TECHNO INDIA UNIVERSITY
WESTBENGAL

Data Structures & Algorithms
(PCC-CS 301)

Dr. Debashis Das
Associate Professor
Department of CSE
Techno India University, Kolkata

Topics Covered

1. Data Sorting
    1.1. Introduction
    1.2. Algorithms and Properties
2. Comparison based Sort
    2.1. Bubble sort
    2.2. Selection sort

Sorting

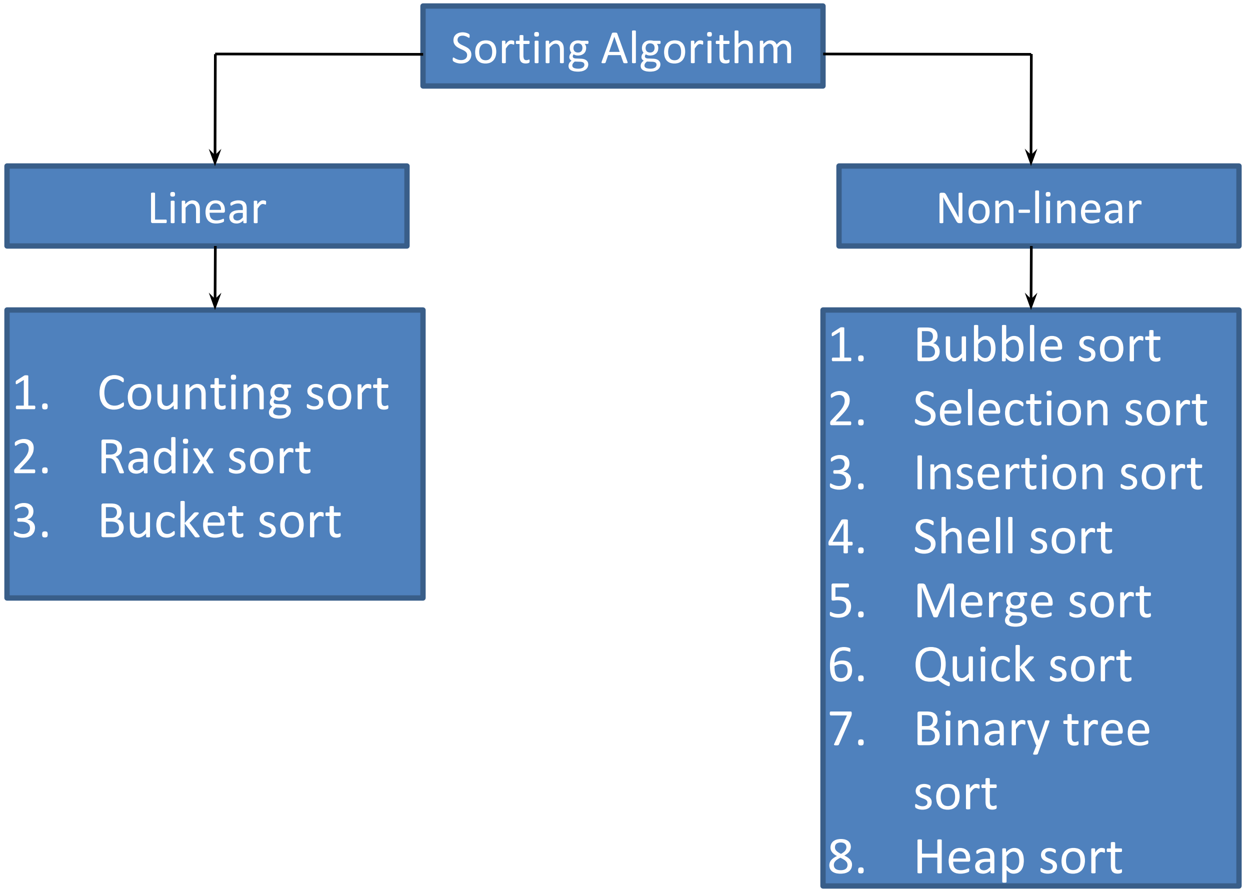
- Introduction

  - Arranging of a data set in a specific sequence
    - Arrangement in ascending order
    - Arrangement in descending order

Input data set	10	5	25	6	3	9	12	30	15	20
Sorted (ascending)	3	5	6	9	10	12	15	20	25	30
Sorted (descending)	30	25	20	15	12	10	9	6	5	3

