**CS3364 Design and Analysis of Algorithms**

Project 3: Algorithms

Instructor: Prof. Victor Sheng

Team Members:

Dhruv Maniar(R11713343)

Atharva Dalvi(R11765481)

Apoorv Rana(R11723071)

Alex Murangira(R11648884)

Computer Science, *Texas Tech University*

Table of Contents

[1. Project Description 1](#_Toc1244118283)

[1.1 ACQUIREMENT 1](#_Toc931109300)

[1.2 Data for Input 1](#_Toc524902518)

[1.3 Objectives 1](#_Toc1555461991)

[2. team contribution 1](#_Toc882307001)

[3. Interpretation of problem 1](#_Toc490811403)

[4. Methodology 1](#_Toc931887496)

[5. Screenshots 1](#_Toc168949224)

[6. Conclusions 1](#_Toc657985539)

# Project Description

The task involves determining the shortest path between two intersections on a city map or between a specified starting point and destination. The project employs the Dijkstra algorithm to compute the optimal shortest path between two locations. Additionally, the Bellman-Ford algorithm is utilized due to its versatility, particularly its capability to handle graphs featuring negative edge weights, in order to identify the shortest path.

## 1.1 ACQUIREMENT

In this group project, we are collectively responsible for implementing and testing both a greedy algorithm (Dijkstra’s Algorithm) and a dynamic programming algorithm (Bellman Ford), as they represent fundamental strategies in algorithm design. Given the widespread use of graphs in modeling real-world applications, we have the option to extend the prototype of the abstract data type (ADT) graph developed in a prior project.

## 1.2 Data for Input

In this project, we were provided with a word file which has all the school buildings and the distances between the school buildings.

## 1.3 Objectives

1. Simplify and solve real world problems.
2. Apply the existing algorithms into real world applications.
3. Design reasonable software solutions.
4. Implement and verify potential solutions.
5. Collaborate with team members.

# team contribution

|  |  |  |  |
| --- | --- | --- | --- |
| Team Members | Research | Implementation | Documentation |
| Atharva Dalvi | * Research on how the Linked list can be implemented. | * Extraction of course names and their prerequisites into the string and the LinkedList. * Adjacency list | * Outlining and explaining different sections of the report document and code documentation. |
| Dhruv Maniar | * Complete knowledge of DFS using stacks and post-visit numbers. | * Adjacency list, Graph * Topological sort using DFS | * Outlining and explaining different sections of the report document and code documentation. |
| Apoorv Rana | * Importing files and how pass by references and value works for Python. | * Graph | * Creating Different Classes to increase the readability and scalability. |
| Atharva Dalvi  Dhruv Maniar  Apoorv Rana  Alex Muraingira | * Usage of Dijkstra Algorithm * Usage of Bellman Ford Algorithm | * Graph | * Outlining and explaining different sections of the report document and code documentation. |

# Interpretation of problem

This project is based on a real-world application. Shortest path algorithms are applied to find the directions between physical locations, such as driving directions on websites like Google Maps. For this specific project, we are finding the shortest path to go to the other buildings from Computer Science buildings.

# Methodology

We have implemented a greedy algorithm (Dijkstra’s Algorithm) and a dynamic programming algorithm (Bellman Ford) to find the shortest path.

