AMRITA VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY UNDER SECTION 3 OF UGC ACT, 1956
SCHOOL OF COMPUTING

Design s Analysis of Algorithm

CSE211

Date: 27/11/2025

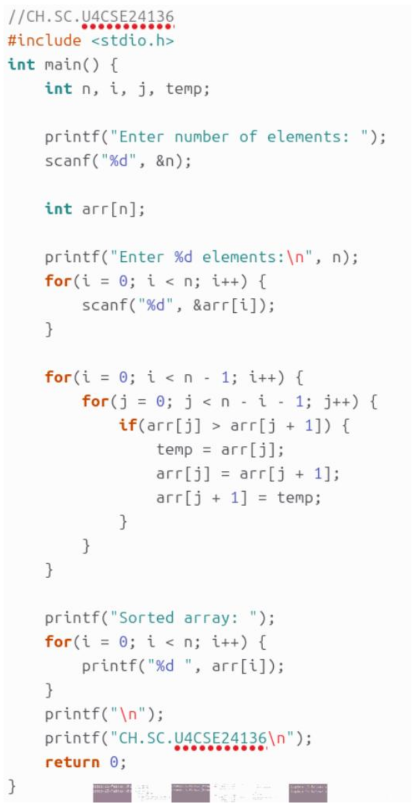
Week-2

CH.SC.U4CSE24136

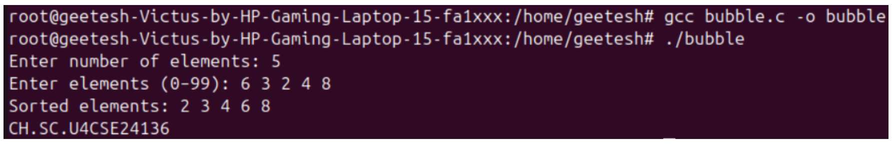
P. Geetesh

Bubble Sort

Code:



```c
//CH.SC.U4CSE24136
#include <stdio.h>
int main() {
    int n, i, j, temp;

    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]);
    }

    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("Sorted array: ");
    for(i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
    printf("CH.SC.U4CSE24136\n");
    return 0;
}
```



```
root@geetesh-Victus-by-HP-Gaming-Laptop-15-fa1xxx:/home/geetesh# gcc bubble.c -o bubble
root@geetesh-Victus-by-HP-Gaming-Laptop-15-fa1xxx:/home/geetesh# ./bubble
Enter number of elements: 5
Enter elements (0-99): 6 3 2 4 8
Sorted elements: 2 3 4 6 8
CH.SC.U4CSE24136
```

Space Complexity and it's Justification:

Time Complexity

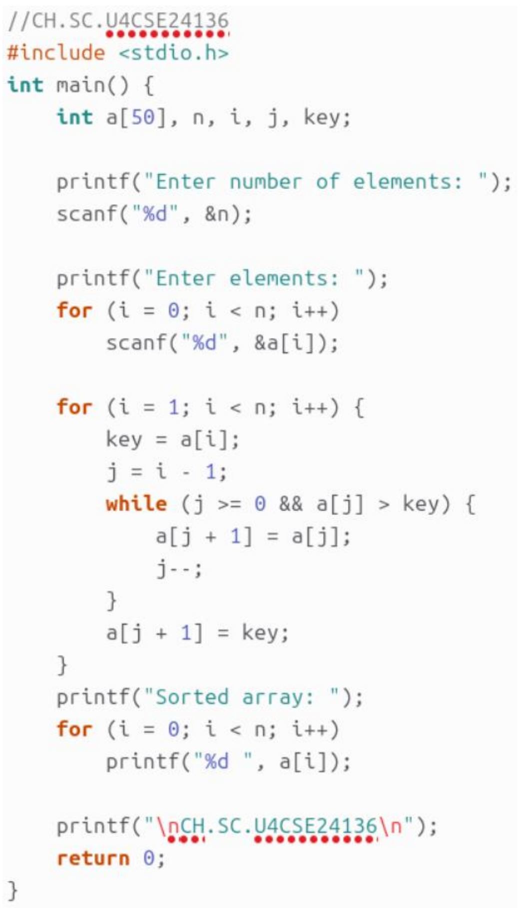
- Best case: O(n^2)
  → No optimization (no early break used).
- Average case: O(n^2)
- Worst case: O(n^2)
  → Nested loops compare all adjacent pairs.

