Hello There User!

This program demonstrates Dijkstra's Algorithm for finding the shortest path and Prims Algorithm for finding the minimum spanning tree. The algorithms will be used to search through a randomly generated sensor network.

Upon execution, you will be prompted to enter information for the network.

- **Width** -> The width of the area in which the sensors will be located
- **Height** -> The height of the area in which the sensors will be located.
- **Depth** -> The depth of the area in which the sensors will be located.
- **Number of Nodes** -> The number of nodes to generate.
- **Threshold** -> The transmission range of each node.
- **Number of Data Nodes**-> The number of nodes to mark as storage depleted.
- **Number of Data Packets**-> The number of packets each Data Node has. Same for all nodes.
- **Storage Capacity**-> The storage capacity of each Storage Node.

After providing the required data, the program will first check if there is enough storage space available in the network. If not, you will be notified and prompted to provide new data. To ensure that the data generated does not exceed the storage capacity, lower the data packets for each data node, or lower the number of data nodes.
The program will then perform a Depth First Search on the network to determine connectivity. If the graph is not fully connected, you will be prompted to provide new data for the network. To ensure connectivity, increase the transmission range of each node.

If the network is connected, you will be provided with a list that contains the ID's and locations of each node and prompted to choose a starting and destination node. Choose a starting node and a destination node from that list.

You will then be prompted to choose a traversal algorithm.

Enter "1" for Dijkstra Algorithm .
OR
Enter "2" for Prim Algorithm.

The program will then calculate and output the shortest path using Dijkstra's Algorithm or the minimum spanning tree using Prim's Algorithm.

EXECUTION EXAMPLES

1) **Not enough storage in network (Unconnected Network).**
   a) **Data Nodes > Number of Nodes**

i)

```
======================================================================
        Hello,
        This Program Calculates the shortest path between
        two nodes using dijkstra's or prims algorithm.
======================================================================

Enter Width (Integer): 1000
Enter Height (Integer): 1000
Enter Depth (Integer): 0
Enter number of Nodes (Integer): 30
Enter Threshhold (Integer): 600
Enter number of  Data Nodes (Integer): 35
Data Nodes greater than total number of nodes.
Number of Data Nodes needs to be <= 30

Enter number of  Data Nodes (Integer):
```

ii)

```
Enter number of  Data Nodes (Integer): 50
Data Nodes greater than total number of nodes.
Number of Data Nodes needs to be <= 30

Enter number of  Data Nodes (Integer):
```

iii)

```
Enter number of  Data Nodes (Integer): 12
Enter number of data packets for each Data Node:
```

iv) **Data Nodes* Data Packet > Storage Capacity**

```
Enter Width (Integer): 100
Enter Height (Integer): 100
Enter Depth (Integer): 0
Enter number of Nodes (Integer): 30
Enter Threshhold (Integer): 90
Enter number of  Data Nodes (Integer): 12
Enter number of Data Packets for each Data Node: 500
Enter Storage Capacity of each Storage Node: 100
There is not enough storage in the network.
Please provide new data for:
 Number of Data Nodes...
 Number of Data Packets...
 and Storage Capcity of each Storage Node...

Enter number of  Data Nodes (Integer):
```

2) **Transmission range too low (Unconnected Network).**

a)

```
======================================================================
        Hello,
        This Program Calculates the shortest path between
        two nodes using dijkstra's or prims algorithm.
======================================================================

Enter Width (Integer): 100
Enter Height (Integer): 100
Enter Depth (Integer): 0
Enter number of Nodes (Integer): 20
Enter Threshhold (Integer): 5
Enter number of  Data Nodes (Integer): 5
Enter number of Data Packets for each Data Node: 100
Enter Storage Capacity of each Storage Node: 150
----------------------------------------------------------------------

Node ID        Node Type      Location
----------------------------------------------------------------------
0              Storage Node   (47.34,0.14,0.00)
1              Data Node      (92.85,57.45,0.00)
2              Data Node      (71.07,0.17,0.00)
3              Data Node      (88.06,27.44,0.00)
4              Data Node      (39.46,58.79,0.00)
5              Data Node      (87.60,97.59,0.00)
6              Storage Node   (74.73,69.26,0.00)
7              Storage Node   (35.03,15.31,0.00)
8              Storage Node   (82.43,27.60,0.00)
9              Storage Node   (81.67,30.60,0.00)
10             Storage Node   (60.10,99.45,0.00)
11             Storage Node   (94.63,60.86,0.00)
12             Storage Node   (66.92,61.51,0.00)
13             Storage Node   (74.59,78.41,0.00)
14             Storage Node   (54.94,83.84,0.00)
15             Storage Node   (71.39,73.31,0.00)
16             Storage Node   (93.22,89.81,0.00)
17             Storage Node   (60.62,43.82,0.00)
18             Storage Node   (31.54,29.78,0.00)
19             Storage Node   (69.73,54.05,0.00)
==================================
Graph is not connected, please input new values. Aim for a tranmission range around 80% of
 the maximum between graph width,height, and depth
Retrying......
Enter Width (Integer):
```

3) **Successful Run**

```
==================================================================
        Hello,
        This Program Calculates the shortest path between
        two nodes using dijkstra's or prims algorithm.
==================================================================

Enter Width (Integer): 200
Enter Height (Integer): 200
Enter Depth (Integer): 0
Enter number of Nodes (Integer): 20
Enter Threshhold (Integer): 120
Enter number of  Data Nodes (Integer): 6
Enter number of Data Packets for each Data Node: 5
Enter Storage Capacity of each Storage Node: 10
------------------------------------------------------------------

Node ID        Node Type      Location
------------------------------------------------------------------
0              Data Node      (62.07,19.98,0.00)
1              Data Node      (55.99,20.27,0.00)
2              Data Node      (141.12,83.75,0.00)
3              Data Node      (154.00,48.39,0.00)
4              Data Node      (160.80,161.14,0.00)
5              Data Node      (173.55,106.72,0.00)
6              Data Node      (75.88,59.87,0.00)
7              Sensor Node    (163.19,28.73,0.00)
8              Sensor Node    (4.70,160.41,0.00)
9              Sensor Node    (156.10,31.72,0.00)
10             Sensor Node    (18.17,187.40,0.00)
11             Sensor Node    (20.96,107.34,0.00)
12             Sensor Node    (63.15,87.59,0.00)
13             Sensor Node    (18.86,14.16,0.00)
14             Sensor Node    (27.08,179.88,0.00)
15             Sensor Node    (48.40,115.26,0.00)
16             Sensor Node    (58.71,88.10,0.00)
17             Sensor Node    (56.46,192.90,0.00)
18             Sensor Node    (102.19,183.50,0.00)
19             Sensor Node    (107.35,103.79,0.00)

##################################################################################################################
Network Successfully Generated...
        Each node has a random x and y location. The list of nodes will be organized based on Node type.
        If there are any Data Nodes, they will be listed first for clarity.

##################################################################################################################



------------------------------
CONNECTION STATUS: CONNECTED
------------------------------



Enter 0 for Dijkstra.
Enter 1 for Bellman-Ford.
Enter 2 for Prim        ->choice: 0
Please choose Starting Node and Destination Node.
^^^^Refer to the above printout for Node ID^^^^
```

