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

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.

2. 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.

3. INTERPRETATION OF PROBLEM

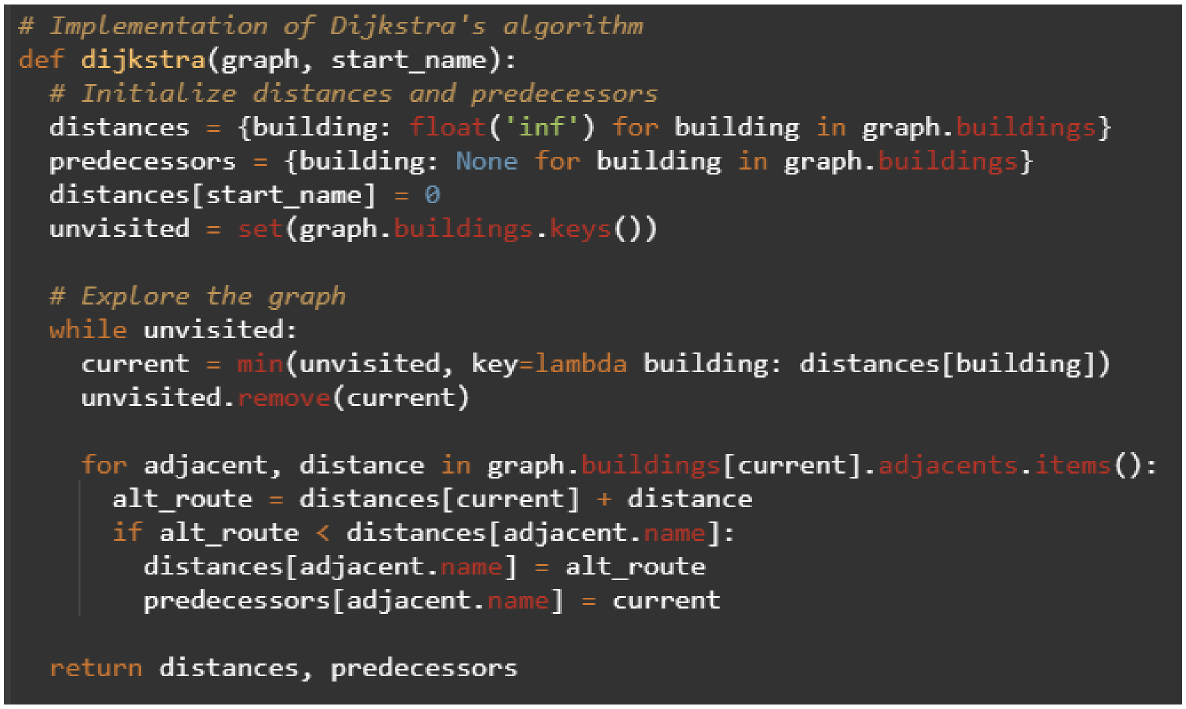
This project is based on a real-world application. Shortest path algorithms are applied to find the directions between physical locations, such as driving directions on websites like Google Maps. For this specific project, we are finding the shortest path to go to the other buildings from Computer Science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

