# PROGRAMMING FOR PROBLEM SOLVING -- PPS

## SUBJECT CODE- BTPS-101-18

*By*

*Dr.A.Deepa,AP,Chandigarh Engineering College*

# PPS SYLLABUS

***Unit 4***

**Basic Algorithms (6 lectures)**

Searching, Basic Sorting Algorithms (Bubble, Insertion and Selection), Finding roots of equations, notion of order of complexity through example programs (no formal definition required)

*By Dr.A.Deepa,AP,Chandigarh Engineering College*

# Sorting

- The process of arranging the elements in a list in either ascending or descending order.
- Types of Sorting:

- *Bubble Sort*
- *Insertion Sort*
- *Selection Sort*
- *Merge Sort*
- *Quick sort*
- *Heap Sort*
- *Radix Sort*

# Bubble Sorting

- Bubble sort is the easiest sorting algorithm to implement.
- It is inspired by observing the behaviour of air bubbles over foam.
- It is an in-place sorting algorithm.
- It uses no auxiliary data structures (extra space) while sorting.

# How Bubble Sort Works?

- Bubble sort uses multiple passes (scans) through an array.
- In each pass, bubble sort compares the adjacent elements of the array.
- It then swaps the two elements if they are in the wrong order.
- In each pass, bubble sort places the next largest element to its proper position.
- In short, it bubbles down the largest element to its correct position.

*By Dr.A.Deepa,AP,Chandigarh Engineering College*

# Bubble Sort Algorithm-

```
for(int pass=1 ; pass<=n-1 ; ++pass) // Making passes through array
{for(int i=0 ; i<=n-2 ; ++i)
{if(A[i] > A[i+1]) // If adjacent elements are in wrong order
swap(i,i+1,A); // Swap them
}}
//swap function : Exchange elements from array A at position x,y
void swap(int x, int y, int[] A)
{int temp = A[x];
A[x] = A[y];
A[y] = temp;
return ;
}// pass : Variable to count the number of passes that are done till now
// n : Size of the array
// i : Variable to traverse the array A
// swap() : Function to swap two numbers from the array
// x,y : Indices of the array that needs to be swapped
```

# Bubble Sort Example-

- Consider the following array A-Now, we shall implement the above bubble sort algorithm on this array.

A : Given Array

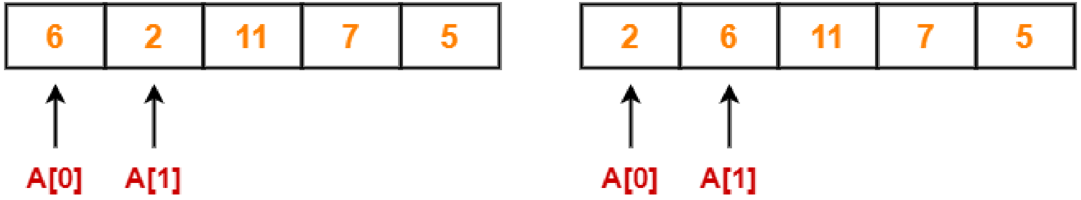
| | | | | |
|---|---|---|---|---|
| 6 | 2 | 11 | 7 | 5 |

**Step-01:**

We have pass=1 and i=0.

We perform the comparison A[0] > A[1] and swaps if the 0th element is greater than the 1th element.

Since 6 > 2, so we swap the two elements.

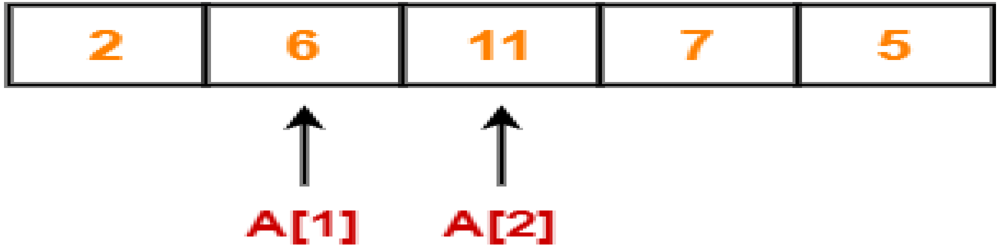


Before Swapping

After Swapping

# Bubble Sort Example-

- Step-02:
-
- We have pass=1 and i=1.
- We perform the comparison A[1] > A[2] and swaps if the 1th element is greater than the 2th element.
- Since 6 < 11, so no swapping is required.



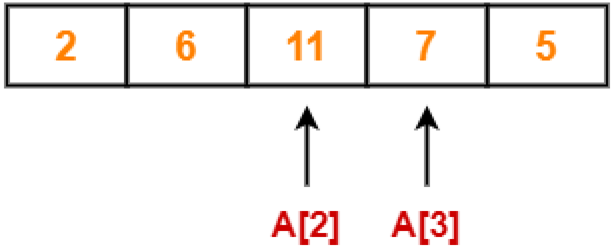
| | | | | |
|---|---|---|---|---|
| 2 | 6 | 11 | 7 | 5 |

A[1] A[2]

No Swapping Required

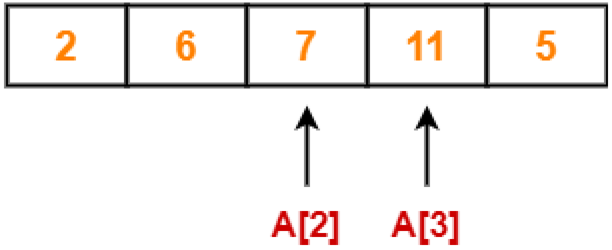
# Bubble Sort Example-

- Step-03:
-
- We have pass=1 and i=2.
- We perform the comparison A[2] > A[3] and swaps if the 2nd element is greater than the 3rd element.
- Since 11 > 7, so we swap the two elements.



