hard Work and Perseverance

N. Ravitha Rajalakshmi

Learning Objective

- Analyze Quick Sort
- Improvised Versions of Quick Sort

Time Complexity

- Input size : n (number of elements)
- Basic Operation : Comparison
- Size of partition is unknown [Specifics of the input]
- If we consider that partition happens at the middle of the subarray (balanced partition) [Best Case]

$$T(n) = 2T(n/2) + n$$

$$T(n) = O(n \log n)$$

Time Complexity

- What if the input array is sorted ?
  - Choose the first element as pivot
  - Results in two partitions with 1 and n-1 elements respectively

- Recurrence relation

$$T(n) = T(n-1) + n, \qquad n > 1$$
$$T(1) = 1$$

$$T(n) = O(n^2)$$

Randomized Quicksort

- Pick a random element as pivot element [This will ensure that partition happens in the middle most of the times]

```
Function RandomizedQuickSort(A, low, high)
# If the array contains more than one element
if  (low <high) {
        Pick a random element r
        Swap A[low] with random element
        # Divide the array into two subarrays
        m = Partition (A, low, high)
        QuickSort(A, low, m-1)
        QuickSort(A, m+1, high)
}
```

Pause & Think

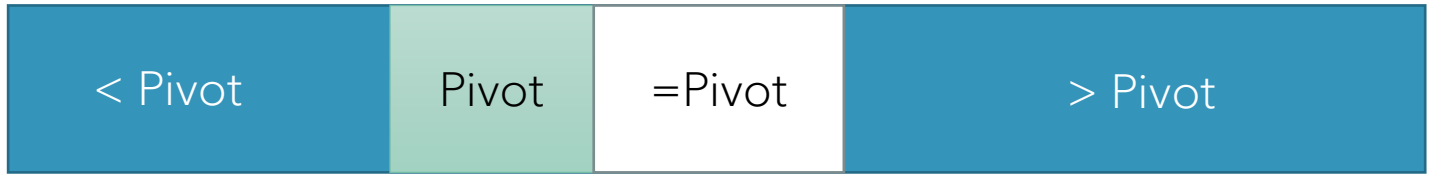
- What if there are many equal elements in the array?
  - Quick sort will bring all the equal elements to left partition

- Is it a good solution
  - No
  - Knowing that elements are equal to pivot it can be removed from further competition

Quick Sort – Partition3

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:



Pause & Think

- Maintain two pointers one to accumulate elements smaller than pivot and other to accumulate elements larger than pivot

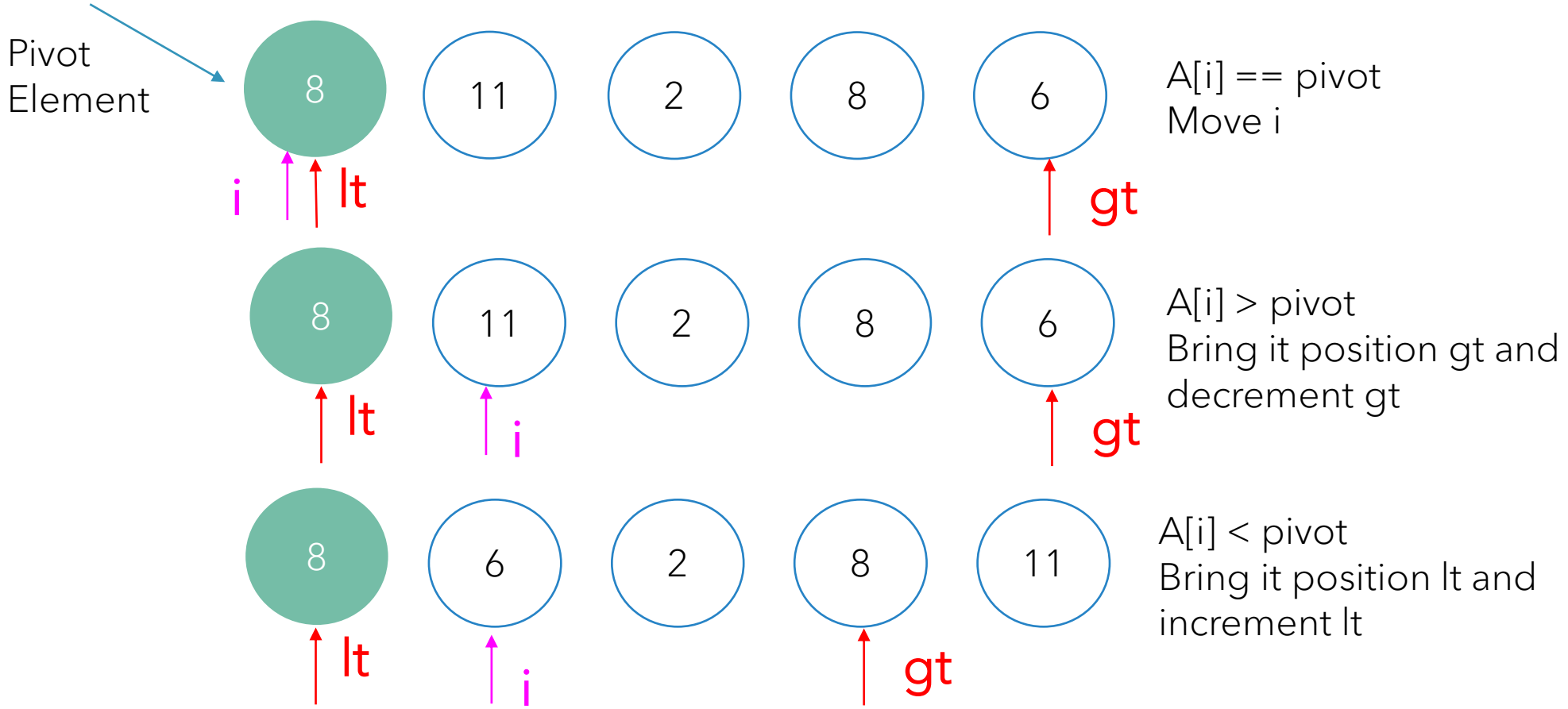
After the partition procedure,
All elements less than position lt to be lesser than pivot
All elements greater than position gt to be greater than pivot
All elements between lt and gt to hold same value as that of pivot

