Design and Analysis of Algorithm (KCS503)

Implementation and Analysis of Insertion Sort through Head recursion Approach

Lecture -9

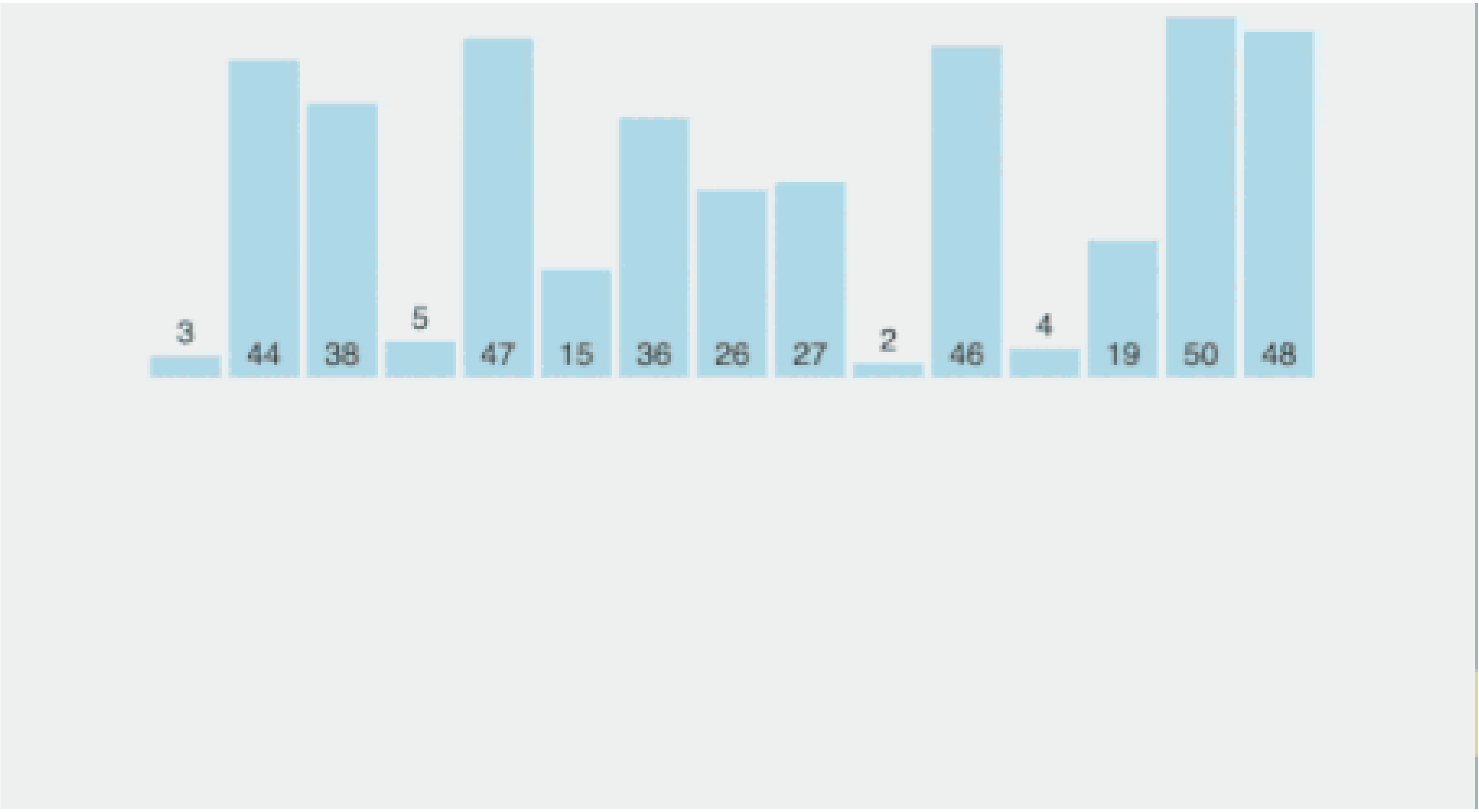
Objective

- Able to learn and apply the head recursive approach

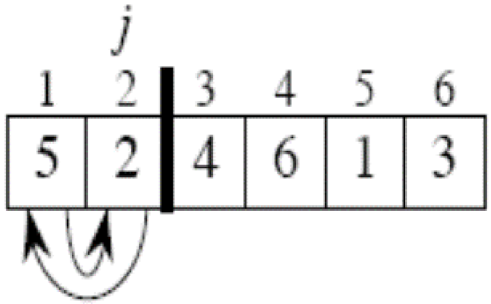
  $$(\,\boldsymbol{T(n)} = \boldsymbol{T(n-1)} + \boldsymbol{O(n)}, \boldsymbol{T(1)} = \boldsymbol{1})$$

  of Insertion sort with analysis and analyse its complexity.