Space Complexity

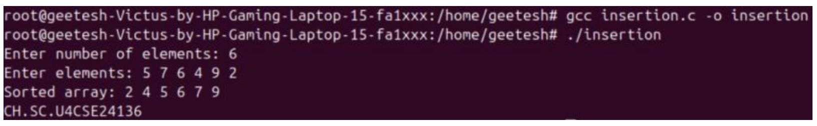
- O(1)
  → Only one temporary variable used (in-place sorting).

Insertion Sort

Code:



```c
//CH.SC.U4CSE24136
#include <stdio.h>
int main() {
    int a[50], n, i, j, key;

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

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

    for (i = 1; i < n; i++) {
        key = a[i];
        j = i - 1;
        while (j >= 0 && a[j] > key) {
            a[j + 1] = a[j];
            j--;
        }
        a[j + 1] = key;
    }
    printf("Sorted array: ");
    for (i = 0; i < n; i++)
        printf("%d ", a[i]);

    printf("\nCH.SC.U4CSE24136\n");
    return 0;
}
```



```
root@geetesh-Victus-by-HP-Gaming-Laptop-15-fa1xxx:/home/geetesh# gcc insertion.c -o insertion
root@geetesh-Victus-by-HP-Gaming-Laptop-15-fa1xxx:/home/geetesh# ./insertion
Enter number of elements: 6
Enter elements: 5 7 6 4 9 2
Sorted array: 2 4 5 6 7 9
CH.SC.U4CSE24136
```

Space Complexity and it's Justification:

Time Complexity

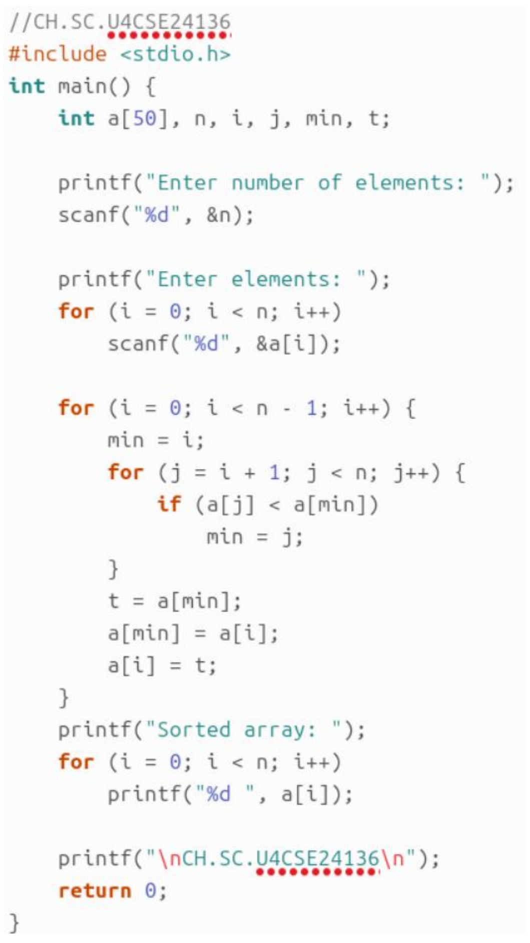
- Best: O(n)
  Justification:
  - Already sorted array → only one comparison per element
- Average / Worst: O(n^2)
  Justification:
  - Each element may shift through the entire sorted part

Space Complexity

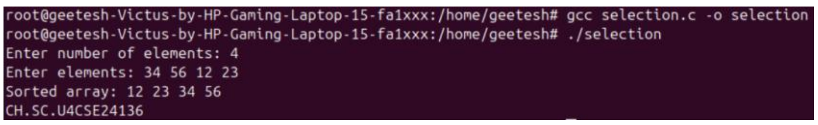
- O(1)
  Justification:
  - Uses only one temporary variable (key)

Selection Sort

Code:



```c
//CH.SC.U4CSE24136
#include <stdio.h>
int main() {
    int a[50], n, i, j, min, t;

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

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

    for (i = 0; i < n - 1; i++) {
        min = i;
        for (j = i + 1; j < n; j++) {
            if (a[j] < a[min])
                min = j;
        }
        t = a[min];
        a[min] = a[i];
        a[i] = t;
    }
    printf("Sorted array: ");
    for (i = 0; i < n; i++)
        printf("%d ", a[i]);

    printf("\nCH.SC.U4CSE24136\n");
    return 0;
}
```



```
root@geetesh-Victus-by-HP-Gaming-Laptop-15-fa1xxx:/home/geetesh# gcc selection.c -o selection
root@geetesh-Victus-by-HP-Gaming-Laptop-15-fa1xxx:/home/geetesh# ./selection
Enter number of elements: 4
Enter elements: 34 56 12 23
Sorted array: 12 23 34 56
CH.SC.U4CSE24136
```

