CHAPTER 5
IMPLEMENTATION AND TESTING

5.1 Implementation

Data will be stored into a arraylist. Data that will be inputed into program such as start node, end node, and cost. Arraylist to store data will be created manually, so data should be add into linked list first. Then, an array of nodes should be created. Each list data is placed then into array of nodes, based on its index.

Array is required to store list data. To create an arraylist of graph, each node that is inputed should be put on index of array. Then, to place destination node, program will doing a search on each index until it match. After it is matched, list will be added.

Method **insertArray** is a method that is used to put node into index of array. Before data is added, it will doing search on each index of array. After it match, list data will be added in array. Arraylist of graph is already created.

After graph is generated, the next step is find minimum spanning tree using Kruskal and Prim algorithm. Kruskal algorithm is created first in this program. Before searching Kruskal's spanning tree, a method to sort all edge of graph from smallest cost will be required. Sorting method will be used to find minimum spanning tree using Kruskal algorithm, because Kruskal's spanning tree is started from smallest cost.

Each edge that will be stored on new object of **ListGraph** namely **sortedEdge**. There are three condition that stated on the sorting method. If current edge is larger than next edge, list will be added on the last. List of edge will be added on first if current edge is smaller than next edge or if the **sortedEdge** is still empty. List will be added on the middle when else condition applied. This method will return the value of **sortedEdge**. This **sortedEdge** will be used in finding Kruskal minimum spanning tree.

```
    if(found[0] && found[1]==false){
        tempNode[index].add(edge.getData().getStart(),edge.getData().getEnd(),edge.getData().getCost());
        tempNode[i] = new ListNode();
        tempNode[i].add(edge.getData().getEnd(),edge.getData().getStart(),edge.getData().getCost());
        i=i+1;
    }
    else if(found[1] && found[0]==false){
        tempNode[index].add(edge.getData().getEnd(),edge.getData().getStart(),edge.getData().getCost());
        tempNode[i] = new ListNode();
        tempNode[i].add(edge.getData().getStart(),edge.getData().getEnd(),edge.getData().getCost());
        i=i+1;
    }
    else if(!found[0] && !found[1]){
        tempNode[i] = new ListNode();
        tempNode[i].add(edge.getData().getStart(),edge.getData().getEnd(),edge.getData().getCost());

        tempNode[i+1] = new ListNode();
        tempNode[i+1].add(edge.getData().getEnd(),edge.getData().getStart(),edge.getData().getCost());
        i=i+2;
    }
}
edge=edge.getNext();
```

Illustration 5.1: Code from Kruskal Method

Piece of code above is from Kruskal method. A **found[]** is a boolean type which contains condition if edge is already added or not. Then, the IF condition above is initiated. Each edge which meet the condition will be added to **tempNode[]**. **TempNode** is an array used to store minimum spanning tree result.

Prim method will also uses **sortedEdge** value. Prim's spanning tree is started with one of the nodes will be arbitrary node. For this program, the first node in **sortedEdge** list will be set as arbitrary node. A method named **minVal()** is created to set edge with lower cost. This **minVal** method later will be used in finding minimum spanning tree using Prim algorithm.

```
while(curr!=null){
    if(curr.getData().getStart().equals(edge.getData().getStart())){
        temp.add(curr.getData().getStart(),curr.getData().getEnd(),curr.getData().getCost());
    }
    if(curr.getData().getEnd().equals(edge.getData().getStart())){
        temp.add(curr.getData().getStart(),curr.getData().getEnd(),curr.getData().getCost());
    }
    curr = curr.getNext();
}
Node minVal=minVal(temp);
exist=false;
tempVisit=visited.getData();
while(tempVisit!=null){
    if(tempVisit.getStart().equals(minVal.getStart()) &&
        tempVisit.getEnd().equals(minVal.getEnd())){
        exist=true;
        break;
    }
    tempVisit=tempVisit.getNext();
}
if(!exist){
    visited.add(minVal.getStart(),minVal.getEnd(),minVal.getCost());
}
edge=edge.getNext();
```

Illustration 5.2: Code from Prim Method

Piece of code above is from Prim method. The if condition on the top side is used to add edge data into temporary list named **temp**. This **temp** then used by **minVal**. The edge that creating minimum spanning tree is stored in **tempVisit**, while **exist** is used to check is node has visited or not.

