

CH.SC.U4CSE24161

Data Analytics Algorithm

Task -2

Dhusyanth R

CH.SC.U4CSE24161

CSE 2nd YEAR

1.)Bubble Sort:

Code:

```
1 #include<stdlib.h>
2 #include<stdio.h>
3
4 int main(){
5     int n;
6     scanf("%d",&n);
7     int arr[n];
8     for(int i=0;i<n;i++){
9         scanf("%d",&arr[i]);
10    }
11    int temp;
12    for(int i=0;i<n-1;i++){
13        for(int j=0;j<n-i-1;j++){
14            if(arr[j]>arr[j+1]){
15                temp=arr[j];
16                arr[j]=arr[j+1];
17                arr[j+1]=temp;
18            }
19        }
20    }
21
22    for(int i=0;i<n;i++){
23        printf("%d\n",arr[i]);
24    }
25
26    return 0;
27 }
```

Output:

```
amma@amma55:~$ gedit sort.c
amma@amma55:~$ gcc -o sort sort.c
amma@amma55:~$ ./sort
5
1
4
2
5
3
12453amma@amma55:~$ gedit sort.c
amma@amma55:~$ gcc -o sort sort.c
amma@amma55:~$ ./sort
5
1
6
2
7
3
1
2
3
6
7
amma@amma55:~$ gedit sort.c
```

2.)Insertion Sort:

```
1 #include<stdlib.h>
2 #include<stdio.h>
3
4 int main(){
5     int n;
6     scanf("%d",&n);
7     int arr[n];
8     for(int i=0;i<n;i++){
9         scanf("%d",&arr[i]);
10    }
11    int key;
12    for(int i=1;i<n;i++){
13        int key=arr[i];
14        int j=i-1;
15
16        while(j>=0 && arr[j]>key){
17            arr[j+1]=arr[j];
18            j--;
19        }
20        arr[j+1]=key;
21    }
22
23    for(int i=0;i<n;i++){
24        printf("%d\n",arr[i]);
25    }
26    return 0;
27 }
```

Output:

```
amma@amma55:~$ ./insertion_sort
5
2
6
8
4
7
2
4
6
7
8
```

3.)Selection Sort:

```
1 #include <stdio.h>
2 void selectionSort(int arr[], int n) {
3     int i, j, minIndex, temp;
4     for (i = 0; i < n - 1; i++) {
5         minIndex = i;
6         for (j = i + 1; j < n; j++) {
7             if (arr[j] < arr[minIndex]) {
8                 minIndex = j;
9             }
10        }
11        if (minIndex != i) {
12            temp = arr[i];
13            arr[i] = arr[minIndex];
14            arr[minIndex] = temp;
15        }
16    }
17 }
18 void printArray(int arr[], int size) {
19     for (int i = 0; i < size; i++) {
20         printf("%d ", arr[i]);
21     }
22     printf("\n");
23 }
24
25 int main() {
26     int arr[] = {64, 25, 12, 22, 11};
27     int n = sizeof(arr) / sizeof(arr[0]);
28     printf("Original array: \n");
29     printArray(arr, n);
30     selectionSort(arr, n);
31     printf("Sorted array: \n");
32     printArray(arr, n);
33     return 0;
34 }
35
```

Output:

```
dhusyanth@Ubuntu:~$ gedit selection_sort.c

(gedit:7366): tepl-WARNING **: 15:52:43.647: Failed to load metadata: Err
ine 1 char 1: Document was empty or contained only whitespace
dhusyanth@Ubuntu:~$ gcc -o selection_sort sleection_sort.c
cc1: fatal error: sleection_sort.c: No such file or directory
compilation terminated.
dhusyanth@Ubuntu:~$ gcc -o selection_sort selection_sort.c
dhusyanth@Ubuntu:~$ .\selection_sort.c
.selection_sort.c: command not found
dhusyanth@Ubuntu:~$ .\selection_sort
.selection_sort: command not found
dhusyanth@Ubuntu:~$ ./selection_sort
Original array:
64 25 12 22 11
Sorted array:
11 12 22 25 64
```

