CHAPTER 4
ANALYSIS AND DESIGN

4.1 Analysis

To be able to implement the minimum spanning tree, a graph tree that is undirected and has a weight is needed. In a graph tree there are three parts, namely vertex or nodes, edges, and weights. For testing purposes in this project in order to analyze the comparison of the Reverse-Delete and Boruvka algorithm, some samples from the graph are needed.

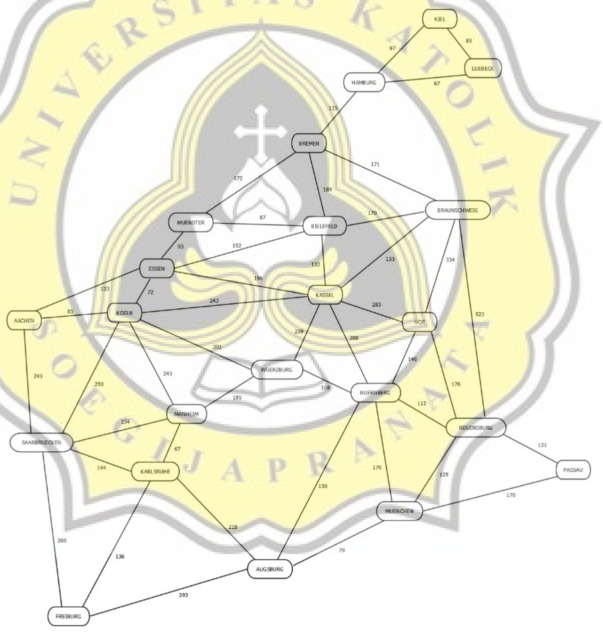
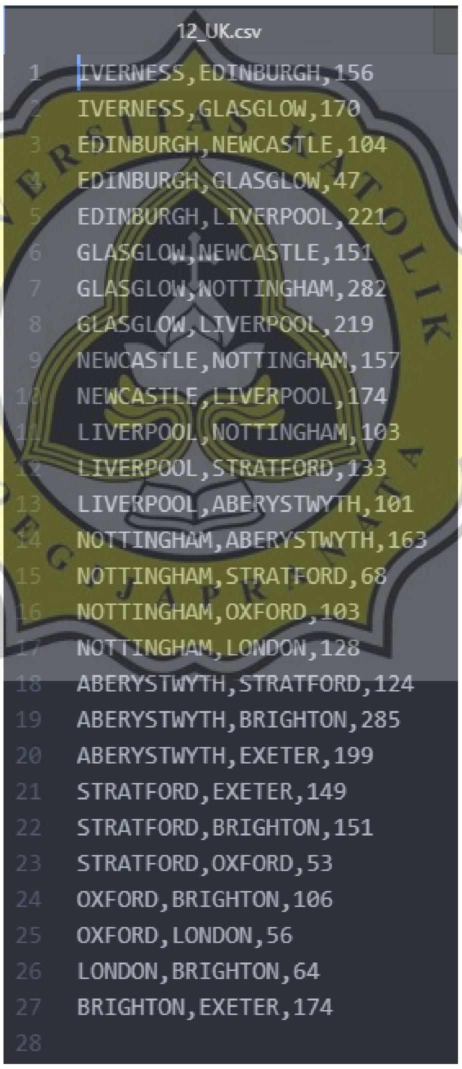


Illustration 4.1: Example of tree graph that will be used in testing

The image above is an example of a graph that will be used in testing, the image above is a graph of 22 cities in West Germany along with 45 edges that connecting each city. The table bellow is a detail of the data that will be used in testing.

Table 4.1: Graph sample test detail

Sample Test	Node	Edge
1	12	27
2	22	45
3	49	120
4	31	60



12_UK.csv

```
IVERNESS,EDINBURGH,156
IVERNESS,GLASGLOW,170
EDINBURGH,NEWCASTLE,104
EDINBURGH,GLASGLOW,47
EDINBURGH,LIVERPOOL,221
GLASGLOW,NEWCASTLE,151
GLASGLOW,NOTTINGHAM,282
GLASGLOW,LIVERPOOL,219
NEWCASTLE,NOTTINGHAM,157
NEWCASTLE,LIVERPOOL,174
LIVERPOOL,NOTTINGHAM,103
LIVERPOOL,STRATFORD,133
LIVERPOOL,ABERYSTWYTH,101
NOTTINGHAM,ABERYSTWYTH,163
NOTTINGHAM,STRATFORD,68
NOTTINGHAM,OXFORD,103
NOTTINGHAM,LONDON,128
ABERYSTWYTH,STRATFORD,124
ABERYSTWYTH,BRIGHTON,285
ABERYSTWYTH,EXETER,199
STRATFORD,EXETER,149
STRATFORD,BRIGHTON,151
STRATFORD,OXFORD,53
OXFORD,BRIGHTON,106
OXFORD,LONDON,56
LONDON,BRIGHTON,64
BRIGHTON,EXETER,174
```

Illustration 4.2: Example of Data That Has Been Entered Into CSV File Format

UNIVERSITAS KATOLIK SOEGIJAPRANATA

From the image above, in the CSV file there is information about the starting node, destination node, and also the distance of the edge that connects the two nodes. This information is needed by the program to be able to generate a weighted graph, before it is finally processed by each algorithm.