a)

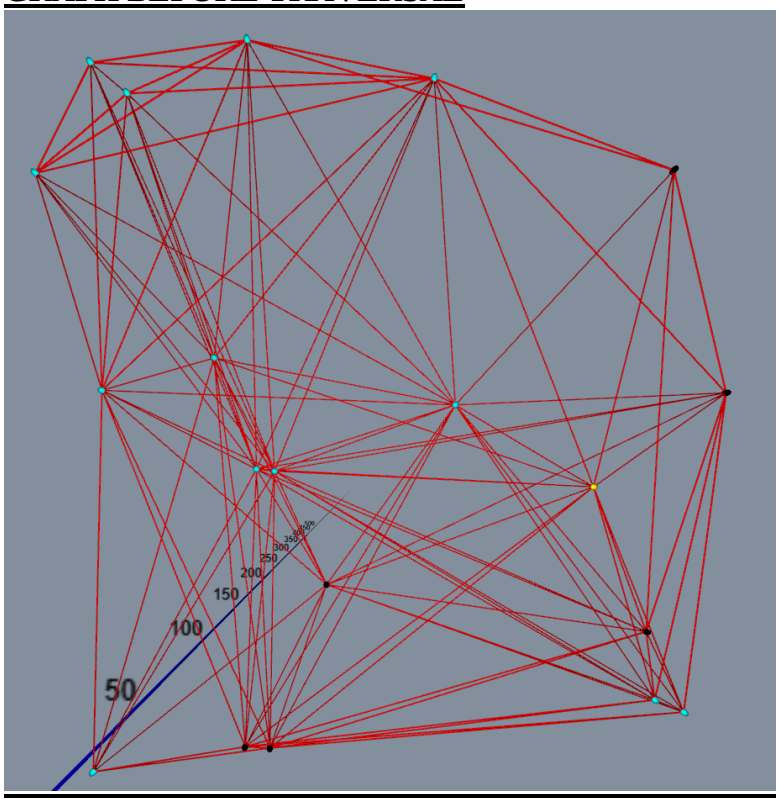
i) Dijkstra's Algorithm

```
------------------------------
CONNECTION STATUS: CONNECTED
------------------------------



Enter 0 for Dijkstra.
Enter 1 for Bellman-Ford.
Enter 2 for Prim        ->choice: 0
Please choose Starting Node and Destination Node.
^^^^Refer to the above printout for Node ID^^^^

Starting Node ID: 2
Destination Node ID: 18
```

ii)

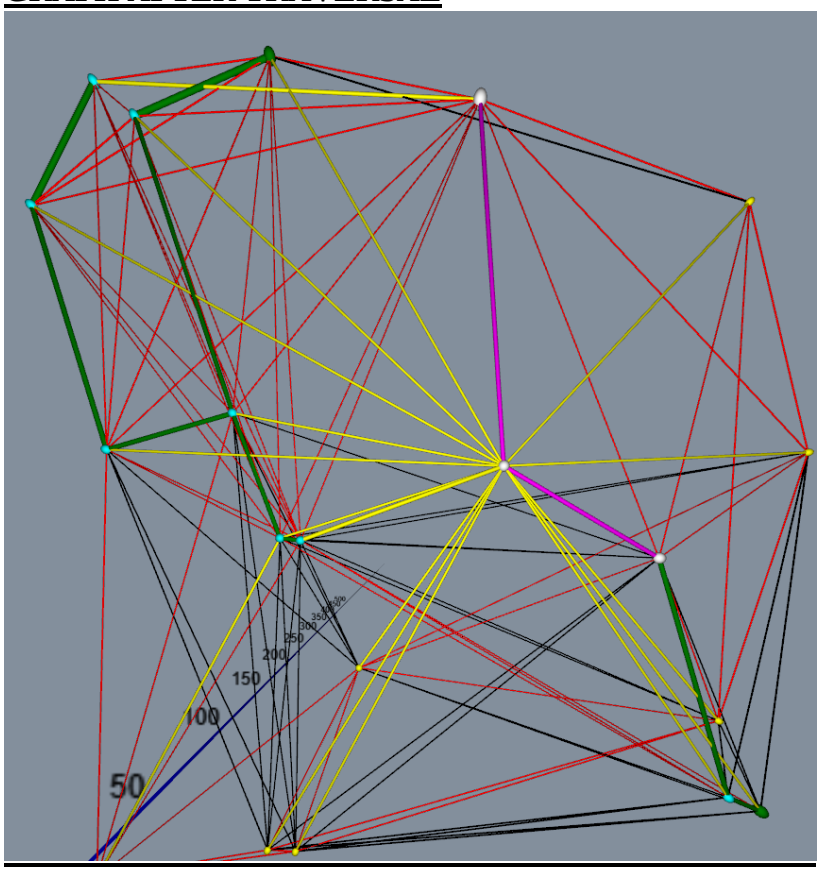
```
#########################################################################################################################
Destination Node Found.....
----------------------------
               Shortest Path To DN_2  from SNode_18
=======================================================================
        Vertices                                Edge Cost
=======================================================================
[DN_2 ,SNode_19 ]                               772197.46

[SNode_19 ,SNode_18 ]                           3963752.72

TOTAL ENERGY COST: 3963752.7231088104

#########################################################################################################################
        All  VISITED EDGES
=======================================================================
        Vertices                                Edge Cost
=======================================================================

[SNode_9 ,SNode_7 ]                             1497404.30

[SNode_11 ,SNode_8 ]                            4265164.24

[SNode_15 ,SNode_11 ]                           2724104.20

[SNode_12 ,SNode_16 ]                           1892125.78

[DN_2 ,SNode_9 ]                                1466797.60

[SNode_15 ,SNode_14 ]                           4631461.89

[SNode_16 ,SNode_15 ]                           2315266.12

[SNode_19 ,SNode_18 ]                           3963752.72

[SNode_14 ,SNode_17 ]                           5148771.93

[DN_2 ,SNode_19 ]                               772197.46

[SNode_19 ,SNode_12 ]                           1881140.25

[SNode_8 ,SNode_10 ]                            4721160.64
```

