

## ` Analysis on Algorithm

### Practical no 1:

#### Insertion Sort

Code :

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

```
#include<conio.h>
```

```
void insertionSort(int A[], int n) {
```

```
    for (int j = 1; j < n; j++) {
```

```
        int key = A[j];
```

```
        int i = j - 1;
```

```
        while (i >= 0 && A[i] > key) {
```

```
            A[i + 1] = A[i];
```

```
            i = i - 1;
```

```
        }
```

```
        A[i + 1] = key;
```

```
    }
```

```
}
```

```
int main() {
```

```
    int A[] = {2, 13, 5, 18, 14};
```

```
    int n = sizeof(A) / sizeof(A[0]);
```

```
    insertionSort(A, n);
```

```
    printf("Sorted array: ");
```

```
    for (int i = 0; i < n; i++) {
```

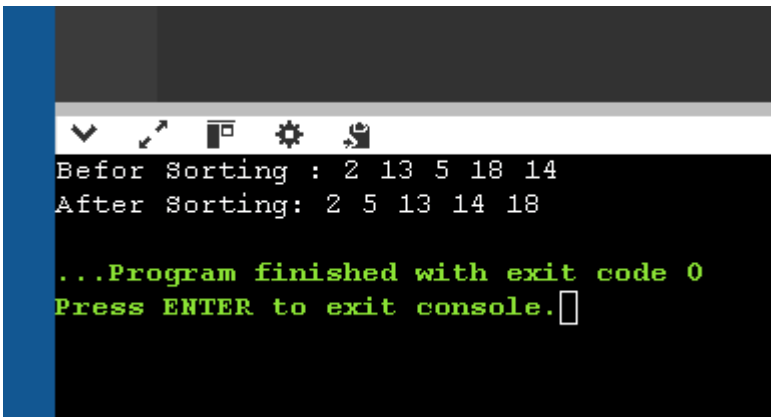
```
        printf("%d ", A[i]);
```

```
    }
```

```
    return 0;
```

```
}
```

Output :



```

Befor Sorting : 2 13 5 18 14
After Sorting: 2 5 13 14 18

...Program finished with exit code 0
Press ENTER to exit console.
```

Analysis of Insertion sort :

#### Code Analysis: Insertion Sort Implementation in C

The provided code implements the insertion sort algorithm in C. Below is a detailed analysis of the code, covering its structure, functionality, and potential improvements.

#### Code Structure

##### 1. Insertion Sort Function:

- **Parameters:** The function takes an array A and its size n.
- **Logic:**
  - The outer loop iterates from the second element to the last element of the array.
  - The inner loop shifts elements that are greater than key to the right, making space for the key to be inserted in its correct position.
  - The key is then placed at the correct position in the sorted part of the array.

##### 2. Main Function:

- An integer array A is initialized with unsorted values.
- The size of the array is calculated using sizeof.
- The program prints the array before and after sorting by calling the insertionSort function.

#### Complexity Analysis

- **Time Complexity:**
  - Best Case:  $O(n)$  when the array is already sorted.
  - Average Case:  $O(n^2)$
  - Worst Case:  $O(n^2)$  when the array is sorted in reverse order.
- **Space Complexity:**  $O(1)$  since it uses a constant amount of additional space.

## Selection Sort

Code :

```
class SelectionSort {  
    public static void main(String[] args) {  
        int[] arr = {64, 34, 25, 12, 22};  
        int n = arr.length;  
        System.out.println("Before Sorting : ");  
        for(int i = 0; i < n; i++)  
            System.out.print(arr[i] + " ");  
  
        for (int i=0; i<n-1; i++) {  
            int min = i;  
            for (int j=i+1; j<n; j++) {  
                if(arr[j] < arr[min])  
                    min = j;  
            }  
            int temp = arr[min];  
            arr[min] = arr[i];  
            arr[i] = temp;  
        }  
        System.out.println("");  
        System.out.println("After Sorting : ");  
        for(int i=0; i<n; i++)  
            System.out.print(arr[i] + " ");  
  
    }  
}
```

Output :

```
Before Sorting :  
64 34 25 12 22  
After Sorting :  
12 22 25 34 64
```

Analysis :

#### Code Analysis: Selection Sort Implementation in Java

The provided code implements the selection sort algorithm in Java. Below, we will analyze the structure, functionality, and performance of the code, along with suggestions for improvement.

#### Code Structure

##### 1. Main Method:

- An integer array `arr` is initialized with unsorted values.
- The size of the array is obtained using `arr.length`.
- The program prints the array before sorting.

##### 2. Selection Sort Logic:

- The outer loop iterates through each element of the array except the last one.
- The inner loop finds the index of the minimum element in the unsorted portion of the array.
- After finding the minimum element, it swaps it with the first unsorted element.

##### 3. Output After Sorting:

- The program prints the sorted array after completing the selection sort process.

#### Complexity Analysis

##### • Time Complexity:

- Best Case:  $O(n^2)$  (the same as average and worst cases).
- Average Case:  $O(n^2)$
- Worst Case:  $O(n^2)$  when the array is sorted in reverse order.

- **Space Complexity:**  $O(1)$  since selection sort operates in place and uses a constant amount of additional space for variables.