23CSE211-DESIGN AND ANALYSIS OF ALGORITHM

LAB – 2

1.Bubble sort

Code:

```c
#include <stdio.h>
void bubbleSort(int arr[], int n) {
  int i, j, temp;
    int swapped;
    for (i = 0; i < n - 1; i++) {
        swapped = 0;
        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;
                swapped = 1;
            }
        }
        if (swapped == 0)
           break;
    }
}

void printArray(int arr[], int size) {
    int i;
    for (i = 0; i < size; i++)
        printf("%d ", arr[i]);
    printf("\n");
}
int main() {
    int arr[] = {64, 34, 25, 12, 22, 11, 90};
    int n = sizeof(arr) / sizeof(arr[0]);
    printf("array before bubble sort:");
    printArray(arr, n);
    bubbleSort(arr, n);
    printf("array after bubble sort:");
    printArray(arr, n);
    return 0;
}
```

Output:

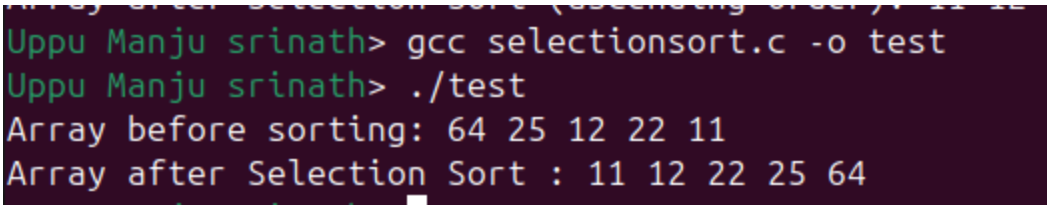
```
Uppu Manju srinath> gcc bubbleSort.c -o test
Uppu Manju srinath> ./test
array before bubble sort:64 34 25 12 22 11 90
array after bubble sort:11 12 22 25 34 64 90
Uppu Manju srinath>
```

2.Selection sort

Code:

```c
#include <stdio.h>
void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}
void selectionSort(int arr[], int n) {
    int i, j, min_idx;
    for (i = 0; i < n - 1; i++) {
        min_idx = i;
        for (j = i + 1; j < n; j++) {
            if (arr[j] < arr[min_idx]) {
                min_idx = j;
            }
        }
        if (min_idx != i) {
            swap(&arr[min_idx], &arr[i]);
        }
    }
}
void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}
int main() {
    int data[] = {64, 25, 12, 22, 11};
    int n = sizeof(data) / sizeof(data[0]);
    printf("Array before sorting: ");
    printArray(data, n);
    selectionSort(data, n);
    printf("Array after Selection Sort : ");
    printArray(data, n);
    return 0;
}
```

Output:

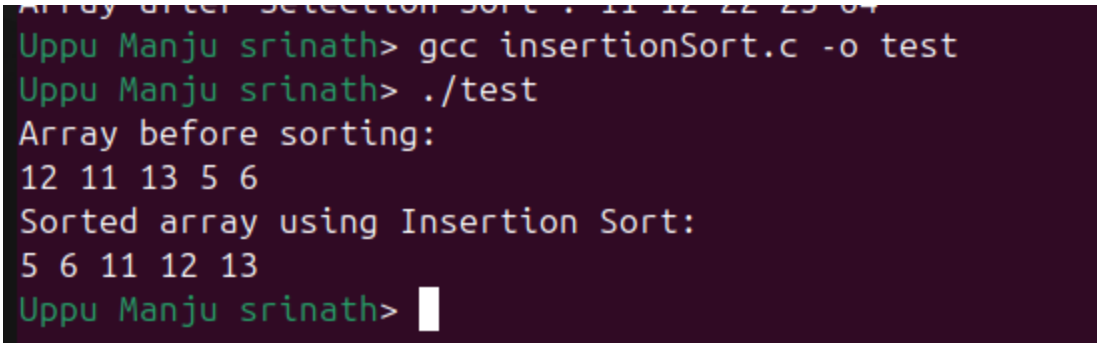


3.Insertion sort

Code:

```c
#include <stdio.h>
void insertionSort(int arr[], int n) {
    int i, key, j;
    for (i = 1; i < n; i++) {
        key = arr[i];
        j = i - 1;
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}
void printArray(int arr[], int size) {
    for (int k = 0; k < size; k++) {
        printf("%d ", arr[k]);
    }
    printf("\n");
}
int main() {
    int data[] = {12, 11, 13, 5, 6};
    int n = sizeof(data) / sizeof(data[0]);
    printf("Array before sorting:\n");
    printArray(data, n);
    insertionSort(data, n);
    printf("Sorted array using Insertion Sort:\n");
    printArray(data, n);
    return 0;
}
```

Output:



4.Bucket sort

Code:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
typedef struct Node {
    float data;
    struct Node* next;
} Node;

void insertSorted(Node** head_ref, float new_data) {
    Node* new_node = (Node*)malloc(sizeof(Node));
    new_node->data = new_data;
    new_node->next = NULL;

    if (*head_ref == NULL || (*head_ref)->data >= new_data) {
        new_node->next = *head_ref;
        *head_ref = new_node;
    } else {
        Node* current = *head_ref;
        while (current->next != NULL && current->next->data < new_data) {
            current = current->next;
        }
        new_node->next = current->next;
        current->next = new_node;
    }
}
void bucketSort(float arr[], int n) {
    Node* buckets[n];
    for (int i = 0; i < n; i++) {
        buckets[i] = NULL;
    }
    for (int i = 0; i < n; i++) {
        int bucket_index = (int)(n * arr[i]);
        insertSorted(&buckets[bucket_index], arr[i]);
    }
```

```
    int index = 0;
    for (int i = 0; i < n; i++) {
        Node* current = buckets[i];
        while (current != NULL) {
            arr[index++] = current->data;
            Node* temp = current;
            current = current->next;
            free(temp);
        }
    }
}
void printArray(float arr[], int size) {
    for (int i = 0; i < size; i++) {
        printf("%.4f ", arr[i]);
    }
    printf("\n");
}
int main() {
    float arr[] = {0.897, 0.565, 0.656, 0.123, 0.665, 0.343};
    int n = sizeof(arr) / sizeof(arr[0]);
    printf("Original array:\n");
    printArray(arr, n);
    bucketSort(arr, n);
    printf("Sorted array:\n");
    printArray(arr, n);
    return 0;
}
```

Output:

```
Uppu Manju srinath> gcc bucketsort.c -o test
Uppu Manju srinath> ./test
Original array:
0.8970 0.5650 0.6560 0.1230 0.6650 0.3430
Sorted array:
0.1230 0.3430 0.5650 0.6560 0.6650 0.8970
Uppu Manju srinath>
```

5.Heap sort

Code:

```c
#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);
    }
}
void printArray(int arr[], int n) {
    for (int i = 0; i < n; ++i)
        printf("%d ", arr[i]);
    printf("\n");
}
```

```c
int main() {
    int arr[] = {12, 11, 13, 5, 6, 7};
    int n = sizeof(arr) / sizeof(arr[0]);
    printf("Original array:\n");
    printArray(arr, n);
    heapSort(arr, n);
    printf("Sorted array:\n");
    printArray(arr, n);

    return 0;
}
```

Output:

```
Uppu Manju srinath> gcc heapsort.c -o test
Uppu Manju srinath> ./test
Original array:
12 11 13 5 6 7
Sorted array:
5 6 7 11 12 13
Uppu Manju srinath>
```