PES UNIVERSITY ONLINE

DESIGN AND ANALYSIS OF ALGORITHMS

Surabhi Narayan
Department of Computer Science & Engineering

DESIGN AND ANALYSIS OF ALGORITHMS

DECREASE AND CONQUER

Surabhi Narayan
Department of Computer Science & Engineering

- **Decrease** or reduce problem instance to smaller instance of the same problem and extend solution.
- **Conquer** the problem by solving smaller instance of the problem.
- **Extend** solution of smaller instance to obtain solution to original problem .
- **Exploit** the relationship between a solution to a given instance of a problem and a solution to its smaller instance.

- Can be implemented either top-down or bottom-up
- Also referred to as *inductive* or *incremental* approach

3 Types of Decrease and Conquer

Decrease by a constant (usually by 1):
- insertion sort
- graph traversal algorithms (DFS and BFS)
- topological sorting
- algorithms for generating permutations, subsets

Decrease by a constant factor (usually by half)
- binary search and bisection method
- exponentiation by squaring
- multiplication à la russe

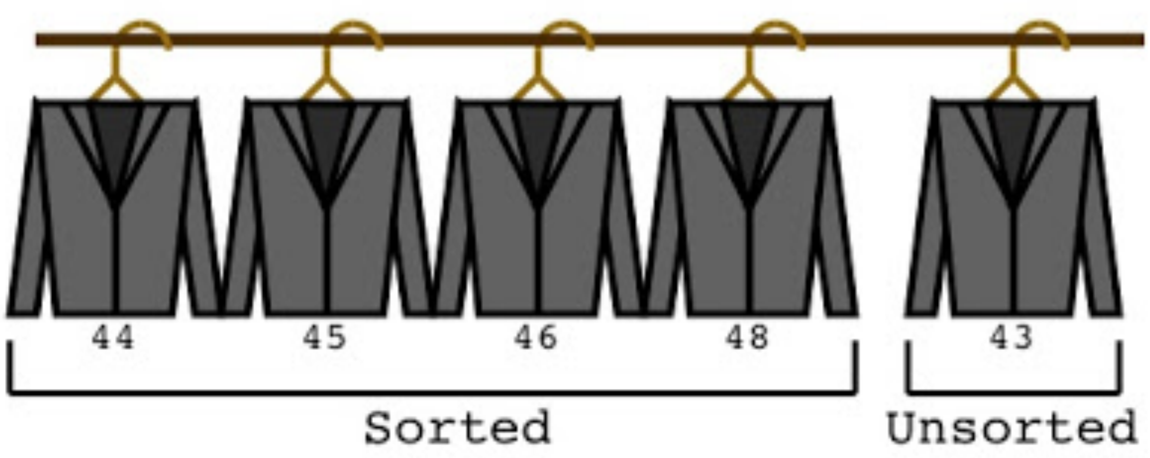
Variable-size decrease
- Euclid's algorithm
- selection by partition
- Nim-like games

This usually results in a recursive algorithm.

Insertion Sort

Imagine a card game

Cards in your hand are sorted.
The dealer hands you exactly one new card.
How would you rearrange your cards



Insertion Sort

- Insertion sort is based on the idea that one element from the input elements is consumed in each iteration to find its correct position i.e, the position to which it belongs in a sorted array.
- grows the sorted array at each iteration
- compares the current element with the largest value in the sorted array.
- If the current element is greater, then it leaves the element in its place and moves on to the next element else it finds its correct position in the sorted array and moves it to that position.
- This is done by shifting all the elements, which are larger than the current element, in the sorted array to one position ahead

