SEMESTER PROJECT

Design and Analysis Of Algorithms

GRAPH ALGORITHMS

Section: P

Muhammad Kashif	I21-0851
Usman Nazeer	I21-0556
Bilal Saleem	I21-0464
Muhammad Huzaifa	I21-2460

Project Report: FAST Social Network Analysis

Executive Summary

The FAST Social Network project aimed to design and implement a C++ program to model and analyse a social network within the FAST-NUCES university setting. The focus was on representing relationships between students, faculty, and staff using graph algorithms to gain insights into the social structure. The team consisted of Muhammad Kashif, Bilal Saleem, Usman Nazeer, and Muhammad Huzaifa, each contributing expertise in different aspects of graph algorithms and analysis.

Project Contributions

1. **Usman:**
    - Graph representation
    - Breadth-First Search (BFS)
    - Topological Sort
    - Highest Degree Centrality
    - Academic Collaborations
2. **Kashif:**
    - Adjacency Matrix
    - Prim's algorithm
    - Influential Individuals
    - Community Detection
3. **Bilal:**
    - Depth-First Search (DFS)
    - Event Attendance Analysis
    - Dynamic Programming
4. **Huzaifa:**
    - Find Interdisciplinary Collaborations
    - Kruskal's algorithm
    - Dijkstra's algorithm
5. **Collective Efforts:**
    - Relationship Analysis
    - Communication Patterns
    - Collaboration Networks
    - Project Report

Project Implementation Overview

1. Graph Representation

The team designed a Graph class to represent the university social network, where nodes correspond to individuals and edges represent various relationships such as friendships and academic collaborations. Usman played a crucial role in defining the overall structure of the graph.

2. Relationship Analysis

Functionality to add individuals, establish relationships, and visualize the network graph was implemented. Different types of relationships, including academic collaborations, mentorship, and extracurricular involvement, were explored.

3. Communication Patterns

Algorithms were implemented to analyse communication patterns, identifying individuals with the highest communication centrality and groups with strong communication ties.

4. Collaboration Networks

The team developed algorithms to identify collaboration networks, emphasizing academic collaborations between students and faculty, as well as interdisciplinary collaboration between different departments.

5. Influence Analysis

Algorithms were implemented to determine influential individuals within the university, considering metrics such as academic impact, leadership roles, and involvement in university events.

6. Community Detection

A community detection algorithm was implemented to identify social clusters within the university, revealing tightly knit groups based on shared interests, projects, or academic pursuits.

7. Event Attendance Analysis

The program was extended to include information about university events, with algorithms analysing attendance patterns to identify popular events and individuals who frequently participate.

8. Dynamic Network

The system allows for dynamic changes in the social network, simulating the evolving nature of social connections as individuals join or leave the university.

Insights and Challenges

The project uncovered several insights into the university's social structure, revealing intricate communication patterns, collaboration networks, and influential individuals. However, the implementation process faced challenges in balancing algorithmic efficiency with real-world complexities, especially in handling dynamic network changes.

Future Enhancements

While the current implementation successfully addressed the project requirements, future enhancements could involve refining algorithms for even better performance, incorporating additional metrics for influence analysis, and expanding the dynamic network capabilities.

Algorithm Performance Analysis

While the current implementation successfully addressed the project requirements, future enhancements could involve refining algorithms for even better performance, incorporating additional metrics for influence analysis, and expanding the dynamic network capabilities.

The following table presents the exact time taken by implemented algorithms on different input sizes (50, 100, 150, 200, 250):

Algorithm	**Input Size 50**	**Input Size 100**	**Input Size 150**	**Input Size 200**	**Input Size 250**
BFS	1.91203	1.12866	0.911246	0.0324056	0.0269148
DFS	0.609043	0.604887	2.52733	0.0432466	0.83273
Topological Sort	0.0100612	0.0215539	0.0311689	0.965577	1.92263
Prim's algorithm	0.0926264	0.274612	0.260722	1.43423	0.0401921
Degree Centrality	0.0939081	0.165923	0.260856	0.356869	0.441448
Academic Collaborations	0.0759087	0.164839	0.243217	0.331063	0.482711
Interdisciplinary Collaborations	1.42793	5.1184	11.457	0.338732	0.412868
Influential Individuals	0.0072224	0.0100697	0.0111359	0.0027074	0.0033055
Kruskal's algorithm	0.105339	0.263634	0.779347	15.8038	20.9308
Dijkstra's algorithm	1.30308	1.29068	1.51517	1.66493	4.37204
Attendance Analysis	8.48534	6.93902	6.27818	1.63552	1.53491
Bellman's Shortest Path	1.20708	1.0093	0.831699	6.15724	8.72003
Community Detection	0.010403	0.0195582	0.0324056	0.636565	0.786033

Time Complexity Analysis

The time complexities of each algorithm are as follows:

- BFS: O(V + E)
- DFS: O(V + E)
- Topological Sort: O(V + E)
- Prim's algorithm: O(V^2)
- Kruskal's algorithm: O(E log V)

- Dijkstra's algorithm: O(V^2)
- Bellman's Shortest Path: O(VE)

The observed execution times generally align with the expected time complexities, confirming the efficiency of the implemented algorithms. Further analysis and optimization could be explored for algorithms with higher time complexities in larger input sizes.

Dataset Description

The dataset used for the project consisted of nodes representing individuals within the university and edges denoting relationships. The graph was directed and weighted, capturing the strength or significance of relationships. The structure of nodes was designed to include information about students, faculty, and staff. Here is the code we used for data generation:

```
#include <fstream>

#include <iostream>

#include <random>

#include <string>


// Function to generate random integer within a range

int generateRandom(int min, int max) {

static std::random_device rd;

static std::mt19937 gen(rd());

std::uniform_int_distribution<> dis(min, max);

return dis(gen);

}


int main() {

std::ofstream outputFile("data.csv"); // Create/open a file for writing


if (!outputFile.is_open()) {

std::cout << "Error opening the file." << std::endl;

return 1;

}


const int MAX_NODES = 500;

outputFile << "Name,Role,Department,AcademicImpact,LeadershipRoles,Involvement\n";
// Writing headers
for (int i = 0; i < MAX_NODES; ++i) {
std::string nodeName = "Node_" + std::to_string(i + 1);
std::string nodeRole = (i < 3) ? "Faculty" : "Student";
std::string nodeDepartment = "Department_" + std::to_string(i % 3 + 1);
int academicImpact = generateRandom(0, 100);
int leadershipRoles = generateRandom(0, 10);
int involvement = generateRandom(0, 30);

outputFile << nodeName << "," << nodeRole << "," << nodeDepartment << ","
<< academicImpact << "," << leadershipRoles << "," << involvement << "\n";
}

outputFile.close(); // Close the file
std::cout << "File 'data.csv' generated successfully." << std::endl;
return 0;
}
```

Conclusion

The FAST Social Network project provided a comprehensive exploration of graph algorithms to model and analyse the university's social structure. Each team member's contribution was instrumental in achieving a well-rounded and efficient implementation. The insights gained from this analysis can be valuable for understanding and enhancing social dynamics within the university setting.