4.1.1 Reverse Delete Algorithm

The way Reverse Delete algorithm work is, take the edge that has the biggest weight and then the edge is deleted. If after the edge is deleted and doesn't disconnect the graph, then it will proceed to the next biggest edge. However, if after deleting the edge will cause the graph disconnected, then the edge will not be deleted and will continue to the next biggest edge until there are no more edges that can be removed from the graph. The following bellow is an ilustration.

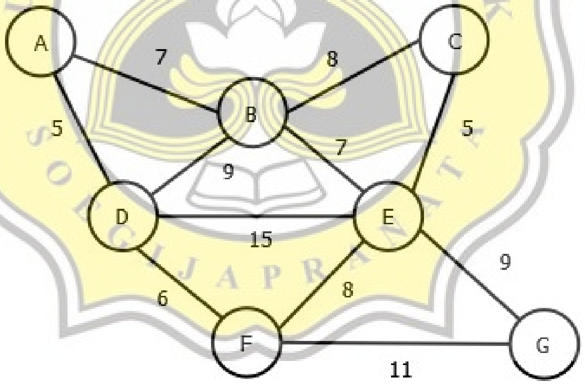


Illustration 4.3: Initial graph

From the picture above, it can be seen that the edge connecting between D and E has the biggest value, so the edge will be deleted. The following bellow is an ilustration.

UNIVERSITAS KATOLIK SOEGIJAPRANATA

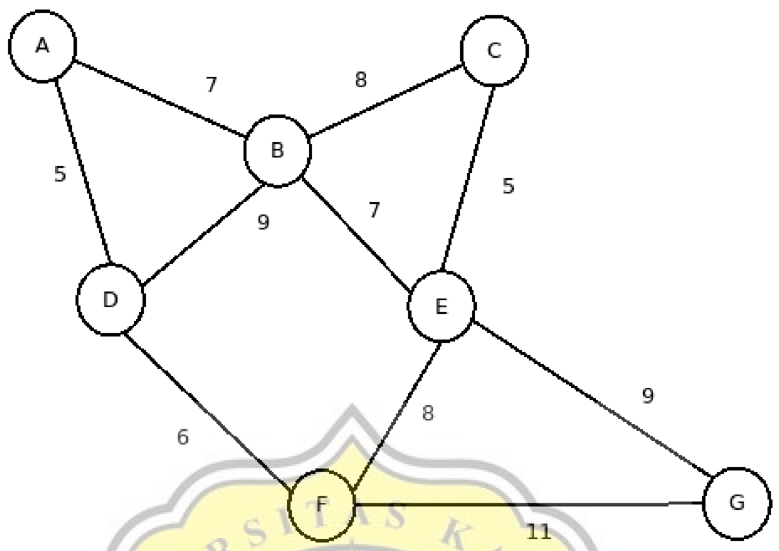


Illustration 4.4: Picture 1

After the edge is removed, it can be seen from picture above that the graph is still connected, therefore it will continue to the next biggest edge. The next biggest edge is the one that connects node F and G, so the edge will be removed from graph. The following bellow is an ilustration.

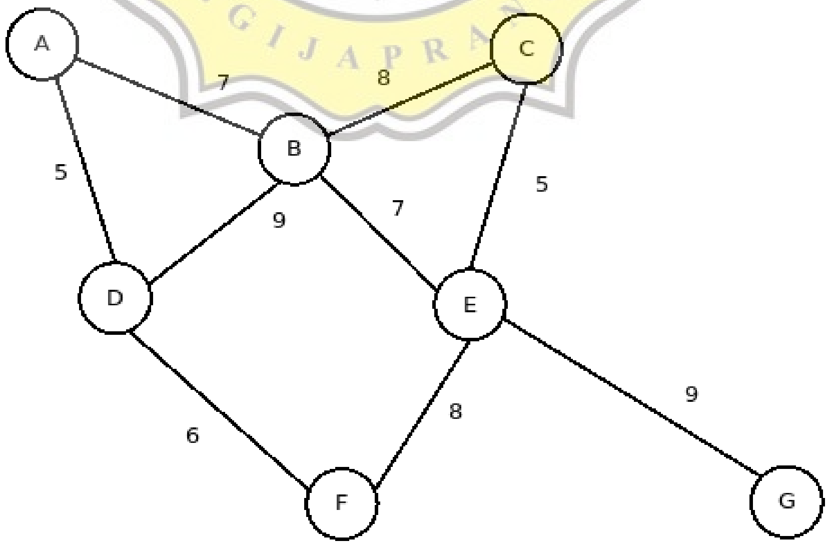


Illustration 4.5: Picture 2

UNIVERSITAS KATOLIK SOEGIJAPRANATA

Go to the next biggest edge. From the image above it can be seen that there are two edges that have the same weight, namely the edge that connects between node E and G, and the edge that connects between B and D. Both have the same weight with the value of nine. If there are two edges that have the same weight, then one of the two edges can be freely selected to delete first. Here what will be removed is the edge that connects between node B and D, because if the edges of E and G are removed, the graph will be disconnected then, therefore the E and G edges are not deleted. The following bellow is the ilustration.