5.2 Testing

There are four samples of graph that will be used for testing. Each graph sample has different number of node and edge. The purpose of this test is each minimum spanning tree result using Kruskal and Prim algorithm can be compared. The number of node and edges on each graph can be seen on table below.

Table 5.1: Graph Sample Data

Sample	Number of Nodes	Number of Edges
I	4	5
II	5	7
III	6	10
IV	7	11

Sample I

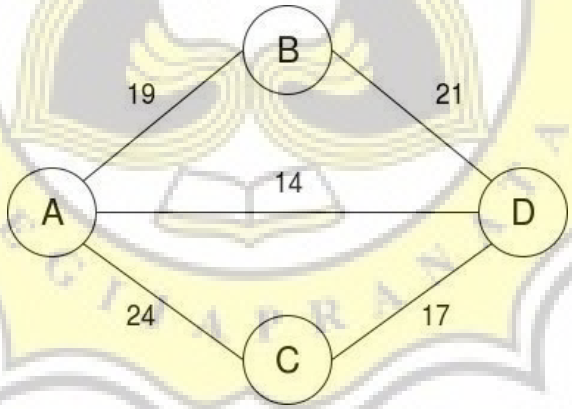

Illustration 5.3: Graph Sample I

Table 5.2: Data Sample I

Start Node	End Node	Cost
A	B	19
A	C	24
B	D	21
C	D	17
D	A	14

The data above is stored in text file. Program will read the data then generate it into graph. To see if the graph is done or not, graph will be represented as adjacency list. After graph is generated properly, Kruskal and Prim algorithm can be run. The result of MST program using Kruskal and Prim algorithm is shown on figure below.

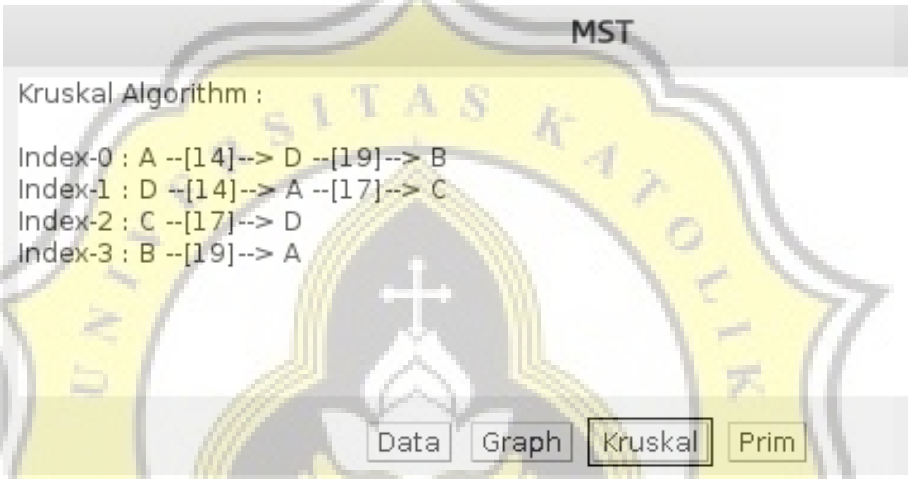

Illustration 5.4: MST Kruskal Result I

Illustration 5.5: MST Result Prim I

The figures above shows MST result using Kruskal and Prim algorithm. Based on the program, both algorithm generate same result. The MST result are A-D, C-D, and B-A.

To ensure that the results of the program are correct, MST is also done manually. MST completion manually result is shown on table below.

Table 5.3: MST Forming Steps from Graph I

Step	**Kruskal Algorithm**		**Prim Algorithm**	
	Selected Edge	**Inserted Node**	**Selected Node**	**Formed Edge**
0	A-D	A, D	A	-
1	C-D	A, C, D	D	A-D
2	A-B	A, B, C, D	C	A-D, C-D
3	B-D	*skipped*	B	A-D, C-D, A-B

The table above shows MST forming process step by step. Kruskal and Prim have different process in determining MST. On Kruskal algorithm, the first step that need to be done is select edge with the lowest cost. Then, nodes from the selected edge will be inserted into the tree. This step is done until all the nodes have been inserted into tree. On the step 3, edge B-D is skipped because it will create a cycle, and also all the node is already inserted before.