| | | | | |
|---|---|---|---|---|
| 2 | 6 | 11 | 7 | 5 |

Before Swapping

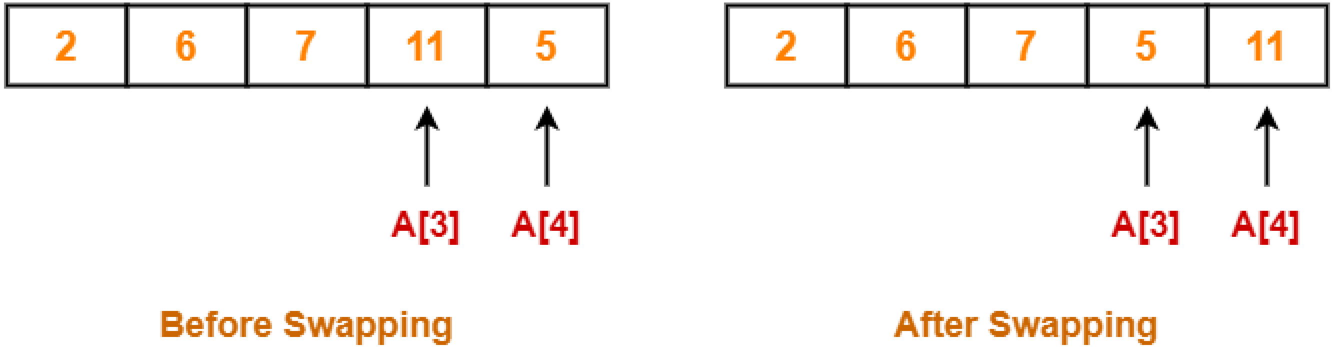


| | | | | |
|---|---|---|---|---|
| 2 | 6 | 7 | 11 | 5 |

After Swapping

# Bubble Sort Example-

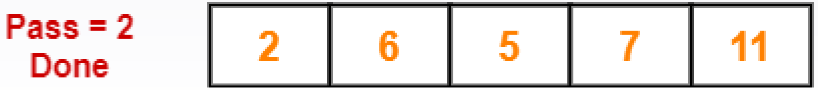
- Step-04:
-
- We have pass=1 and i=3.
- We perform the comparison A[3] > A[4] and swaps if the 3rd element is greater than the 4th element.
- Since 11 > 5, so we swap the two elements.



Finally after the first pass, we see that the largest element 11 reaches its correct position

# Bubble Sort Example-

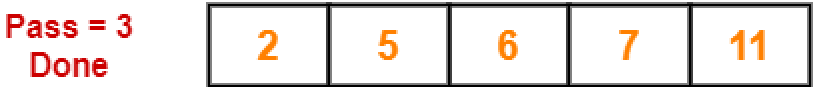
- Step-05:
- Similarly after pass=2, element 7 reaches its correct position.
- The modified array after pass=2 is shown below-



Pass = 2
Done

|   |   |   |   |   |
|---|---|---|---|---|
| 2 | 6 | 5 | 7 | 11 |

- Step-06:
- Similarly after pass=3, element 6 reaches its correct position.
- The modified array after pass=3 is shown below-



Pass = 3
Done

|   |   |   |   |   |
|---|---|---|---|---|
| 2 | 5 | 6 | 7 | 11 |

# Bubble Sort Example-

- Step-07:
- No further improvement is done in pass=4.
- This is because at this point, elements 2 and 5 are already present at their correct positions.
- The loop terminates after pass=4.
- Finally, the array after pass=4 is shown below-

Pass = 4
Done

| | | | | |
|---|---|---|---|---|
| 2 | 5 | 6 | 7 | 11 |

Array is Sorted

## Optimization Of Bubble Sort Algorithm-

- If the array gets sorted after a few passes like one or two, then ideally the algorithm should terminate.
- But still the above algorithm executes the remaining passes which costs extra comparisons.

```
for (int pass=1 ; pass<=n-1 ; ++pass)
{
flag=0 // flag denotes are there any swaps done in pass
for (int i=0 ; i<=n-2 ; ++i)
{
if(A[i] > A[i+1])
{
swap(i,i+1,A);
flag=1 // After swap, set flag to 1
}
}
if(flag == 0) break; // No swaps indicates we can terminate loop
}
void swap(int x, int y, int[] A)
{
int temp = A[x];
A[x] = A[y];
A[y] = temp;
return;
}
```

# Optimization Of Bubble Sort Algorithm-

- To avoid extra comparisons, we maintain a flag variable.
- The flag variable helps to break the outer loop of passes after obtaining the sorted array.
- The initial value of the flag variable is set to 0.
- The zero value of flag variable denotes that we have not encountered any swaps.
- Once we need to swap adjacent values for correcting their wrong order, the value of flag variable is set to 1.
- If we encounter a pass where flag == 0, then it is safe to break the outer loop and declare the array is sorted.

# Bubble Sort Algorithm- Time Complexity

| Bubble Sort Algorithm | Time Complexity |
|---|---|
| Best Case | $O(n)$ |
| Average Case | $\Theta(n^2)$ |
| Worst Case | $O(n^2)$ |

# Properties- Bubble Sort Algorithm

- Some of the important properties of bubble sort algorithm are-
- Bubble sort is a stable sorting algorithm.
- Bubble sort is an in-place sorting algorithm.
- The worst case time complexity of bubble sort algorithm is O(n2).
- The space complexity of bubble sort algorithm is O(1).
- Number of swaps in bubble sort = Number of inversion pairs present in the given array.
- Bubble sort is beneficial when array elements are less and the array is nearly sorted.

# Thank You

Queries????