* **Dijkstra’s Algorithm:**

**Function Dijkstra(Graph, StartName):**

**Initialize distances dictionary with all buildings from Graph with value infinity**

**Initialize predecessors dictionary with all buildings from Graph with value None**

**Set distance of StartName in distances to 0**

**Create an unvisited set with all building names from Graph**

**While unvisited is not empty:**

**Set current to the unvisited building with the smallest distance**

**Remove current from unvisited**

**For each adjacent building and distance in adjacents of current building:**

**Calculate alternate route distance as sum of current distance and edge distance**

**If alternate route distance < distance to adjacent building:**

**Update distance to adjacent building**

**Update predecessor of adjacent building**

**Return distances and predecessors**

**A screen shot of a computer program

Description automatically generated**

* **Bellman- Ford Algorithm:**

**Function BellmanFord(Graph, StartName):**

**Initialize distances dictionary with all buildings from Graph with value infinity**

**Initialize predecessors dictionary with all buildings from Graph with value None**

**Set distance of StartName in distances to 0**

**For each building in Graph (repeat len(Graph.buildings) - 1 times):**

**For each building in Graph.buildings:**

**For each adjacent building and distance in building.adjacents:**

**If distance to adjacent building > distance to current building + edge distance:**

**Update distance to adjacent building**

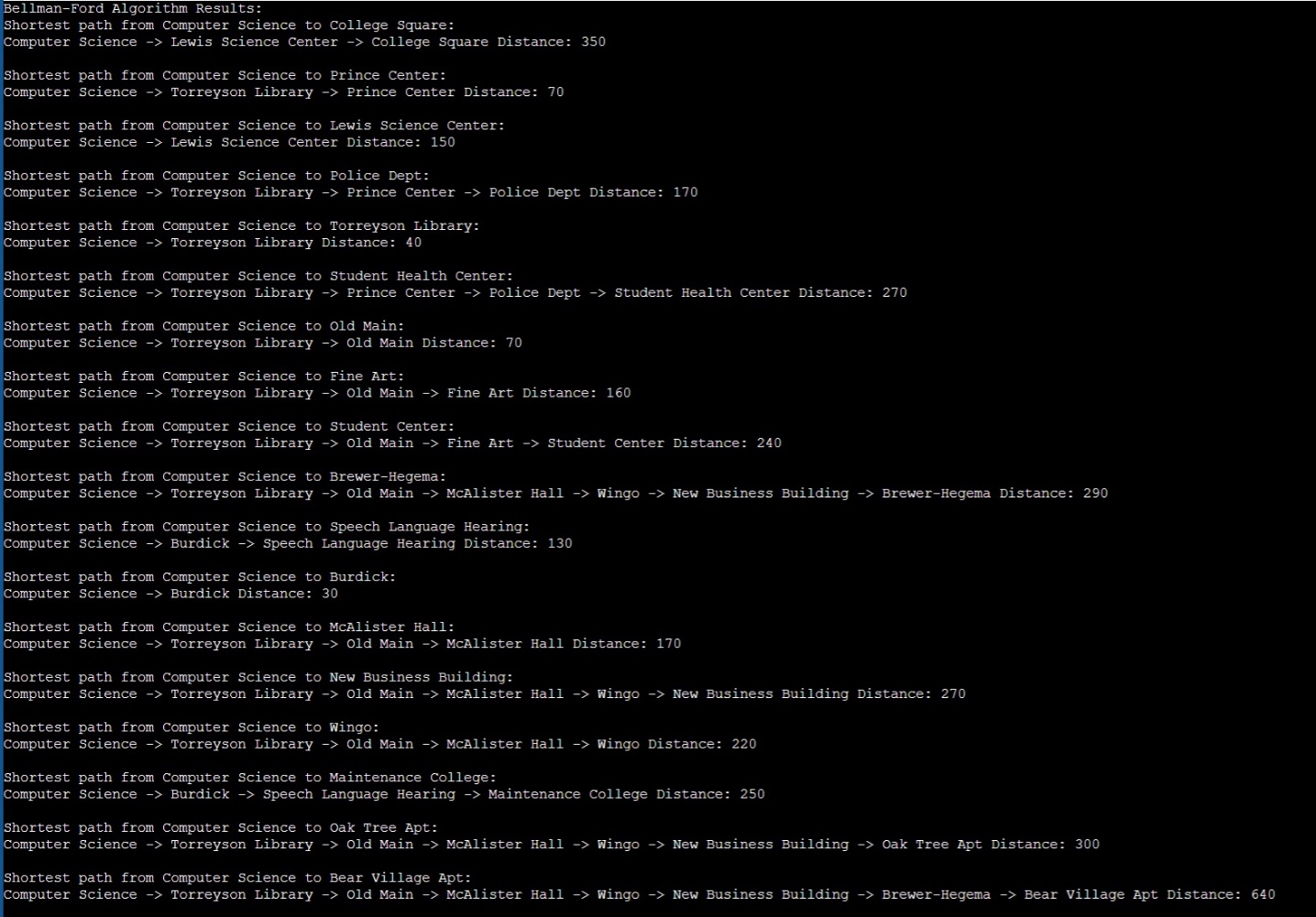
**Update predecessor of adjacent building**

**Return distances and predecessors**

A computer screen shot of text

Description automatically generated

# Screenshots



# Conclusions

Our project underscored the significance of algorithm selection in solving real-world problems. By experimenting with different sorting algorithms and analyzing source reliability, we gained insights into the practical implications of algorithmic efficiency and accuracy. The project's future direction involves optimizing the program for larger datasets and enhancing performance with more advanced data structures and programming techniques. This endeavor highlighted the dynamic nature of computer science, where continual improvement and adaptation are key to technological advancement.