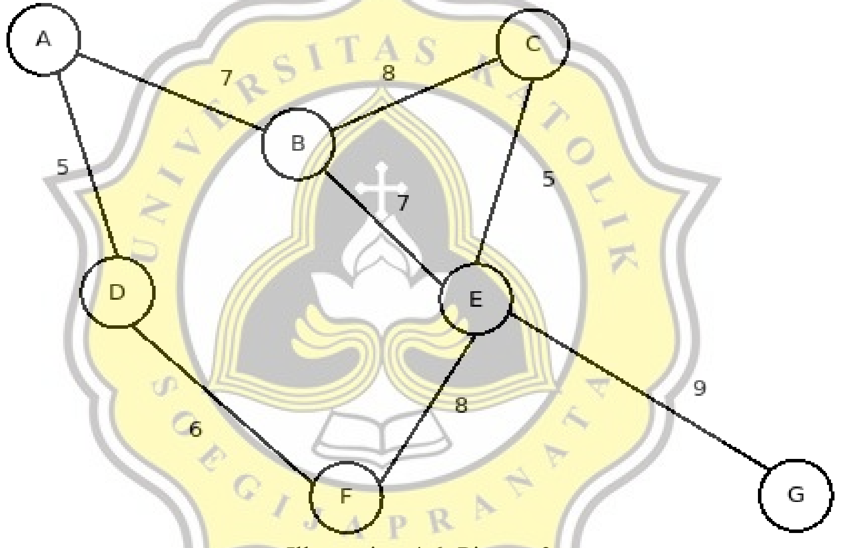


Illustration 4.6: Picture 3

Then move again to the next biggest edge. From the picture above, it can be seen that there are two edges that have the same weight, namely the edge that connects B and C, and the edge that connects F and E. Because if edge B and C or edge F and E deleted will not make the graph disconnected, so one edge can be randomly selected to be deleted first. Here the edges between F and E will be deleted first. The following bellow is the ilustration.

UNIVERSITAS KATOLIK SOEGIJAPRANATA

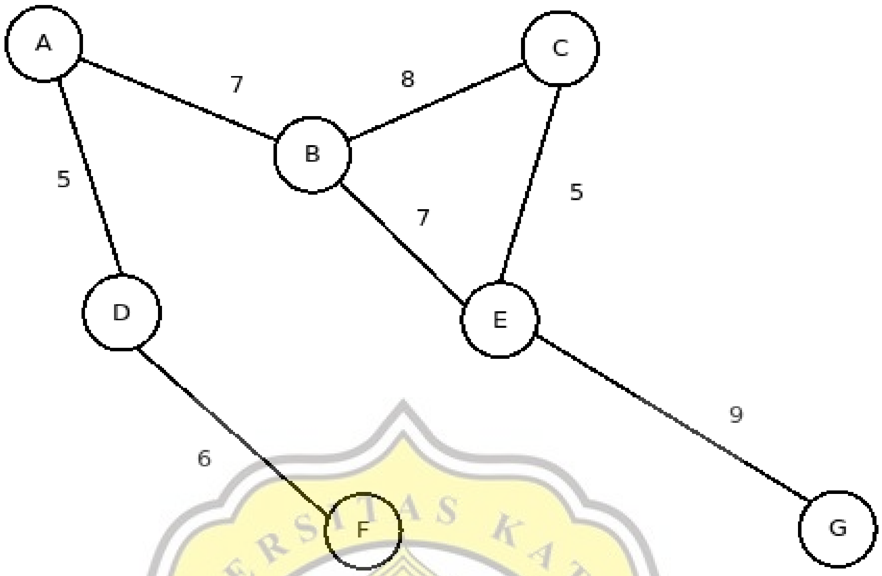


Illustration 4.7: Picture 4

Then move to next biggest edge. From the picture above it can be seen that the edge that connects between node B and C has the biggest weight with the value of eight, the same as the previously deleted edge, and because if the edge is removed it won't disconnect the graph then it will be deleted . The following bellow is the ilustration.

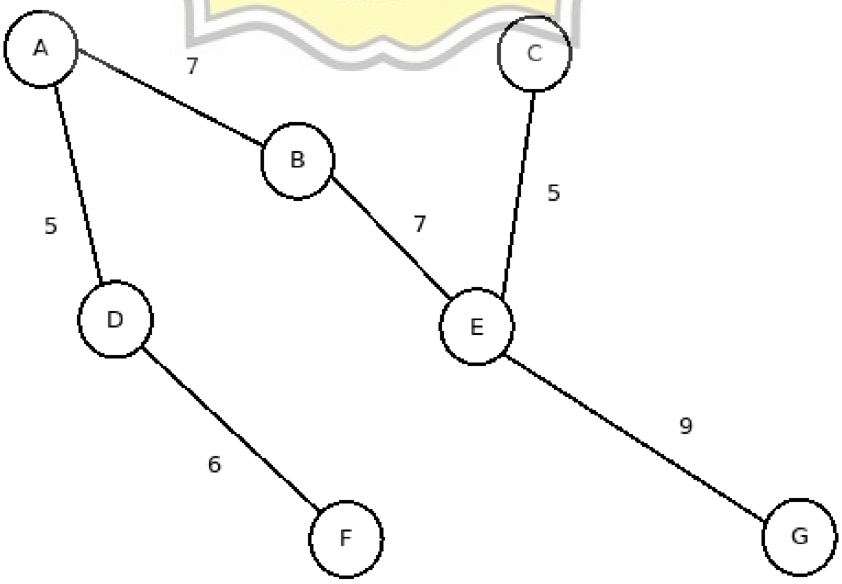


Illustration 4.8: Picture 5

It can be seen from the picture above that there are no more edges that can be removed, So the remaining edge will form the Minimum Spanning Tree, and the Reverse Delete algorithm ends.

