NOTRE DAME UNIVERSITY BANGLADESH

Department of Computer Science and Engineering

Project Report

Course Code:	0613-2108
Course Title:	Data Structures Lab
Project No:	01
Topic :	Graph
Date of Submission:	10-06-2024
Semester:	Spring 2024
Batch:	CSE 21

Submitted To-

Humayara Binte Rashid

Lecturer

Department of CSE

Submitted By-

Aysha Islam

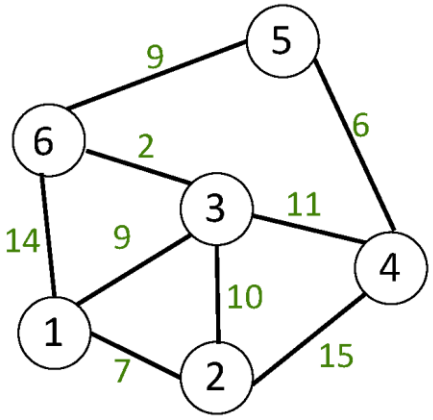
ID:- 0692310005101001

Graph

Definition of Graph:

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.

Graph data structures are a powerful tool for representing and analyzing complex relationships between objects or entities.



★ The graph is denoted by G(V,E)

- V : set of vertices (nodes)
- E : set of edges (links)

Types of graph :

- Undirected : An undirected graph is a type of graph where the edges have no specified direction assigned to them.

  In this case, E{u,v} is same as E{v,u}

- Directed : A directed graph is a type of graph where the edges have specified direction assigned to them.

  In this case, E{u,v} is not the same as E{v,u}.

Advantages of Graph Data Structure:

1) **Modeling Relationships:** Graphs are an effective way to represent and model relationships between different entities. This makes them suitable for applications such as social networks, transportation networks, and communication systems.
2) **Flexibility:** Graphs can represent a wide variety of relationships, including one-to-one, one-to-many, and many-to-many relationships. This flexibility allows for versatile data modeling.
3) **Traversal and Path-finding:** Graph algorithms enable efficient traversal and path-finding, which is useful in applications such as GPS navigation, network routing, and recommendation systems.
4) **Cyclic Relationships:** Graphs can represent cyclic relationships, making them suitable for modeling processes with cyclical behavior, such as scheduling and resource allocation.
5) **Data Organization:** Graphs can be used to organize and represent data in a non-linear structure, which can be advantageous for certain types of data and operations.
6) **Complex Analysis:** Graph algorithms enable complex analysis of relationships and structures within the data, allowing for insights and optimizations in various domains.
7) **Network analysis:** Graphs are commonly used in network analysis to study relationships between individuals or organizations, as well as to identify important nodes and edges in a network. This is useful in a variety of fields, including social sciences, business, and marketing.

8) **Visualization:** Graphs are highly visual, making it easy to communicate complex data and relationships in a clear and concise way. This makes them useful for presentations, reports, and data analysis.

Disadvantages of Graph:

1) **Limited representation:** Graphs can only represent relationships between objects, and not their properties or attributes. This means that in order to fully understand the data, it may be necessary to supplement the graph with additional information.
2) **Difficulty in interpretation:** Graphs can be difficult to interpret, especially if they are large or complex. This can make it challenging to extract meaningful insights from the data, and may require advanced analytical techniques or domain expertise.
3) **Scalability issues:** As the number of nodes and edges in a graph increases, the processing time and memory required to analyze it also increases. This can make it difficult to work with large or complex graphs.
4) **Data quality issues:** Graphs are only as good as the data they are based on, and if the data is incomplete, inconsistent, or inaccurate, the graph may not accurately reflect the relationships between objects.
5) **Lack of standardization:** There are many different types of graphs, and each has its own strengths and weaknesses. This can make it difficult to compare graphs from different sources, or to choose the best type of graph for a given analysis.
6) **Privacy concerns:** Graphs can reveal sensitive information about individuals or organizations, which can raise privacy concerns, especially in social network analysis or marketing.

Uses And Applications:

- One most likely utilizes social networking platforms such as Facebook, LinkedIn, Instagram, and others. A wonderful example of a graph in usage is social media. Graphs are used in social media to hold information about each user. Every user is a node in this case, just like in Graph.

  Users on Facebook are referred to as vertices, and if they are friends, there is an edge connecting them. The Friend Suggestion system on Facebook is based on graph theory.