iii) **GRAPH BEFORE TRAVERSAL**



iv)

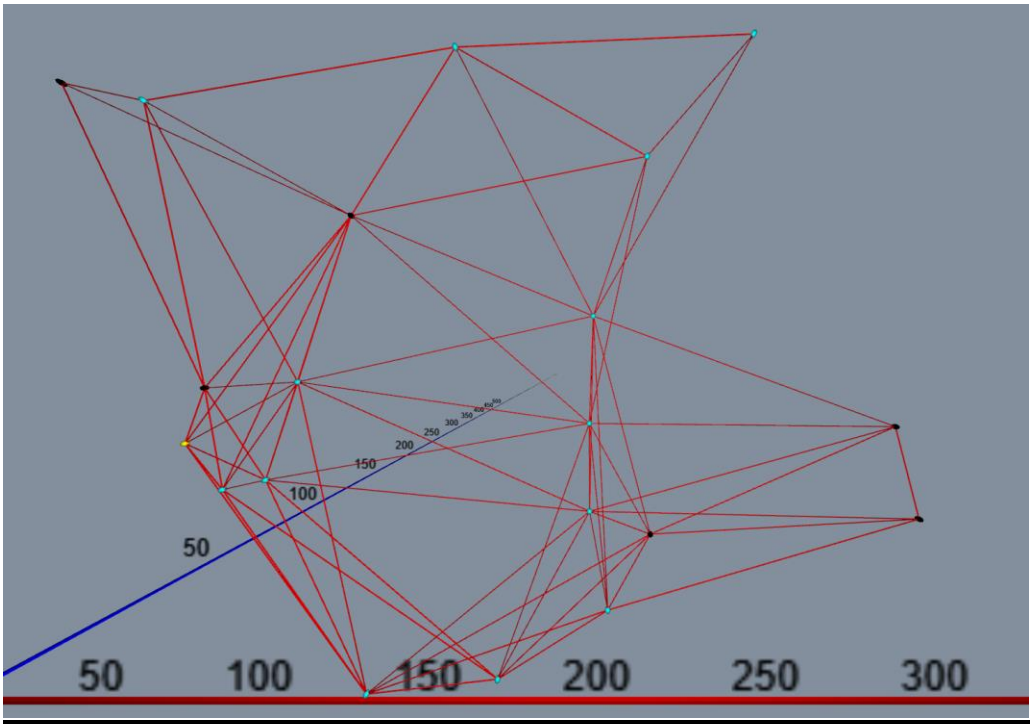
v) **GRAPH AFTER TRAVERSAL**



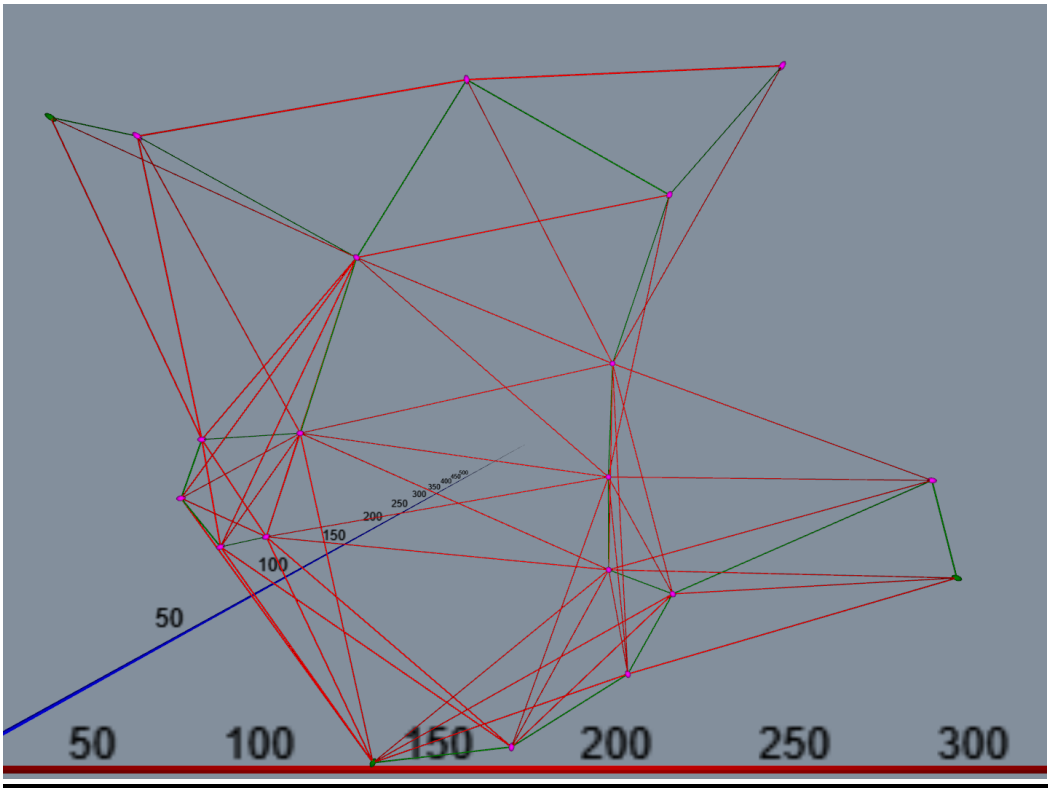
vi)

