GRAPH THEORY AND ITS APPLICATIONS

UE20CS323

ASSIGNMENT 2&3

DONE BY: GAURAV DNYANESH MAHAJAN PES1UG20CS150 SECTION C

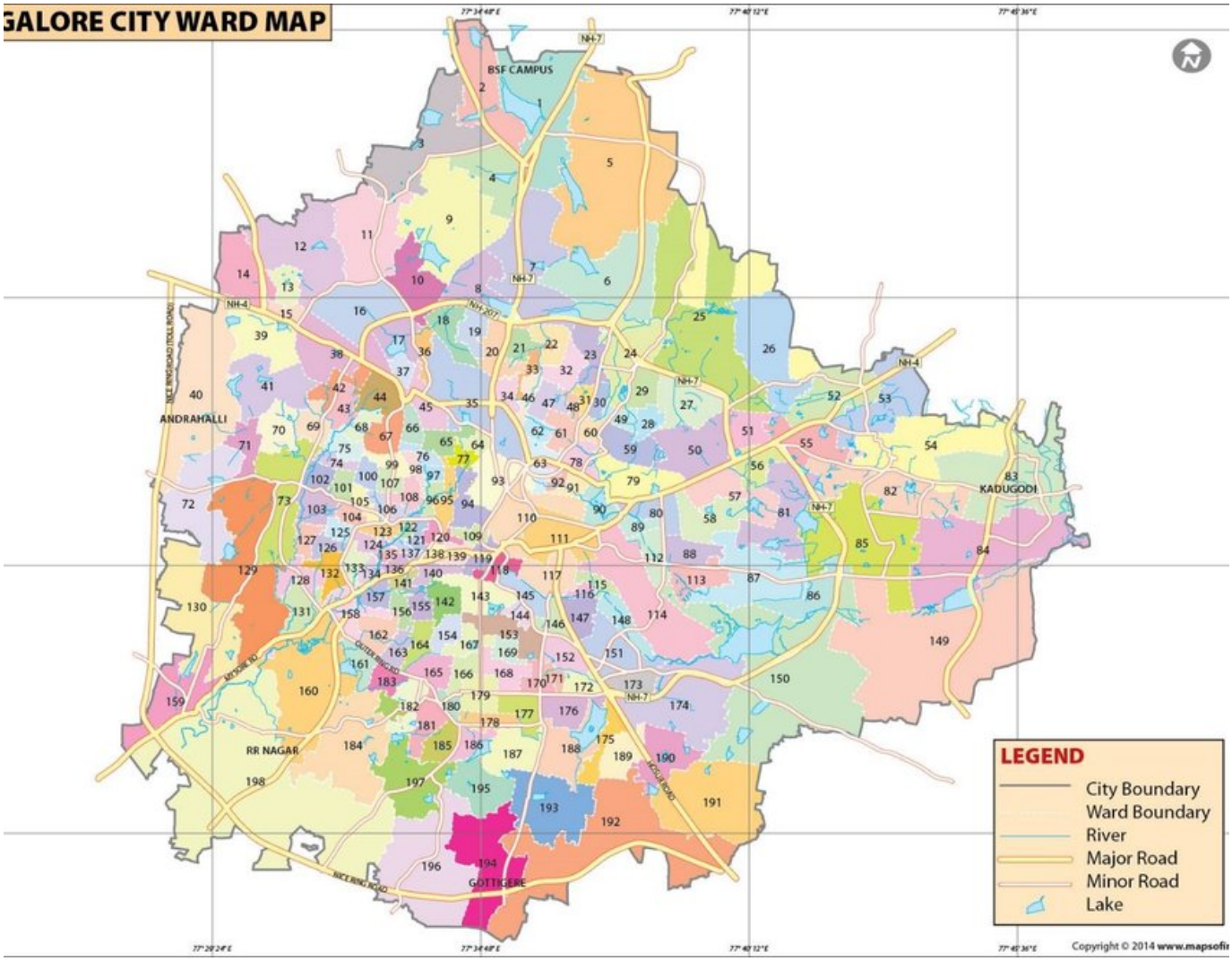
PROBLEM STATEMENT 8

You are a city planner and you are given the road network with n neighbourhoods and m edges that are connecting these. Identify bottlenecks in terms of neighbourhoods and suggest additional links/edges to have better reliable connectivity

IMPORTING PACKAGES

```
import networkx as nx
import matplotlib.pyplot as plt
```

