Green University of Bangladesh

Department of Computer Science and Engineering (CSE)
Semester: (Fall, Year: 2023), B.Sc. in CSE (Day)

Path Finding and Navigation System

Course Title: Algorithm Lab
Course Code: 221 D11
Section: CSE 206

Students Details

Name	ID
Md. Mazharul Islam Shehab	221002534

Submission Date: 22.11.23
Course Teacher's Name: Mr. Montaser Abdul Quader

[For teachers use only: Don't write anything inside this box]

Lab Project Status	
Marks:	**Signature:**
Comments:	**Date:**

Contents

Chapter 1

Introduction

1.1 Overview

In this project, I'll try to create a pathfinding system that calculates the shortest path between two points on a map by applying Dijkstra's algorithm. Algorithm coding, graph representation, and possibly user interface integration for interaction and visualization are all part of implementation.

1.2 Motivation

I picked this project to broaden my knowledge of graph algorithms and how they are used in practical settings like navigational systems. I can explore complex problem-solving while improving my coding skills by creating a pathfinding tool. The project's significance lies in its capacity to develop effective routing solutions, which could have an impact on GPS technology, gaming, logistics, and other industries.

1.3 Problem Definition

1.3.1 Problem Statement

The goal is to solve the problem of locating the best routes between locations on a map by creating a reliable pathfinding and navigation system with Dijkstra's algorithm. The project aims to address real-world navigation challenges by improving route planning in a variety of industries, including gaming, logistics, and transportation, with a focus on efficient graph traversal. The challenge is developing an algorithmic solution that can determine the shortest routes, minimize travel time, and enhance users' overall navigational experiences in a variety of situations.

1.4 Design Goals/Objectives

My projects objectives are given below:

- Effective Pathfinding: Create a system that can determine the shortest path between any two points on a map in a timely manner .
- User-Friendly Interface: Provide a user-friendly interface so that users can interact with the system, enter beginning and ending locations, see routes, and comprehend how to navigate.
- Algorithmic Accuracy: Verify that the algorithm computes optimal routes accurately, taking into account edge cases, obstacles, or map modifications without sacrificing efficiency.
- Testing and Validation: To confirm the system's precision, effectiveness, and dependability in identifying the best routes, carry out thorough testing across a range of map configurations and scenarios.

1.5 Application

The pathfinding and navigation system utilizing Dijkstra's algorithm holds significant applications across various real-world scenarios:

- Logistics and Transportation
- GPS and Navigation Services
- Gaming and Virtual Environments
- Network Routing
- Robotics and Autonomous Vehicles
- Urban Planning and Emergency Services
- Healthcare Navigation

Chapter 2

Design/Development/Implementation of the Project

2.1 Introduction

With a grid-based graphical user interface, this Java-based project provides an interactive navigation and pathfinding system. On the grid, users can add start/finish locations, barriers, and erase parts in addition to choosing between the Dijkstra and A* algorithms. Via the generation of random obstacles and the manipulation of grid size, speed, and density, it facilitates the investigation of many scenarios. The program gives users a hands-on, user-friendly experience with pathfinding by visually representing the algorithm's exploration and shortest path.

2.2 Project Details

The project uses the Swing library to create a graphical user interface for a Java-based pathfinding and navigation system. It makes Dijkstra and A* algorithm experimentation easier in a grid-based setting.

2.2.1 Key Features

- Algorithm Selection: Users can choose between Dijkstra and A* algorithms for pathfinding.
- Interactive Grid: The grid-based canvas enables placing start, finish, walls, and erasing elements using mouse interactions.
- Map Generation/Clearing: Functionality to randomly generate obstacles and reset the map for different scenarios.
- User Control: Sliders for adjusting grid size, speed/delay, and obstacle density provide user control.

- Visual Representation: The program visually represents the exploration, visited nodes, and shortest path on the grid.

2.2.2 Components

- PathFinding Class: Main class managing the GUI, algorithm execution, map generation, and user interactions.
- Node Class: Represents nodes on the grid with properties like type, hops, coordinates, and methods for exploration.
- Map Class: Handles graphical representation and interaction on the grid canvas using Swing's JPanel.

2.2.3 Functionality Breakdown

- Initialization: Sets up the GUI, UI components, event listeners, and initializes the grid environment.
- Map Interaction: Allows users to interact with the grid by placing start, finish, walls, or erasing elements.
- Algorithm Execution: Implements Dijkstra and A* algorithms for pathfinding and exploration on the grid.
- User Control: Sliders and dropdowns enable users to control grid parameters, algorithm selection, and map generation.
- Visualization: Utilizes the Graphics class to visualize the exploration, visited nodes, and shortest path on the grid.

