Design and Analysis of Algorithms

Week-2 Assignment

1. BUBBLE SORT:

PROGRAM:

```
#include <stdio.h> int
main() {

int n;

printf("ENTER NO.OF.ELEMENTS:");

scanf("%d",&n); int
arr[n];

printf("ENTER THE VALUES:");

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

}

   for (int i = 0; i < n - 1; i++) {

      for (int j = 0; j < n - i - 1; j++) { if
         (arr[j] > arr[j + 1]) {

            int temp = arr[j]; arr[j] =
            arr[j + 1]; arr[j + 1] =
            temp;

         }

      }

   }

   printf("Sorted array: "); for (int i
   = 0; i < n; i++) {

      printf("\t%d\t,",arr[i]);

   }

   return 0;
```

}

OUTPUT:

```
amma@amma11:~$ gcc bubble_sort.c -o bubble_sort
amma@amma11:~$ ./bubble_sort
enter element 0: 6
enter element 1: 9
enter element 2: 8
enter element 3: 1
enter element 4: 3
1
3
6
8
9
```

2. Insertion Sort :

PROGRAM:

#include <stdio.h> int main() {

int n;

printf("ENTER NO.OF ELEMENTS");

scanf("%d",&n); int arr[n];

printf("ENTER THE ELEMENTS:");

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

}

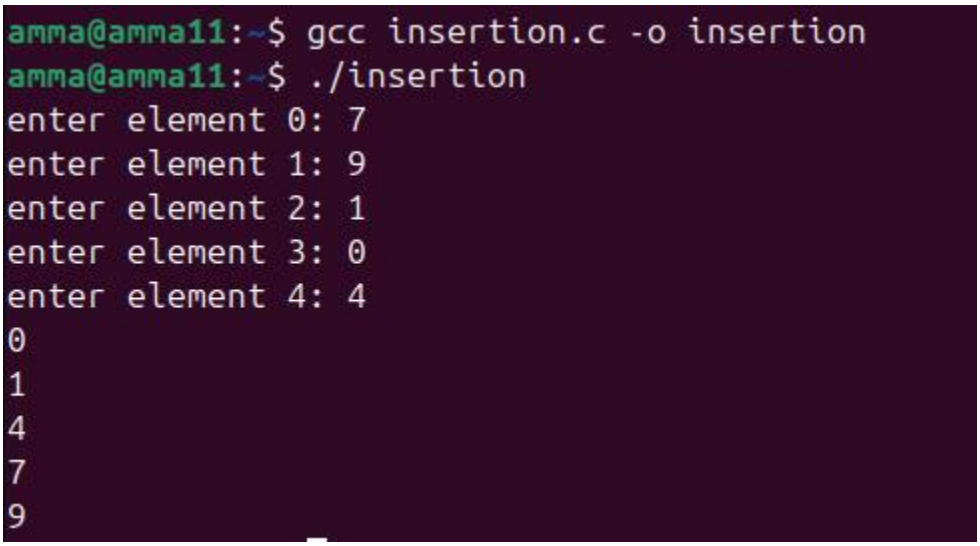
for (int i = 1; i < n; i++) { int key = arr[i];

int j = i - 1;

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

```
j--;
}
arr[j + 1] = key;
}
printf("Sorted array: "); for (int i = 0; i < n; i++) {
printf("%d ", arr[i]);
}
return 0;
}
```

OUTPUT:



3. Selection Sort

PROGRAM:

```
int main() {
int n;
```

```
printf("ENTER NO. OF ELEMENTS: ");
scanf("%d", &n); int arr[n];
printf("ENTER THE VALUES: ");
for (int i = 0; i < n; i++) { scanf("%d", &arr[i]);
}
for (int i = 0; i < n - 1; i++) { int minIndex = i;
for (int j = i + 1; j < n; j++) { if (arr[j] < arr[minIndex]) {
minIndex = j;
}
}
int temp = arr[i];
arr[i] = arr[minIndex]; arr[minIndex] = temp;
}
printf("Sorted array: "); for (int i = 0; i < n; i++) {
printf("%d ", arr[i]);
}
return 0;
}
```

0UTPUT:

```
amma@amma11:~$ gcc selection_sort.c -o selection_sort
amma@amma11:~$ ./selection_sort
enter element 0: 9
enter element 1: 1
enter element 2: 0
enter element 3: 3
enter element 4: 6
0
1
3
6
9
```

4. Bucket Sort

PROGRAM:

```
#include <stdio.h>

int main() { int n;


printf("ENTER NO. OF ELEMENTS: ");


scanf("%d", &n); int arr[n];


printf("ENTER THE VALUES (0 to 100): ");


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


}
```

```
int bucket[101] = {0}; for (int i = 0; i < n; i++) {

bucket[arr[i]]++;

}

printf("Sorted array: ");

for (int i = 0; i <= 100; i++) { while (bucket[i] > 0) {

printf("%d ", i);

bucket[i]--;
}
}
return 0;
}
```