LET US CONSIDER THE EXAMPLE OF BANGALORE



LET US CONSIDER A FEW AREAS LIKE

115-VANNARPET,

147-ADUGODI,

116-NEELASANDRA,

148-EJIPURA,

114-AGARAM,

112-DOMLUR,

89-JOGUPALYA,

80-HOYSALANAGAR,

88-JEEVANBHIMA NAGAR,

113-KONENA AGRAHARA,

87-HAL AIRPORT,

58-NEW THIPPASANDRA,

57-CV RAMAN NAGAR,

81-VIGNANANAGAR,

86-MARATHAHALLI,

79-SARVAGNANAGAR

50-BENNIGANAHALLI

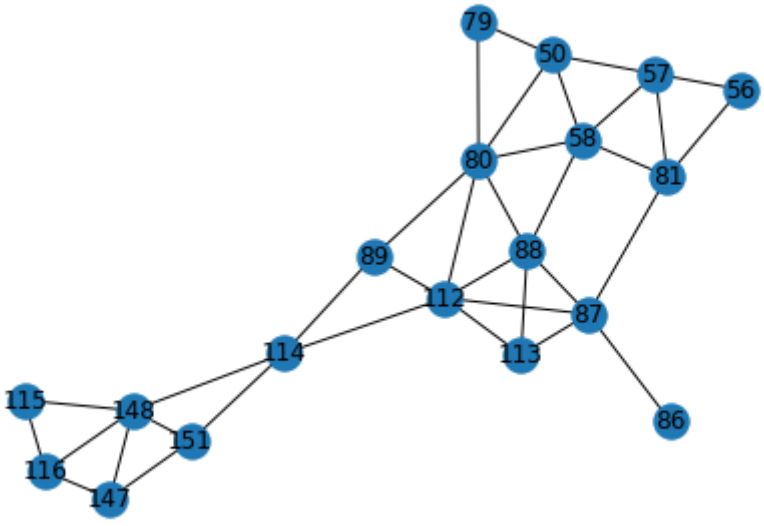
56-NARAYANPURA

151-KORAMANGALA

```
G=nx.Graph()
G.add_node(115)
G.add_node(147)
G.add_node(116)
G.add_node(148)
G.add_node(114)
G.add_node(112)
G.add_node(89)
G.add_node(80)
G.add_node(88)
G.add_node(113)
G.add_node(87)
G.add_node(58)
G.add_node(57)
G.add_node(81)
G.add_node(86)
G.add_node(79)
G.add_node(50)
G.add_node(56)
nx.draw(G,with_labels=True)
```


```
G.add_edge(115,116)
G.add_edge(115,148)
G.add_edge(116,148)
```

```
G.add_edge(116,147)
G.add_edge(147,148)
G.add_edge(147,151)
G.add_edge(148,151)
G.add_edge(148,114)
G.add_edge(151,114)
G.add_edge(114,112)
G.add_edge(114,89)
G.add_edge(112,89)
G.add_edge(112,80)
G.add_edge(89,80)
G.add_edge(112,113)
G.add_edge(112,88)
G.add_edge(80,88)
G.add_edge(113,88)
G.add_edge(113,87)
G.add_edge(88,87)
G.add_edge(112,87)
G.add_edge(87,86)
G.add_edge(87,81)
G.add_edge(88,58)
G.add_edge(81,58)
G.add_edge(80,58)
G.add_edge(80,79)
G.add_edge(79,50)
G.add_edge(80,50)
G.add_edge(58,50)
G.add_edge(50,57)
G.add_edge(58,57)
G.add_edge(57,56)
G.add_edge(81,56)
G.add_edge(81,57)
nx.draw(G,with_labels=True)
```



The graph as been plotted with all the areas as mentioned.

The bottlenecks will be in the form of Articulation Points. An articulation point or cut vertex is any node whose removal (along with all its incident edges) increases the number of disconnected components of a graph.

In order to find an articulation point we need to see if the graph is biconnected.A graph is biconnected if, and only if, it cannot be disconnected by removing only one node (and all edges incident on that node).

If removing a node increases the number of disconnected components in the graph, that node is called an articulation point, or cut vertex.

A biconnected graph has no articulation points

Checking if the graph is biconnected

```
print(nx.is_biconnected(G))
```

```
False
```

The graph is not bi-connected which means there are articulation points or bottlenecks in our network.

Hence displaying those points

```
l=list(nx.articulation_points(G))
print(l)
```

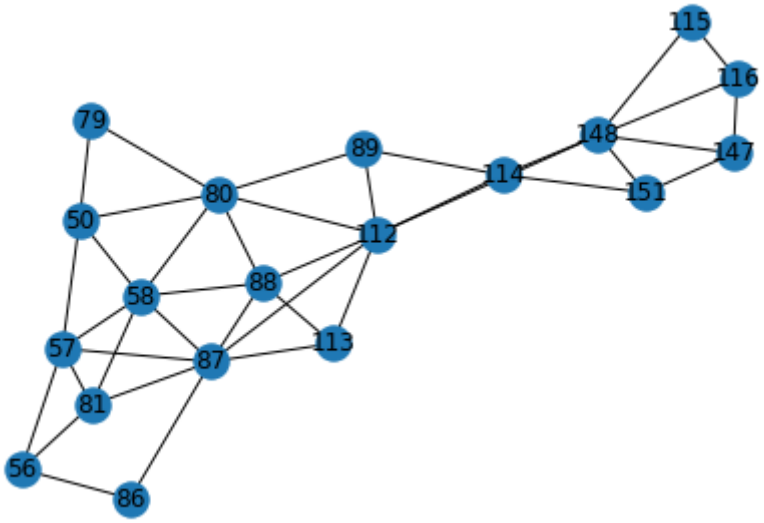
```
[87, 114]
```

87 - HAL AIRPORT, 114 - AGARAM are the bottlenecks we have

SUGGESTING EDGES

```
G.add_edge(112,148)
G.add_edge(86,56)
```

```
nx.draw(G,with_labels=True)
```



IF WE CAN HAVE A ROAD THAT CONNECTS

112 [DOMLUR] TO 148 [EJIPURA]

86 [MARATHAHALLI] TO 56 [NARAYANPURA]

THE BOTTLENECKS CAN BE REMOVED

NOW , LETS CHECK IF THE GRAPH HAS BECOME BI-CONNECTED OR NOT

```
print(nx.is_biconnected(G))
```

```
True
```

That means, there are no more articulation points.

```
len(list(nx.articulation_points(G)))
print(list(nx.articulation_points(G)))
```

```
[]
```

Colab paid products - Cancel contracts here

✓ 0s completed at 8:54 AM