Graph

Course Code : 0613 – 2108
Course Title : Data Structure Lab
Topic : Graph
Batch : CSE21
Semester : Spring 2024
Date of Submission : 10- 06-2024

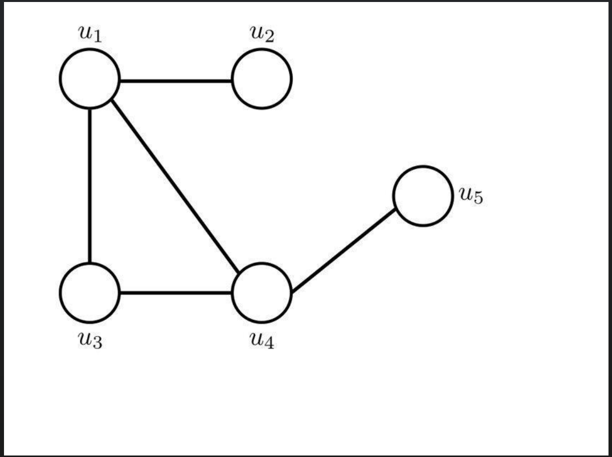
Submitted By :
Aysha Islam
ID : 0692310005101001

Joynob Bint Jamal
ID : 0692310005101002

Submitted To :
Humayara Binte Rashid
Lecturer
Department of CSE
Notre Dame University Bangladesh

What is Graph Data Structure?

Graphs in data structures are non-linear data structures made up of a finite number of nodes or vertices and the edges that connect them.

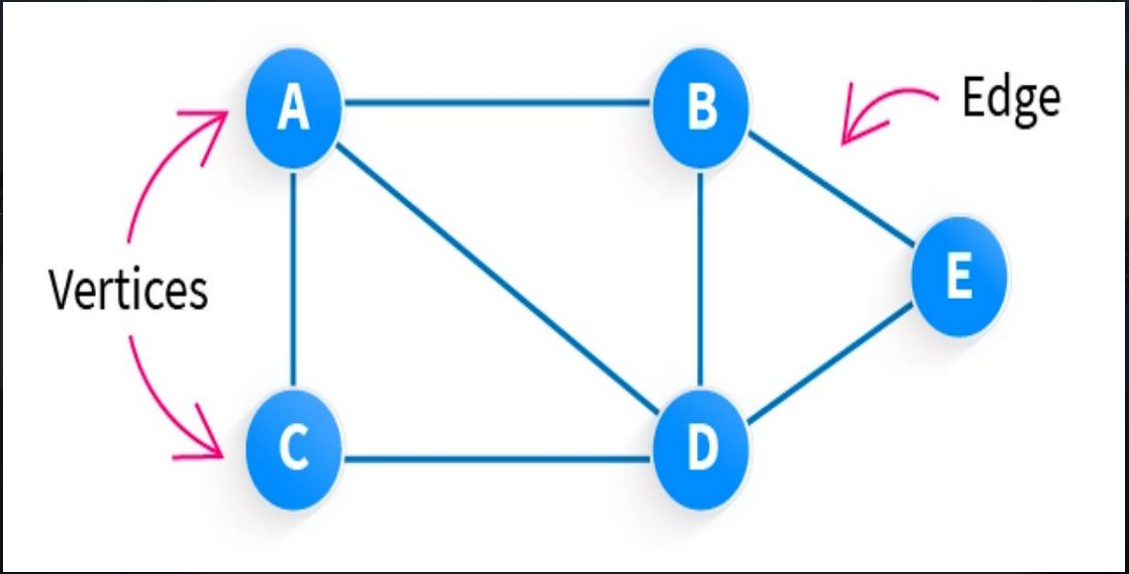


Uses of Graph

- Social networking platforms such as Facebook,LinkedIn, Instagram, and others
- Google Maps
- Web pages are referred to as vertices on the World Wide Web
- Finding shortest route

