**EXPERIMENT NO. 2**

|  |
| --- |
| **Student Name and Roll Number:** Piyush Gambhir – 21CSU349 |
| **Semester /Section:** Semester-V – AIML-V-B (AL-3) |
| **Link to Code:** [NCU-Lab-Manual-And-End-Semester-Projects/NCU-CSL347-AAIES-Lab\_Manual at main · Piyush-Gambhir/NCU-Lab-Manual-And-End-Semester-Projects (github.com)](https://github.com/Piyush-Gambhir/NCU-Lab-Manual-And-End-Semester-Projects/tree/main/NCU-CSL347-AAIES-Lab_Manual) |
| **Date:** 19.08.2023 |
| **Faculty Signature:** |
| **Grade:** |

|  |
| --- |
| **Objective(s):**   * Understand what breadth first Search (BFS) and Depth first Search (DFS) is. * Study about different uninformed searching approaches. * Implement BFS and DFS for solving a real-world problem. |
| **Outcome:**  Students would be familiarized with BFS and DFS.  Students would be able to make a comparison between the two algorithms. |
| **Problem Statement:**  Implement Breadth-First Search (BFS) and Depth-First Search (DFS) algorithms in Python to analyze a simple social network. The program aims to explore social connections between users and offer insights into their relationships.  The program should be able to:   * Find groups of users who are directly or indirectly connected to the influencer “*Brenda*”. * Determine if there is a path between two users “*Sally”* and “*Ted”*   Use the graph as provided below for the assignment:layout - How best to display a network of friends? - User Experience Stack  Exchange |
| **Background Study:**  Breadth-First Search (BFS) and Depth-First Search (DFS) are graph traversal algorithms. BFS explores a graph level by level, visiting all neighboring nodes before moving deeper, utilizing a queue data structure. On the other hand, DFS explores as deep as possible along each branch before backtracking, using a stack data structure. Both algorithms are fundamental in graph analysis and have various applications in tasks like path finding, cycle detection, and social network analysis. |
| **Question Bank:**   1. What do you understand by blind search algorithms?   Blind search algorithms are methods used in computer science to explore and traverse problem spaces without having any specific information about the structure or characteristics of the space. These algorithms rely solely on the information available during the search process and do not incorporate domain-specific knowledge.   1. What is *Breadth-First Search* and *Depth-First Search*?   Blind search algorithms are methods used in computer science to explore and traverse problem spaces without having any specific information about the structure or characteristics of the space. These algorithms rely solely on the information available during the search process and do not incorporate domain-specific knowledge.   1. What are the appropriate scenarios for using BFS and DFS algorithms?   Breadth-First Search (BFS) is suitable for scenarios where finding the shortest path is important, like in navigation systems or puzzle-solving. It guarantees the shortest path but can require more memory due to the need to store the entire level's nodes.  Depth-First Search (DFS) is appropriate for scenarios where memory efficiency is crucial, such as solving mazes or searching through large trees. It doesn't guarantee the shortest path and can get stuck in infinite loops in some cases, but it's memory efficient as it explores one branch deeply before backtracking. |

**Student Work Area**

**Algorithm/Flowchart/Code/Sample Outputs**

**Pseudocode**

**A white background with red text

Description automatically generated**

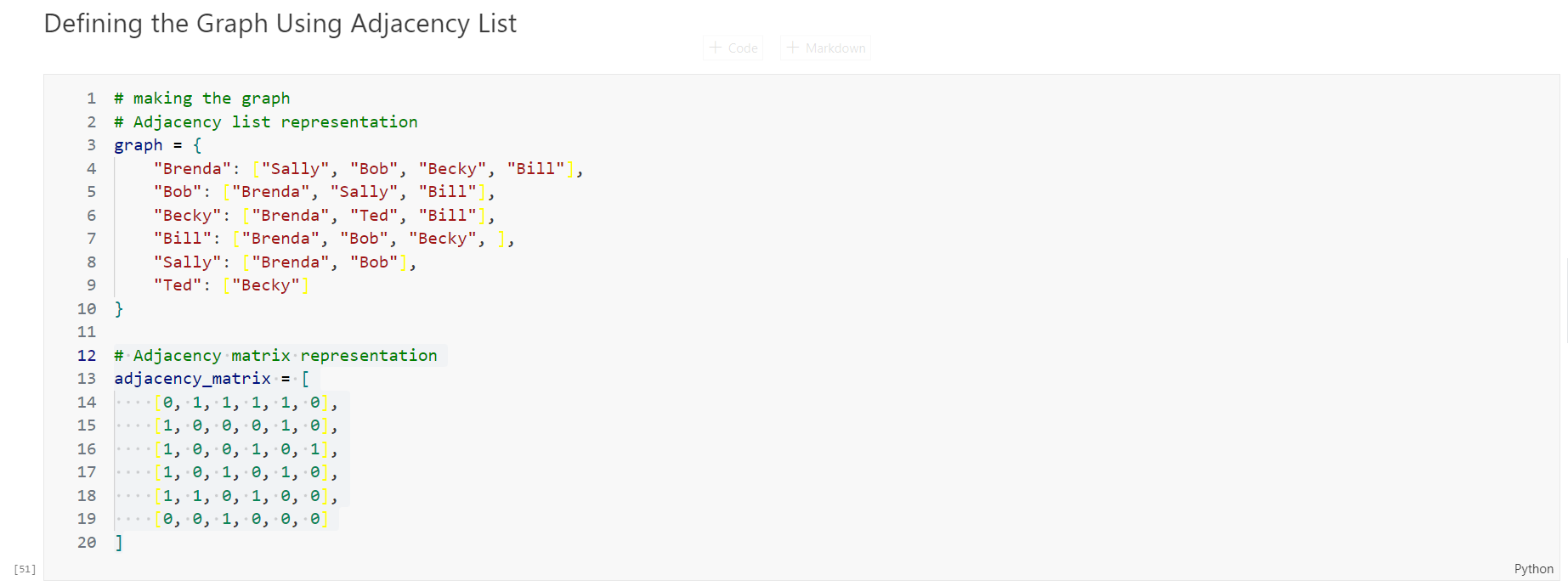
**A white background with red text

Description automatically generated**

**Code:**

**A screenshot of a computer

Description automatically generated**

****

**A white background with a black and white flag

Description automatically generated with medium confidence**

**A close-up of a computer screen

Description automatically generated**

**A white background with text

Description automatically generated**

**Output:**

**A white background with black text

Description automatically generated**