4. 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
```

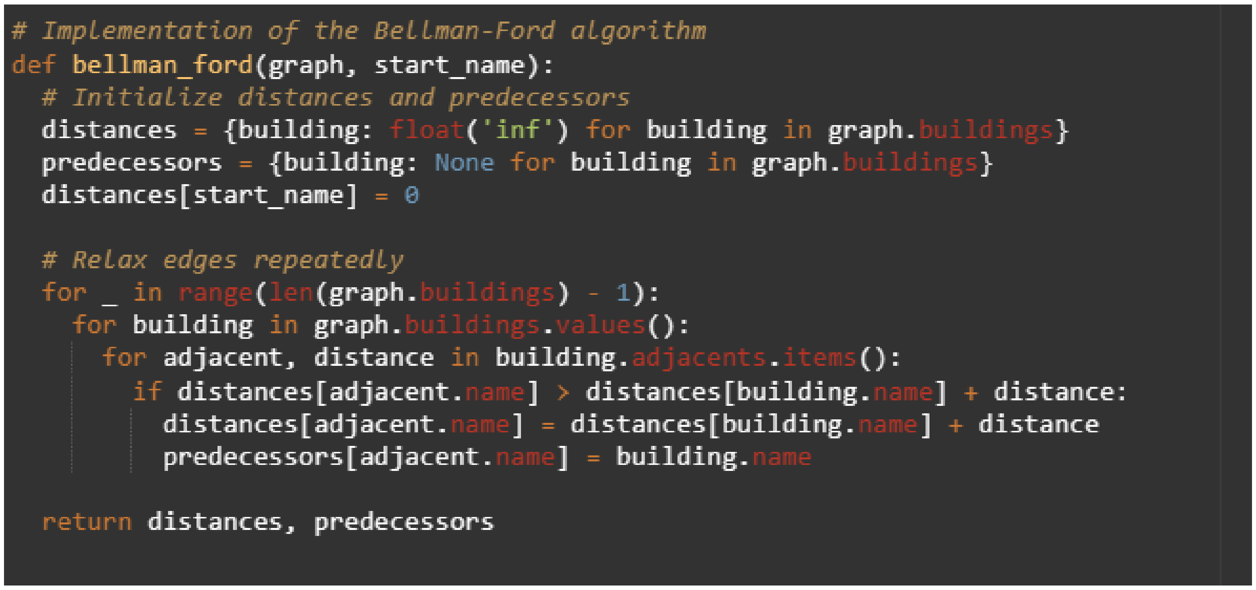


```python
# Implementation of Dijkstra's algorithm
def dijkstra(graph, start_name):
  # Initialize distances and predecessors
  distances = {building: float('inf') for building in graph.buildings}
  predecessors = {building: None for building in graph.buildings}
  distances[start_name] = 0
  unvisited = set(graph.buildings.keys())

  # Explore the graph
  while unvisited:
    current = min(unvisited, key=lambda building: distances[building])
    unvisited.remove(current)

    for adjacent, distance in graph.buildings[current].adjacents.items():
      alt_route = distances[current] + distance
      if alt_route < distances[adjacent.name]:
        distances[adjacent.name] = alt_route
        predecessors[adjacent.name] = current