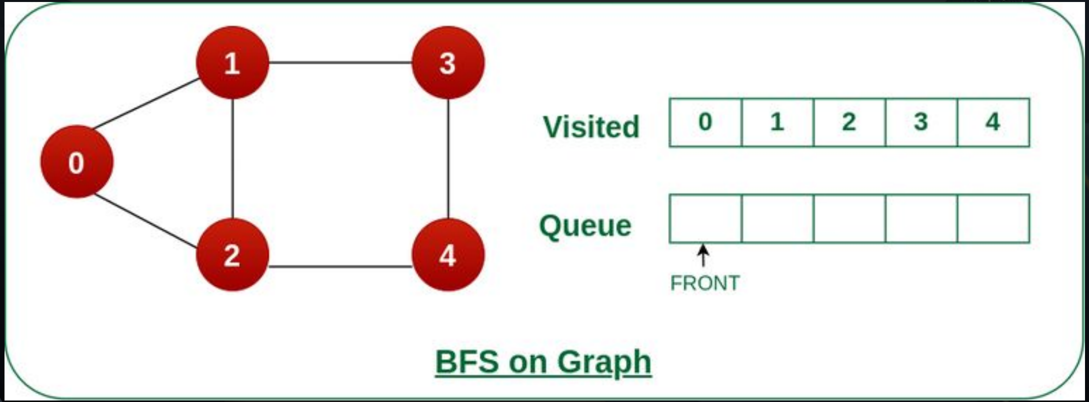
Graph Representation

A **Graph** is a non-linear data structure consisting of vertices and edges.



Project Topic

This project is about one of the graph traversing techniques, Breadth-First Search algorithm, where you select a random initial node (source or root node) and start traversing the graph layer-wise in such a way that all the nodes and their respective children nodes are visited and explored.
We have worked on the application “Find the Shortest Path & Minimum Spanning Tree for an unweighted graph”.



BFS on Graph

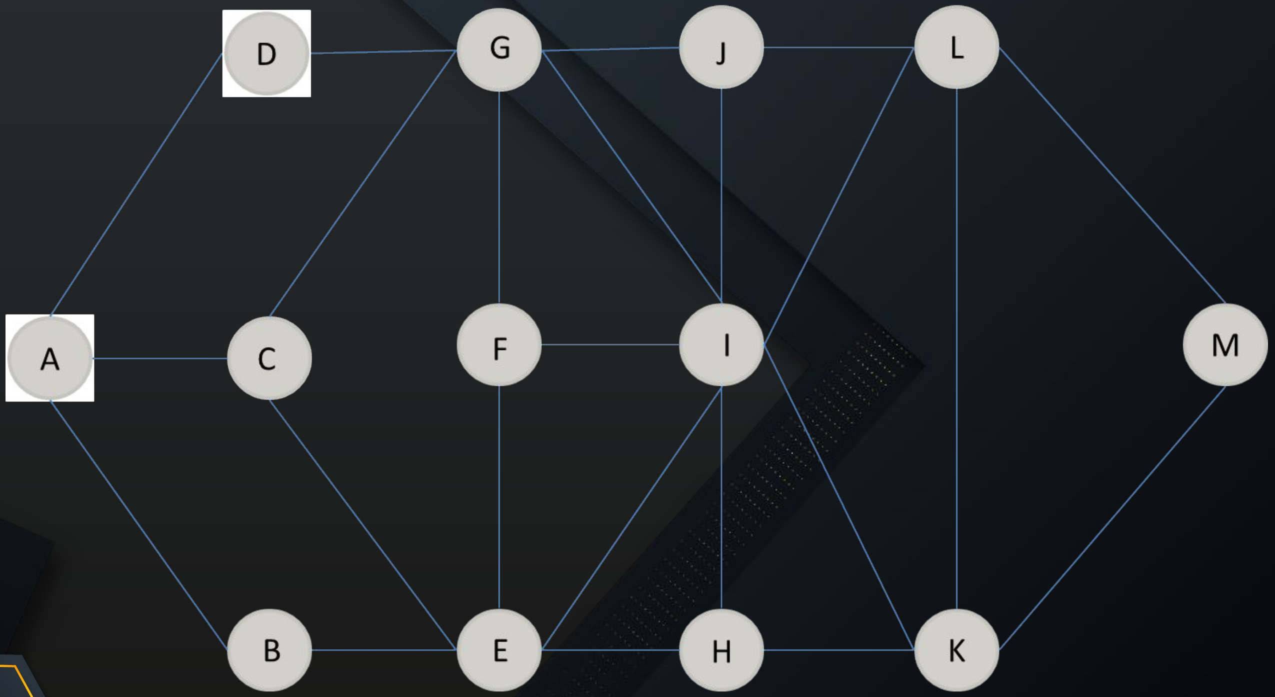
Project Objective

- Efficient Gift Delivery
- Minimize Travel Distance
- Optimize Route Planning
- Practical Application of BFS
- Enhance Problem-Solving Skills