4.1.2 Boruvka Algorithm

For Boruvka algorithm, the way it works is the input of undirected graph is defined as an individual component for each nodes, so that the spanning tree is still empty. After that, for each node an edge will be searched with the lowest weight that connects the node and the other node. After that, add the edge if it has not been added into the graph. This will continue until all nodes are connected to each other and form a Minimum Spanning Tree. The following bellow is the ilustration.

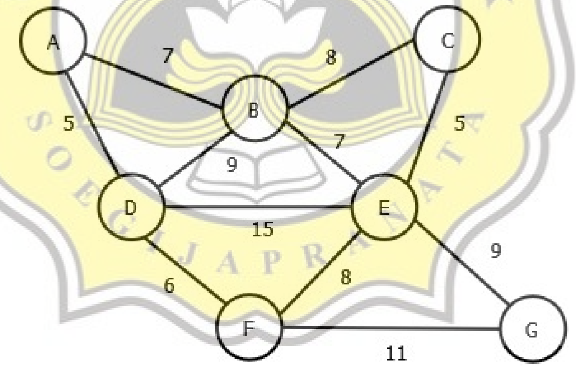


Illustration 4.9: Initial graph

From the ilustration above. The graph will be defined as individual and empty components between each node. The following bellow is the ilustration.

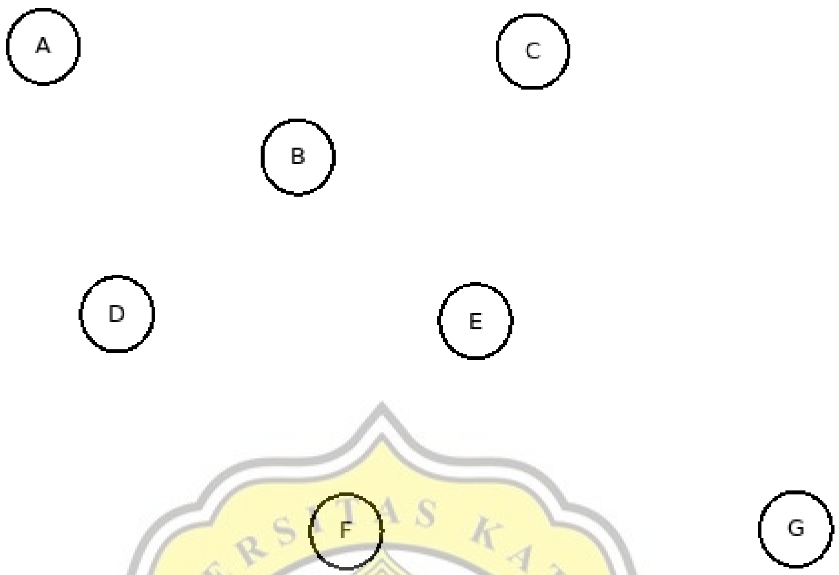


Illustration 4.10: Picture 1 Boruvka

To find the Minimum Spanning Tree, it can be started from any node, but here it will be start from the node in alphabetical order, so it will started from node A first. From the initial graph, it can be seen that node A has two connected edge components. From the two edges, it can be seen that the edge that connects to the node D has the smallest weight, therefore the edge will be added into the graph. Following bellow is the ilustration.

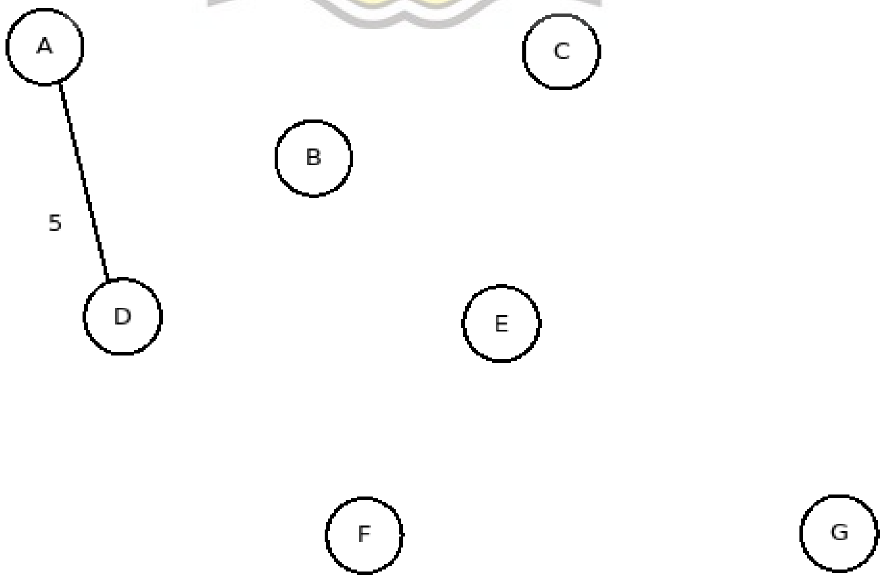


Illustration 4.11: Picture 2 Boruvka

From the picture above. After node A is connected to node D, the search will continue to the next node, namely node B. From the initial graph it can be seen that node B has four connected edges, namely the edge that connects with node A, node B, node C, and node E. It can be seen from the four edges, there are two edges that have the same weight, namely the edge that connects nodes A and B and the edge that connects node B and E, both have the same weight. Of the two edges is free to take one of them to be included into the list, here that will be taken is the edge that connects between node B and E. The following bellow is the ilustration.