MST forming using Prim begins with selecting one of the node from the graph. On graph I, the first selected node is node A. There is no formed edge yet in the first step, because it has only one node. The next node that should be selected is the node which connected with node A, and have the lowest cost. Node D is selected, because edge A-D have the lowest cost in the graph I. Edge A-D then will be inserted into the tree. This step is done until all the nodes have been visited.

Based on the MST results above, both Kruskal and Prim algorithm generate same result. The MST path are A-D, C-D, and A-B. Visual form of MST results from sample I can be seen on the figures below.

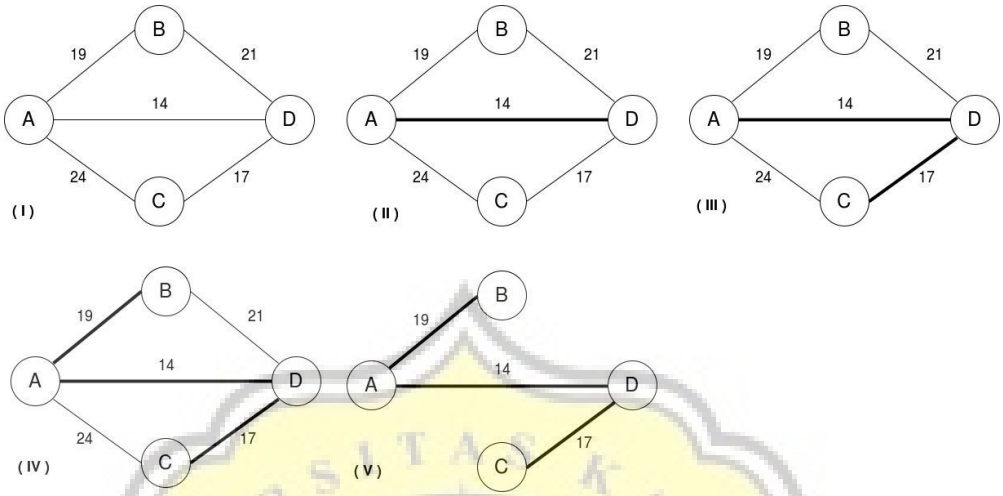

Illustration 5.6: MST Kruskal Form from Graph I

Illustration above shows MST forming step with Kruskal algorithm. Thick line means that edge is selected. After all the nodes is visited, the unselected edge the will be removed. The final form is shown on step (V).

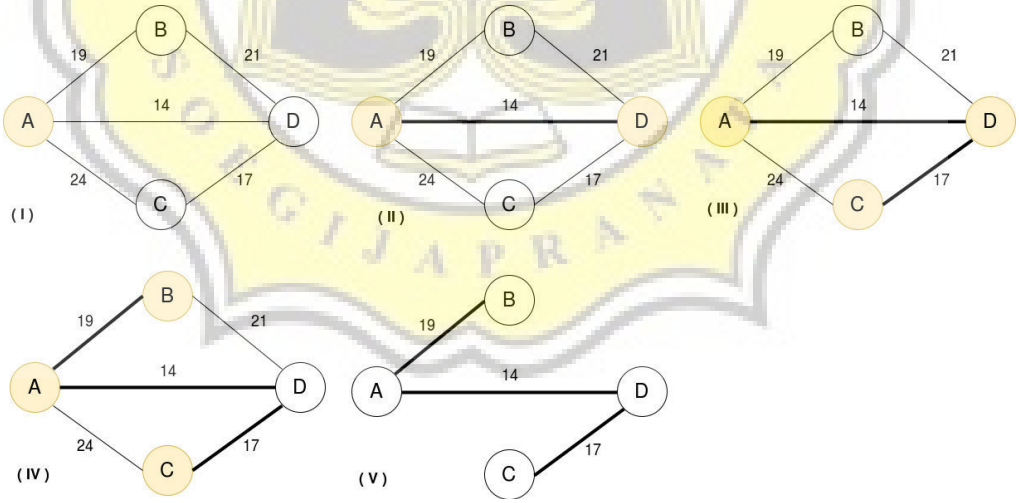

Illustration 5.7: MST Prim Form from Graph I

Illustration above shows MST forming step with Prim algorithm. The colored nodes means selected node, while the thick lines means that edge is inserted into the tree. The final form is shown on step (V).