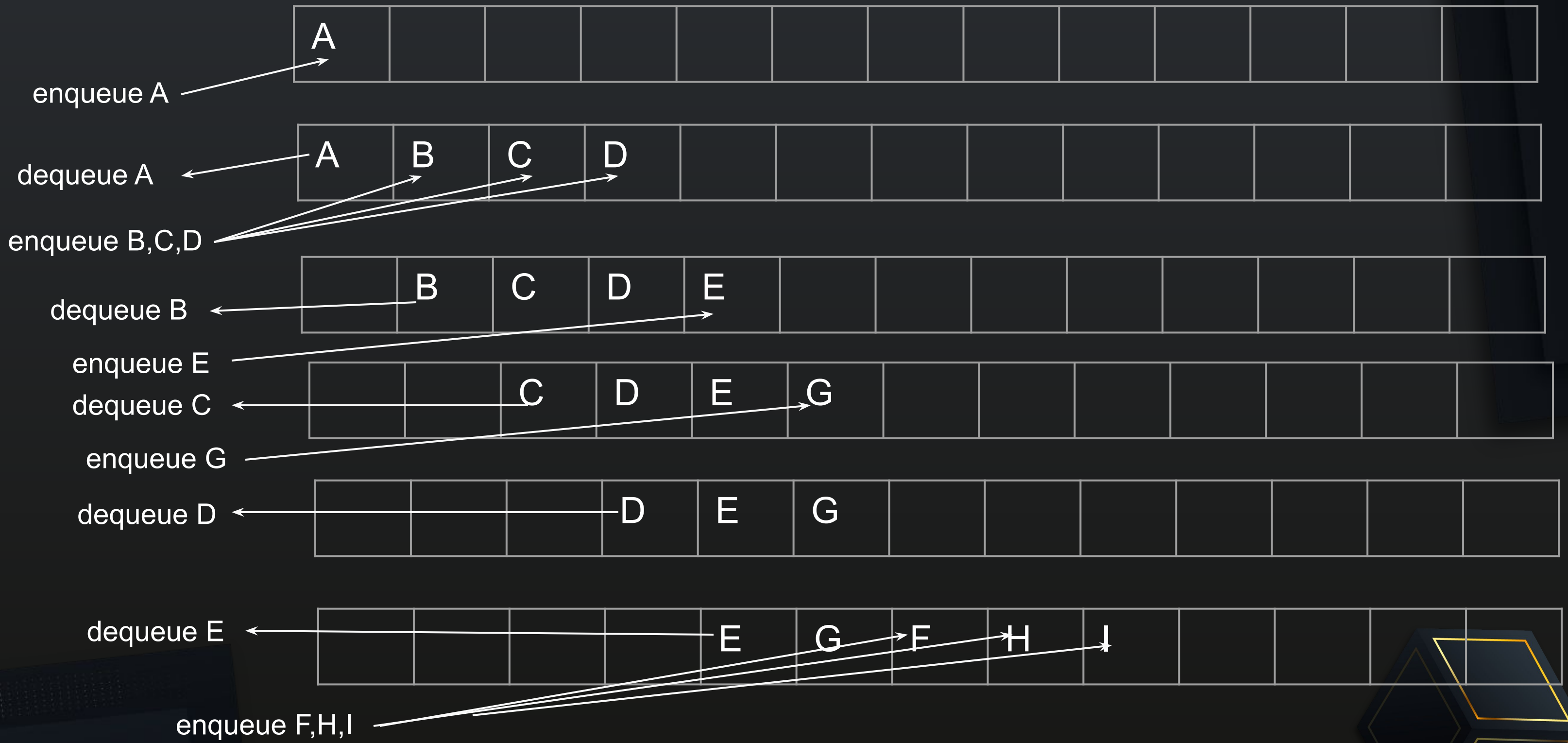
Problem Statement

Eid Ul Adha 2024 is coming soon. Me, with another Friend want to send an Eid gift package to our other friends. We prepared it nicely. Now the problem is we have to go to each of our friend's house so that the minimum span of paths are needed and using the minimum number of ways we can reach every house and send the
gift packs. Through solving this problem described above, we have made a solution by using the BFS graph traversal algorithm.