Space Complexity and it's Justification:

Time Complexity

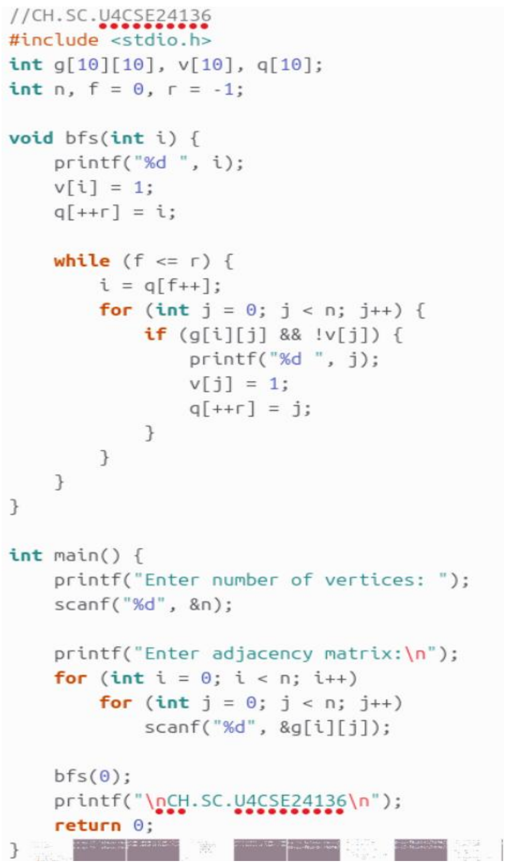
- **Best / Average / Worst: O(n^2)**
  Justification:
  - Always compares every element to find minimum
  - Number of comparisons does not depend on input order

Space Complexity

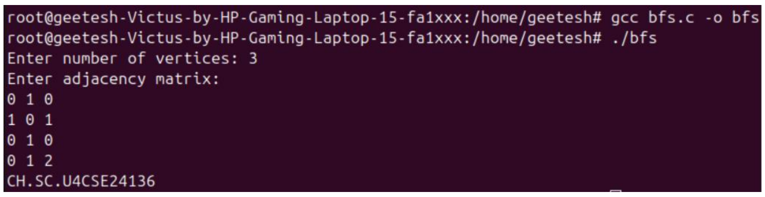
- **O(1)**
  Justification:
  - Sorting done **in-place**, no extra memory used

BFS

Code:



```c
//CH.SC.U4CSE24136
#include <stdio.h>
int g[10][10], v[10], q[10];
int n, f = 0, r = -1;

void bfs(int i) {
    printf("%d ", i);
    v[i] = 1;
    q[++r] = i;

    while (f <= r) {
        i = q[f++];
        for (int j = 0; j < n; j++) {
            if (g[i][j] && !v[j]) {
                printf("%d ", j);
                v[j] = 1;
                q[++r] = j;
            }
        }
    }
}

int main() {
    printf("Enter number of vertices: ");
    scanf("%d", &n);

    printf("Enter adjacency matrix:\n");
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            scanf("%d", &g[i][j]);

    bfs(0);
    printf("\nCH.SC.U4CSE24136\n");
    return 0;
}
```



```
root@geetesh-Victus-by-HP-Gaming-Laptop-15-fa1xxx:/home/geetesh# gcc bfs.c -o bfs
root@geetesh-Victus-by-HP-Gaming-Laptop-15-fa1xxx:/home/geetesh# ./bfs
Enter number of vertices: 3
Enter adjacency matrix:
0 1 0
1 0 1
0 1 0
0 1 2
CH.SC.U4CSE24136
```