  return distances, predecessors
```

- **Bellman- Ford Algorithm:**

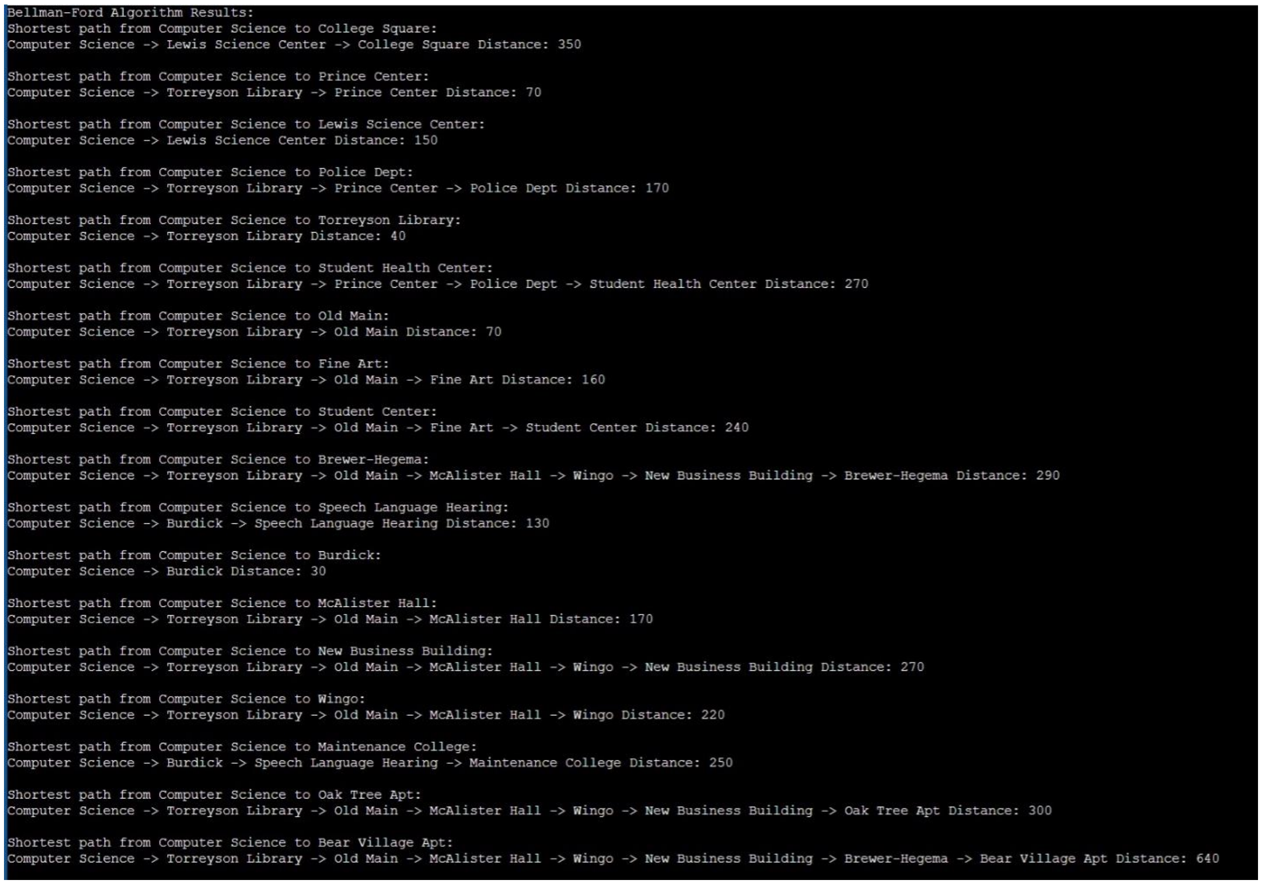
```
Function BellmanFord(Graph, StartName):
    Initialize distances dictionary with all buildings from Graph with value infinity
    Initialize predecessors dictionary with all buildings from Graph with value None
    Set distance of StartName in distances to 0

    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



```
# Implementation of the Bellman-Ford algorithm
def bellman_ford(graph, start_name):
  # Initialize distances and predecessors
  distances = {building: float('inf') for building in graph.buildings}
  predecessors = {building: None for building in graph.buildings}
  distances[start_name] = 0

  # Relax edges repeatedly
  for _ in range(len(graph.buildings) - 1):
    for building in graph.buildings.values():
      for adjacent, distance in building.adjacents.items():
        if distances[adjacent.name] > distances[building.name] + distance:
          distances[adjacent.name] = distances[building.name] + distance
          predecessors[adjacent.name] = building.name

  return distances, predecessors
```

