

CERTIFICATE

*This is to certify that Mr./Ms. **Hemil...Chovatiya**..... with enrolment no.**200303108003**..... has successfully completed **his/her** laboratory experiments in the **.....Design and Analysis of Algorithms.....** from the department of **.....Information Technology(5ITA1).....** during the academic year **.....2022-2023.....***



Date of Submission:

Staff In charge:

Head of Department:

PRACTICAL-1

Aim:- Implementation and Time analysis of Bubble, Selection and Insertion sorting algorithms for best case, average case & worst case.

1) Bubble Sorting:-

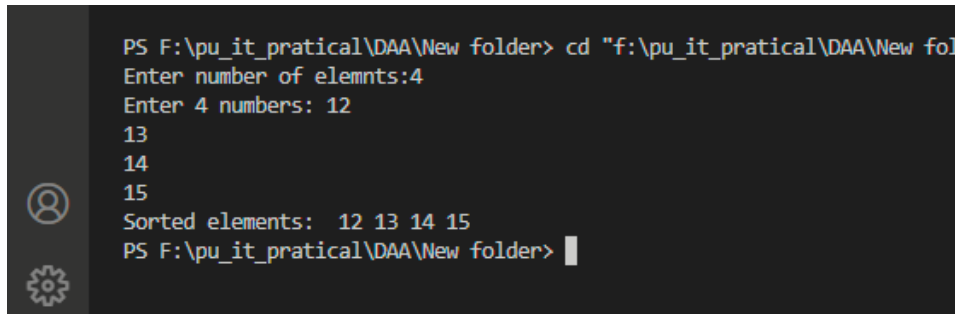
Algorithm:

1. begin BubbleSort(arr)
2. for all array elements
3. if arr[i] > arr[i+1]
4. swap(arr[i], arr[i+1])
5. end if
6. end for
7. return arr
8. end BubbleSort

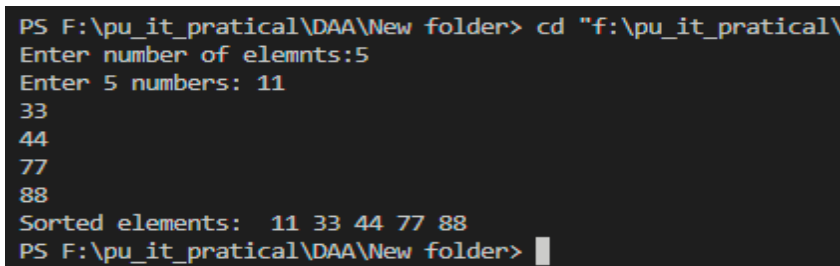
Code:-

```
#include<stdio.h>
#include<conio.h>
int main(){
int n, temp, i, j, number[30];
printf("Enter number of elemnts:");
scanf("%d",&n);
printf("Enter %d numbers: ",n);
for(i=0;i<n;i++){
scanf("%d",&number[i]);
for(i=n-2;i>=0;i--){
for(j=0;j<=i;j++){
if(number[j]>number[j+1]){
temp=number[j];
number[j]=number[j+1];
number[j+1]=temp;
} } }
```

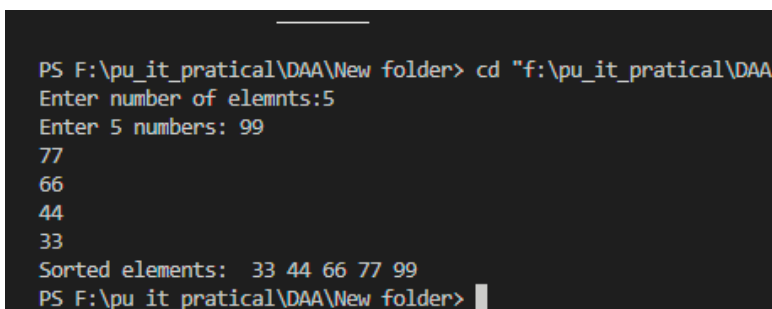
```
printf("Sorted elements: ");  
for(i=0;i<n;i++)  
printf(" %d",number[i]);  
return 0;  
}
```

OUTPUT:Best Case:O(n)

```
PS F:\pu_it_practical\DAA\New folder> cd "f:\pu_it_practical\DAA\New fo  
Enter number of elemnts:4  
Enter 4 numbers: 12  
13  
14  
15  
Sorted elements: 12 13 14 15  
PS F:\pu_it_practical\DAA\New folder> |
```

Avg Case:O(n²)

```
PS F:\pu_it_practical\DAA\New folder> cd "f:\pu_it_practical\  
Enter number of elemnts:5  
Enter 5 numbers: 11  
33  
44  
77  
88  
Sorted elements: 11 33 44 77 88  
PS F:\pu_it_practical\DAA\New folder> |
```