Insertion Sort

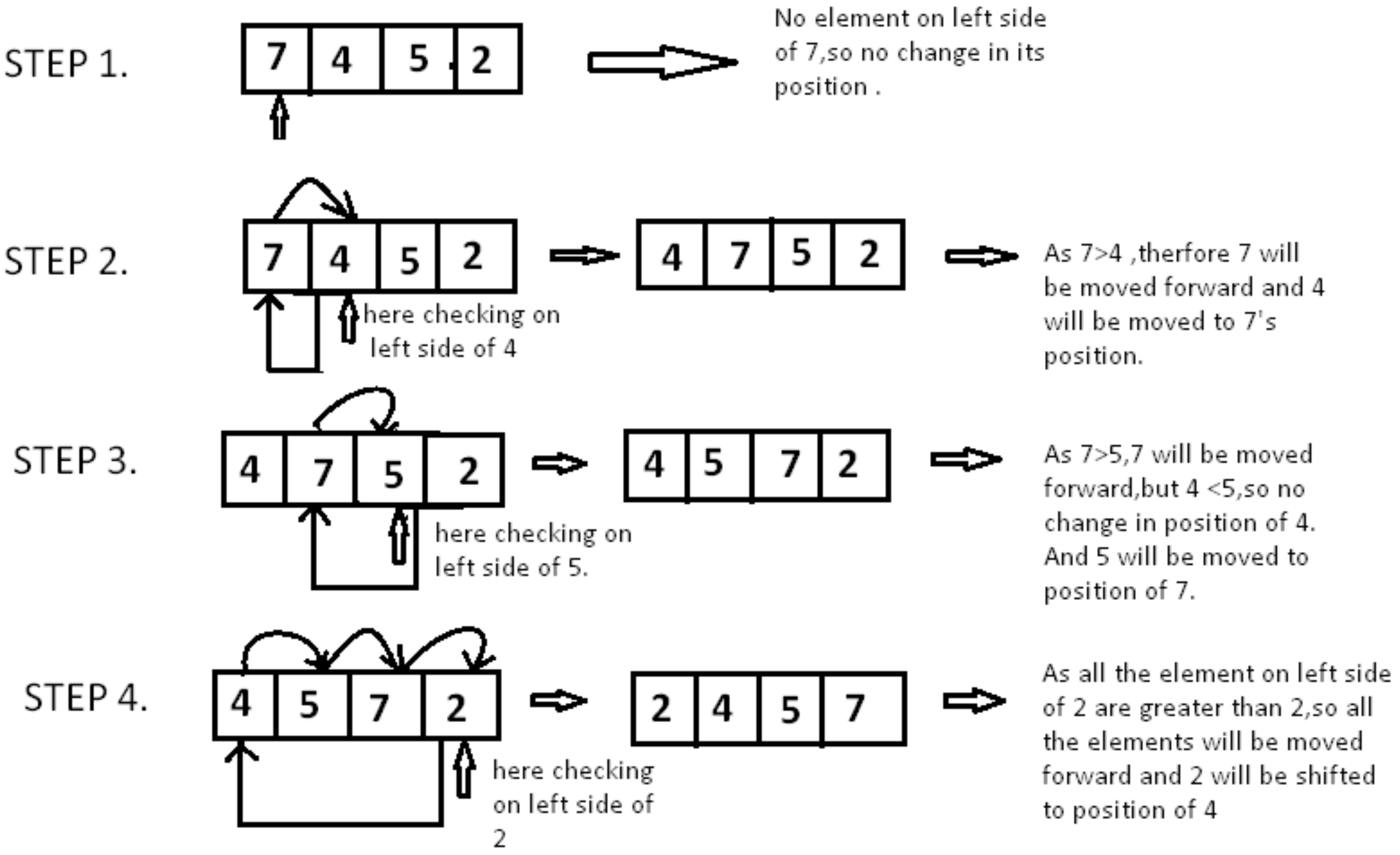
To sort array A[0..n-1], sort A[0..n-2] recursively and then insert A[n-1] in its proper place among the sorted A[0..n-2]

Usually implemented bottom up (nonrecursively)

