2.3 OPTIMIZED BUBBLE SORT FOR A RANDOM ARRAY

Question:

Write code to modify bubble sort function to stop early if the list becomes sorted before all passes are completed.

AIM

To sort a given array using Bubble Sort with an early stopping condition if the array becomes sorted before completing all passes.

ALGORITHM

1. Start from the first element.
2. Compare adjacent elements, swapping them if they are in the wrong order.
3. After each pass, the largest element in the unsorted region moves to its correct position.
4. Keep track of whether any swaps occurred in the current pass.
5. If no swaps are made, stop early since the list is already sorted.

PROGRAM

```
def bubble_sort_optimized(arr):
    n = len(arr)
    for i in range(n):
        swapped = False
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
                swapped = True
        print(f"Pass {i+1}: {arr}")
        if not swapped:
            break
    return arr

def run_bubble_sort():
    arr = list(map(int, input("Enter array elements: ").split()))
    print("Sorted array:", bubble_sort_optimized(arr))

run_bubble_sort()
```

Input:

64 25 12 22 11

Output:

```
Enter array elements: 64 25 12 22 11
Pass 1: [25, 12, 22, 11, 64]
Pass 2: [12, 22, 11, 25, 64]
Pass 3: [12, 11, 22, 25, 64]
Pass 4: [11, 12, 22, 25, 64]
Pass 5: [11, 12, 22, 25, 64]
Sorted array: [11, 12, 22, 25, 64]
>>>
```

RESULT:

Thus the program is successfully executed and the output is verified.

PERFORMANCE ANALYSIS:

- Time Complexity:
  - Best Case (already sorted): O(n)
  - Worst/Average Case: $O(n^2)$
- Space Complexity:
  - Uses only a few variables, so O(1) (constant extra space).