b) Prim's Algorithm

GRAPH BEFORE TRAVERSAL

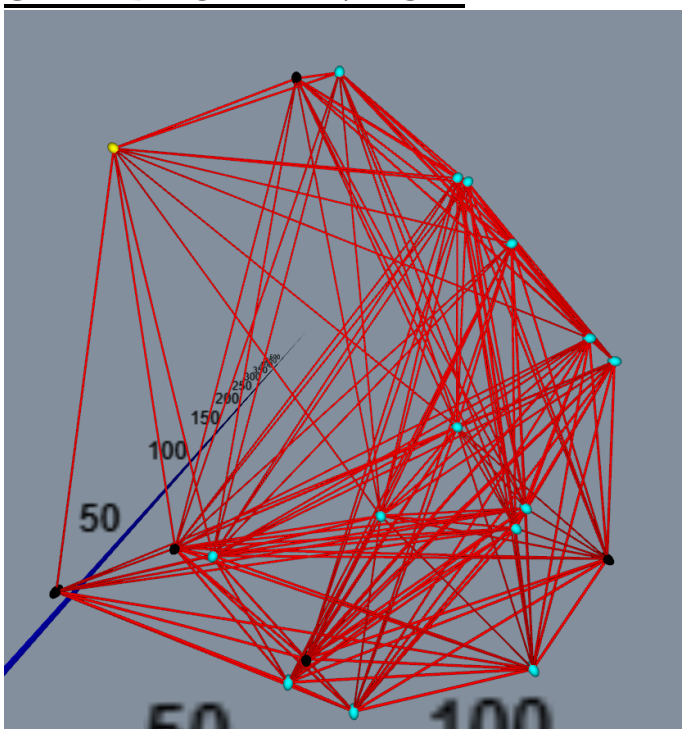


GRAPH AFTER TRAVERSAL

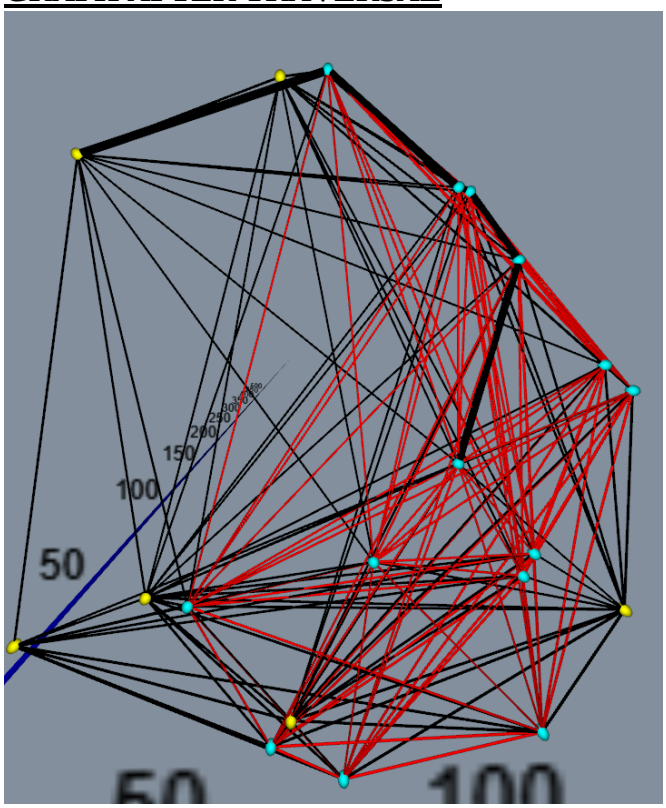


c) **Bellman-Ford**

GRAPH BEFORE TRAVERSAL



GRAPH AFTER TRAVERSAL



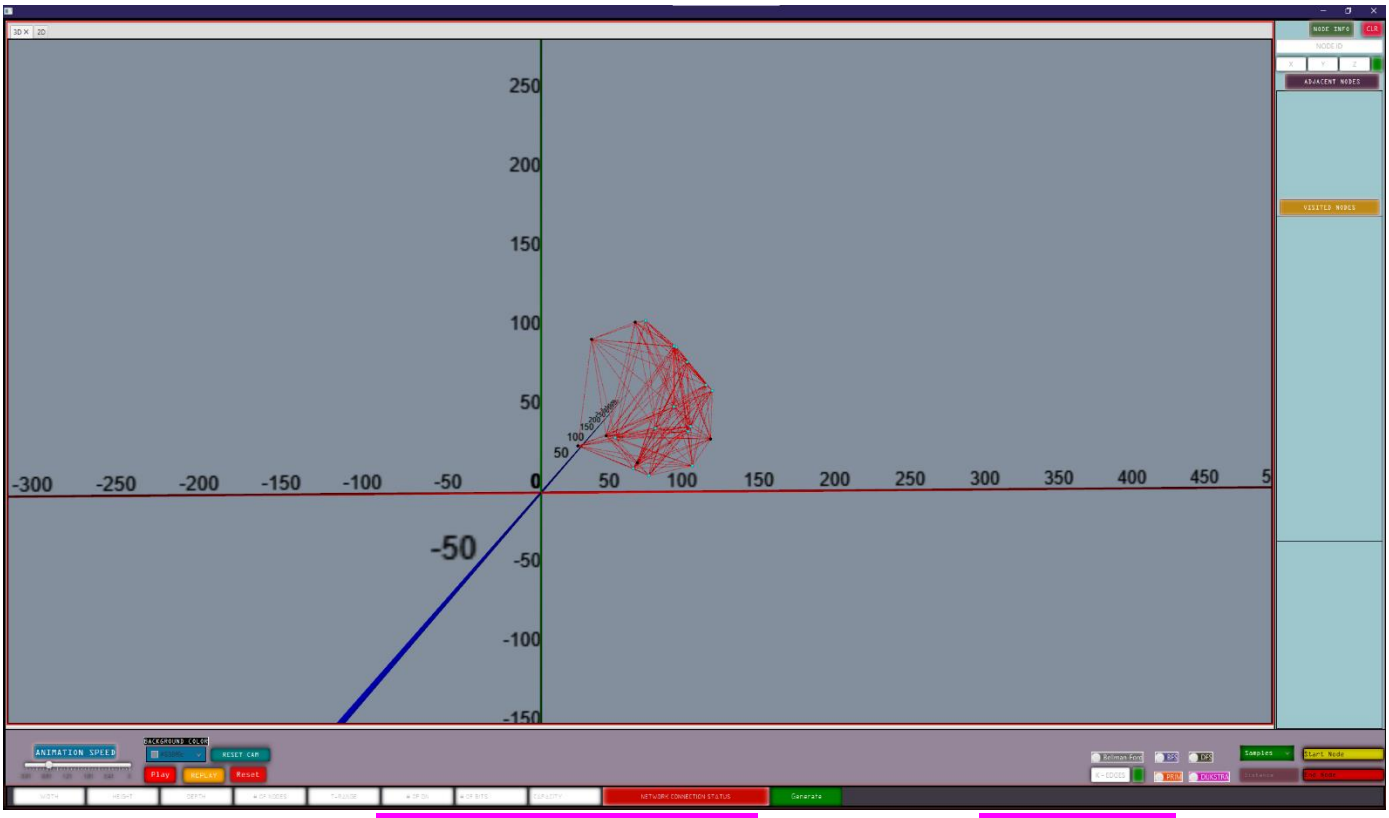
```
                Shortest Path To DN_2  from SNode_18
=====================================================================
        Vertices                                    Edge Cost
=====================================================================
[DN_2 ,SNode_16 ]                                   1341360.74

[SNode_16 ,SNode_10 ]                               601266.34

[SNode_10 ,SNode_13 ]                               4925.23

[SNode_13 ,SNode_9 ]                                140862.57

[SNode_9 ,SNode_18 ]                                887018.78
```