Sorting Algorithm

- Classification: based on time complexity



Sorting Algorithm

Linear

1. Counting sort
2. Radix sort
3. Bucket sort

Non-linear

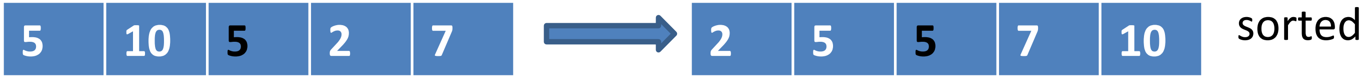
1. Bubble sort
2. Selection sort
3. Insertion sort
4. Shell sort
5. Merge sort
6. Quick sort
7. Binary tree sort
8. Heap sort

Sorting Algorithm

- Classification: based on few properties
  - Comparison based sort
    - data are compared with each other
    - Ex. - bubble, selection, insertion, shell, quick sort
  - Divide-and-Conquer based sort
    - Data set is divided-arranged-combined to perform sorting
    - Ex. – merge, quick sort
  - In-place sort
    - No extra memory is required to sort the data set (internal)
    - Ex. – bubble, selection, insertion, quick, shell

Sorting Algorithm

- Classification: based on few properties
  - External sort
    - Extra memory is required to sort a data set
    - Ex. – merge sort, counting, radix sort
  - Stable sort
    - Data ordering of same value will remain same after sorting
    - Ex. – Insertion, bubble, merge, binary tree sort

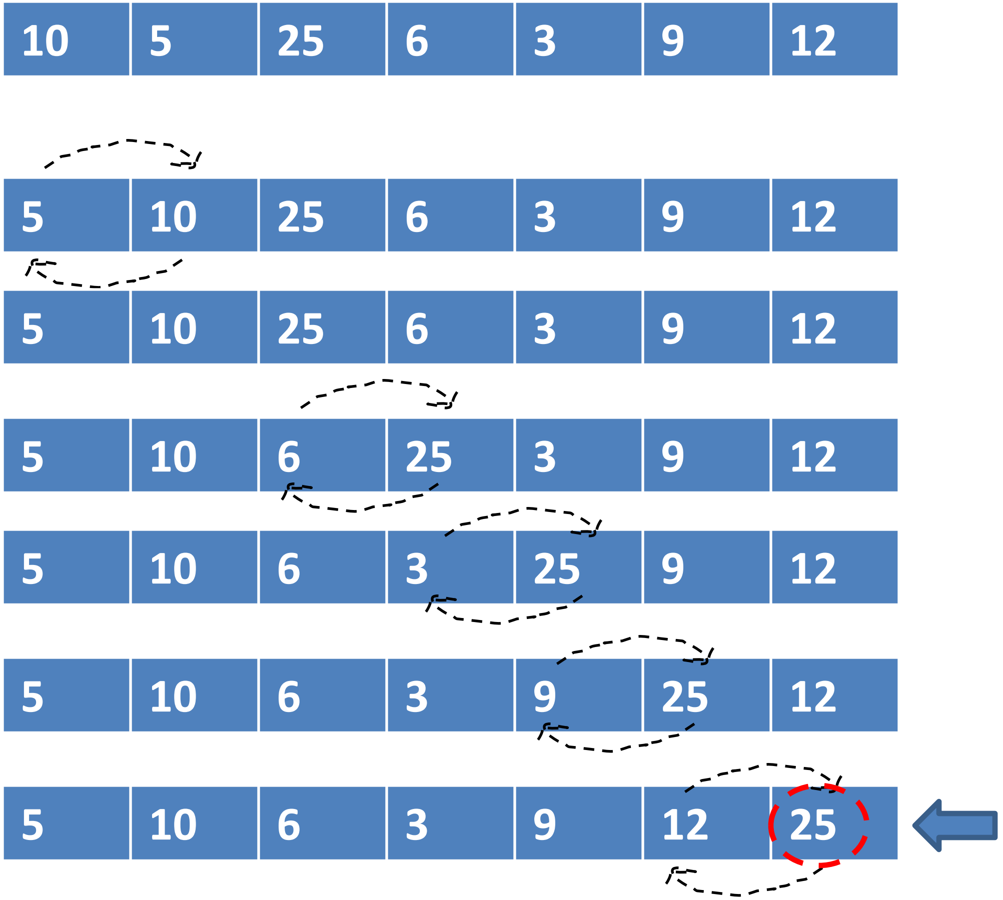


5 | 10 | **5** | 2 | 7 → 2 | 5 | **5** | 7 | 10 sorted

  - Un-stable sort
    - Ordering of same value may not be same in sorted form
    - Ex. – selection, heap, quick sort