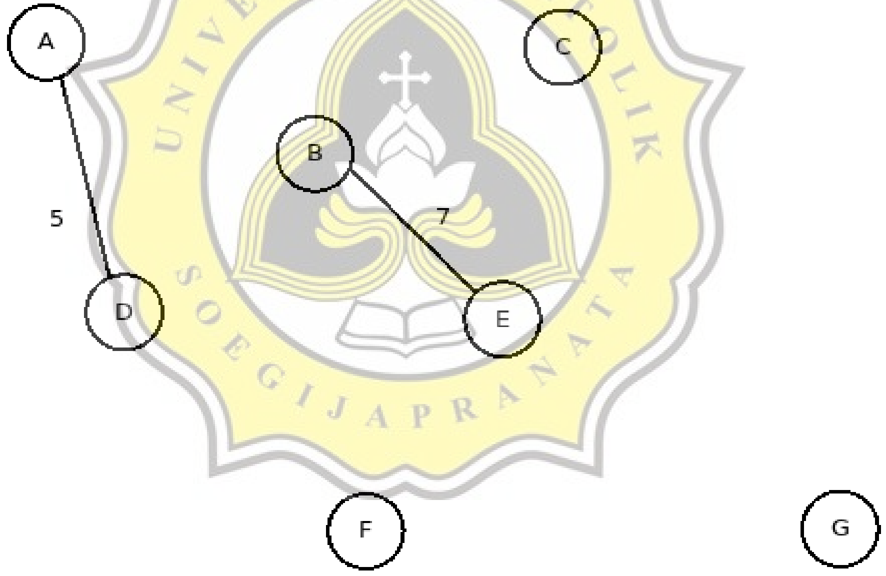


Illustration 4.12: Picture 3 Boruvka

The proceed to the next node, namely node C. From the initial graph, it can be seen that node C has two edges, each of wich connects between node B and C also node C and E. Because the edge that connects C and E has the smallest weight, so that edge is added to the list. The following bellow is the ilustration.

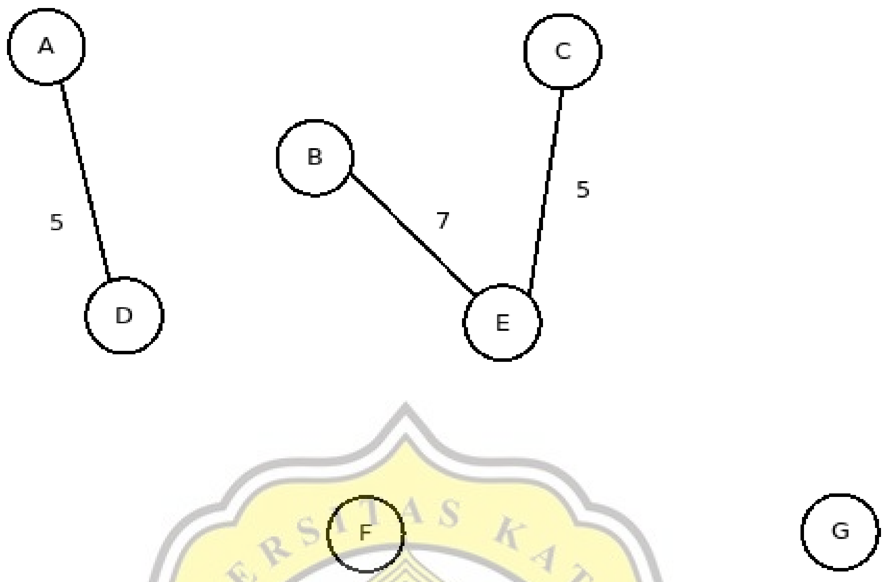


Illustration 4.13: Picture 4 Boruvka

The proceed continue to next node, namely node D. From the initial graph it can be seen that there are four edges connected to node D, from the four edges it can be seen that the edge with the smallest weight belongs to the edge that connecting nodes A and D . Because the edge has already been added to the list, it’s not picked up again, and will continue to the next node. The next node is E, on the node E there are five connected edges. Of the five connected edges, the edge with the smallest weight is the edge that connects between node C and node E. Because the case is the same as node D before, the edge will not be taken, and will continue to the next node, namely node F. There are three edge that connect to node F. Edge that connects between node D and node F has the smallest weight among the three edges, because the edge has not been added to the list, it will be taken. The following bellow is the ilustration.

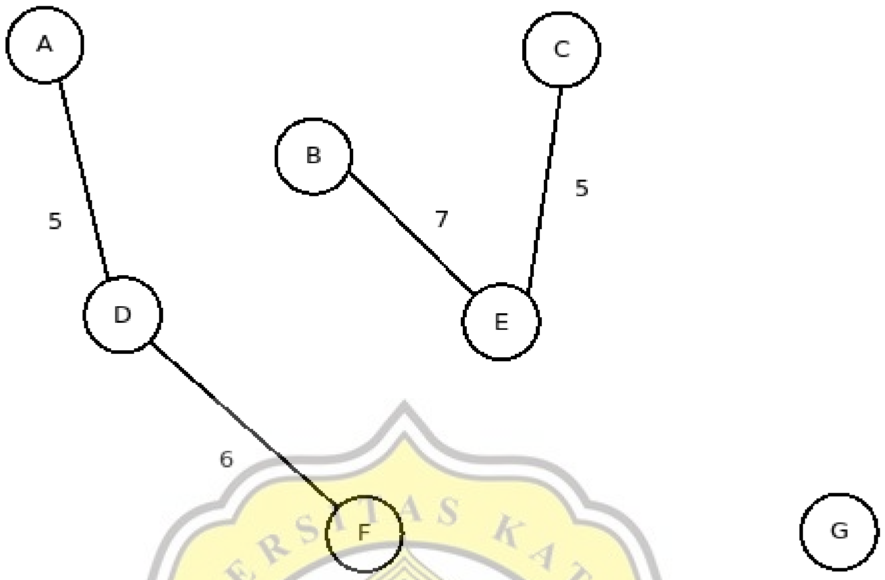


Illustration 4.14: Picture 5 Boruvka

Then go to node G, at node G it has two connected edges, the edge with least weight is on the edge that connecting between node E and G with a weight of nine, because the edge has not been taken, the edge will be added to the list. The following bellow is the ilustration.