GUI Instructions

To use the GUI without entering all the information on the console, comment out the line "setUp(0,0,0,0,0,0,0.0,0.0);" in the initialize method (Located in GraphController). Build in NetBeans or any IDE that has JavaFx support. You can then run the project and it will launch the GUI.

After a successful build and launch, you will be shown this screen.

Zoom in for Details

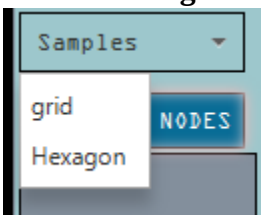


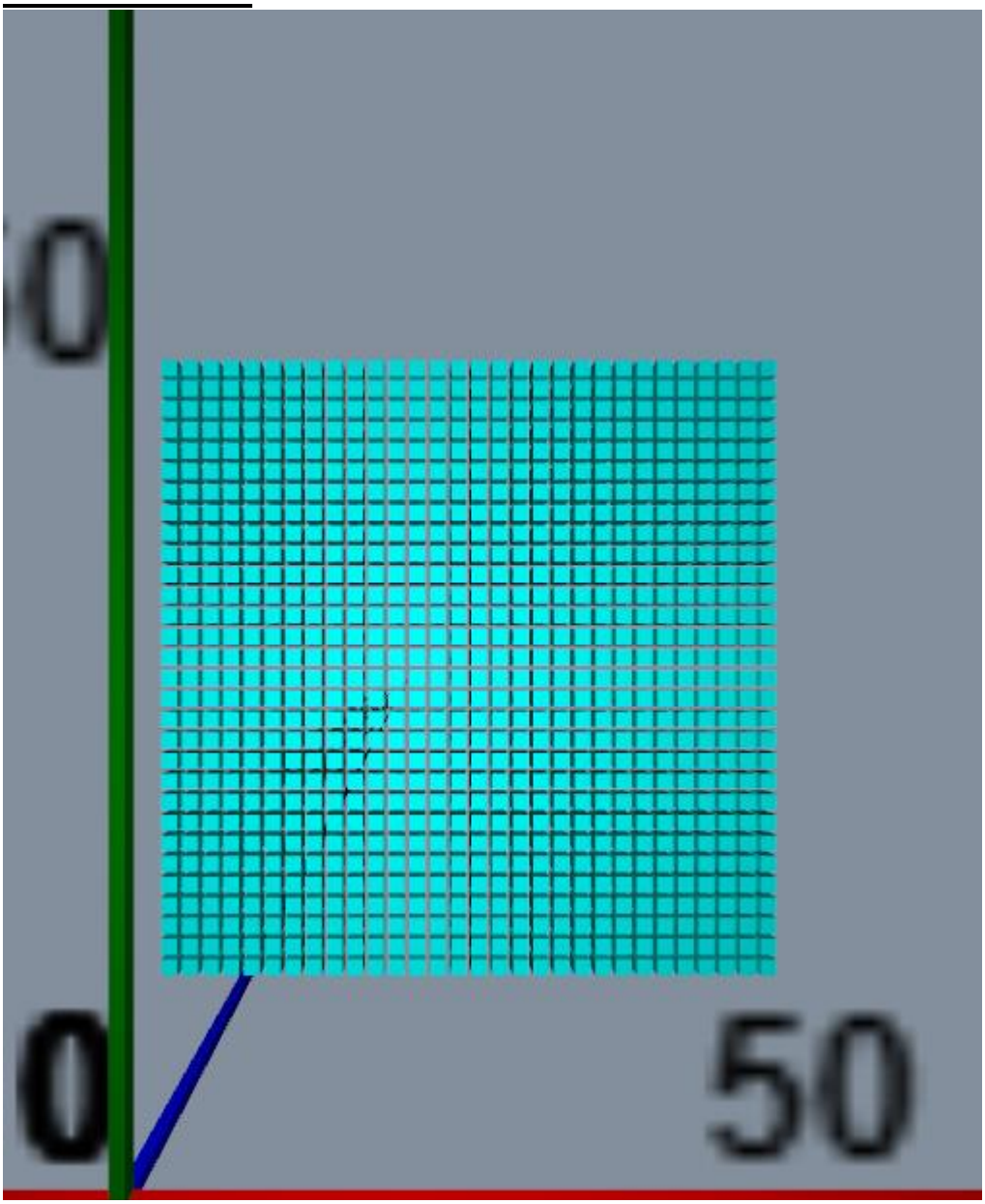
- You can pan the scene using MIDDLE MOUSE BUTTON, rotate using the LEFT MOUSE BUTTON, and zoom in using the RIGHT MOUSE BUTTON. STARTING NODE is selected by clicking on a node with the MIDDLE MOUSE BUTTON and DESTINATION NODE is selected by the RIGHT MOUSE BUTTON.
- There are two sample networks included. A grid network and a hexagon network. You can also generate your own network after providing:
  - Width, Height, Depth, # Of Nodes, Tr, # Data Nodes, # of Bits , and Storage Capacity.
- After loading your chosen network, choose from one of the four algorithms by clicking on the buttons labeled BFS, DFS, DIJKSTRA, or PRIM.
- Clicking on the button perform the chosen algorithm on the current network. After the algorithm is done, the Play button will turn green. Use LEFT MOUSE BUTTON repeatedly to play step by step, or RIGHT MOUSE BUTTON to play the entire sequence.