Comparison based Sort

- Bubble sort: mechanism

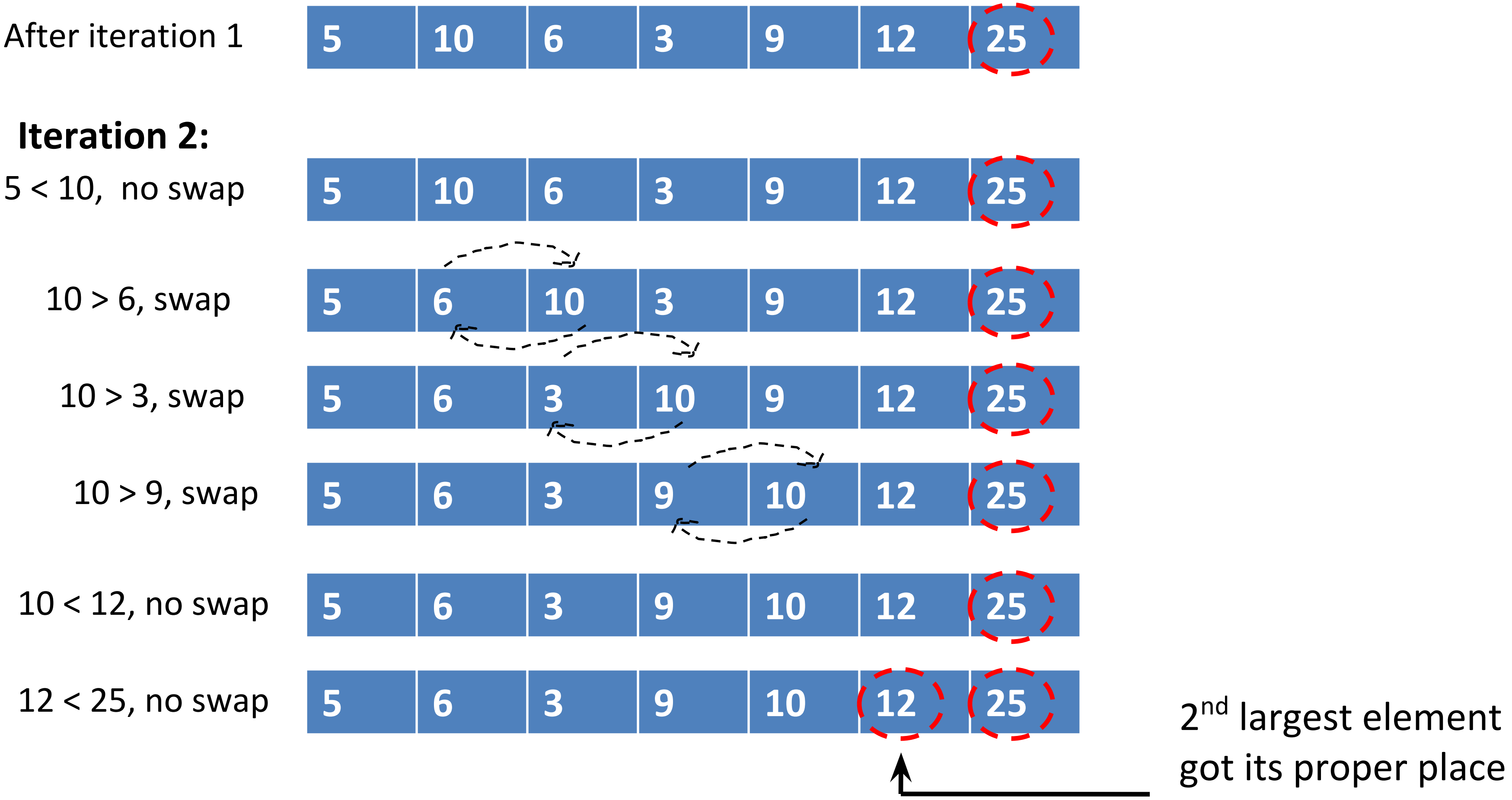


Input data set	10	5	25	6	3	9	12
Iteration 1:							
10 > 5, swap	5	10	25	6	3	9	12
10 < 25, no swap	5	10	25	6	3	9	12
25 > 6, swap	5	10	6	25	3	9	12
25 > 3, swap	5	10	6	3	25	9	12
25 > 9, swap	5	10	6	3	9	25	12
25 > 12, swap	5	10	6	3	9	12	25

Largest element got its proper place

Comparison based Sort

- Bubble sort: mechanism



After iteration 1	5	10	6	3	9	12	25
Iteration 2:							
5 < 10, no swap	5	10	6	3	9	12	25
10 > 6, swap	5	6	10	3	9	12	25
10 > 3, swap	5	6	3	10	9	12	25
10 > 9, swap	5	6	3	9	10	12	25
10 < 12, no swap	5	6	3	9	10	12	25
12 < 25, no swap	5	6	3	9	10	12	25