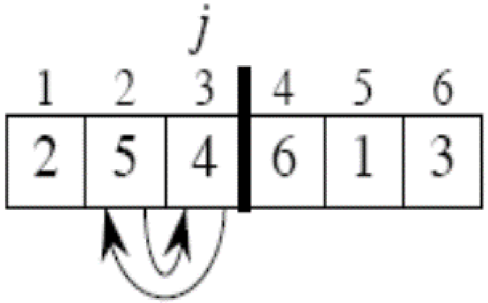
A Sorting Problem
(Insertion Sort Algorithm with Head Recursive Approach)



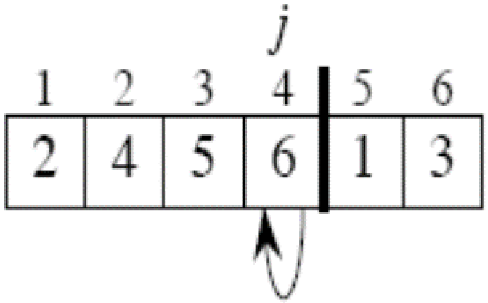
A Sorting Problem
(Insertion Sort Algorithm with Head Recursive Approach)



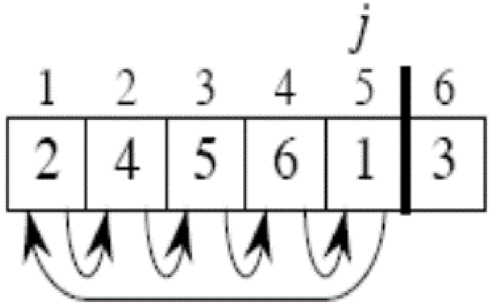
1st Iteration



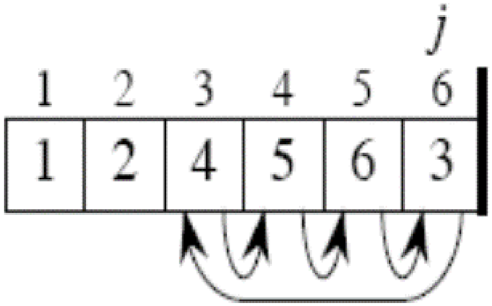
2nd Iteration



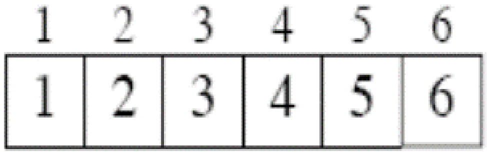
3rd Iteration



4th Iteration



5th Iteration



6th Iteration

A Sorting Problem
(Insertion Sort Algorithm with Head Recursive Approach)

A = [5, 2, 4, 6, 1, 3], size = 6

```
insertion-sort(A, 0)
    insert 5
    return A = [5]

insertion-sort(A, 1)
    insert 2
    return A = [2, 5]

insertion-sort(A, 2)
    insert 4
    return A = [2, 4, 5]

insertion-sort(A, 3)
    insert 6
    return A = [2, 4, 5, 6]

insertion-sort(A, 4)
    insert 1
    return A = [1, 2, 4, 5, 6]

insertion-sort(A, 5)
    insert 3
    return A = [1, 2, 3, 4, 5, 6]
```

A Sorting Problem
(Insertion Sort Algorithm with Head Recursive Approach)

It was observed that the concept is:

Insert an element in a previously sorted array named as ‘A’.

Solution Steps:

- Base Case: If array size is 1 or smaller, return.
- Recursively sort first n-1 elements.
- Insert the last element at its correct position in the sorted array.