OUTPUT:

```
amma@amma11:~$ gcc bucket_sort.c -o bucket_sort
amma@amma11:~$ ./bucket_sort
enter element 0: 8
enter element 1: 2
enter element 2: 0
enter element 3: 1
enter element 4: 7
0
1
2
7
8
```

5. Heap Sort

PROGRAM:

#include<stdio.h> int main() {

int n;

printf("ENTER NO. OF ELEMENTS: ");

scanf("%d", &n); int arr[n];

printf("ENTER THE VALUES: ");

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

}

for (int i = 1; i < n; i++) { int child = i;

while (child > 0) {

int parent = (child - 1) / 2;

if (arr[parent] < arr[child]) { int temp = arr[parent]; arr[parent] = arr[child]; arr[child] = temp;

child = parent;

```
} else {
break;
}
}
}
for (int i = n - 1; i > 0; i--) { int temp = arr[0];
arr[0] = arr[i]; arr[i] = temp; int parent = 0; while (1) {
int left = 2 * parent + 1; int right = 2 * parent + 2; int largest = parent;
if (left < i && arr[left] > arr[largest]) largest = left;
if (right < i && arr[right] > arr[largest])
largest = right;
if (largest != parent) {
int temp2 = arr[parent]; arr[parent] = arr[largest]; arr[largest] = temp2; parent =
largest;
} else {
break;
}
}
}
printf("Sorted array: "); for (int i = 0; i < n; i++) {
printf("%d ", arr[i]);
}
```

return 0;

}

OUTPUT:

```
amma@amma11:~$ gcc heap_sort.c -o heap_sort
amma@amma11:~$ ./heap_sort
enter element 0: 9
enter element 1: 1
enter element 2: 8
enter element 3: 3
enter element 4: 6
1
3
6
8
9
```

6. BFS

PROGRAM:

```
#include <stdio.h>


#define MAX 100


void bfs(int graph[MAX][MAX], int n, int start) {
    int queue[MAX];
    int front = 0, rear = 0;
    int visited[MAX] = {0};


    visited[start] = 1;
    queue[rear++] = start;


    while (front < rear) {
        int node = queue[front++];
```

```
        printf("%d ", node);

        for (int i = 0; i < n; i++) {
            if (graph[node][i] == 1 && !visited[i]) {
                visited[i] = 1;
                queue[rear++] = i;
            }
        }
    }
}

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

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

```
    int start;
    printf("Enter starting node: ");
    scanf("%d", &start);

    printf("BFS Traversal: ");
    bfs(graph, n, start);

    return 0;
}
```

OUTPUT:

```
BFS Traversal: 2 root@amma52:/home/amma/Documents# ./bfs
Enter number of nodes: 3
Enter adjacency matrix:
0 1 0 1 1 1 1 1 1
Enter starting node: 0

BFS Traversal: 0 1 2 root@amma52:/home/amma/Documents#
```

7. DFS

PROGRAM :

```
#include <stdio.h>

#define MAX 100

int visited[MAX] = {0};
```

```
void dfs(int graph[MAX][MAX], int n, int node) {
    printf("%d ", node);
    visited[node] = 1;

    for (int i = 0; i < n; i++) {
        if (graph[node][i] == 1 && !visited[i]) {
            dfs(graph, n, i);
        }
    }
}

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

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

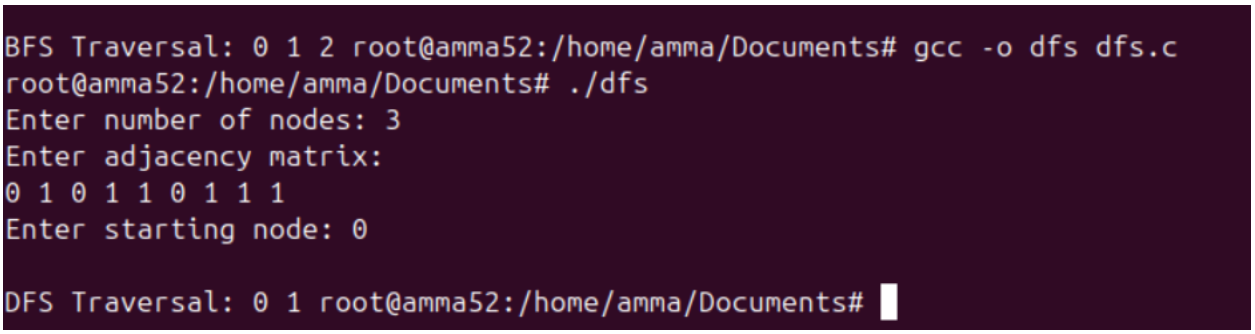
```
    }

    int start;
    printf("Enter starting node: ");
    scanf("%d", &start);

    printf("DFS Traversal: ");
    dfs(graph, n, start);

    return 0;
}
```

OUTPUT:



NAME: TEJA PRATHAP VARMA

ROLL NUMBER: CH.SC.U4CSE24117