

Exp-2.3

Title:

Optimized Bubble Sort with Early Stop

Aim:

To implement a Bubble Sort algorithm that detects if the list is already sorted during any pass and stops early to improve efficiency.

Procedure:

1. Take input size n and the array.
2. Perform Bubble Sort with a flag to monitor swaps during each pass.
3. For each pass, compare adjacent elements and swap if needed.
4. If no swaps occur during a pass, it means the list is sorted, so stop early.
5. Print the sorted array.

Algorithm:

1. Start
2. For each pass from 0 to $n-1$:
 - Set a flag swapped to False.
 - For each adjacent pair in unsorted part of the array:
 - Swap if the order is wrong.
 - Set swapped to True.
 - If swapped is False after the pass, break early (list is sorted).
3. End with sorted array.

Input:

6

5 1 4 2 8 0

Output:

0 1 2 4 5 8

Program:

```
def bubble_sort(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
        # If no two elements were swapped in the inner loop, break
        if not swapped:
            break
    return arr

n = int(input("Enter number of elements: "))
arr = list(map(int, input(f"Enter {n} elements separated by space: ").split()))

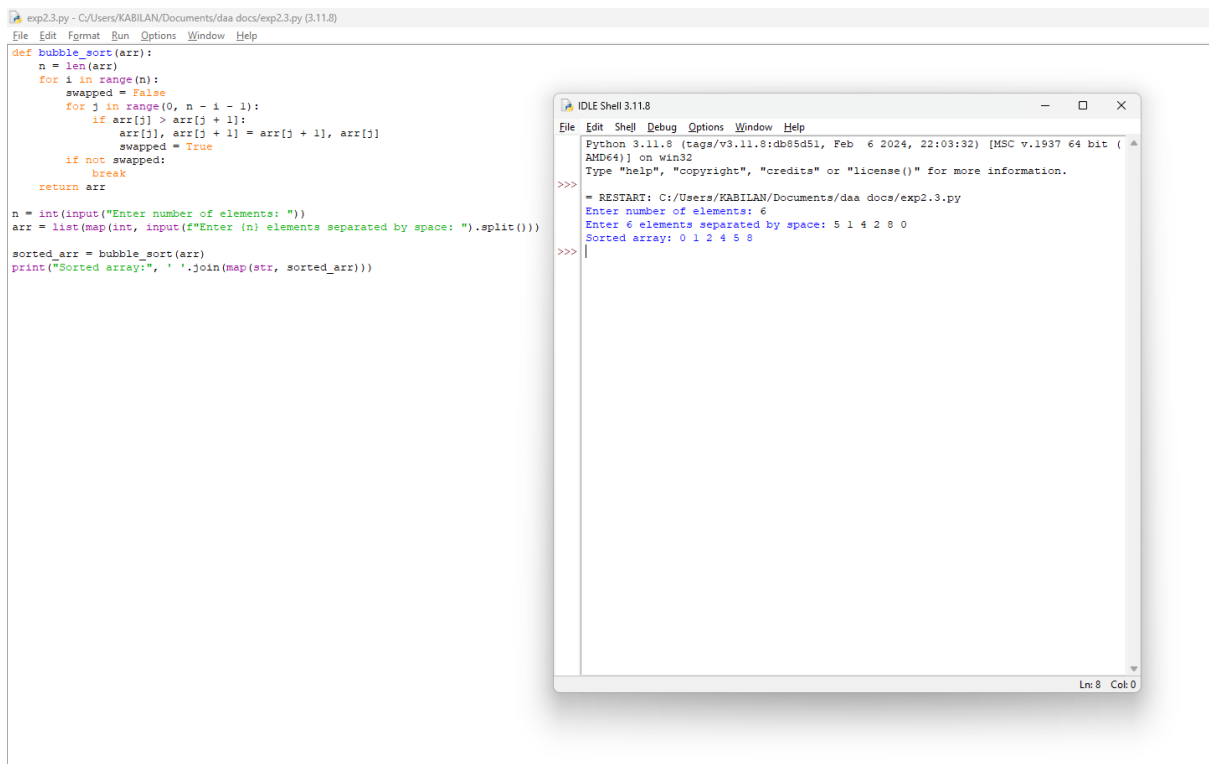
sorted_arr = bubble_sort(arr)
print("Sorted array:", ' '.join(map(str, sorted_arr)))
```

Performance Analysis:

Time Complexity: $O(n)$. Best cases , $O(n^2)$ worst cases

Space Complexity: $O(1)$

Program Output:



The image shows a Python IDE window titled 'exp2.3.py - C:/Users/KABILAN/Documents/daa docs/exp2.3.py (3.11.8)'. The code implements an optimized bubble sort. It defines a function 'bubble_sort(arr)' that iterates through the array, swapping adjacent elements if they are in the wrong order. The optimization includes a 'swapped' flag and a 'break' statement to stop the loop if no swaps occurred. The main part of the code prompts the user to enter the number of elements and the elements themselves, then prints the sorted array.

```
def bubble_sort(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
        if not swapped:
            break
    return arr

n = int(input("Enter number of elements: "))
arr = list(map(int, input(f"Enter {n} elements separated by space: ").split()))

sorted_arr = bubble_sort(arr)
print("Sorted array:", ' '.join(map(str, sorted_arr)))
```

The output window shows the execution of the program. It prompts for the number of elements (6) and the elements (5 1 4 2 8 0). The sorted array is displayed as 0 1 2 4 5 8.

```
IDLE Shell 3.11.8
Python 3.11.8 (tags/v3.11.8:db85d51, Feb  6 2024, 22:03:32) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/KABILAN/Documents/daa docs/exp2.3.py
Enter number of elements: 6
Enter 6 elements separated by space: 5 1 4 2 8 0
Sorted array: 0 1 2 4 5 8
>>>
```

Result:

Thus the given program Optimized Bubble Sort is executed and got output successfully.