<u>Space Complexity and it’s Justification:</u>

Time Complexity

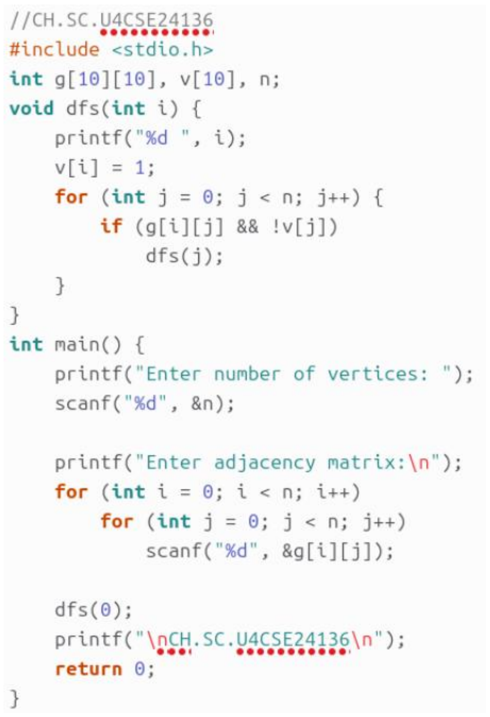
- **O(V^2)**
  Justification:
  - Uses **adjacency matrix**
  - For each vertex, all V vertices are checked

Space Complexity

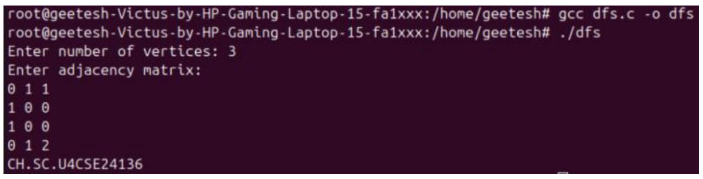
- **O(V)**
  Justification:
  - Queue array + visited array store at most V vertices

DFS

Code:



```c
//CH.SC.U4CSE24136
#include <stdio.h>
int g[10][10], v[10], n;
void dfs(int i) {
    printf("%d ", i);
    v[i] = 1;
    for (int j = 0; j < n; j++) {
        if (g[i][j] && !v[j])
            dfs(j);
    }
}
int main() {
    printf("Enter number of vertices: ");
    scanf("%d", &n);

    printf("Enter adjacency matrix:\n");
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            scanf("%d", &g[i][j]);

    dfs(0);
    printf("\nCH.SC.U4CSE24136\n");
    return 0;
}
```



```
root@geetesh-Victus-by-HP-Gaming-Laptop-15-fa1xxx:/home/geetesh# gcc dfs.c -o dfs
root@geetesh-Victus-by-HP-Gaming-Laptop-15-fa1xxx:/home/geetesh# ./dfs
Enter number of vertices: 3
Enter adjacency matrix:
0 1 1
1 0 0
1 0 0
0 1 2
CH.SC.U4CSE24136
```

<u>Space Complexity and it's Justification:</u>

Time Complexity

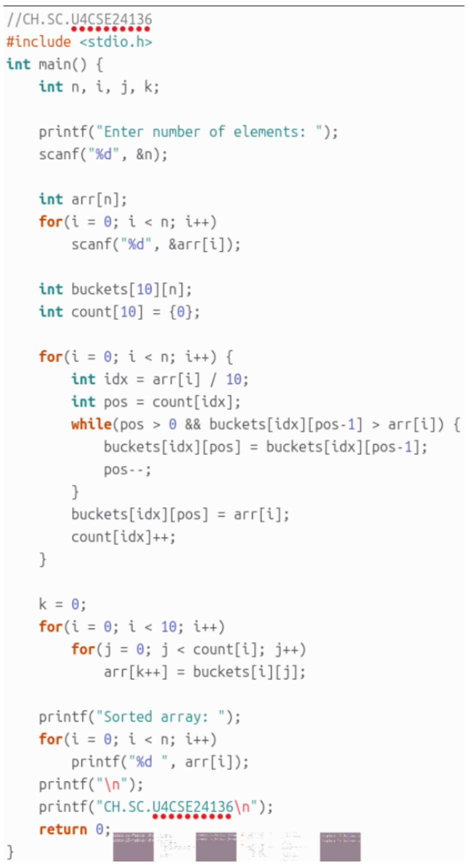
- O(V^2)
  Justification:
  - Adjacency matrix representation
  - Each DFS call scans all vertices

Space Complexity

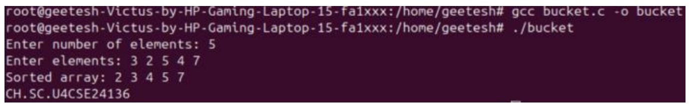
- O(V)
  Justification:
  - Recursive call stack + visited array