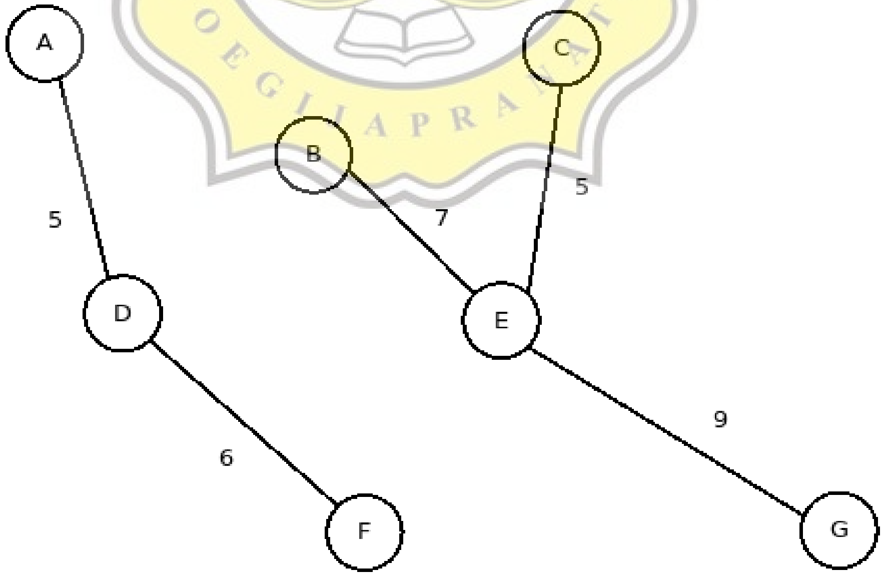


Illustration 4.15: Picture 6 Boruvka

After this process, it can be seen from ilustration above that there are two separate graph, then look for the edge with lowest weight that can unite the two graph into

UNIVERSITAS KATOLIK SOEGIJAPRANATA

one graph. From the initial graph, it can be seen that, the edge connecting A and B has a lower weight than the other edge that has not been added, then the edge will be added to the list. The following bellow is the ilustration.

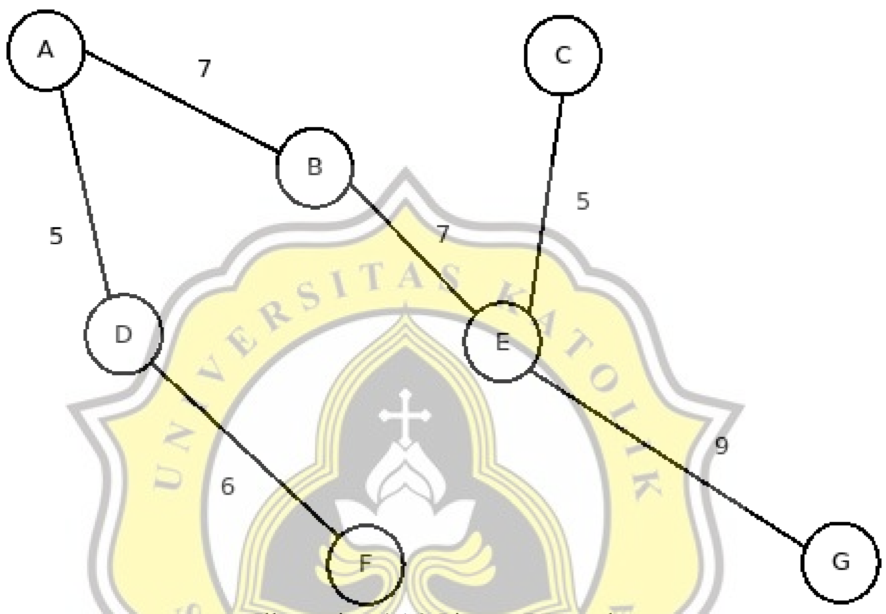


Illustration 4.16: Picture 7 Boruvka

From the ilustration above, it can be seen that the graph is all connected, so the search for the Minimum Spanning Tree using Boruvka algorithm is complete. From the initial total weight of ninety, after being processed using the Boruvka algorithm the total weight becomes thirty-nine.