4.)Bucket Sort:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #define BUCKET_COUNT 10
4 void insertionSort(float arr[], int n) {
5     int i, j;
6     float temp;
7     for (i = 1; i < n; i++) {
8         temp = arr[i];
9         j = i - 1;
10        while (j >= 0 && arr[j] > temp) {
11            arr[j + 1] = arr[j];
12            j--;
13        }
14        arr[j + 1] = temp;
15    }
16}
17 void bucketSort(float arr[], int n) {
18     float buckets[BUCKET_COUNT][n];
19     int bucketSizes[BUCKET_COUNT]; // To track the size of each
20
21     for (int i = 0; i < BUCKET_COUNT; i++) {
22         bucketSizes[i] = 0;
23     }
24     for (int i = 0; i < n; i++) {
25         int bucketIndex = arr[i] * BUCKET_COUNT; // Get the index
26         buckets[bucketIndex][bucketSizes[bucketIndex]] = arr[i];
27         bucketSizes[bucketIndex]++;
28     }
29     for (int i = 0; i < BUCKET_COUNT; i++) {
30         if (bucketSizes[i] > 0) {
31             insertionSort(buckets[i], bucketSizes[i]);
32         }
33     }
34     int index = 0;
35     for (int i = 0; i < BUCKET_COUNT; i++) {
36         for (int j = 0; j < bucketSizes[i]; j++) {
37             arr[index++] = buckets[i][j];
38         }
39     }
40 }
```

```
41 void printArray(float arr[], int n) {
42     for (int i = 0; i < n; i++) {
43         printf("%f ", arr[i]);
44     }
45     printf("\n");
46 }
47 int main() {
48     float arr[] = {0.42, 0.32, 0.53, 0.87, 0.12, 0.43, 0.34};
49     int n = sizeof(arr) / sizeof(arr[0]);
50     printf("Original array: \n");
51     printArray(arr, n);
52     bucketSort(arr, n);
53     printf("Sorted array: \n");
54     printArray(arr, n);
55
56     return 0;
57 }
```

Output:

```
dhusyanth@Ubuntu:~$ ./bucket_sort
Original array:
0.420000 0.320000 0.530000 0.870000 0.120000 0.430000 0.340000
Sorted array:
0.120000 0.320000 0.340000 0.420000 0.430000 0.530000 0.870000
```

5.)Heap Sort:

```

1 #include <stdio.h>
2 void heapify(int arr[], int n, int i) {
3     int largest = i;
4     int left = 2 * i + 1;
5     int right = 2 * i + 2;
6     if (left < n && arr[left] > arr[largest]) {
7         largest = left;
8     }
9     if (right < n && arr[right] > arr[largest]) {
10        largest = right;
11    }
12    if (largest != i) {
13        int temp = arr[i];
14        arr[i] = arr[largest];
15        arr[largest] = temp;
16        heapify(arr, n, largest);
17    }
18 }
19 void heapSort(int arr[], int n) {
20     for (int i = n / 2 - 1; i >= 0; i--) {
21         heapify(arr, n, i);
22     }
23     for (int i = n - 1; i > 0; i--) {
24         int temp = arr[0];
25         arr[0] = arr[i];
26         arr[i] = temp;
27         heapify(arr, i, 0);
28     }
29 }
30 void printArray(int arr[], int n) {
31     for (int i = 0; i < n; i++) {
32         printf("%d ", arr[i]);
33     }
34     printf("\n");
35 }
36
37 int main() {
38     int arr[] = {4, 10, 3, 5, 1};
39     int n = sizeof(arr) / sizeof(arr[0]);
40
41     printf("Original array: \n");
42     printArray(arr, n);
43 }
```

```
6
7 int main() {
8     int arr[] = {4, 10, 3, 5, 1};
9     int n = sizeof(arr) / sizeof(arr[0]);
0
1     printf("Original array: \n");
2     printArray(arr, n);
3
4     heapSort(arr, n);
5
6     printf("Sorted array: \n");
7     printArray(arr, n);
8
9     return 0;
0 }
```

Output:

