

2025

DESINGN AND ANALYSIS OF ALGORITHM

LAB EXPERIMENT -2

KOLATHUR SURYA [CH.SC.U4CSE24224]

1) BUBBLE SORT

Code:

```
#include <stdio.h>

int main() {
    int n, i, j, temp;
    printf("Name: KOLATHUR SURYA\n");
    printf("Roll No: CH.SC.U4CSE24224\n\n");

    printf("Enter number of elements: ");
    scanf("%d", &n);

    int arr[n];

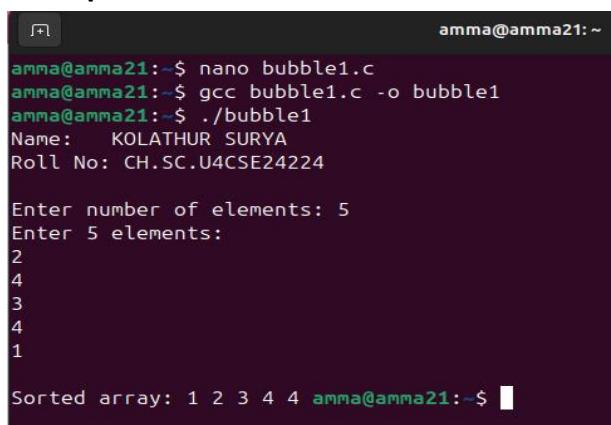
    printf("Enter %d elements:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    // Bubble Sort
    for (i = 0; i < n - 1; i++) {
        for (j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }

    printf("\nSorted array: ");
    for (i = 0; i < n; i++)
        printf("%d ", arr[i]);

    return 0;
}
```

Output:



The terminal window shows the following sequence of commands and output:

```
amma@amma21:~$ nano bubble1.c
amma@amma21:~$ gcc bubble1.c -o bubble1
amma@amma21:~$ ./bubble1
Name: KOLATHUR SURYA
Roll No: CH.SC.U4CSE24224

Enter number of elements: 5
Enter 5 elements:
2
4
3
4
1

Sorted array: 1 2 3 4 4
```

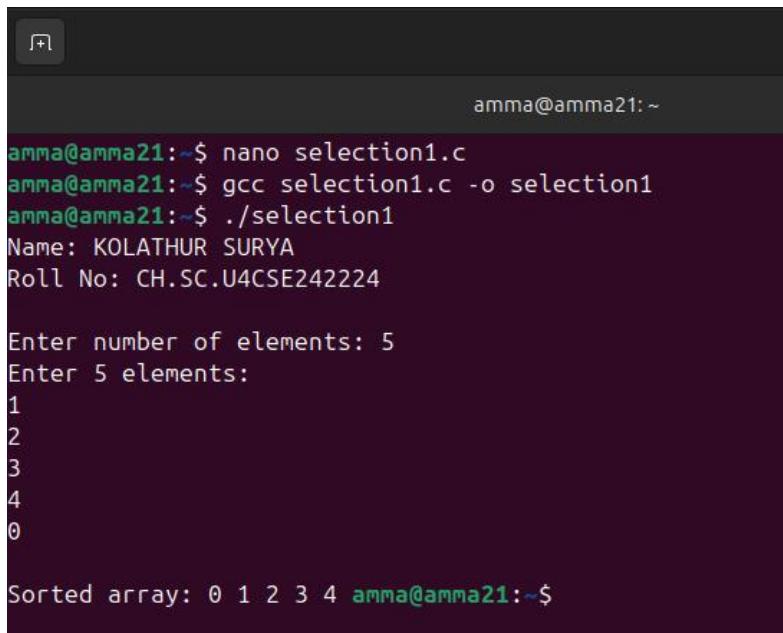
- Repeatedly compares adjacent elements and swaps them if they are in the wrong order.
- Each pass “bubbles” the largest element to the end.

2) SELECTION SORT

Code:

```
#include <stdio.h>
int main() {
    int n, i, j, minIndex, temp;
    printf("Name: KOLATHUR SURYA\n");
    printf("Roll No: CH.SC.U4CSE24224\n\n");
    printf("Enter number of elements: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter %d elements:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    // Selection Sort
    for (i = 0; i < n - 1; i++) {
        minIndex = i;
        for (j = i + 1; j < n; j++) {
            if (arr[j] < arr[minIndex]) {
                minIndex = j;
            }
        }
        temp = arr[minIndex];
        arr[minIndex] = arr[i];
        arr[i] = temp;
    }
    printf("\nSorted array: ");
    for (i = 0; i < n; i++)
        printf("%d ", arr[i]);
}
return 0;
```

OUTPUT:



The terminal window shows the following session:

```
amma@amma21:~$ nano selection1.c
amma@amma21:~$ gcc selection1.c -o selection1
amma@amma21:~$ ./selection1
Name: KOLATHUR SURYA
Roll No: CH.SC.U4CSE242224

Enter number of elements: 5
Enter 5 elements:
1
2
3
4
0

Sorted array: 0 1 2 3 4 amma@amma21:~$
```

- Selects the smallest element from the unsorted part and places it at the beginning.
- Continues selecting and placing until the whole array is sorted.