Sample II

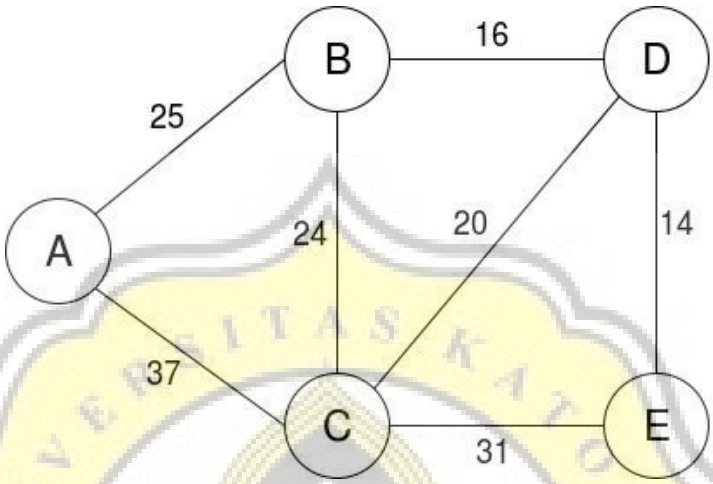

Illustration 5.8: Graph Sample II

Table 5.4: Data Sample II

Start Node	End Node	Cost
A	B	25
A	C	37
B	C	24
B	D	16
C	D	20
C	E	31
D	E	14

The results of MST program using Kruskal and Prim algorithm is shown on figures below.

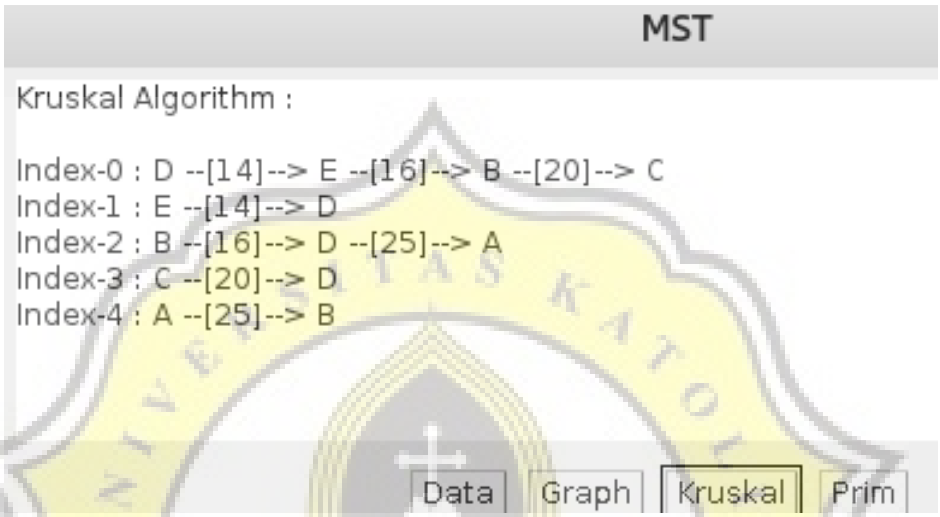

Illustration 5.9: MST Result Kruskal II

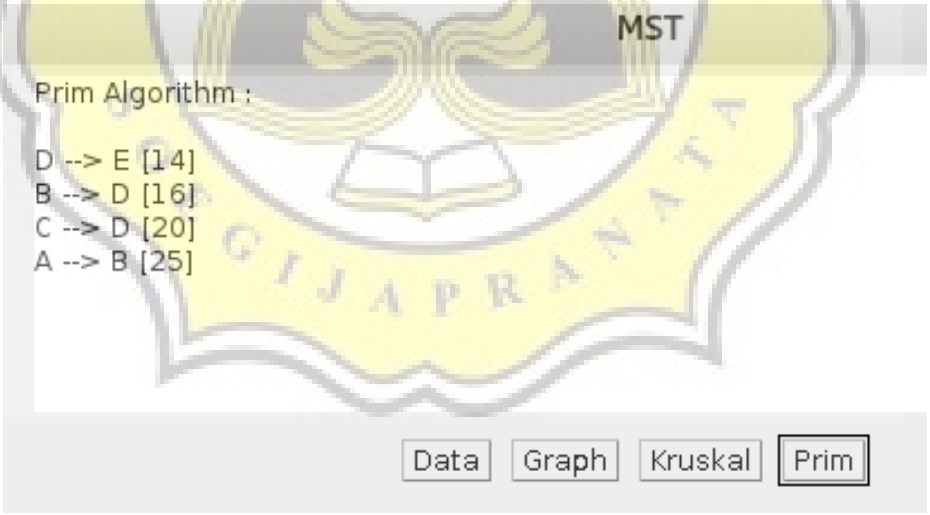

Illustration 5.10: MST Result Prim II

The figures above shows MST result using Kruskal and Prim algorithm. Based on the program, both algorithm generate the same result. The MST result are D-E, B-D, C-D, and A-B.