- Google Maps is another application that makes use of graphs. In the case of Google Maps, each place is referred to as a node, and the roads that connect them are referred to as edges.
- Web pages are referred to as vertices on the World Wide Web. Suppose there is a link from page A to page B that can represent an edge. This application is an illustration of a directed graph.

Basically, theory and applications of graphs can be described enormously in real life.

Project Report

Introduction:

This Project is based on BFS (Breadth first search) . BFS is one of the graph traversal techniques . BFS is based on a queue which follows FIFO (First in first out) structure. We made this project to give a simple concept about real life examples . It means this algorithm applies behind the real life example .

Applications of BFS:

- ➔ Crawlers in Search Engines: Breadth-First Search is one of the main algorithms used for indexing web pages. The algorithm starts traversing from the source page and follows all the links associated with the page. Here each web page will be considered as a node in a graph.
- ➔ GPS Navigation systems: Breadth-First Search is one of the best algorithms used to find neighboring locations by using the GPS system.
- ➔ Find the Shortest Path & Minimum Spanning Tree for an unweighted graph: When it comes to an unweighted graph, calculating the shortest path is quite simple since the idea behind shortest path is to choose a path with the least number of edges. Breadth-First Search can allow this by traversing a minimum number of nodes starting from the source node. Similarly, for a spanning tree, we can use either of the two, Breadth-First Search or Depth-first traversal methods to find a spanning tree.

- ➔ Broadcasting: Networking makes use of what we call as packets for communication. These packets follow a traversal method to reach various networking nodes. One of the most commonly used traversal methods is Breadth-First Search. It is being used as an algorithm that is used to communicate broadcasted packets across all the nodes in a network.
- ➔ Peer to Peer Networking: Breadth-First Search can be used as a traversal method to find all the neighboring nodes in a Peer to Peer Network. For example, BitTorrent uses Breadth-First Search for peer to peer communication.

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" mentioned above.

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.

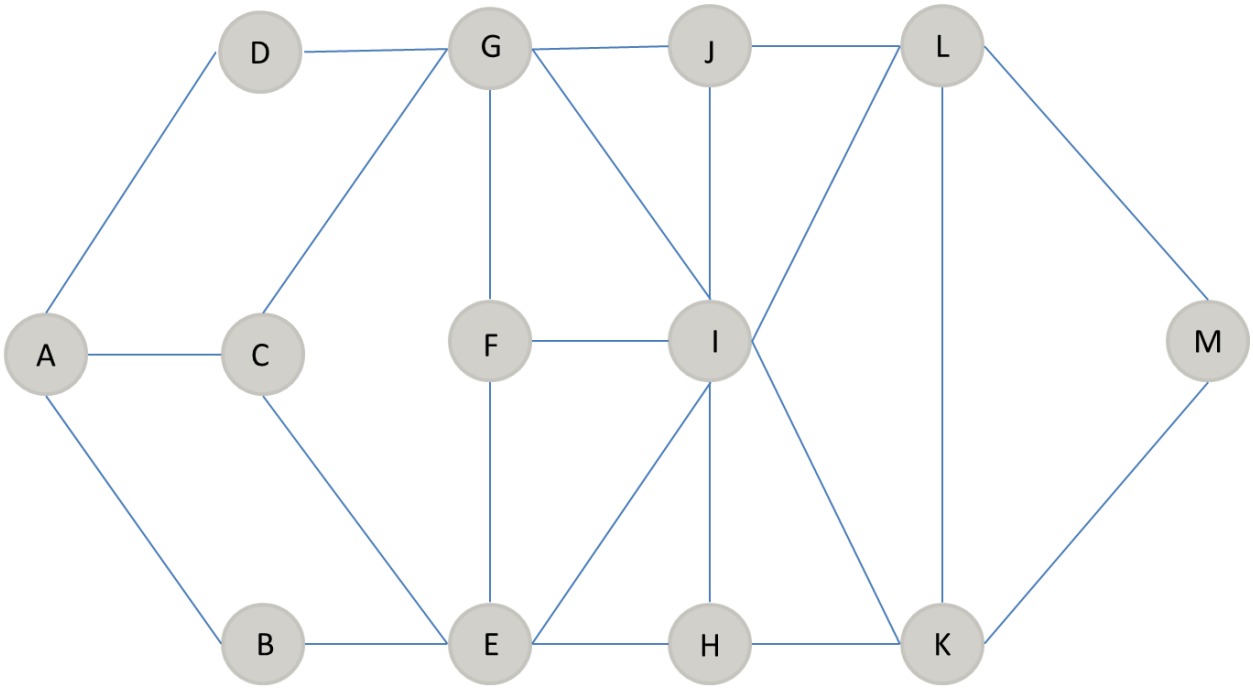
Through solving this problem described above, we have made a solution by using the BFS graph traversal algorithm.

