Green University of Bangladesh
Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering

Semester: (Fall, Year:2023), B.Sc. in CSE (Day)

Lab Report NO: 02

Course Title	**: Algorithms Lab**
Course Code	**: CSE 204**
Section	**: 221 D9**

Lab Experiment Name: **Sorting in an Directed Graph using BFS aka Topological Sorting.**

Student Details

	Name	ID
1.	Jahidul Islam	221002504

Lab Date	**: 09-10-2023**
Submission Date	**: 16-10-2023**
Course Teacher's Name	**: Md. Abu Rumman Refat**

Lab Report Status

Marks: ………………………………… **Signature:.....................**

Comments:... **Date:..............................**

1. TITLE: Topological sort using BFS.

ABSTRACT

This lab report presents a C++ program that implements topological sorting in a directed acyclic graph (DAG) using Breadth-First Search (BFS). Topological sorting is a crucial algorithm for tasks with dependencies, and the program provides an example of how to apply BFS to achieve this task in a directed acyclic graph.

2. INTRODUCTION:

Topological sorting is a fundamental algorithm used in graph theory to arrange the vertices of a directed acyclic graph (DAG) in a linear order, ensuring that for every directed edge (u, v), vertex u appears before vertex v in the ordering. This linear order is essential for scheduling tasks with dependencies, building systems, and solving various real-world problems.

we implement topological sorting using BFS in a C++ program. The program takes an example directed acyclic graph and returns the topological order.

3. Equipment and Software Used:

1. C++ Programming Language
2. IDE – Code Blocks.
3. Directed acyclic graph
4. Graph Visualization Tools

4. PROCEDURE:

1. **Graph Representation:** The program uses an adjacency list to represent the directed graph.

2. **Breadth-First Search (BFS):**

- Initialize an array to keep track of in-degrees for each vertex.
- Calculate the in-degrees for all vertices by iterating through the adjacency list.
- Enqueue vertices with in-degrees of 0 into a queue.
- Process the vertices in the queue:
  - Decrement the in-degrees of adjacent vertices.
  - Enqueue vertices with in-degrees of 0.
- The result queue contains the topological sort order.

5. IMPLEMENTATION:

The C++ program consists of the following components:

- "Graph" class: Defines a directed acyclic graph with methods for adding edges and performing topological sorting.
- "addEdge()" method: Adds directed edges to the graph.
- "topologicalSort()" method: Implements the BFS-based topological sorting algorithm.

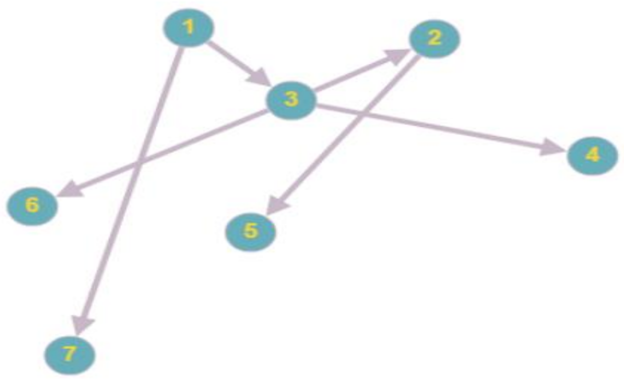
6. Code in C++

```cpp
//jahidulZaid
 #include <bits/stdc++.h>
 using namespace std;
 #define optimize() ios_base::sync_with_stdio;cin.tie(0);cout.tie(0);
 #define endl '\n'

 const int MAX = 1e5+12;
 vector<int> adj[MAX];
 vector<int> inDegree(MAX, 0);

 vector<int> TopologicalSort(int n) {
     vector<int> result; // Store the topological order.

     queue<int> q;

     // Initialize the queue with vertices having in-degree 0.
     for (int i = 1; i <= n; i++) {
         if (inDegree[i] == 0) {
             q.push(i);
         }
     }

     while (!q.empty()) {
         int u = q.front();
         q.pop();
         result.push_back(u);

         // Process adjacent vertices.
         for (int v : adj[u]) {
             inDegree[v]--;
             if (inDegree[v] == 0) {
                 q.push(v);
             }
         }
     }

     return result;
 }

 int main() {

     int n, m;
     cout << "Enter the number of Node and Edges: ";
     cin >> n >> m;

     cout << "Enter the Edges (X Y):" << endl;
     for (int i = 0; i < m; i++) {
         int u, v;
         cin >> u >> v;
         adj[u].push_back(v);
         inDegree[v]++;
     }

     // Call for topological sort and get the result.
     vector<int> result = TopologicalSort(n);

     // Check if the graph is a DAG (no cycles).
     if (result.size() == n) {
         cout << "Topological Order: ";
         for (int vertex : result) {
             cout << vertex << " ";
         }
     } else {
         cout << "Error! The graph contains cycle." << endl;
     }
     return 0;
 }
```