If MST is done manually, the results is as follows:

Table 5.5: MST Forming Steps from Graph II

Step	**Kruskal Algorithm**		**Prim Algorithm**	
	Selected Edge	**Inserted Node**	**Selected Node**	**Formed Edge**
0	D-E	D, E	D	-
1	B-D	B, D, E	E	D-E
2	C-D	B, C, D, E	B	B-D, D-E
3	B-C	*skipped*	C	B-D, C-D, D-E
4	A-B	A, B, C, D, E	A	A-B, B-D, C-D, D-E

The table above shows MST forming process step by step. On Kruskal algorithm, the first step that need to be done is select edge with the lowest cost. The first selected edge is D-E. On step 3, edge B-C is skipped because it caused a cycle. The step is done at step 4 when all the nodes have been inserted into tree.

MST forming using Prim begins with selecting one of the node from the graph. On graph II, the first selected node is node D. There is no formed edge yet in the first step, because it has only one node. The next selected node is E, because edge D-E have the lowest cost. This step will be done until all the nodes have been visited.

Based on the MST results above, both Kruskal and Prim algorithm generate same result. The MST path are D-E, B-D, C-D, and A-B. Visual form of MST results from sample II can be seen on figures below.

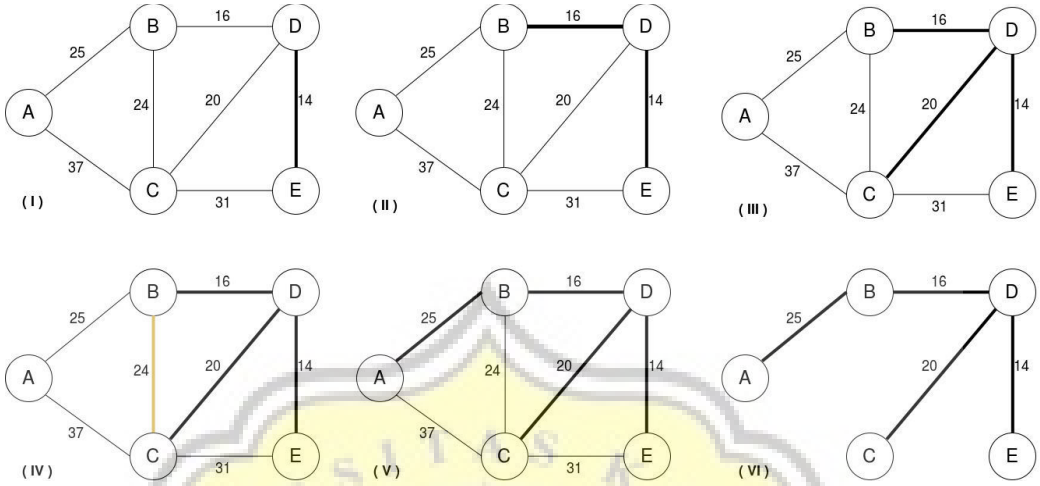

Illustration 5.11: MST Kruskal Form from Graph II

Illustration above shows MST forming step with Kruskal algorithm. Thick line means that edge is selected. The colored line indicates that edge is creating a cycle, so it will be skipped. After all the nodes is visited, the unselected edge the will be removed. The final form is shown on step (V).

