Bubble Sort

Bubble Sort is a simple algorithm which is used to sort a given set of n elements provided in form of an array with n number of elements. Bubble Sort compares all the element one by one and sort them based on their values.

Implementation

```
#include<stdio.h>

int n;
void Sort(int ara[])
{
    for(int i=0; i<n; i++)
        for(int j=0; j<(n-i-1); j++)
        {
            if(ara[j]>ara[j+1])
            {
                int p=ara[j];
                ara[j]=ara[j+1];
                ara[j+1]=p;
            }
        }

}


int main()
{
    int ara[]= {5,1,4,2,8};
    n=6;
    printf("Array before swap: ");
    for(int i=0; i<n; i++)
        printf("%d ",ara[i]);
    printf("\n");
    Sort(ara);
    printf("Array after swap: ");
    for(int i=0; i<n; i++)
        printf("%d ",ara[i]);
    printf("\n");
}
```

S.M. Khasrul Alam Shakil
191-15-12180

Analysis:

Let’s consider an array: (5, 1, 4, 2, 8)

We want to sort it in ascending order:

So,

1st Iteration:
(**5 1** 4 2 8) –> (**1 5** 4 2 8), Here, algorithm compares the first two elements, and swaps since 5 > 1.
(1 **5 4** 2 8) –> (1 **4 5** 2 8), Swap since 5 > 4
(1 4 **5 2** 8) –> (1 4 **2 5** 8), Swap since 5 > 2
(1 4 2 **5 8**) –> (1 4 2 **5 8**), Now, since these elements are already in order (8 > 5), algorithm does not swap them.

2nd Iteration:
(**1 4** 2 5 8) –> (**1 4** 2 5 8)
(1 **4 2** 5 8) –> (1 **2 4** 5 8), Swap since 4 > 2
(1 2 **4 5** 8) –> (1 2 **4 5** 8)
(1 2 4 **5 8**) –> (1 2 4 **5 8**)
Now, 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

In Bubble Sort, (n-1) comparisons will be done in the 1st pass, (n-2) in 2nd pass, (n-3) in 3rd pass and so on. So the total number of comparisons will be,

(n-1) + (n-2) + (n-3) + (n-4) + …… + 3 + 2 + 1

= (n-1)*n/2

Ignoring the constant co-efficient, we can say that the complexity is: O(n^2)

By using the following process we have to do same number operation for best case, worst case and average case.

So complexity is: O(n^2)

Optimized Bubble sort

Sample :

```
void bubbleOpt(int ara[])
{
        bool flg;
        for(int i=0; i<n; i++)
        {
         flg=true;
         for(int j=0; j<(n-i-1); j++)
                if(ara[j]>ara[j+1])
                        swap(ara[j],ara[j+1]),flg=false;
        if(!flg)
         break;

        }
}
```

It can be optimized by stopping the algorithm if inner loop didn't cause any swap.

<u>Worst and Average Case</u>: O(n*n). Worst case occurs when array is reverse sorted.

<u>Best Case</u>: O(n). Best case occurs when array is already sorted.

S.M. Khasrul Alam Shakil
191-15-12180