Quick Sort – Partition3

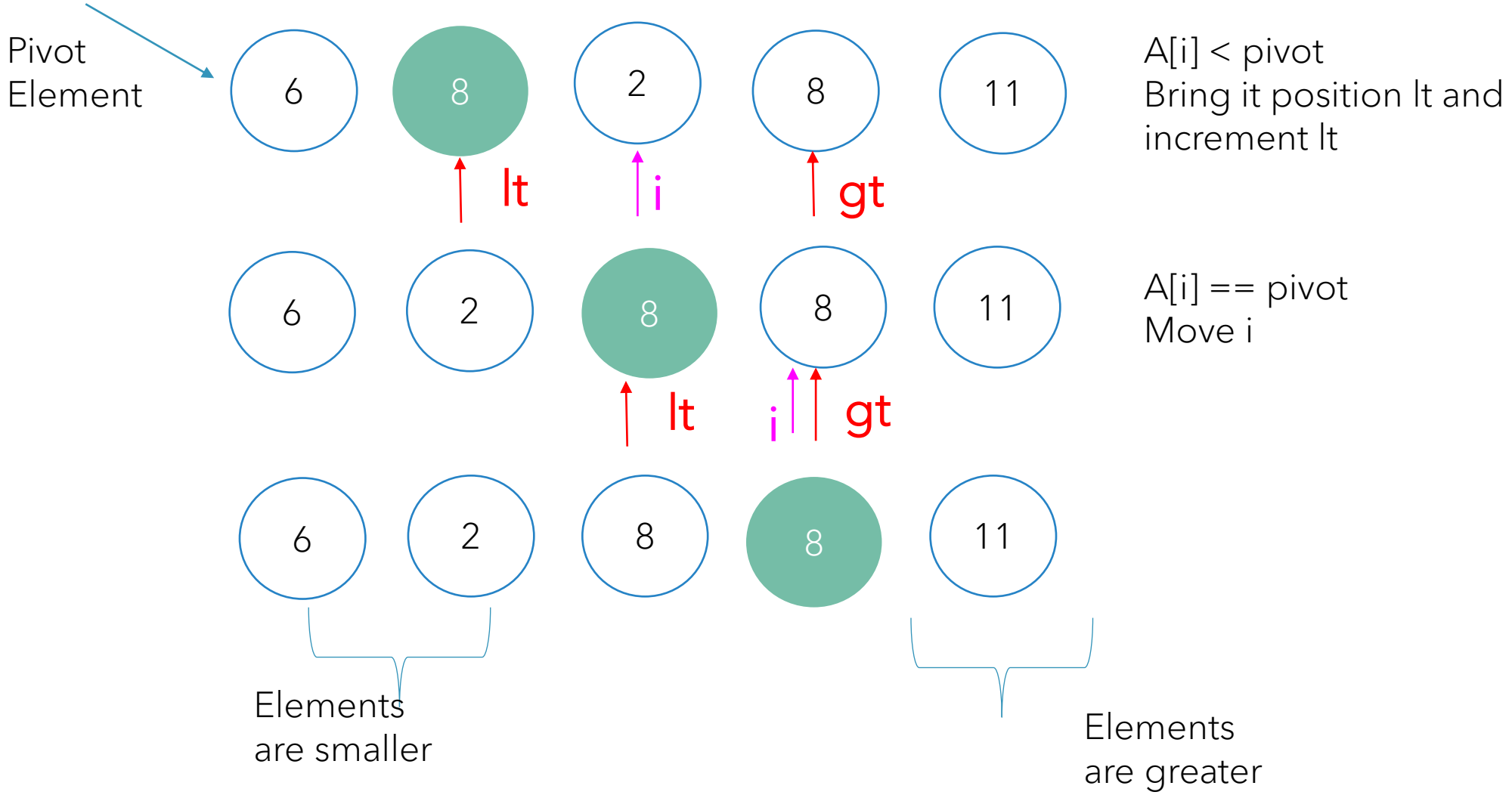
Function Partition3(A, low, high)

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

Example – Partition3



Example – Partition3



Summary

- Discussed Variants of Quick Sort

Thank You

Happy Learning

Success is always inevitable with Hard Work and Perseverance