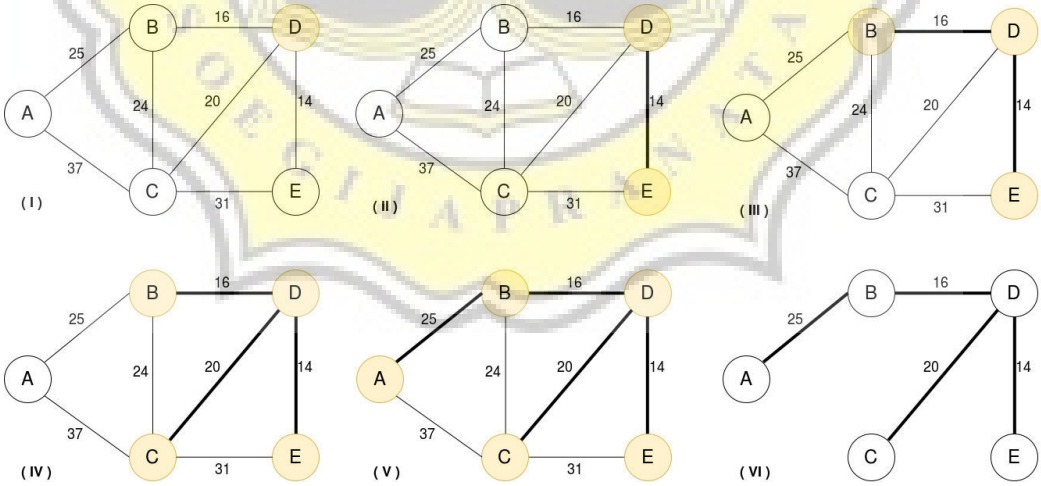

Illustration 5.12: MST Prim Form from Graph II

Illustration above shows MST forming step with Prim algorithm. The colored nodes means selected node, while the thick lines means that edge is inserted into the tree. The final form is shown on step (V).

Sample III

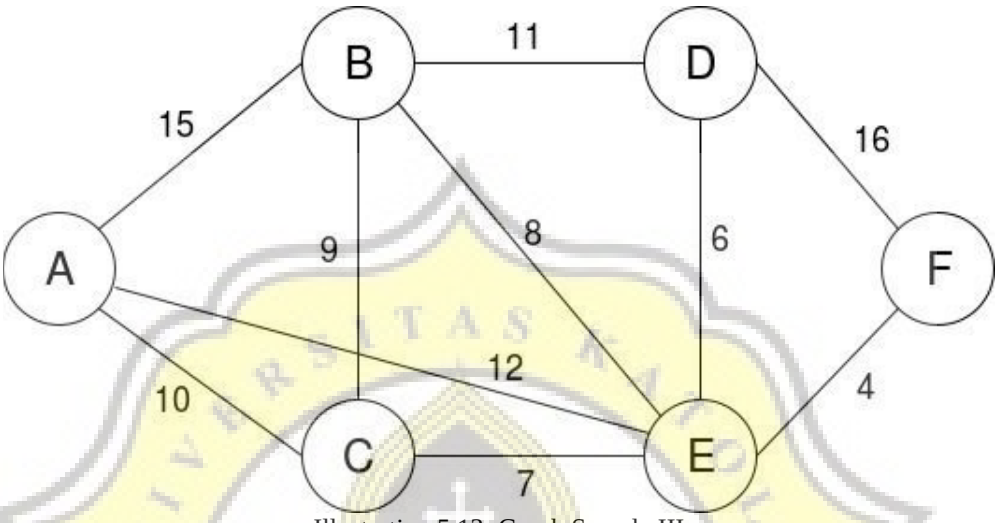

Illustration 5.13: Graph Sample III

Table 5.6: Data Sample III

Start Node	End Node	Cost
A	B	15
A	C	10
B	C	9
B	D	11
B	E	8
C	E	7
D	E	6
E	F	4
D	F	16
A	E	12

The result of MST using Kruskal and Prim algorithm is shown on figure below.