2.3 Implementation

2.3.1 The workflow

- Initialization and Setup: The `PathFinding` class initializes the JFrame, sets up UI components, and creates the grid canvas (`'Map'`).
- User Interaction: Users interact with the grid by placing start/finish points, walls, and triggering actions using buttons and sliders.
- Algorithm Execution: Upon user request, the selected algorithm (Dijkstra or A*) explores the grid, finding the shortest path.
- Visualization: The grid updates in real-time, showing explored nodes, path progression, and the final shortest path.
- Control and Adjustment: Users control grid parameters (size, speed, density) to observe different scenarios and behaviors of algorithms.

2.3.2 Tools and Libraries

- Java: Core language used for development.
- Swing Library: Utilized for creating the graphical user interface, including buttons, sliders, and canvas.
- Graphics: Leveraged to paint and update the grid's visual representation based on algorithmic exploration.
-

Figure 2.1: Java Language

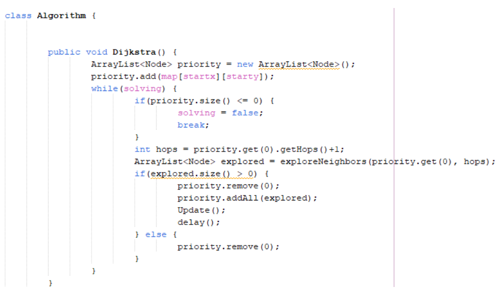
Figure 2.2: Netbeans

2.4 Algorithms

Algorithm 1: Sample Algorithm

```
Input: Your Input
Output: Your output
Data: Testing set x
1 function Dijkstra(Graph, source): dist[source] := 0 for each vertex v in Graph:
    if v  source dist[v] := infinity add v to Q
2 while Q is not empty: // The main loop v := vertex in Q with min dist[v]
    remove v from Q
3 for each neighbor u of v: // where neighbor u has not yet been removed from Q.
    alt := dist[v] + length(v, u) if alt < dist[u]: // A shorter path to u has been
    found dist[u] := alt // Update distance of u
4 return dist[] end function
```

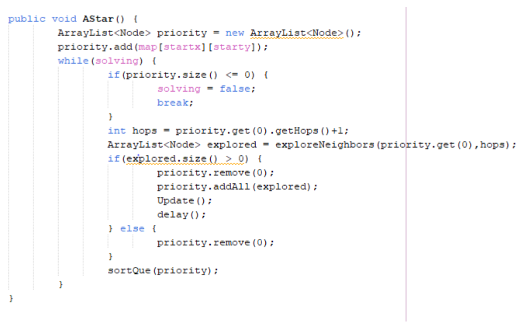
Implementation details (with screenshots and programming codes)



```
class Algorithm {


        public void Dijkstra() {
                ArrayList<Node> priority = new ArrayList<Node>();
                priority.add(map[startx][starty]);
                while(solving) {
                        if(priority.size() <= 0) {
                                solving = false;
                                break;
                        }
                        int hops = priority.get(0).getHops()+1;
                        ArrayList<Node> explored = exploreNeighbors(priority.get(0), hops);
                        if(explored.size() > 0) {
                                priority.remove(0);
                                priority.addAll(explored);
                                Update();
                                delay();
                        } else {
                                priority.remove(0);
                        }
                }
        }
```

Figure 2.3: Implementation-1



```
public void AStar() {
        ArrayList<Node> priority = new ArrayList<Node>();
        priority.add(map[startx][starty]);
        while(solving) {
                if(priority.size() <= 0) {
                        solving = false;
                        break;
                }
                int hops = priority.get(0).getHops()+1;
                ArrayList<Node> explored = exploreNeighbors(priority.get(0),hops);
                if(explored.size() > 0) {
                        priority.remove(0);
                        priority.addAll(explored);
                        Update();
                        delay();
                } else {
                        priority.remove(0);
                }
                sortQue(priority);
        }
}
```

