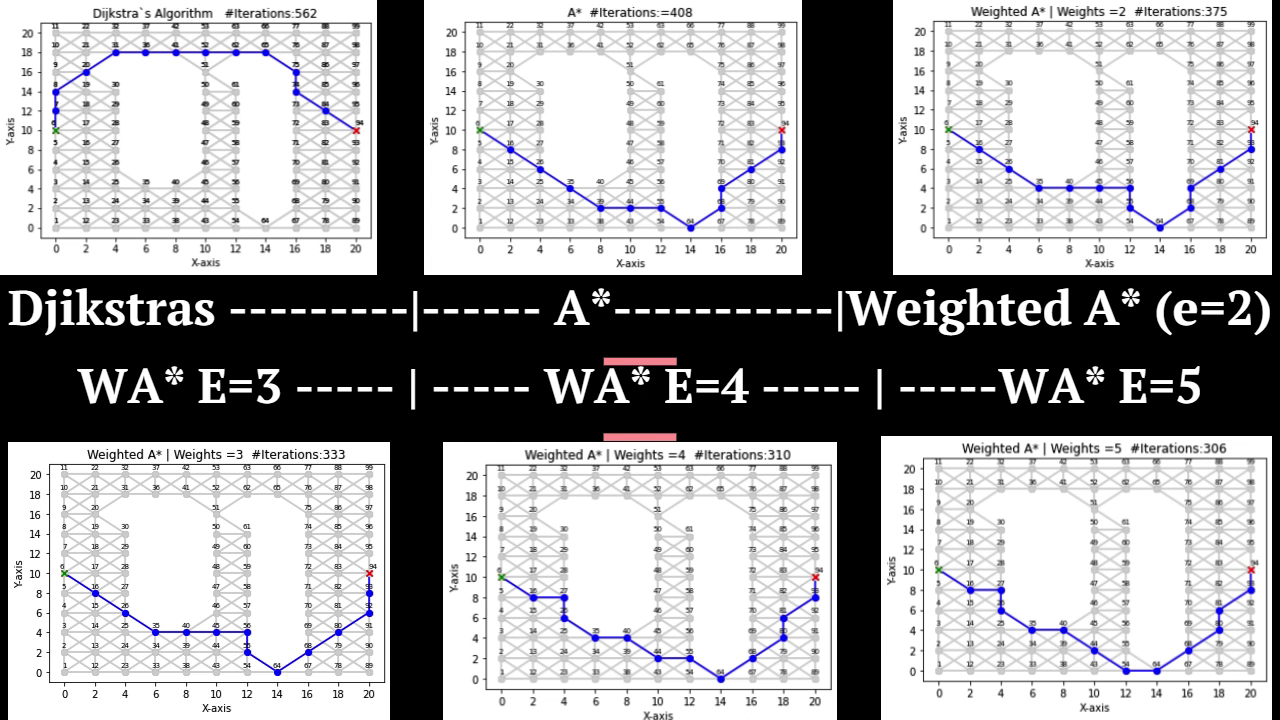
Assignment 2

**Implementation Report for Shortest Path Finding Program**

**Results:**

****

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Algorithm** | **Dijkstaras** | **A\*** | **WA e=2** | **WA e=3** | **WA e=4** | **WA e=5** |
| **Final Cost** | 30.1421 | 30.6274 | 31.7990 | 31.7990 | 31.7990 | 31.7990 |
| **Iteration** | 562 | 408 | 375 | 333 | 310 | 306 |

**Files**

* **input.txt**: Contains details about the graph (number of vertices, starting and target vertices).
* **coords.txt**: Holds the coordinates of graph vertices.
* **FormattedInput.txt**: Contains organized input data.
* **output.txt**: The output file where results are stored.

**1. Input Processing**

* Reads **input.txt** to get important details like the number of vertices, starting point, and target point.

**2. Coordinate Handling**

* Takes coordinates from **coords.txt** and stores them for later use in plotting.

**3. Graph Initialization**

* Sets up the graph structure and fills it with edges from **FormattedInput.txt**.

**4. Dijkstra's Algorithm**

* The program employs this algorithm to find the shortest path from the starting point to all other points.

**5. Weighted A\* Algorithm**

* Offers an alternative algorithm that can be influenced by different weightings.

**6. Heuristic Dictionary**

* Calculates heuristic distances for the Weighted A\* algorithm.

**7. Plotting and Visualization**

* Uses **matplotlib** to create visual representations of the graph and algorithm steps.

**8. Output File Generation**

* Adds results to **output.txt**, including the path and distances.

**Running the Program**

* Make sure **input.txt**, **FormattedInput.txt**, **coords.txt**, and **output.txt** are available.
* Run the program to see the algorithms in action.
* The output will be saved in **output.txt**.

**Usage Examples**

* This program is great for learning about graphs and algorithms. You can experiment with different setups to see how the algorithms behave.

**Conclusion**

This Python program provides an easy-to-follow demonstration of two methods for finding the shortest path in a graph. The combination of clear visuals and detailed output makes it a valuable learning tool.

**Dijkstra's Algorithm (dijkstra)**

Function Signature

def dijkstra(self, start):

Description

This function implements Dijkstra's algorithm to find the shortest path from a given starting vertex to all other vertices in the graph.

Parameters

* **self**: Refers to the current instance of the **Graph** class.
* **start**: The starting vertex for the algorithm.

Returns

* **dist**: A dictionary containing the shortest distances from the starting vertex to all other vertices.
* **parent**: A dictionary containing the parent vertices that lead to the shortest path.

**Shortest Path with Distances (shortest\_path\_with\_distances)**

Function Signature

def shortest\_path\_with\_distances(self, start, end, parent):

Description

This function calculates the shortest path and distances from the starting vertex to the target vertex using the parent vertices obtained from Dijkstra's algorithm.

Parameters

* **self**: Refers to the current instance of the **Graph** class.
* **start**: The starting vertex.
* **end**: The target vertex.
* **parent**: A dictionary containing the parent vertices obtained from Dijkstra's algorithm.

Returns

* **path**: A list representing the shortest path from the starting vertex to the target vertex.
* **distances\_between**: A dictionary containing distances between vertices in the shortest path.

**Weighted A\* Algorithm (weighted\_a\_star)**

Function Signature

def weighted\_a\_star(self, start, weight=1):

Description

This function implements the weighted A\* algorithm to find the shortest path from a given starting vertex to all other vertices in the graph. It allows for the influence of different weights on the search process.

Parameters

* **self**: Refers to the current instance of the **Graph** class.
* **start**: The starting vertex for the algorithm.
* **weight**: A multiplier influencing the heuristic function (default is 1).

Returns

* **dist**: A dictionary containing the shortest distances from the starting vertex to all other vertices.
* **parent**: A dictionary containing the parent vertices that lead to the shortest path.