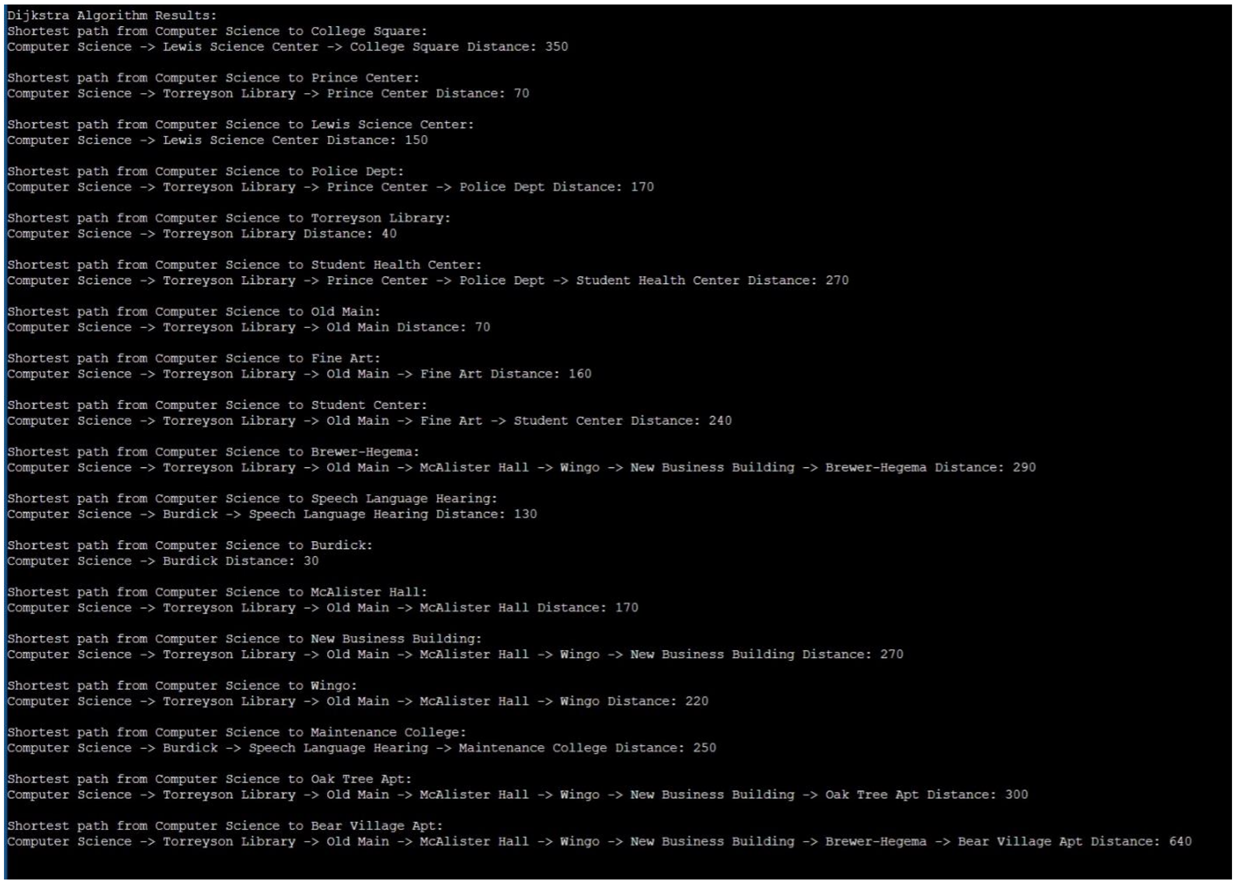
5. SCREENSHOTS



```
Bellman-Ford Algorithm Results:
Shortest path from Computer Science to College Square:
Computer Science -> Lewis Science Center -> College Square Distance: 350

Shortest path from Computer Science to Prince Center:
Computer Science -> Torreyson Library -> Prince Center Distance: 70

Shortest path from Computer Science to Lewis Science Center:
Computer Science -> Lewis Science Center Distance: 150

Shortest path from Computer Science to Police Dept:
Computer Science -> Torreyson Library -> Prince Center -> Police Dept Distance: 170

Shortest path from Computer Science to Torreyson Library:
Computer Science -> Torreyson Library Distance: 40

Shortest path from Computer Science to Student Health Center:
Computer Science -> Torreyson Library -> Prince Center -> Police Dept -> Student Health Center Distance: 270

Shortest path from Computer Science to Old Main:
Computer Science -> Torreyson Library -> Old Main Distance: 70

Shortest path from Computer Science to Fine Art:
Computer Science -> Torreyson Library -> Old Main -> Fine Art Distance: 160

Shortest path from Computer Science to Student Center:
Computer Science -> Torreyson Library -> Old Main -> Fine Art -> Student Center Distance: 240

Shortest path from Computer Science to Brewer-Hegema:
Computer Science -> Torreyson Library -> Old Main -> McAlister Hall -> Wingo -> New Business Building -> Brewer-Hegema Distance: 290

Shortest path from Computer Science to Speech Language Hearing:
Computer Science -> Burdick -> Speech Language Hearing Distance: 130

Shortest path from Computer Science to Burdick:
Computer Science -> Burdick Distance: 30

Shortest path from Computer Science to McAlister Hall:
Computer Science -> Torreyson Library -> Old Main -> McAlister Hall Distance: 170

Shortest path from Computer Science to New Business Building:
Computer Science -> Torreyson Library -> Old Main -> McAlister Hall -> Wingo -> New Business Building Distance: 270

Shortest path from Computer Science to Wingo:
Computer Science -> Torreyson Library -> Old Main -> McAlister Hall -> Wingo Distance: 220

Shortest path from Computer Science to Maintenance College:
Computer Science -> Burdick -> Speech Language Hearing -> Maintenance College Distance: 250

Shortest path from Computer Science to Oak Tree Apt:
Computer Science -> Torreyson Library -> Old Main -> McAlister Hall -> Wingo -> New Business Building -> Oak Tree Apt Distance: 300

Shortest path from Computer Science to Bear Village Apt:
Computer Science -> Torreyson Library -> Old Main -> McAlister Hall -> Wingo -> New Business Building -> Brewer-Hegema -> Bear Village Apt Distance: 640
```