==========**BELOW ARE EXAMPLES OF EACH NETWORK** ==========

Grid Network (Using BFS)

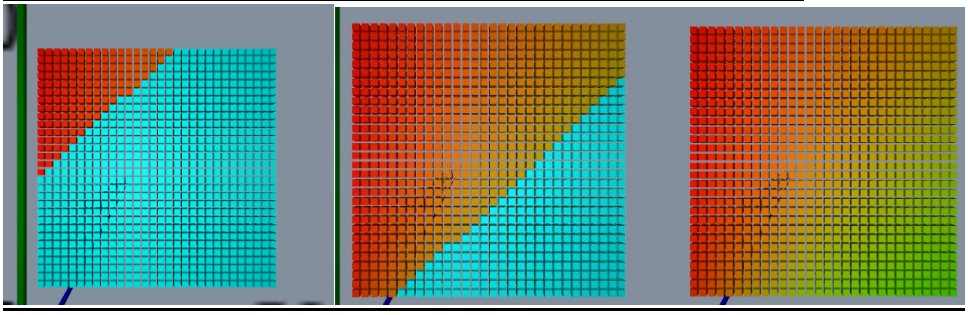
These examples will show the result of BFS on the grid network. Click on the BFS button after choosing at least a starting node to perform BFS.



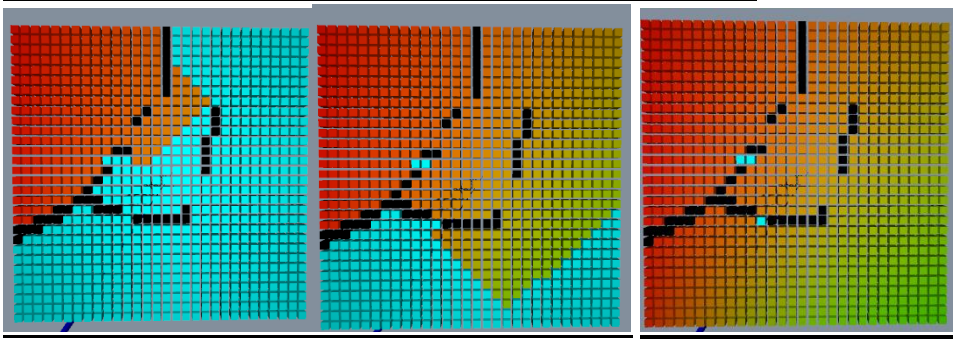


Holding control while moving the mouse over the grid creates barriers. This helps to visualize what the algorithm does as it evaluates each neighbor.

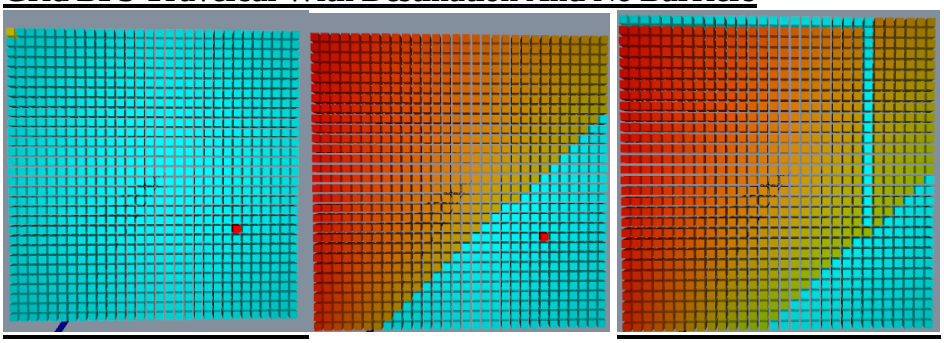
Grid BFS Traversal With No Destination And No Barriers



Grid BFS Traversal With No Destination And Barriers

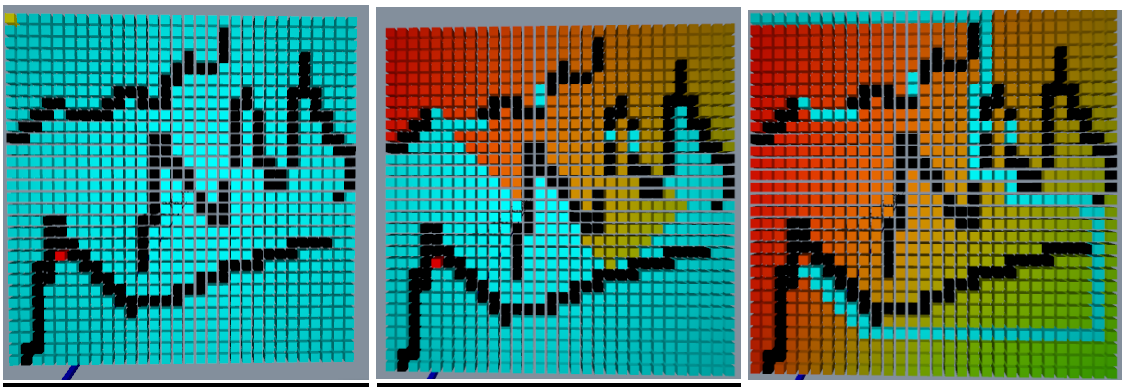


Grid BFS Traversal With Destination And No Barriers

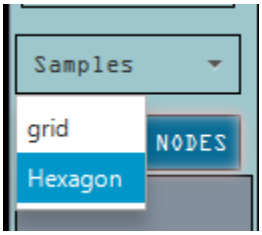


- After destination is discovered, a path is drawn to it.