Reason for Choosing This Project:

Our project is aimed to solve real life problems. Since we are working on graphs and in graphs, traversing is an important thing. While working on graph traversal, BFS can be the most useful to finding efficient ways of route planning in less timing.
This project introduces us with a Real Life scenario. Maximum people see that something is happening in reality but we tried the algorithmic process used behind the scenery to express to them . The reason that we chose this project is to show people and to give a simple concept about the uses of algorithms behind the real life example.

Project Objectives :

- Efficient Gift Delivery: The main objective of this project was to use a system that ensures efficient delivery of Eid gift packages to our friends' houses.
- Minimize Travel Distance: By using the BFS graph traversal algorithm, our aim was to minimize the total distance traveled, also reducing time and energy consumption.
- Optimize Route Planning: We have worked on an algorithm that plans the route to cover all friends' houses with minimal travel.
- Practical Application of BFS: This project aimed to demonstrate the real-world application of graph traversal algorithms.
- Enhance Problem-Solving Skills: By solving this gift delivery problem, we aimed to enhance our problem-solving skills through algorithms.



Source Code and 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");

   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);

   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]);
```

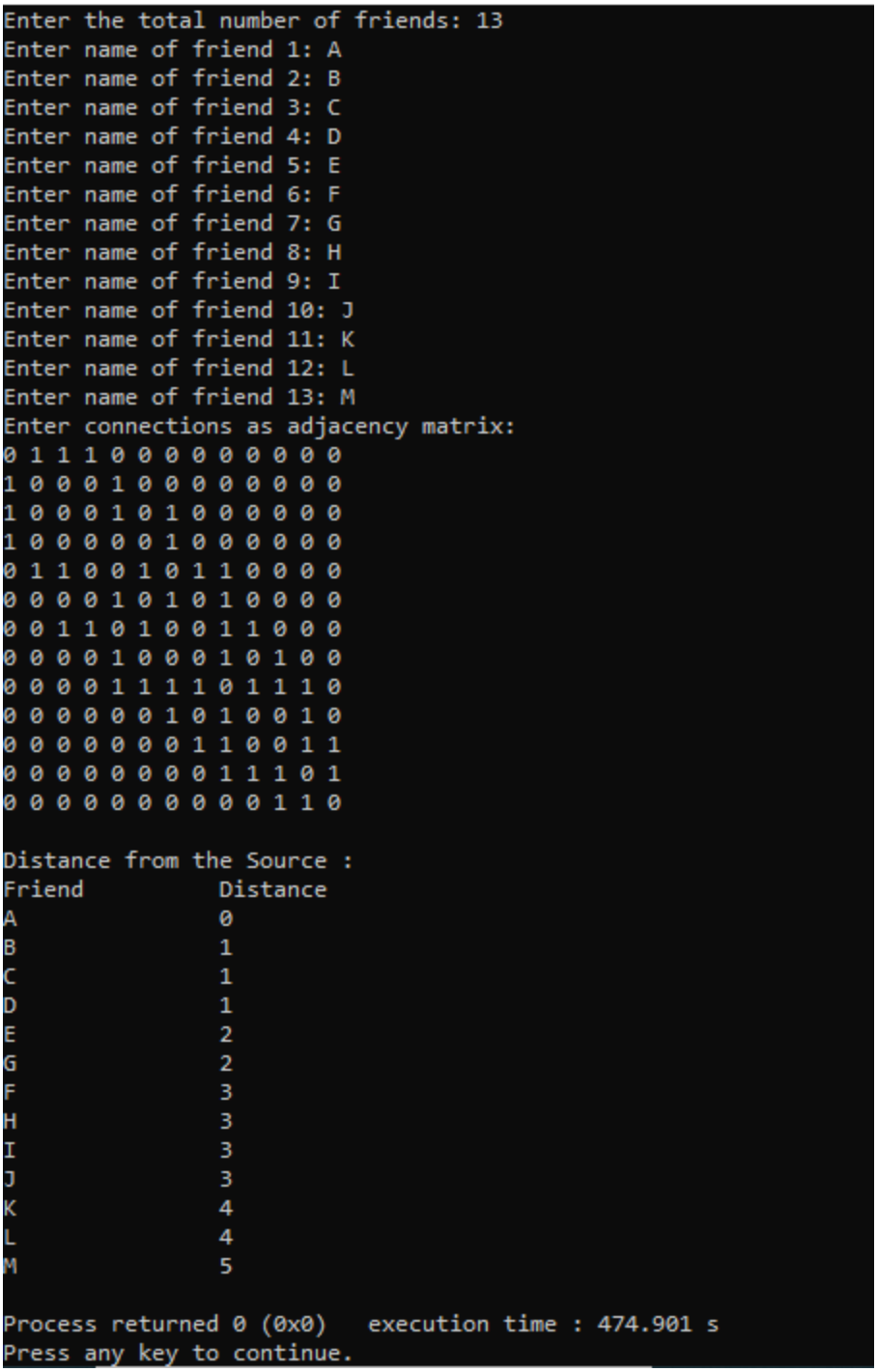
```
    }