3) INSERTION SORT

CODE:

```
#include <stdio.h>

int main() {
    int n, i, j, key;

    printf("Name: KOLATHUR SURYA\n");
    printf("Roll No: CH.SC.U4CSE24224\n\n");

    printf("Enter number of elements: ");
    scanf("%d", &n);

    int arr[n];

    printf("Enter %d elements:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

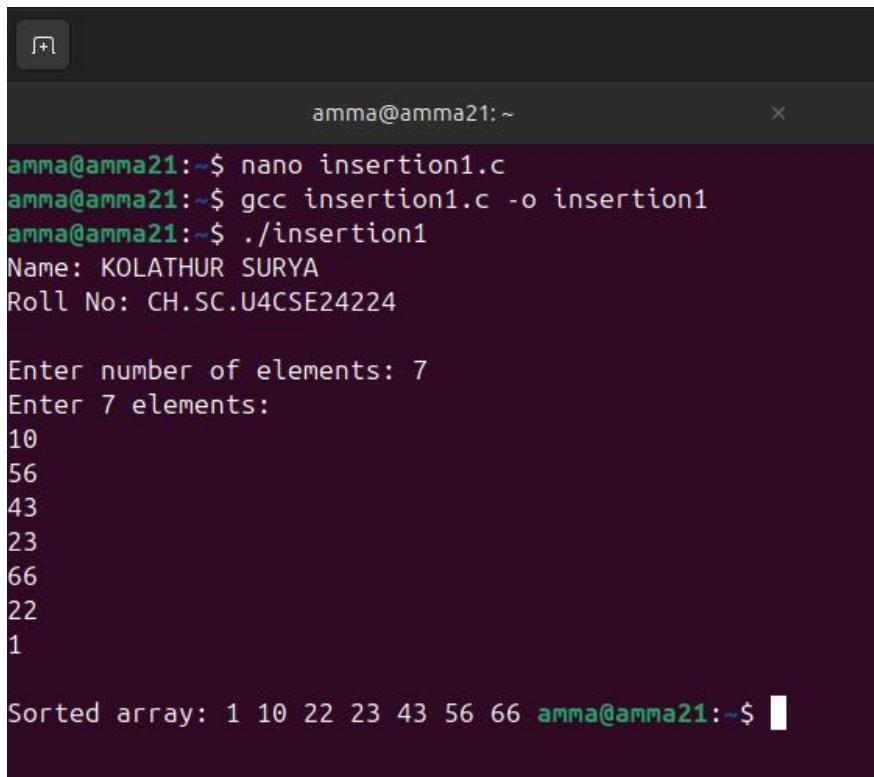
    // Insertion Sort
    for (i = 1; i < n; i++) {
        key = arr[i];
        j = i - 1;

        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key;
    }

    printf("\nSorted array: ");
    for (i = 0; i < n; i++)
        printf("%d ", arr[i]);

    return 0;
}
```

OUTPUT:



A terminal window showing the execution of a C program named `insertion1.c`. The terminal prompt is `amma@amma21:~`. The user enters the command `gcc insertion1.c -o insertion1`, compiles it, and runs the resulting executable `./insertion1`. The program outputs the user's name and roll number, then prompts for the number of elements and the elements themselves. Finally, it prints the sorted array.

```
amma@amma21:~$ nano insertion1.c
amma@amma21:~$ gcc insertion1.c -o insertion1
amma@amma21:~$ ./insertion1
Name: KOLATHUR SURYA
Roll No: CH.SC.U4CSE24224

Enter number of elements: 7
Enter 7 elements:
10
56
43
23
66
22
1

Sorted array: 1 10 22 23 43 56 66 amma@amma21:~$
```

- Builds the sorted list one item at a time by inserting each element into its correct position.
- Works great for small or nearly sorted arrays.