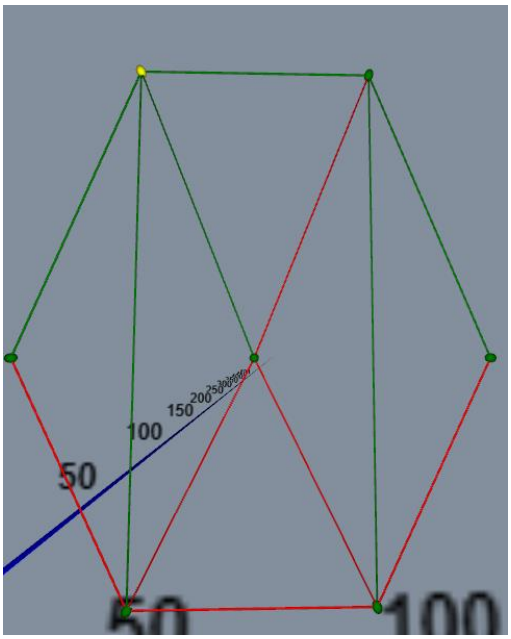
Grid BFS Traversal With Destination And Barriers



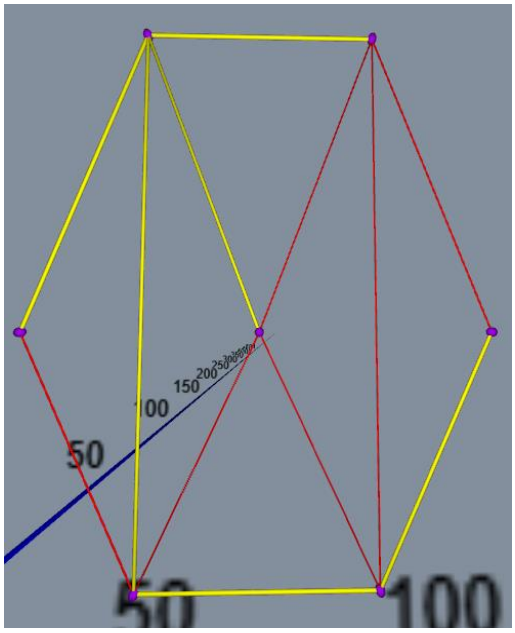
Hexagon Network



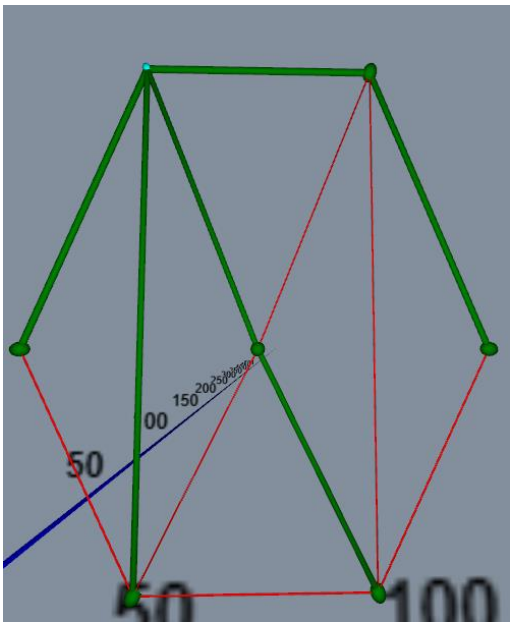
BFS



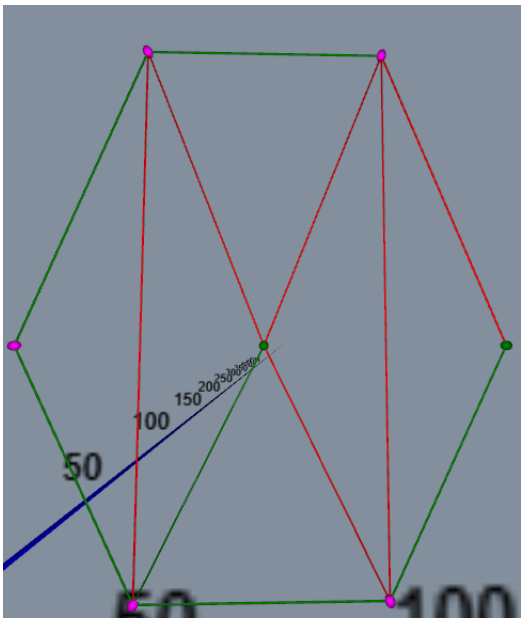
DFS



DIJKSTRA



PRIM



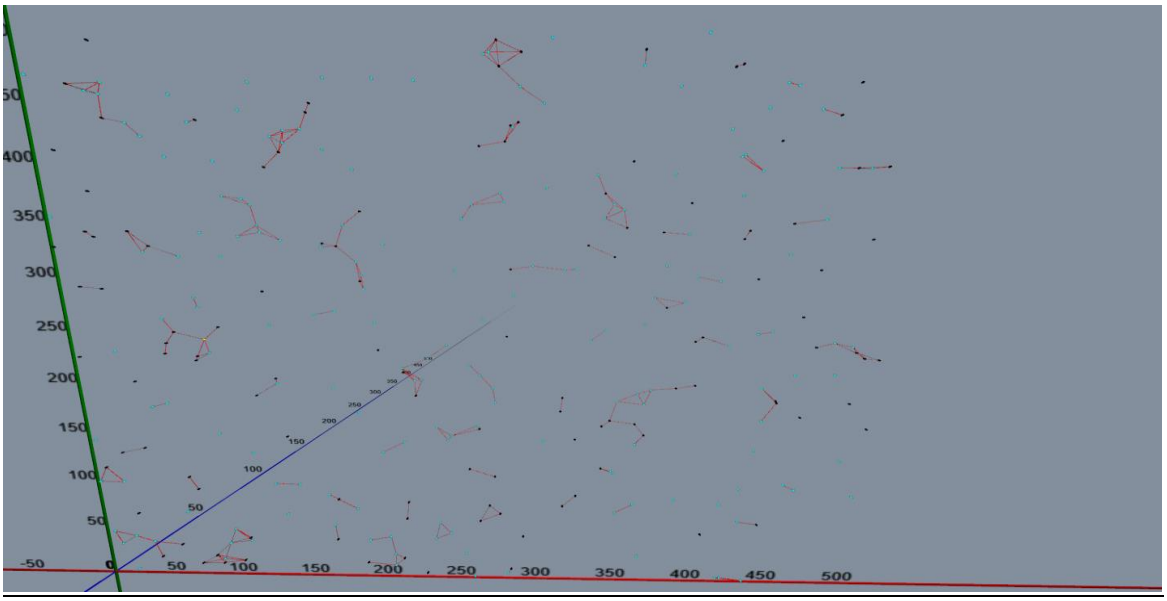
Randomly Generated Network

First, provide values for each field by clicking on the textbox and entering a value.

