OCTOBER 10, 2021

BSSE 4B MORNING

ROLL NO 12302

ANALYSIS OF ALGORITHM

ASSIGNMENT #01

SUBMITTED BY: M FARHAN MOBEEN
SUBMITTED TO: MAM IQRA SHAHZAD

Question : Calculate the time and space complexity of Bubble sort algorithm

a. write a pseudocode for bubble sorting of an array
size n= 7 ; diagram depiction.

b. Find its best average and worst cases with respect to
the question statement.

ANSWER

BUBBLE SORT

An in-place sorting algorithm that finds max. element in each cycle and puts it in appropriate position in list by performing swapping adjacent elements. In bubble sort, we continue swapping adjacent elements until they are in correct order.

PSEUDO CODE

```
Initialize n = Length of Array

BubbleSort(Array, n)

{

   for i = 0 to n-2

   {

      for j = 0 to n-2

      {

         if Array[j] > Array[j+1]

         {

            swap(Array[j], Array[j+1])

         }

}

   }

}
```

CONSTANT TIME COMPLEXITY

To remain constant the algorithms should not conta

in loops, recursions or calls to any other functions

the time complexity of Bubble Sort is O(n^2).

Time and Space complexity for the Bubble Sort algorithm.

- Worst Case Time Complexity [Big-O]: O(n^2)
- Best Case Time Complexity [Big-omega]: O(n)
- Average Time Complexity [Big-theta]: O(n^2)
- Space Complexity: O(1)

The main advantage of Bubble Sort is the simplicity of the algorithm.

The space complexity for Bubble Sort is O(1), because only a single additional memory space is required

Also, the best case time complexity will be O(n), it is when the list is already sorted.

BEST CASE TIME COMPLEXITY

Let's start with the most straightforward case: If the numbers are already sorted in ascending order, the algorithm will determine in the first iteration that no number pairs need to be swapped and will then terminate immediately.

The algorithm must perform n-1 comparisons; therefore:

The best-case time complexity of Bubble Sort is: *O(n)*

WORST AND AVERAGE CASE TIME COMPLEXITY:

O(n*n). Worst case occurs when array is reverse sorted. And when the targeted value is at the end of the array or worst will be when the targeted value is not present in the array.

5 is greater then 1 so interchange

5	1	3	2	6	7	4

3 is less then 5 so interchange

1	5	3	2	6	7	4

5 is greater then 2 so interchange

1	3	5	2	6	7	4

2 is less then 3 so swap

1	3	2	5	6	7	4

7 is greater then 4 so interchange

1	2	3	5	6	7	4

6 is greater then 4 so swap

1	2	3	5	6	4	7

5 is greater then 4 so swap

1	2	3	5	4	6	7

The array is sorted

1	2	3	4	5	6	7