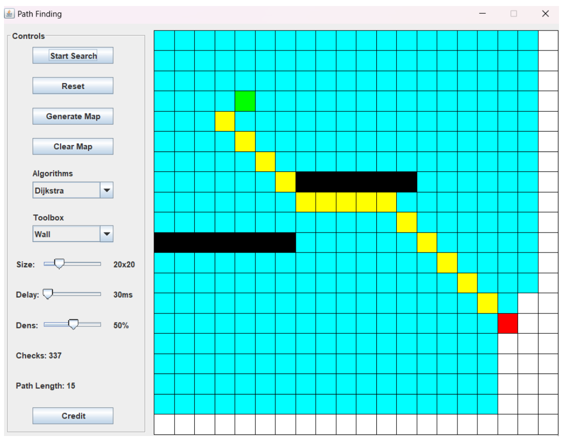
Figure 2.4: Implementation-2

Chapter 3

Performance Evaluation

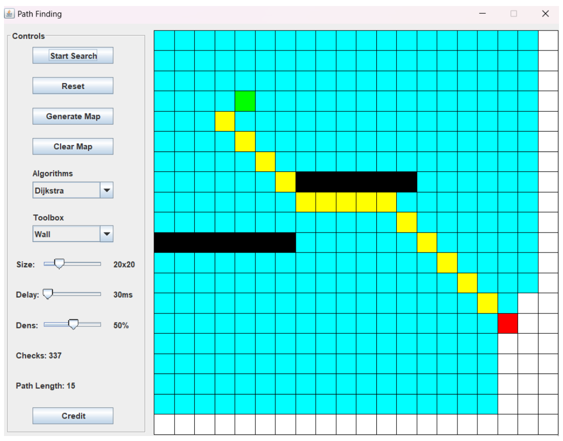
3.1 Results Analysis/Testing

3.1.1 Result_portion_1



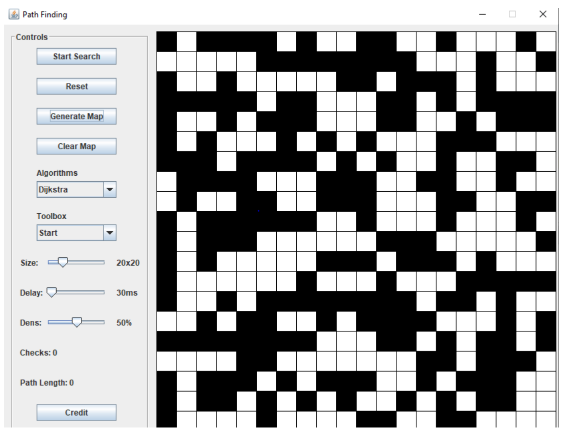
Algorithms: Dijkstra
Toolbox: Wall
Size: 20x20
Delay: 30ms
Dens: 50%
Checks: 337
Path Length: 15

Figure 3.1: Output 1



Algorithms: Dijkstra
Toolbox: Wall
Size: 20x20
Delay: 30ms
Dens: 50%
Checks: 337
Path Length: 15

Figure 3.2: Output



Algorithms: Dijkstra
Toolbox: Start
Size: 20x20
Delay: 30ms
Dens: 50%
Checks: 0
Path Length: 0

Figure 3.3: Output 2

Chapter 4

Conclusion

4.1 Discussion

This project presents a Java-based pathfinding and navigation system using Swing for the graphical interface. It offers users the ability to interactively explore pathfinding algorithms like Dijkstra and A* within a grid-based environment. The system allows for placing start/finish points, obstacles, and adjusting grid parameters, providing a hands-on experience with pathfinding concepts. Throughout exploration, users witness real-time updates of algorithmic progress, visually represented on the grid. The results showcase successful navigation from start to finish, highlighting the shortest path found by the algorithms. Observations reveal the efficiency and effectiveness of each algorithm in finding optimal paths, while also demonstrating the impact of grid parameters on algorithmic behavior and pathfinding outcomes. Overall, the project serves to understand and experiment with pathfinding algorithms in a visually intuitive manner.

4.2 Limitations

While this Pathfinding and Navigation System offers an engaging way to explore Dijkstra and A* algorithms within a grid-based environment, it has limitations that impact its depth and applicability. The project's reliance on a simplified grid representation limits its real-world applicability, lacking complexity to handle dynamic obstacles or varied terrains. Additionally, the visualization's simplicity may hinder a deeper understanding of algorithmic steps, and the project's dependency on grid parameters for algorithm behavior exploration might limit its scope. Expanding algorithm options, incorporating more sophisticated visualizations, handling more varied terrains, and offering deeper educational context could address these limitations, enhancing the system's educational and practical value

4.3 Scope of Future Work

In future I will expand this Pathfinding and Navigation System that involves integrating a wider array of algorithms such as BFS, DFS, and more advanced A* variants to offer users a comprehensive understanding of pathfinding strategies. Enhancing the grid system to handle complex terrains, dynamic obstacles, and varying traversal costs would simulate real-world scenarios more accurately. Advanced visualizations showcasing step-by-step algorithmic processes, along with an in-depth analysis of parameter sensitivity, would deepen user comprehension. Moreover, augmenting educational resources within the system, optimizing user interface elements for larger grids, and aiming for real-time adaptability in applications would significantly enhance the project's educational and practical value, catering to a broader audience

Chapter 5

Reference

https://www.javatpoint.com/java-swing
https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/
https://www.programiz.com/dsa/dijkstra-algorithm
https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/a-star-algorithm#:~:text=The%20A*%20algorithm%20is%20widely,obstacles%20and%20find%20optimal%20paths.
https://www.geeksforgeeks.org/a-search-algorithm/