Wroost Case:O(n²)

```
PS F:\pu_it_practical\DAA\New folder> cd "f:\pu_it_practical\DAA  
Enter number of elemnts:5  
Enter 5 numbers: 99  
77  
66  
44  
33  
Sorted elements: 33 44 66 77 99  
PS F:\pu_it_practical\DAA\New folder> |
```

2) Selection Sorting:-**Algorithm:**

1. SELECTION SORT(arr, n)
2. Step 1: Repeat Steps 2 and 3 for i = 0 to n-1
3. Step 2: CALL SMALLEST(arr, i, n, pos)

4. Step 3: SWAP arr[i] with arr[pos]
5. [END OF LOOP]
6. Step 4: EXIT
7. SMALLEST (arr, i, n, pos)
8. Step 1: [INITIALIZE] SET SMALL = arr[i]
9. Step 2: [INITIALIZE] SET pos = i
10. Step 3: Repeat for j = i+1 to n
11. if (SMALL > arr[j])
12. SET SMALL = arr[j]
13. SET pos = j
14. [END OF if]
15. [END OF LOOP]
16. Step 4: RETURN pos

Code:-

```
#include<stdio.h>
#include<conio.h>
int main()
{
int a[100], n, i, j, position, swap;
printf("enter the number of inputs:");
scanf("%d", &n);
printf("Enter %d Numbers:", n);
for (i = 0; i < n; i++)
scanf("%d", &a[i]);
for(i = 0; i < n - 1; i++)
{
position=i;
for(j = i + 1; j < n; j++)
{
if(a[position] > a[j])
```

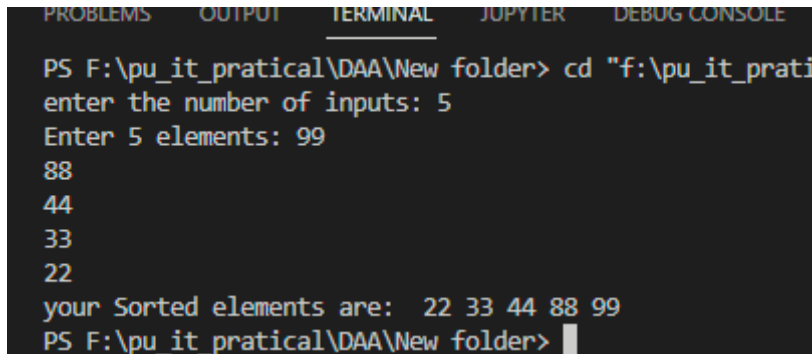
```
position=j;
}
if(position != i)
{
swap=a[i];
a[i]=a[position];
a[position]=swap;
}
}
printf("Sorted Array:");
for(i = 0; i < n; i++)
printf("%d\n", a[i]);
return 0;
}
```

OUTPUT: Best Case:($O(n^2)$)

```
enter the number of inputs: 5
Enter 5 elements: 1
2
4
5
8
your Sorted elements are: 1 2 4 5 8
PS F:\pu_it_practical\DAA\New folder>
```

Avg Case: $O(n^2)$

```
PS F:\pu_it_practical\DAA\New folder> cd "f:\pu_it_practical\DAA\New f
enter the number of inputs: 5
Enter 5 elements: 11
12
49
18
12
your Sorted elements are: 11 12 12 18 49
PS F:\pu_it_practical\DAA\New folder>
```

Worst Case: $O(n^2)$ 

```
PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE
PS F:\pu_it_practical\DAA\New folder> cd "f:\pu_it_prati
enter the number of inputs: 5
Enter 5 elements: 99
88
44
33
22
your Sorted elements are: 22 33 44 88 99
PS F:\pu_it_practical\DAA\New folder>
```

3) Insertion Sorting:-**Algorithm:**

- Step 1 - If the element is the first element, assume that it is already sorted. Return 1.
- Step2 - Pick the next element, and store it separately in a key.
- Step3 - Now, compare the key with all elements in the sorted array.
- Step 4 - If the element in the sorted array is smaller than the current element, then move to the next element. Else, shift greater elements in the array towards the right.
- Step 5 - Insert the value.
- Step 6 - Repeat until the array is sorted.