    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;
}
```



Result Analysis :

The result produces the distances from source node to other nodes. Here we take input for the node name i.e vertices. Name of the friends acts as nodes. After that

we are taking input for an adjacency matrix to represent a graph that shows the connections or the existence of edges among the vertices. The program calculates the distance and shows us the result. Through this we can determine whose house is near and where to go first, then next, then next……. . By this process, we can visit everyone's house and as a result we don't need to pass the same node twice.

The final result shows that, our visit sequence may look like :
A -> B -> C -> D -> E -> G -> F -> H -> I -> J -> K -> L -> M

Algorithm:

Step -1 : Define constants for maximum number of friends 'MAX_FRIENDS' and maximum length for their names 'MAX_NAME_LENGTH'.

Step -2 : Declare arrays 'visited[]' and 'distance[]' to keep track

Step -3 : Implement BFS() function :

A. Declare queue[] of MAX_FRIENDS size and initialize variables front and rear to 0.
B. Enqueue the starting node into the queue, mark it as visited, and set its distance as 0.
C. While the queue is not empty:
    I. Dequeue a vertex from the queue.
    II. Print the name of the friend and its distance from the source.
    III. Traverse through all friends (nodes) and enqueue unvisited friends that are connected to the current friend.
D. Check if a friend is not visited and there is an edge to the current friend (graph[vertex][j] == 1), enqueue the friend, mark it as visited=1, and set its distance as one more than the distance of the current friend.

Step -4 : Take inputs for total friends 'total'.

Step -5 : Declare arrays 'names' and 'graph' to store friends' names and adjacency matrix, respectively and take inputs for these.

Step -6 : Initialize 'visited[]' and 'distance[]' as visited[i] = 0 and

distance[i] = -1.

Step -7 : Call BFS() function

Step -8 : End

Limitations :

It can be slow since it expands all the vertices at each level before moving on to the next level. It can sometimes find sub-optimal solutions since it doesn't explore all possible paths through the search tree. The wrong entry of the adjacency matrix can cause the wrong route thus wasting time and energy.

Related Works:

This project related some research papers and web resources are :

- Research Papers:
  - Original papers by Konrad Zuse and early formalizations of BFS.
  - Recent papers on parallel and distributed BFS.
  - The Nature of Breadth-First Search by Jason J Holdsworth
- Web Resources:
  - Online courses (e.g., Coursera, edX) and tutorials on BFS.
  - Open source project documentation (e.g., NetworkX library for Python).

By taking the knowledge of previous literature reviews , we can ensure a comprehensive review of BFS , covering historical context and modern developments .

Conclusion :

In short, our project displays significant improvements in the efficiency of any kind of delivery process. The successful implementation of this project not only facilitated seamless gift delivery but also underscored the practical relevance of algorithmic solutions in everyday scenarios. One of the greatest uses is that we human beings want to solve our real life problems in a simpler way , by using this

algorithm we can simply solve our most complicated problems in a very simpler way in real life .