UNIVERSITAS KATOLIK SOEGIJAPRANATA

4.2 Desain

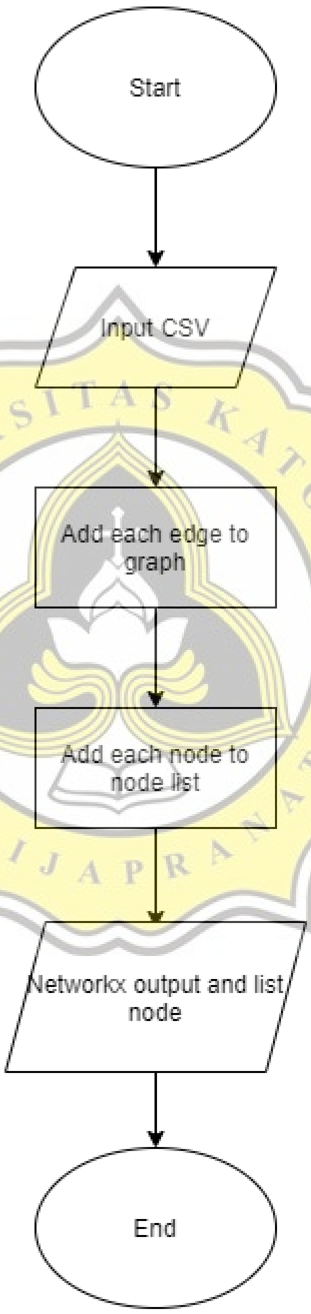


Illustration 4.17: Flowchart for generate tree graph

UNIVERSITAS KATOLIK SOEGIJAPRANATA

The data used to generate the graph will be entered into the CSV file which included information about node one, node two, and their weight or distances. For generate the initial graph, in this project will use the **Networkx** library that available in python 3.

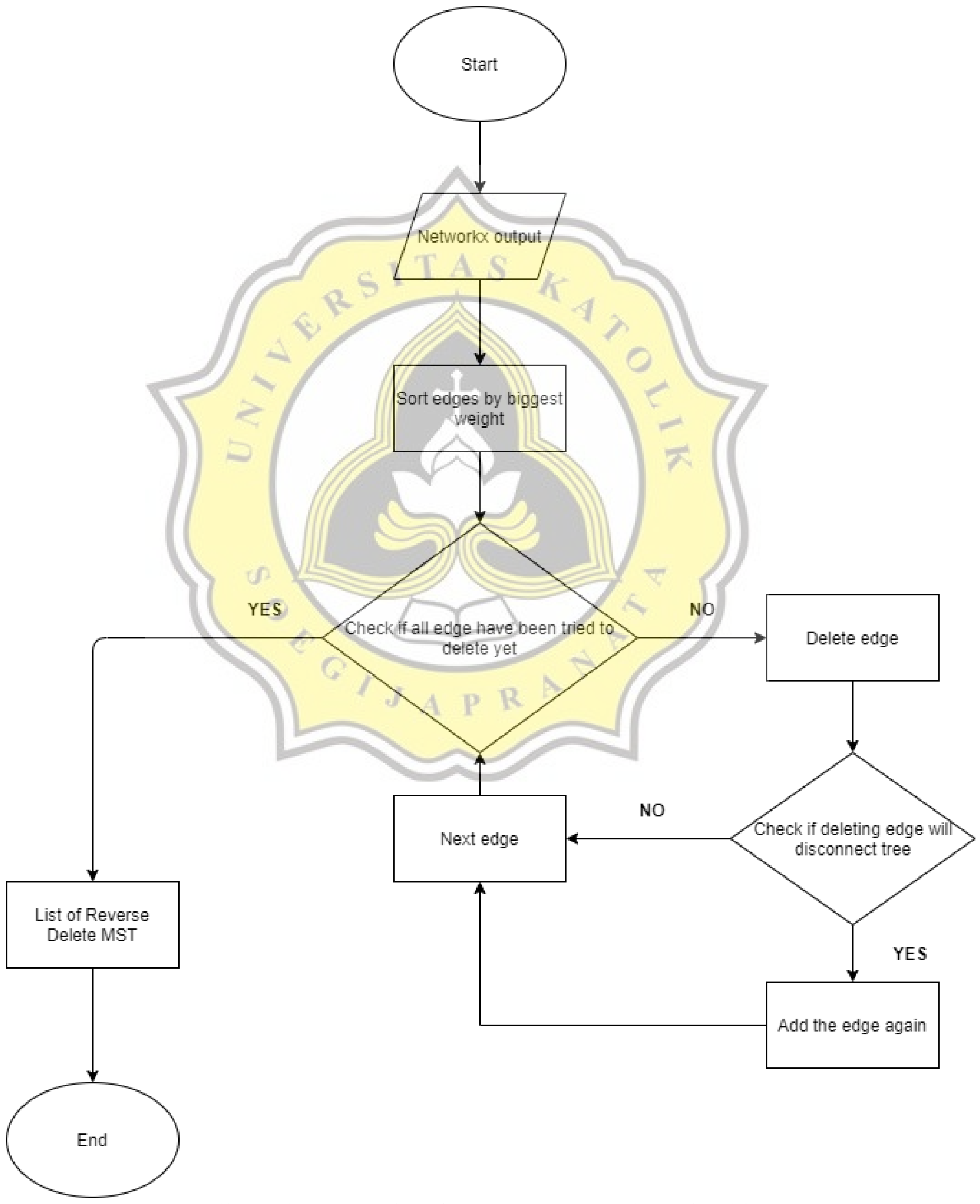

Start
Networkx output
Sort edges by biggest weight
Check if all edge have been tried to delete yet
YES
NO
Delete edge
Check if deleting edge will disconnect tree
NO
YES
Next edge
Add the edge again
List of Reverse Delete MST
End