A Sorting Problem

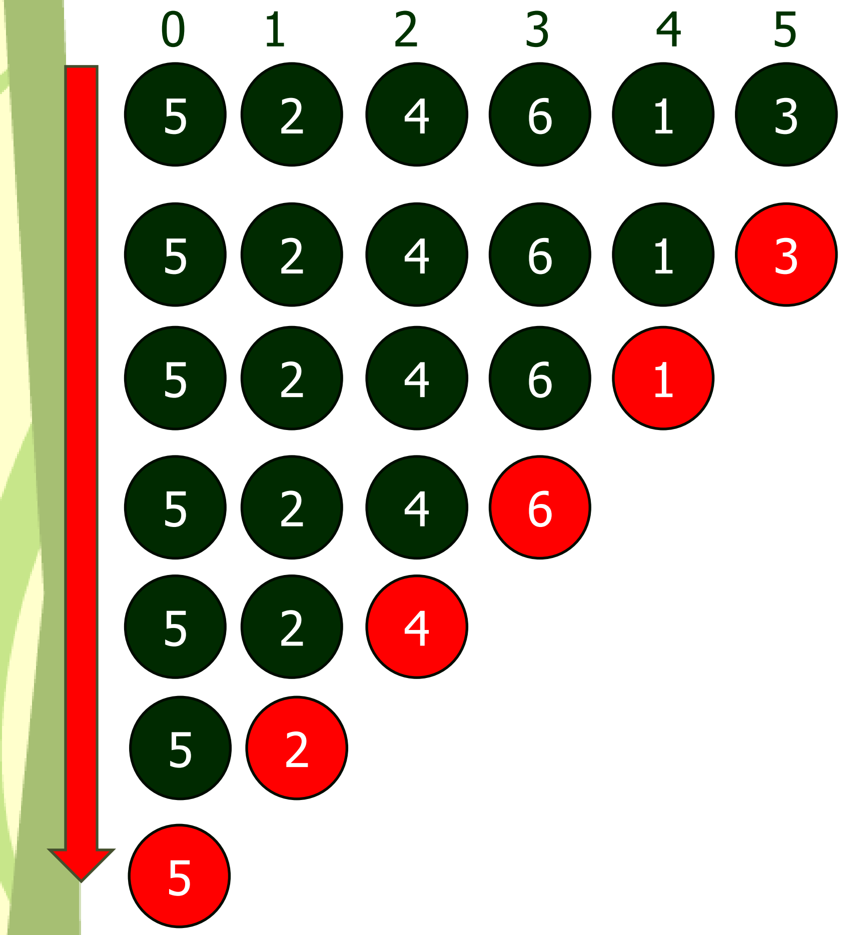
(Insertion Sort Algorithm with Head Recursive Approach)

```
void insertion_sort(A, j) {
//Initially j=length(A)
    // Base case
    if (j <= 1)
        return
    // Sort first i-1 elements
    insertion_sort( A, j-1 )
    insert A[j-1] into A[0…j-2]

}
```