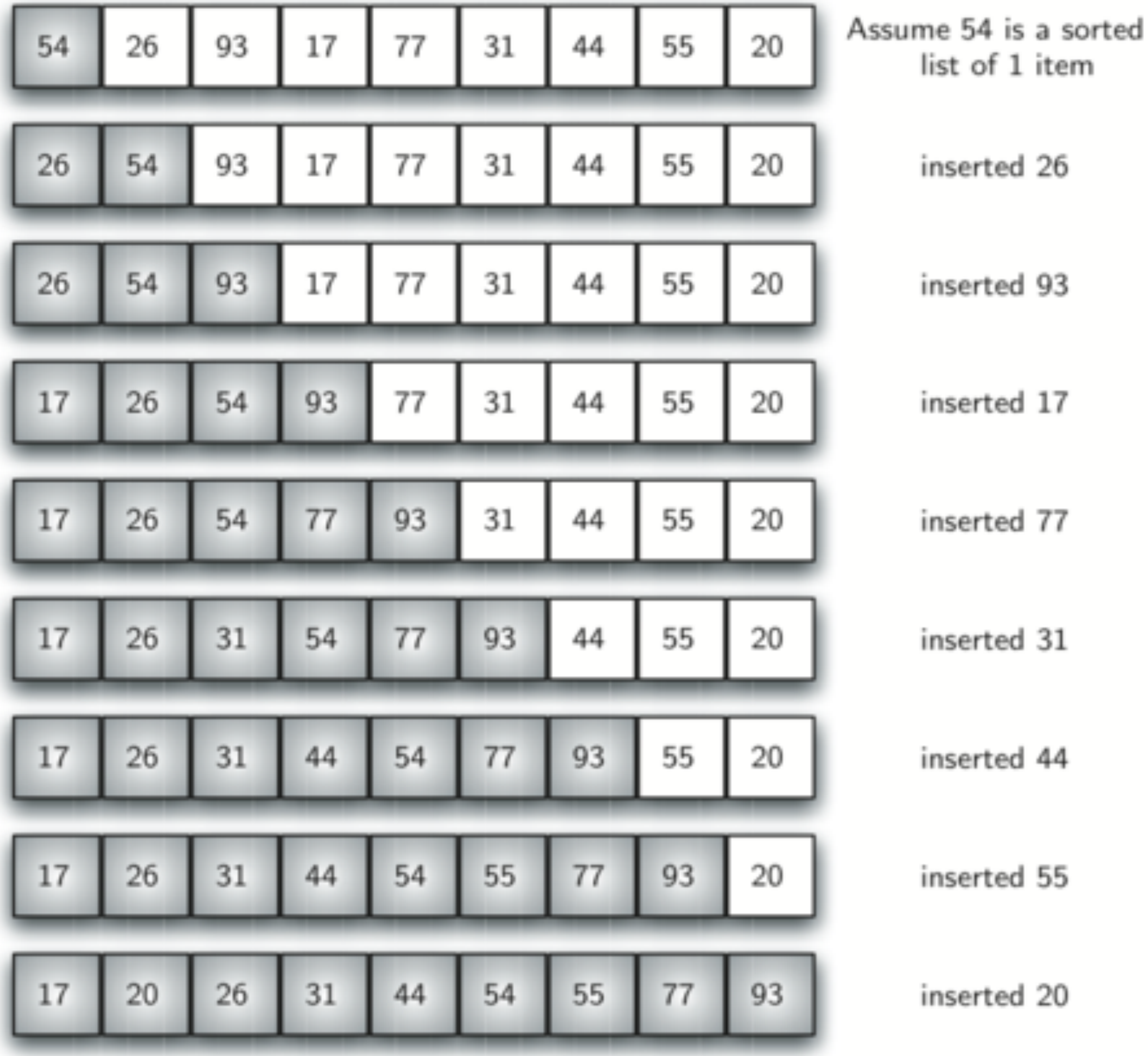
Example: Sort 6, 4, 1, 8, 5

6 | <u>4</u> 1 8 5
4 6 | <u>1</u> 8 5
1 4 6 | <u>8</u> 5
1 4 6 8 | <u>5</u>
1 4 5 6 8

Insertion Sort



Insertion Sort



ALGORITHM *InsertionSort*$(A[0..n-1])$
    //Sorts a given array by insertion sort
    //Input: An array $A[0..n-1]$ of n orderable elements
    //Output: Array $A[0..n-1]$ sorted in nondecreasing order
    for $i \leftarrow 1$ **to** $n-1$ **do**
        $v \leftarrow A[i]$
        $j \leftarrow i-1$
        while $j \geq 0$ **and** $A[j] > v$ **do**
            $A[j+1] \leftarrow A[j]$
            $j \leftarrow j-1$
        $A[j+1] \leftarrow v$

Time efficiency

$C_{worst}(n) = n(n\text{-}1)/2 \in \Theta(n^2)$

$\mathrm{C}_{\mathrm{avg}}(n) \approx n^2/4 \in \Theta(n^2)$

$C_{best}(n) = n - 1 \in \Theta(n)$ (also fast on almost sorted arrays)

Space efficiency: in-place

Stability: yes

Best elementary sorting algorithm overall

Binary insertion sort

THANK YOU

Surabhi Narayan
Department of Computer Science &Engineering
surabhinarayan@pes.edu