Bucket Sort

Code:



```c
//CH.SC.U4CSE24136
#include <stdio.h>
int main() {
    int n, i, j, k;

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

    int arr[n];
    for(i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    int buckets[10][n];
    int count[10] = {0};

    for(i = 0; i < n; i++) {
        int idx = arr[i] / 10;
        int pos = count[idx];
        while(pos > 0 && buckets[idx][pos-1] > arr[i]) {
            buckets[idx][pos] = buckets[idx][pos-1];
            pos--;
        }
        buckets[idx][pos] = arr[i];
        count[idx]++;
    }

    k = 0;
    for(i = 0; i < 10; i++)
        for(j = 0; j < count[i]; j++)
            arr[k++] = buckets[i][j];

    printf("Sorted array: ");
    for(i = 0; i < n; i++)
        printf("%d ", arr[i]);
    printf("\n");
    printf("CH.SC.U4CSE24136\n");
    return 0;
}
```



```
root@geetesh-Victus-by-HP-Gaming-Laptop-15-fa1xxx:/home/geetesh# gcc bucket.c -o bucket
root@geetesh-Victus-by-HP-Gaming-Laptop-15-fa1xxx:/home/geetesh# ./bucket
Enter number of elements: 5
Enter elements: 3 2 5 4 7
Sorted array: 2 3 4 5 7
CH.SC.U4CSE24136
```

Space Complexity and it's Justification:

Time Complexity

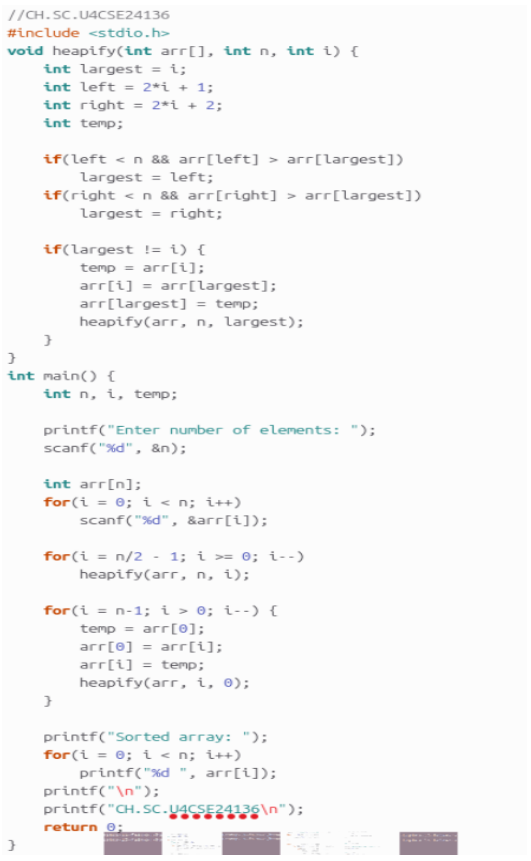
- Best case: O(n)
  → Elements uniformly distributed, minimal shifting inside buckets.
- Average case: O(n)
  → Each bucket has few elements; insertion is fast.
- Worst case: O(n^2)
  → All elements fall into one bucket → insertion sort behavior.

Space Complexity

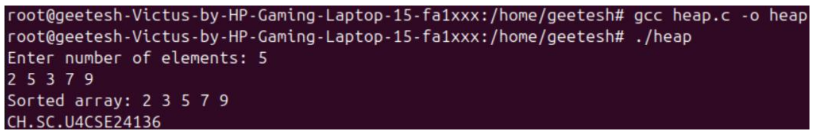
- O(n)
  → Buckets array buckets[10][n] + count array.

Heap Sort

Code:



```c
//CH.SC.U4CSE24136
#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);
    }
}
int main() {
    int n, i, temp;

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

    int arr[n];
    for(i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    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);
    }

    printf("Sorted array: ");
    for(i = 0; i < n; i++)
        printf("%d ", arr[i]);
    printf("\n");
    printf("CH.SC.U4CSE24136\n");
    return 0;
}
```



```
root@geetesh-Victus-by-HP-Gaming-Laptop-15-fa1xxx:/home/geetesh# gcc heap.c -o heap
root@geetesh-Victus-by-HP-Gaming-Laptop-15-fa1xxx:/home/geetesh# ./heap
Enter number of elements: 5
2 5 3 7 9
Sorted array: 2 3 5 7 9
CH.SC.U4CSE24136
```

Space Complexity and it's Justification:

Time Complexity

- Best case: O(n log n)
- Average case: O(n log n)
- Worst case: O(n log n)
  → Heap construction O(n) + n heapify operations O(log n).

Space Complexity

- O(log n)
  → Due to recursive heapify() call stack.