4) BUCKET SORT

CODE:

```
#include <stdio.h>

int main() {
    int n, i, j;

    printf("Name: KOLATHUR SURYA\n");
    printf("Roll No: CH.SC.U4CSE24224\n\n");

    printf("Enter number of elements: ");
    scanf("%d", &n);

    int arr[n];

    printf("Enter %d elements:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    // Find max value
    int max = arr[0];
    for (i = 1; i < n; i++) {
        if (arr[i] > max)
            max = arr[i];
    }

    // Create buckets
    int bucket[max + 1];

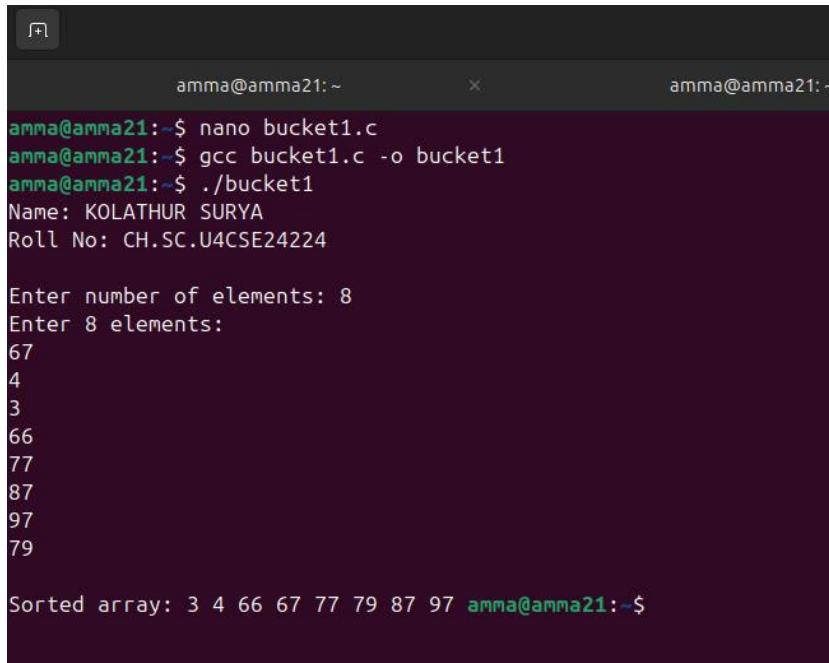
    for (i = 0; i <= max; i++)
        bucket[i] = 0;

    // Fill buckets
    for (i = 0; i < n; i++)
        bucket[arr[i]]++;

    printf("\nSorted array: ");
    for (i = 0; i <= max; i++)
        for (j = 0; j < bucket[i]; j++)
            printf("%d ", i);

    return 0;
}
```

OUTPUT:



The terminal window shows the following session:

```
amma@amma21:~$ nano bucket1.c
amma@amma21:~$ gcc bucket1.c -o bucket1
amma@amma21:~$ ./bucket1
Name: KOLATHUR SURYA
Roll No: CH.SC.U4CSE24224

Enter number of elements: 8
Enter 8 elements:
67
4
3
66
77
87
97
79

Sorted array: 3 4 66 67 77 79 87 97 amma@amma21:~$
```

- Divides elements into multiple buckets based on range.
- Sorts each bucket (usually using another algorithm) and merges them.

5)HEAP SORT

CODE:

```
#include <stdio.h>

void heapify(int arr[], int n, int i) {
    int largest = i;
    int left = 2 * i + 1;
    int right = 2 * i + 2;
    int temp;

    if (left < n && arr[left] > arr[largest])
        largest = left;

    if (right < n && arr[right] > arr[largest])
        largest = right;

    if (largest != i) {
        temp = arr[i];
        arr[i] = arr[largest];
        arr[largest] = temp;

        heapify(arr, n, largest);
    }
}

void heapSort(int arr[], int n) {
    int i, temp;

    for (i = n / 2 - 1; i >= 0; i--)
        heapify(arr, n, i);

    for (i = n - 1; i >= 0; i--) {
        temp = arr[0];
        arr[0] = arr[i];
        arr[i] = temp;

        heapify(arr, i, 0);
    }
}

int main() {
    int n, i;

    printf("Name: KOLATHUR SURYA\n");
    printf("Roll No: CH.SC.U4CSE24224\n\n");

    printf("Enter number of elements: ");
    scanf("%d", &n);

    int arr[n];

    printf("Enter %d elements:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    heapSort(arr, n);

    printf("\nSorted array: ");
    for (i = 0; i < n; i++)
        printf("%d ", arr[i]);

    return 0;
}
```

OUTPUT:

The screenshot shows a terminal window with two tabs. Both tabs have the title "amma@amma21:~". The left tab contains the command history and the output of the sorted array. The right tab is empty.

```
amma@amma21:~$ nano heap1.c
amma@amma21:~$ gcc heap1.c -o heap1
amma@amma21:~$ ./heap1
Name: KOLATHUR SURYA
Roll No: CH.SC.U4CSE24224

Enter number of elements: 5
Enter 5 elements:
23
27
28
26
21

Sorted array: 21 23 26 27 28 amma@amma21:~$
```

- Builds a max-heap/min-heap from the array.
- Repeatedly removes the root (largest/smallest) and rebuilds the heap to get a sorted list.