Mosa. Rabeya

Sub.: _____

I'd : 191-15-12145

SAT SUN MON TUE WED THU FRI
☐   ☐   ☐   ☐   ☐   ☐   ☐

DATE :   /   /

Bubble Sort

Implementation :

```
void   bubble_sort (int arr[],int n )
{
    int i, j, temp, flag;
    for (i=0; i<n-1 ; i++)
    {
    if (arr[j] > arr[j+1] )
    {
    temp = arr[j];
    arr[j]  = arr[j+1];
    arr[j+1] = temp;
    flag = 1;
    }
    }
    if (flag == 0)
        break ;
    }
}
```

Analysis:

Here we have, $arr[5] = \{5, 1, 4, 2, 8\}$

For ascending order:

1st Iteration:

$(5\ 1\ 4\ 2\ 8) \rightarrow (1\ 5\ 4\ 2\ 8)$, Here, algorithm compares the first two elements, and swaps since $5 > 1$.

$(1\ 5\ 4\ 2\ 8) \rightarrow (1\ 4\ 5\ 2\ 8)$, swap since $5 > 4$.

$(1, 4\ 5\ 2\ 8) \rightarrow (1\ 4\ 2\ 5\ 8)$, swap since $5 > 2$

$(1\ 4\ 2\ 5\ 8) \rightarrow (1, 4\ 2\ 5\ 8)$, since these elements are already in order $(8 > 5)$, algorithm does not swap them.

2nd Iteration:

$(1\ 4\ 2\ 5\ 8) \rightarrow (1\ 4\ 2\ 5\ 8)$

$(1\ 4\ 2\ 5\ 8) \rightarrow (1\ 2\ 4\ 5\ 8)$, swap $4 > 2$

$(1\ 2\ 4\ 5\ 8) \rightarrow (1\ 2\ 4\ 5\ 8)$

$(1\ 2\ 4\ 5\ 8) \rightarrow (1\ 2\ 4\ 5\ 8)$

The array is already sorted but our algorithm does not know if it is completed. The algorithm needs one whole pass without any swap to know it is sorted.

3rd iteration:

(1 2 4 5 8) → (1 2 4 5 8)

(1 2 4 5 8) → (1 2 4 5 8)

(1 2 4 5 8) → (1 2 4 5 8)

(1 2 4 5 8) → (1 2 4 5 8)

Time complexity (Bubble sort)

Best case:

1	2	3	4	5

If the array is already sorted, then only inner loop will execute for (n-1) times. Theref in best case the complexity of bubble sort is O(n).

Worst case :

5	4	3	2	1

For reversely sorted array elements, if there are n elements, for all iteration the total checked condition will be $(n-1)+(n-2)$

$+ (n-3) + \ldots + 3 + 2 + 1$

The eqⁿ can be written as,

which is a polynomial eqⁿ. Considering

the highest order of n is 2.

∴ In worst case, the complexity of bubble sort is $O(n^2)$.