Code:-

```
#include <math.h>
#include <stdio.h>

int main(){
    int i, j, count, temp, number[25];
    printf("numbers of input: ");
    scanf("%d",&count);
    printf("Enter %d the numbers: ", count);
    for(i=0;i<count;i++)
        scanf("%d",&number[i]);

    for(i=1;i<count;i++)
    {
        temp=number[i];
        j=i-1;
        while((temp<number[j])&&(j>=0))
```

```
{  
    number[j+1]=number[j];  
    j=j-1;  
}  
number[j+1]=temp;  
}  
printf("your insertion sorting is: ");  
for(i=0;i<count;i++)  
    printf(" %d",number[i]);  
return 0;  
}
```

OUTPUT:Best Case:O(n)

```
0 insertion } , if ($?) { .\insertion }  
numbers of input: 4  
Enter 4 the numbers: 1  
2  
3  
4  
your insertion sorting is: 1 2 3 4  
PS F:\pu_it_practical\DAA\New folder> |
```

Avg Case:O(n^2)

```
{ gcc insertion.c -o insertion } , if ($?) { .\insertion }  
numbers of input: 5  
Enter 5 the numbers: 9  
1  
4  
2  
10  
your insertion sorting is: 1 2 4 9 10
```

Wroost Case:O(n^2)

```
numbers of input: 5  
Enter 5 the numbers: 99  
88  
77  
66  
11  
your insertion sorting is: 11 66 77 88 99
```

Practical 2:

AIM: Implementation and Time analysis of Max-Heap sort algorithm.

Algorithm:

1. HeapSort(arr)
2. BuildMaxHeap(arr)
3. for $i = \text{length}(\text{arr})$ to 2
4. swap arr[1] with arr[i]
5. heap_size[arr] = heap_size[arr] ? 1
6. MaxHeapify(arr,1)
7. End

1. BuildMaxHeap(arr)
2. heap_size(arr) = length(arr)
3. for $i = \text{length}(\text{arr})/2$ to 1
4. MaxHeapify(arr,i)
5. End

1. MaxHeapify(arr,i)
2. L = left(i)
3. R = right(i)
4. if L ? heap_size[arr] and arr[L] > arr[i]
5. largest = L
6. else
7. largest = i
8. if R ? heap_size[arr] and arr[R] > arr[largest]
9. largest = R
10. if largest != i
11. swap arr[i] with arr[largest]
12. MaxHeapify(arr,largest)
13. End

Code:

```
#include <stdio.h>

void swap(int *a, int *b)
{
    int temp = *a;
```



```
*a = *b;
*b = temp; }
void heapify(int arr[], int n, int i)
{   int largest = i;
    int left = 2 * i + 1;
    int right = 2 * i + 2;
    if (left < n && arr[left] > arr[largest])
        largest = left;
    if (right < n && arr[right] > arr[largest])
        largest = right;
    if (largest != i)
    {   swap(&arr[i], &arr[largest]);
        heapify(arr, n, largest);
    } }
void heapSort(int arr[], int n)
{
    for (int i = n / 2 - 1; i >= 0; i--)
        heapify(arr, n, i);
    for (int i = n - 1; i >= 0; i--)
    {   swap(&arr[0], &arr[i]);
        heapify(arr, i, 0);
    } }
int main()
{
    int n,i;
    printf("Enter Array size: ");
    scanf("%d",&n);
    int arr[n];
    for(i=0;i<n;i++)
    {   printf("Enter Element %d: ",i+1);
```

```
scanf("%d",&arr[i]);    }  
heapSort(arr, n);  
printf("\nSorted Heap array:\n");  
i=0;  
for(i=0;i<n;i++)  
{  
printf("%d\t",arr[i]);  
} }
```

OUTPUT: Best Case: $n\log(n)$

```
PS F:\pu_it_practical\DAA\New folder> c  
Enter Array size: 5  
Enter Element 1: 1  
Enter Element 1: 1  
Enter Element 2: 2  
Enter Element 3: 3  
Enter Element 4: 4  
Enter Element 5: 5  
  
Sorted Heap array:  
1      2      3      4      5  
PS F:\pu_it_practical\DAA> |
```

Avg Case: $n\log(n)$

```
PS C:\Users\raj> cd "f:\pu_it_practical\DAA\" ; if ($?) { gcc hea  
Enter Array size: 5  
Enter Element 1: 12  
Enter Element 2: 1  
Enter Element 3: 9  
Enter Element 4: 4  
Enter Element 5: 7  
  
Sorted Heap array:  
1      4      7      9      12  
PS F:\pu_it_practical\DAA> |
```

Worst case: $n \log(n)$

```
PS F:\pu_it_practical\DAA> cd "f:\pu_it_practical\DAA\" ; if ($?) {  
Enter Array size: 5  
Enter Element 1: 99  
Enter Element 2: 77  
Enter Element 3: 66  
Enter Element 4: 55  
Enter Element 5: 44  
  
Sorted Heap array:  
44    55    66    77    99  
PS F:\pu_it_practical\DAA>
```