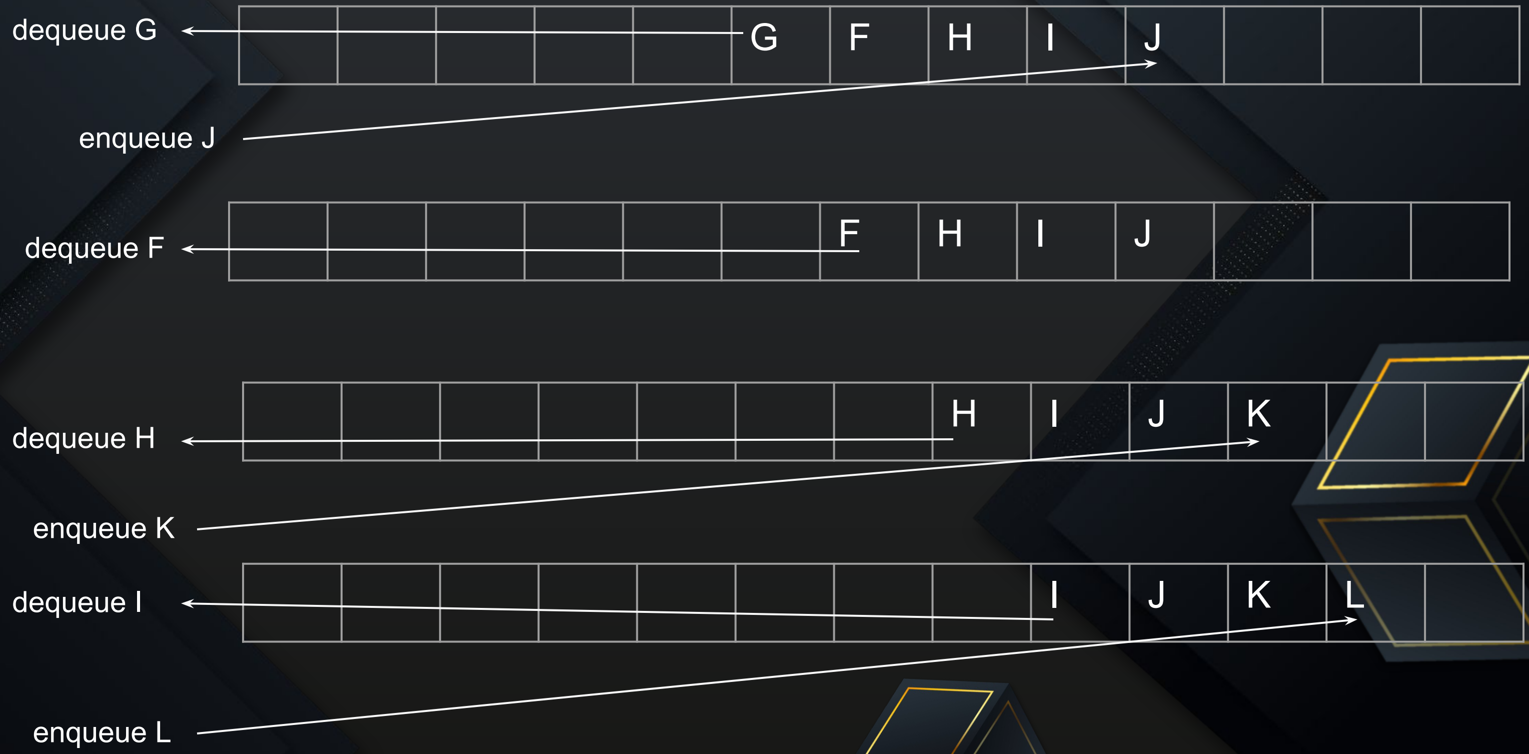
Project Graph



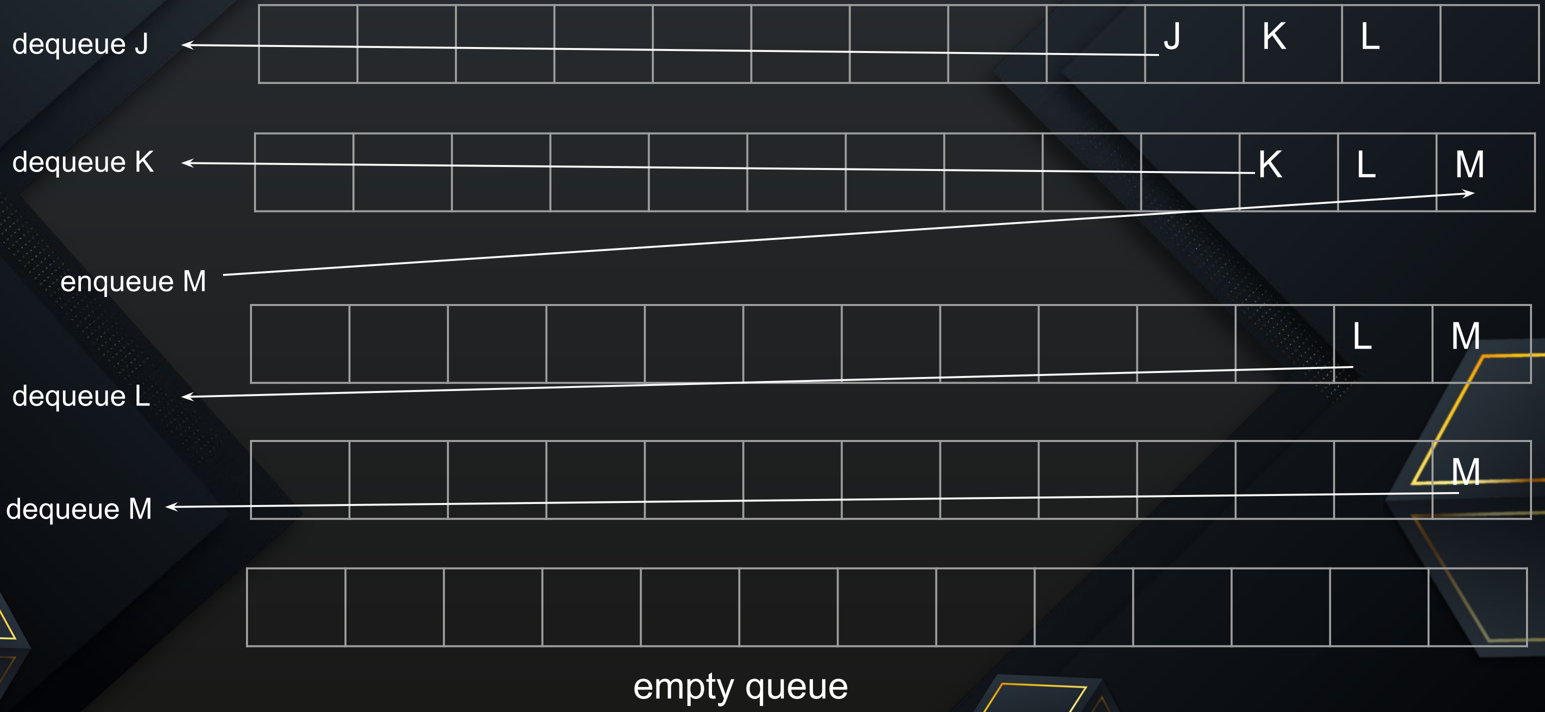
Procedure


A
enqueue A
dequeue A
A
B
C
D
enqueue B,C,D
dequeue B
B
C
D
E
enqueue E
dequeue C
C
D
E
G
enqueue G
dequeue D
D
E
G
dequeue E
E
G
F
H
I
enqueue F,H,I

Procedure


dequeue G
G
F
H
I
J
enqueue J
dequeue F
F
H
I
J
H
I
J
K
dequeue H
enqueue K
dequeue I
I
J
K
L
enqueue L