Then, Click on the Generate button to generate a random sensor network.

TRIAL 1

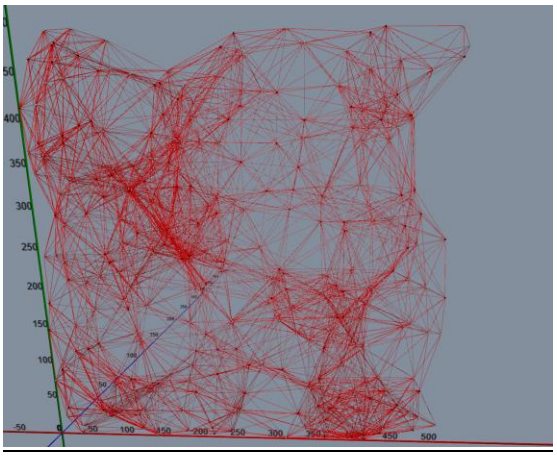
Generated Network



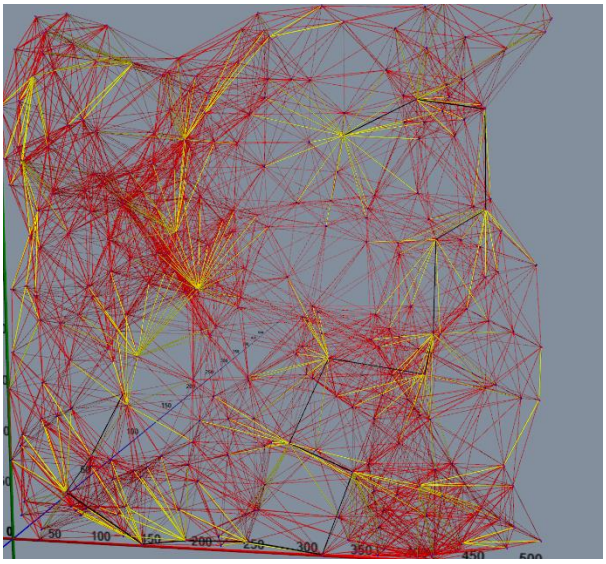
If the graph is connected, the Network Status display will turn green, otherwise, it will stay red.

TRIAL 2

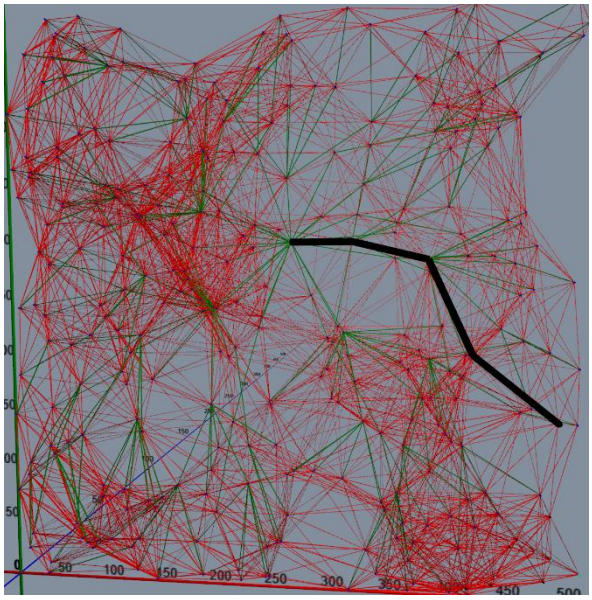
Generated Network



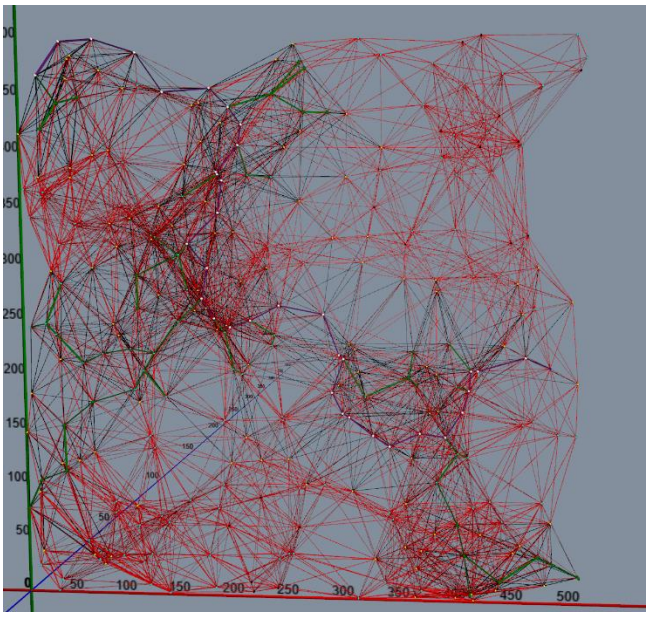
DFS Node- to- Node



BFS Node- to - Node



DIJKSTRA Node- to- Node



PRIM Node-to-Node