6. 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.



```
Dijkstra Algorithm Results:
Shortest path from Computer Science to College Square:
Computer Science -> Lewis Science Center -> College Square Distance: 350

Shortest path from Computer Science to Prince Center:
Computer Science -> Torreyson Library -> Prince Center Distance: 70

Shortest path from Computer Science to Lewis Science Center:
Computer Science -> Lewis Science Center Distance: 150

Shortest path from Computer Science to Police Dept:
Computer Science -> Torreyson Library -> Prince Center -> Police Dept Distance: 170

Shortest path from Computer Science to Torreyson Library:
Computer Science -> Torreyson Library Distance: 40

Shortest path from Computer Science to Student Health Center:
Computer Science -> Torreyson Library -> Prince Center -> Police Dept -> Student Health Center Distance: 270

Shortest path from Computer Science to Old Main:
Computer Science -> Torreyson Library -> Old Main Distance: 70

Shortest path from Computer Science to Fine Art:
Computer Science -> Torreyson Library -> Old Main -> Fine Art Distance: 160

Shortest path from Computer Science to Student Center:
Computer Science -> Torreyson Library -> Old Main -> Fine Art -> Student Center Distance: 240

Shortest path from Computer Science to Brewer-Hegema:
Computer Science -> Torreyson Library -> Old Main -> McAlister Hall -> Wingo -> New Business Building -> Brewer-Hegema Distance: 290

Shortest path from Computer Science to Speech Language Hearing:
Computer Science -> Burdick -> Speech Language Hearing Distance: 130

Shortest path from Computer Science to Burdick:
Computer Science -> Burdick Distance: 30

Shortest path from Computer Science to McAlister Hall:
Computer Science -> Torreyson Library -> Old Main -> McAlister Hall Distance: 170

Shortest path from Computer Science to New Business Building:
Computer Science -> Torreyson Library -> Old Main -> McAlister Hall -> Wingo -> New Business Building Distance: 270

Shortest path from Computer Science to Wingo:
Computer Science -> Torreyson Library -> Old Main -> McAlister Hall -> Wingo Distance: 220

Shortest path from Computer Science to Maintenance College:
Computer Science -> Burdick -> Speech Language Hearing -> Maintenance College Distance: 250

Shortest path from Computer Science to Oak Tree Apt:
Computer Science -> Torreyson Library -> Old Main -> McAlister Hall -> Wingo -> New Business Building -> Oak Tree Apt Distance: 300

Shortest path from Computer Science to Bear Village Apt:
Computer Science -> Torreyson Library -> Old Main -> McAlister Hall -> Wingo -> New Business Building -> Brewer-Hegema -> Bear Village Apt Distance: 640
```