2nd largest element got its proper place

Comparison based Sort

- Bubble sort: algorithm and complexity

```
Bubble_sort( A)   // A is the array
{
  for i= 1 to N      // N is the number of elements
        for j= 1 to N-1
             if A(j) > A(j+1)
                 swap A(j) and A(j+1)
  return A
}
```

You can implement this cleverly

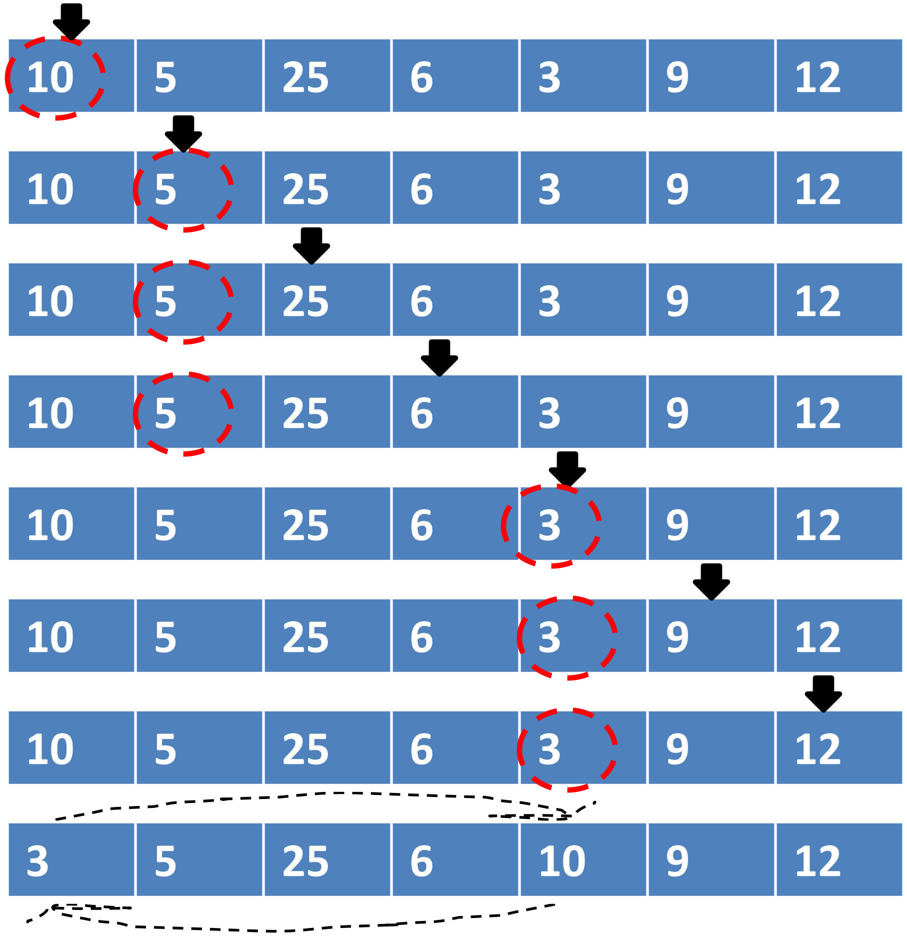
- Bubble sort: complexity

Time Complexity		
Best case	Average case	Worst case
$O(n^2)$	$O(n^2)$	$O(n^2)$

Comparison based Sort

- Selection sort: mechanism

Iteration 1:



Set min=10	10	5	25	6	3	9	12
5 < min, min=5	10	5	25	6	3	9	12
25 > min, min=5	10	5	25	6	3	9	12
6> min, min=5	10	5	25	6	3	9	12
3<min, min=3	10	5	25	6	3	9	12
9 > min, min=3	10	5	25	6	3	9	12
12 > min, min=3	10	5	25	6	3	9	12
Swap 1st and min element	3	5	25	6	10	9	12

Comparison based Sort

- Selection sort: algorithm and complexity

```
Selection_sort( A)   // A is the array
{
 for i= 1 to N     // N is the number of elements
      set min= i
      for j= i+1 to N
          if A(j) < A(min)
              set min = j
      swap A(i) and A(min)
 return A
}
```

- Selection sort: complexity

Time Complexity		
Best case	Average case	Worst case
O(n^2)	O(n^2)	O(n^2)

Iteration1: n times
Iteration 2: n-1 times
…
Iteration n: 1 time
Total: 1+2+…+(n-1)+n = O(n^2)

Queries?