Illustration 4.18: Flowchart for Reverse Delete Algorithm

UNIVERSITAS KATOLIK SOEGIJAPRANATA

From the flowchart above, it can be seen that to be able to use the Reverse Delete algorithm for finding MST, the tree graph that has been generated from CSV using **Networkx** library, must be sorted first from the biggest edge to the smallest edge. After the data is sorted, we check first whether the edge has been tried to delete or not, if not then it will be deleted and continue to the next edge. If after deleting the edge will disconnect the tree, the edge will be added back into the tree and continue to the next edge. The process will stop when all edges are have been tried to be removed. And the remaining edge will be the Minimum Spanning Tree

UNIVERSITAS KATOLIK SOEGIJAPRANATA

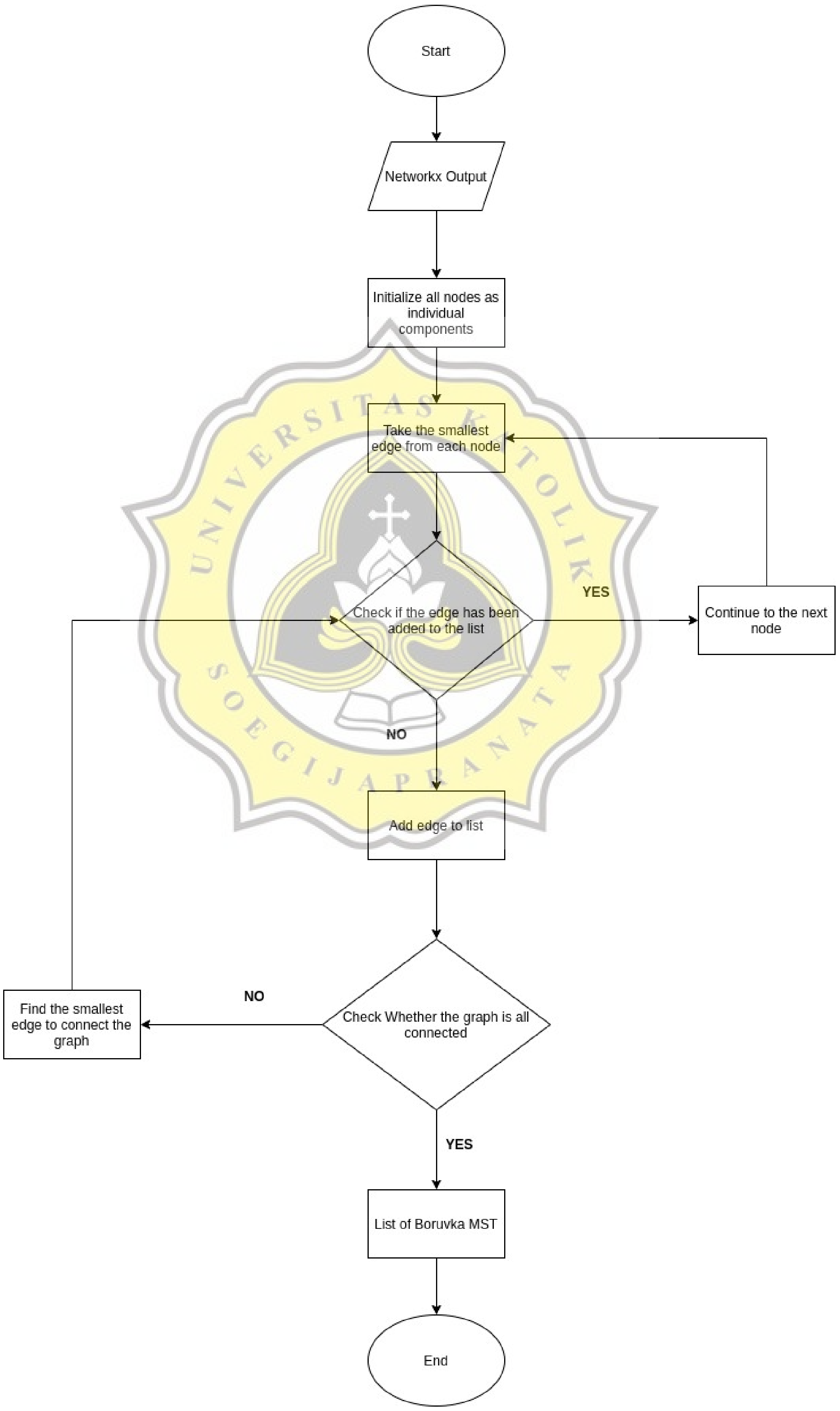

Start
Networkx Output
Initialize all nodes as individual components
Take the smallest edge from each node
Check if the edge has been added to the list
YES
Continue to the next node
NO
Add edge to list
Check Whether the graph is all connected
NO
Find the smallest edge to connect the graph
YES
List of Boruvka MST
End

Illustration 4.19: Flowchart for Boruvka Algorithm

UNIVERSITAS KATOLIK SOEGIJAPRANATA

For Boruvka algorithm the way it work is very different from the Reverse Delete algorithm. From the flowchart above, to be able to use the Boruvka algorithm to find MST, the three graph that has been generated using the **Networkx** library is initialized as individual component for each node. After that for each node the smallest edge is taken. Before the smallest edge is added to the list, the edge will be checked first whether it has been added to the list, if the edge has been added to the list then the search will continue to the next edge. After that, check again if there are still graph that are still not connected, if there are any, the algorithm will look for the smallest edge to connect the graph. If the graph are all connected, then the search process for finding the Minimum Spanning Tree using Boruvka algorithm is complete.