A Sorting Problem

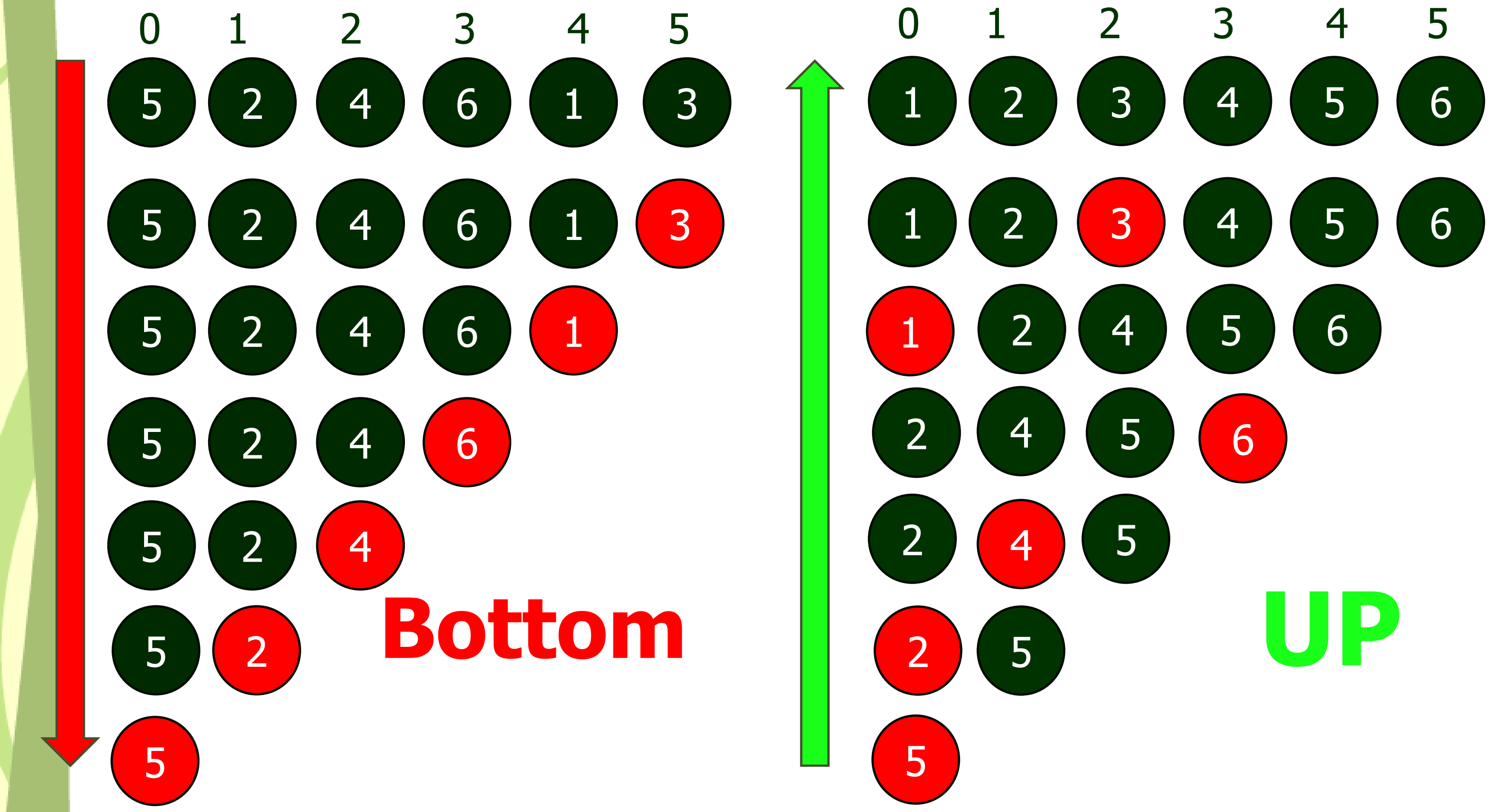
(Insertion Sort Algorithm with Head Recursive Approach)



0	1	2	3	4	5
5	2	4	6	1	3
5	2	4	6	1	3
5	2	4	6	1	
5	2	4	6		
5	2	4			
5	2				
5					

A Sorting Problem

(Insertion Sort Algorithm with Head Recursive Approach)



0	1	2	3	4	5
5	2	4	6	1	3
5	2	4	6	1	3
5	2	4	6	1	
5	2	4	6		
5	2	4			
5	2				
5					

Bottom

0	1	2	3	4	5
1	2	3	4	5	6
1	2	3	4	5	6
1	2	4	5	6	
2	4	5	6		
2	4	5			
2	5				
5					

UP

A Sorting Problem

(Insertion Sort Algorithm with Head Recursive Approach)

```
void insertion_sort(A, j) {
//Initially j=length(A)
    // Base case
    if (j <= 1)
        return
    // Sort first n-1 elements
    insertion_sort( A, j-1 )
    val = A[j-1]
    i = j-2
    while (i >= 0 && A[i] > val) {
        A[i+1] = A[i]
        i = i - 1
    }
    A[i+1] = val
}
```

A Sorting Problem
(Insertion Sort Algorithm with Head Recursive Approach)

Complexity:

$$T(n) = T(n-1) + (n)$$

$$T(n) = T(n-2) + (n-1) + (n)$$

$$T(n) = T(n-3) + (n-2) + (n-1) + (n)$$

....

....

....

$$T(n) = 1 + 2 + 3 + \cdots + \cdots + (n-3) + (n-2) + (n-1) + (n)$$

Hence $T(n) = \frac{n(n+1)}{2}$ → $T(n) = O(n^2)$

A Sorting Problem
(Insertion Sort Algorithm with Head Recursive Approach)

```
void insertion_sort(A, j) {
//Initially j=length(A)
    // Base case
    if (j <= 1)
        return
    // Sort first j-1 elements
    insertion_sort( A, j-1 )
    val = A[j-1]
    i = j-2
    while (i >= 0 && A[i] > val) {
        A[i+1] = A[i]
        i = i - 1
    }
    A[i+1] = val
}
```

Hence, the recursive (Head) method have develop the recurrence equation

$$\boldsymbol{T(n) = T(n-1) + O(n) \in O(n^2)}$$

🙏 Thank You 🙏