Procedure


dequeue J
J
K
L
dequeue K
K
L
M
enqueue M
L
M
dequeue L
M
dequeue M
empty queue

Source Code & Output

```
#include <stdio.h>
#include <string.h>

#define MAX_FRIENDS 100
#define MAX_NAME_LENGTH 50

int visited[MAX_FRIENDS];
int distance[MAX_FRIENDS];
int total;

void BFS(int start, int graph[][MAX_FRIENDS], char names[][MAX_NAME_LENGTH]) {
    int queue[MAX_FRIENDS];
    int front = 0, rear = 0;

    queue[rear] = start;
    rear=rear + 1;
    visited[start] = 1;
    distance[start] = 0;

    printf("Friend\t\tDistance\n");
```

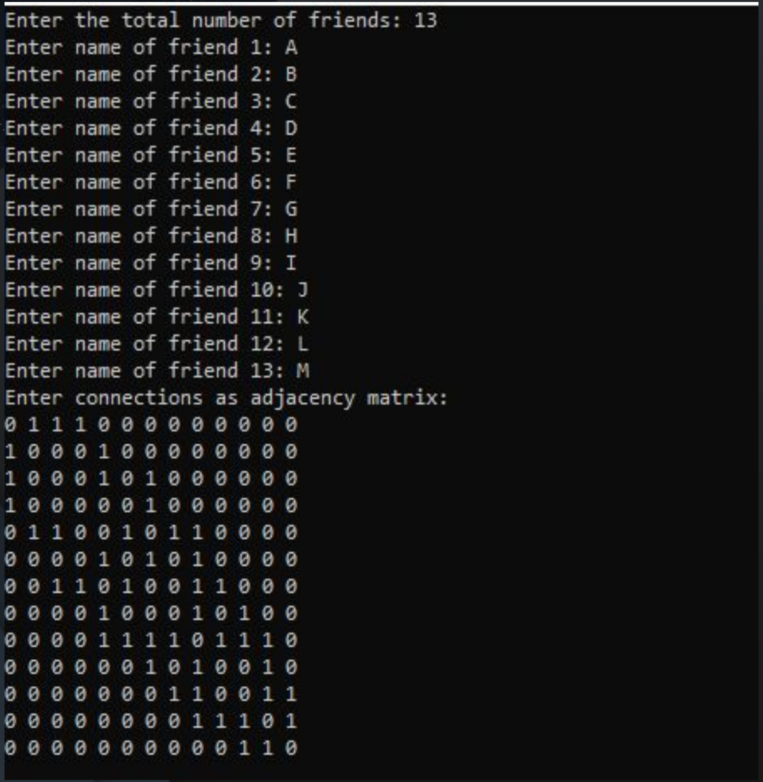
Source Code & Output

```
while (front != rear) {
        int vertex;
        vertex = queue[front];
        front=front+1;

        printf("%s\t\t%d\n", names[vertex], distance[vertex]);

        for (int j = 0; j < total; j++) {
            if (!visited[j] && graph[vertex][j] == 1) {
                queue[rear] = j;
                rear=rear+1;
                visited[j] = 1;
                distance[j] = distance[vertex] + 1;
            }
        }
    }
}

int main() {
    printf("Enter the total number of friends: ");
    scanf("%d", &total);
```

