Design & Analysis of
Algorithms
Pranav N
CH.SC.U4CSE24236

1.BFS:~
CODE:

```
#include <stdio.h>
#define MAX 10
int graph[MAX][MAX];   // Adjacency matrix
int visited[MAX];      // Visited array
int queue[MAX];        // Simple queue for BFS
int front = -1, rear = -1;
int n;                 // Number of vertices
void enqueue(int v) {
   if (rear == MAX - 1) return; // Queue full (not handled for simplicity)
   if (front == -1) front = 0;
   rear++;
   queue[rear] = v;
}
int dequeue() {
   if (front == -1 || front > rear) return -1; // Queue empty
   int v = queue[front];
   front++;
   return v;
}
int isEmpty() {
   if (front == -1 || front > rear) return 1;
   return 0;
}
void BFS(int start) {
   // Initialize visited array to 0
   for (int i = 0; i < n; i++) {
      visited[i] = 0;
   }
   visited[start] = 1;
   enqueue(start);
   printf("BFS Traversal: ");
   while (!isEmpty()) {
      int v = dequeue();
      printf("%d ", v);
      for (int i = 0; i < n; i++) {
         if (graph[v][i] == 1 && visited[i] == 0) {
            visited[i] = 1;
            enqueue(i);
         }
      }
   }
```

```
  printf("\n");
}

int main() {
  printf("Enter number of vertices: ");
  scanf("%d", &n);
  printf("Enter adjacency matrix (%d x %d):\n", n, 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 vertex for BFS: ");
  scanf("%d", &start);
  BFS(start);
  return 0;
}
```

OUTPUT:

```
cd "/Users/pranavneelakandan/DAA/DAA/" && gcc BFS.c -o BFS && "/Users/pranavneelakandan/DAA/DAA/"BFS
pranavneelakandan@Pranavs-MacBook-Air DAA % cd "/Users/pranavneelakandan/DAA/DAA/" && gcc BFS.c -o BFS && "/User
s/pranavneelakandan/DAA/DAA/"BFS
Enter number of vertices: 5
Enter adjacency matrix (5 x 5):
0 1 1 0 1
0 0 1 1 1
1 1 1 0 0
0 1 0 1 0
1 1 1 1 1
Enter starting vertex for BFS: 1
BFS Traversal: 1 2 3 4 0
pranavneelakandan@Pranavs-MacBook-Air DAA %
```

Interpretation:

BFS explores a graph level by level, visiting all neighbors of a node before moving to the next layer of nodes farther away .
Because of this, it naturally finds the shortest path in terms of number of edges from the start node in an unweighted graph

2.DFS:~

CODE:

```
#include <stdio.h>
#define MAX 10
int graph[MAX][MAX];   // adjacency matrix
int visited[MAX];      // track visited nodes
int n;                 // number of vertices
void DFS(int v) {
  visited[v] = 1;           // mark as visited
```

Design & Analysis of
Algorithms
Pranav N
CH.SC.U4CSE24236

```
   printf("%d ", v);          // print the node
   for (int i = 0; i < n; i++) {
      if (graph[v][i] == 1 && !visited[i]) {
         DFS(i);            // recursive call
      }
   }
}
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", &graph[i][j]);
   // Initialize visited[] to 0
   for (int i = 0; i < n; i++) {
      visited[i] = 0;
   }
   printf("DFS starting from node 0:\n");
   DFS(0);
   printf("\n");
   return 0;
}
```

OUTPUT:

```
neelakandan/"DFS
Enter number of vertices: cd "/Users/pranavneelakandan/" && gcc DFS.c -o DFS && "/Users/pranavneelakandan/"DFS
Enter adjacency matrix:
DFS starting from node 0:
0
pranavneelakandan@Pranavs-MacBook-Air ~ % cd "/Users/pranavneelakandan/" && gcc DFS.c -o DFS && "/Users/pranavne
elakandan/"DFS
Enter number of vertices: 4
Enter adjacency matrix:
0 1 0 1
0 1 1 0
0 1 1 1
1 1 1 1
DFS starting from node 0:
0 1 2 3
pranavneelakandan@Pranavs-MacBook-Air ~ %
```

Interpretation:

DFS starts from a chosen node and explores one path as deeply as possible, visiting neighbours, then neighbours of neighbours, before coming back.

It uses a stack-like behaviour (via recursion or an explicit stack) and a visited array to avoid infinite loops while eventually visiting all reachable nodes from the start.