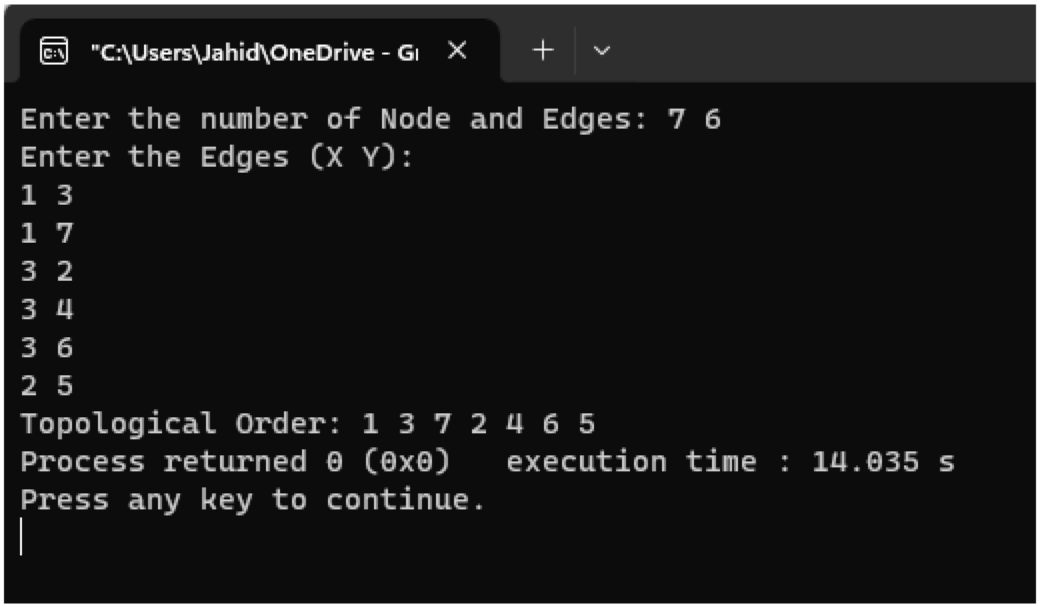
7. INPUT GRAPH:

Node: 7 and Edge: 6
Edges are:

1 → 3, 1 → 7, 3 → 2, 3 → 4, 3 → 6, 2 → 5



8. OUTPUT:



```
Enter the number of Node and Edges: 7 6
Enter the Edges (X Y):
1 3
1 7
3 2
3 4
3 6
2 5
Topological Order: 1 3 7 2 4 6 5
Process returned 0 (0x0)   execution time : 14.035 s
Press any key to continue.
```

Topological Order: 1 3 7 2 4 6 5

9. DISCUSSION:

The program demonstrates that topological sorting using BFS is an efficient way to determine a linear ordering of vertices in a directed acyclic graph. This ordering is crucial for various applications where tasks or components depend on one another.

While this program showcases the concept of topological sorting, it's important to note that there can be multiple valid topological orders for a given DAG. The specific order may vary depending on the problem and the particular graph structure.

10. CONCLUSION:

Topological sorting is a valuable algorithm in graph theory, and the C++ program we implemented in this lab report successfully applies topological sorting using BFS to a directed acyclic graph. By understanding and applying this concept, we can solve problems involving dependencies, scheduling, and more.