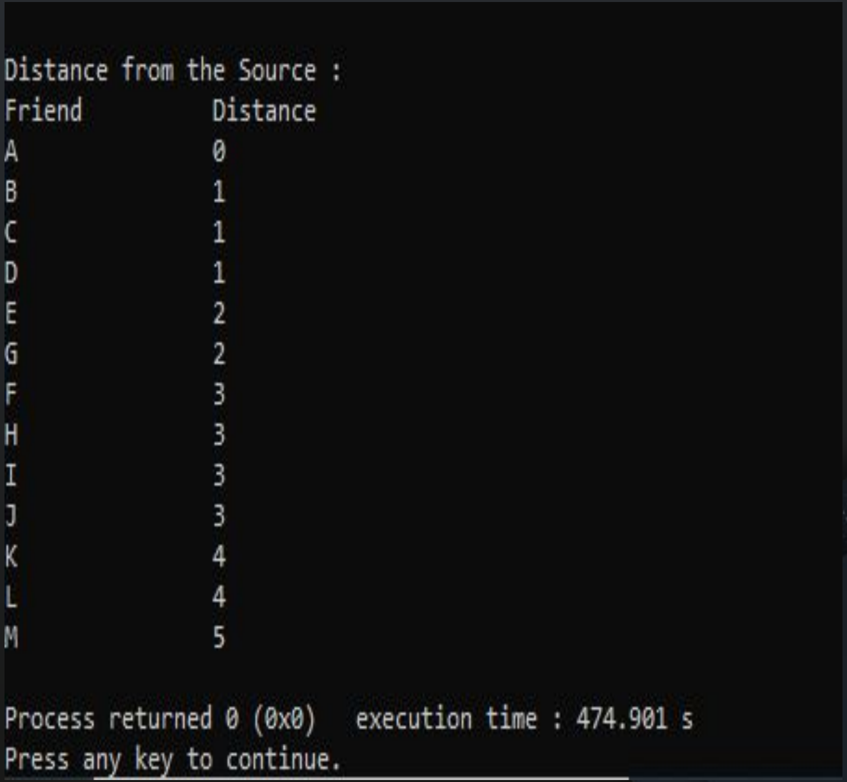
Source Code & Output

```
 char names[MAX_FRIENDS][MAX_NAME_LENGTH];
   int graph[MAX_FRIENDS][MAX_FRIENDS];

for (int i = 0; i <total; i++) {
        printf("Enter name of friend %d: ", i+1);
        scanf("%s",&names[i]);
   }

   printf("Enter connections as adjacency matrix:\n");
   for (int i = 0; i < total; i++) {
       for (int j = 0; j < total; j++) {
           scanf("%d", &graph[i][j]);
       }
   }
   for (int i = 0; i < total; i++) {
       visited[i] = 0;
       distance[i] = -1;
   }
   printf("\nDistance from the Source :\n");
   BFS(0, graph, names);

   return 0;
}
```

Source Code & Output



```
Enter the total number of friends: 13
Enter name of friend 1: A
Enter name of friend 2: B
Enter name of friend 3: C
Enter name of friend 4: D
Enter name of friend 5: E
Enter name of friend 6: F
Enter name of friend 7: G
Enter name of friend 8: H
Enter name of friend 9: I
Enter name of friend 10: J
Enter name of friend 11: K
Enter name of friend 12: L
Enter name of friend 13: M
Enter connections as adjacency matrix:
0 1 1 1 0 0 0 0 0 0 0 0 0
1 0 0 0 1 0 0 0 0 0 0 0 0
1 0 0 0 1 0 1 0 0 0 0 0 0
1 0 0 0 0 0 1 0 0 0 0 0 0
0 1 1 0 0 1 0 1 1 0 0 0 0
0 0 0 0 1 0 1 0 1 0 0 0 0
0 0 1 1 0 1 0 0 1 1 0 0 0
0 0 0 0 1 0 0 0 1 0 1 0 0
0 0 0 0 1 1 1 1 0 1 1 1 0
0 0 0 0 0 0 1 0 1 0 0 1 0
0 0 0 0 0 0 0 1 1 0 0 1 1
0 0 0 0 0 0 0 0 1 1 1 0 1
0 0 0 0 0 0 0 0 0 0 1 1 0
```



```
Distance from the Source :
Friend          Distance
A               0
B               1
C               1
D               1
E               2
G               2
F               3
H               3
I               3
J               3
K               4
L               4
M               5

Process returned 0 (0x0)   execution time : 474.901 s
Press any key to continue.
```

Limitation

- ✔ For large graphs, storing the adjacency matrix can be memory-intensive, especially if the graph is sparse (contains few connections).
- ✔ The BFS algorithm implemented here has a time complexity of O(V + E), where V is the number of vertices (friends) and E is the number of edges (connections). For dense graphs (many connections), this can be inefficient.
- ✔ Printing the distance of each friend might not be practical for very large social networks
- ✔ This project cannot handle disconnected components

Thanks!