```
dhusyanth@Ubuntu:~$ gedit heap_sort.c
dhusyanth@Ubuntu:~$ gcc -o heap_sort heap_sort.c
dhusyanth@Ubuntu:~$ ./heap_sort
Original array:
4 10 3 5 1
Sorted array:
1 3 4 5 10
dhusyanth@Ubuntu:~$ █
```

6.)Bfs:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #define MAX_NODES 100
4 typedef struct Queue {
5     int items[MAX_NODES];
6     int front, rear;
7 } Queue;
8 void initQueue(Queue* q) {
9     q->front = -1;
10    q->rear = -1;
11 }
12 int isEmpty(Queue* q) {
13     return q->front == -1;
14 }
15 void enqueue(Queue* q, int value) {
16     if (q->rear == MAX_NODES - 1)
17         printf("Queue is full\n");
18     else {
19         if (q->front == -1)
20             q->front = 0;
21         q->rear++;
22         q->items[q->rear] = value;
23     }
24 }
25 int dequeue(Queue* q) {
26     if (isEmpty(q)) {
27         printf("Queue is empty\n");
28         return -1;
29     }
30     int value = q->items[q->front];
31     q->front++;
32     if (q->front > q->rear) {
33         q->front = q->rear = -1;
34     }
35     return value;
36 }
37 void BFS(int graph[MAX_NODES][MAX_NODES], int start, int n) {
38     Queue q;
39     initQueue(&q);
40 }
```

```

40
41     int visited[MAX_NODES] = {0};
42     visited[start] = 1;
43     enqueue(&q, start);
44     printf("BFS Traversal starting from node %d:\n", start);
45     while (!isEmpty(&q)) {
46         int current = dequeue(&q);
47         printf("%d ", current);
48         for (int i = 0; i < n; i++) {
49             if (graph[current][i] == 1 && !visited[i]) {
50                 visited[i] = 1;
51                 enqueue(&q, i);
52             }
53         }
54     }
55     printf("\n");
56 }
57
58 int main() {
59     int n = 6;
60     int graph[MAX_NODES][MAX_NODES] = {
61         {0, 1, 1, 0, 0, 0}, // Node 0 (A)
62         {1, 0, 0, 1, 1, 0}, // Node 1 (B)
63         {1, 0, 0, 0, 0, 0}, // Node 2 (C)
64         {0, 1, 0, 0, 0, 1}, // Node 3 (D)
65         {0, 1, 0, 0, 0, 0}, // Node 4 (E)
66         {0, 0, 0, 1, 0, 0} // Node 5 (F)
67     };
68     BFS(graph, 0, n);
69
70     return 0;
71 }

```

Output:

7.)Dfs:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #define MAX_NODES 100
4 typedef struct Stack {
5     int items[MAX_NODES];
6     int top;
7 } Stack;
8 void initStack(Stack* s) {
9     s->top = -1;
10 }
11 int isEmpty(Stack* s) {
12     return s->top == -1;
13 }
14 void push(Stack* s, int value) {
15     if (s->top == MAX_NODES - 1) {
16         printf("Stack overflow\n");
17     } else {
18         s->items[+(s->top)] = value;
19     }
20 }
21 int pop(Stack* s) {
22     if (isEmpty(s)) {
23         printf("Stack underflow\n");
24         return -1;
25     }
26     return s->items[s->top--];
27 }
28 void DFS(int graph[MAX_NODES][MAX_NODES], int start, int n) {
29     Stack s;
30     initStack(&s);
31
32     int visited[MAX_NODES] = {0};
33     push(&s, start);
34     visited[start] = 1;
35     printf("DFS Traversal starting from node %d:\n", start);
36     while (!isEmpty(&s)) {
37         int current = pop(&s);
38         printf("%d ", current);
39         for (int i = 0; i < n; i++) {
40             if (graph[current][i] == 1 && !visited[i]) {
41                 visited[i] = 1;
42                 push(&s, i);
43             }
44         }
45     }
46 }
```

```
45     }
46     printf("\n");
47 }
48
49 int main() {
50     int n = 6;
51     int graph[MAX_NODES][MAX_NODES] = {
52         {0, 1, 1, 0, 0, 0}, // Node 0 (A)
53         {1, 0, 0, 1, 1, 0}, // Node 1 (B)
54         {1, 0, 0, 0, 0, 0}, // Node 2 (C)
55         {0, 1, 0, 0, 0, 1}, // Node 3 (D)
56         {0, 1, 0, 0, 0, 0}, // Node 4 (E)
57         {0, 0, 0, 1, 0, 0} // Node 5 (F)
58     };
59     DFS(graph, 0, n);
60
61     return 0;
62 }
63
```

Output:

```
dhusyanth@Ubuntu:~$ gedit dfs.c
dhusyanth@Ubuntu:~$ gcc -o dfs dfs.c
dhusyanth@Ubuntu:~$ ./dfs
bash: ./: Is a directory
dhusyanth@Ubuntu:~$ ./dfs
DFS Traversal starting from node 0:
0 2 1 4 3 5
dhusyanth@Ubuntu:~$
```