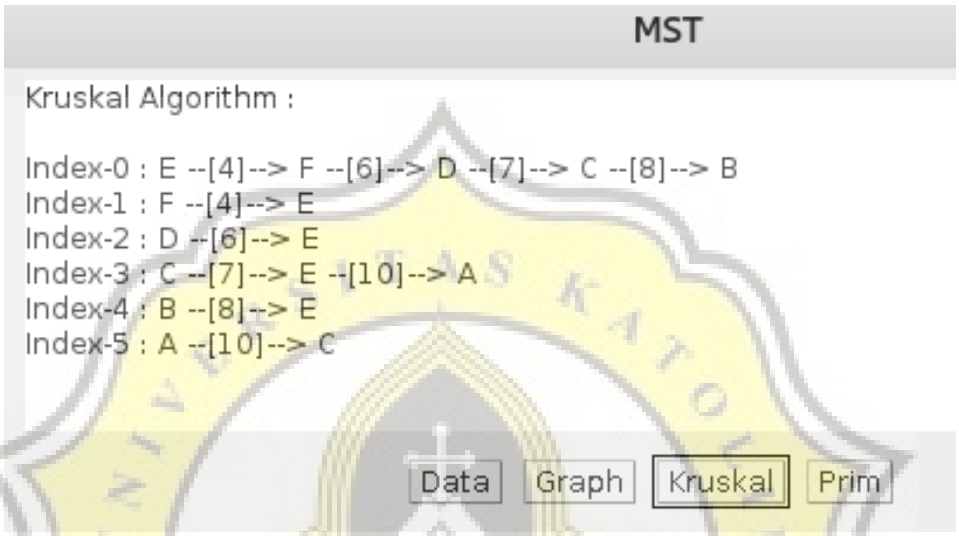

Illustration 5.14: MST Result Kruskal III

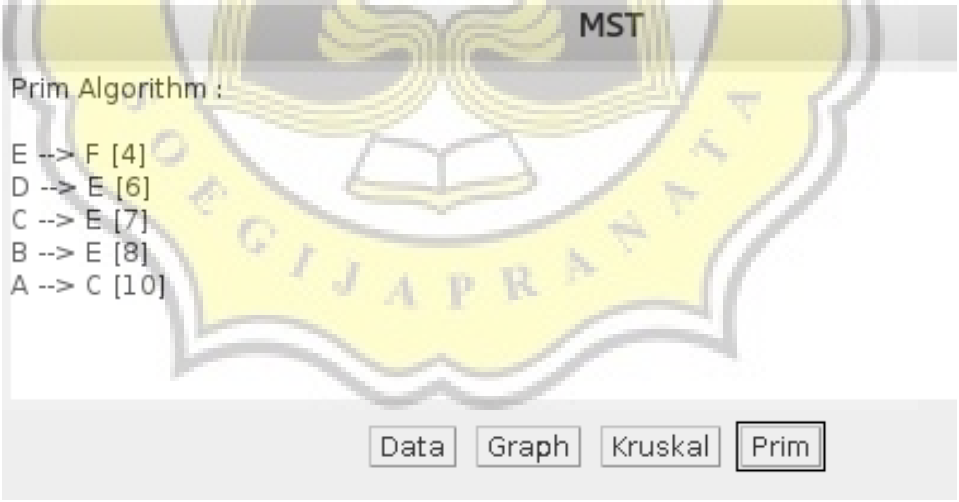

Illustration 5.15: MST Result Prim III

The figures above shows MST result using Kruskal and Prim algorithm. Based on the program, both algorithm generate the same result. The MST result are E-F, D-E, C-E, B-E, and A-C.

If MST is done manually, the results is as follows.

Table 5.7: MST Forming Steps from Graph III

Step	Kruskal Algorithm		Prim Algorithm	
	Selected Edge	**Inserted Node**	**Selected Node**	**Formed Edge**
0	E-F	E, F	E	-
1	D-E	D, E, F	F	E-F
2	C-E	C, D, E, F	D	D-E, E-F
3	B-E	B, C, D, E, F	C	C-E, D-E, E-F
4	B-C	*skipped*	B	B-E, C-E, D-E, E-F
5	A-C	A, B, C, D, E, F	A	A-C, B-E, C-E, D-E, E-F

The table above shows MST forming process step by step. On Kruskal algorithm, the first step that need to be done is select edge with the lowest cost. The first selected edge is E-F. Node E and F then wil be inserted into tree. On the step 4, edge B-C is skipped because it caused a cycle. The step is done at step 5 when all the nodes have been inserted into tree.

MST forming using Prim begins with selecting one of the node from the graph. On graph III, the first selected node is E. There is no formed edge yet in the first step, because it has only one node. The next selected node is F, because edge E-F have the lowest cost. Edge E-F then will be inserted into tree. This step will be done until all the nodes have been visited.

Based on the MST results above, both Kruskal and Prim algorithm generate same result. The MST path are E-F, D-E, C-E, B-E, and A-C. Visual form of MST results from sample III can be seen on figures below.

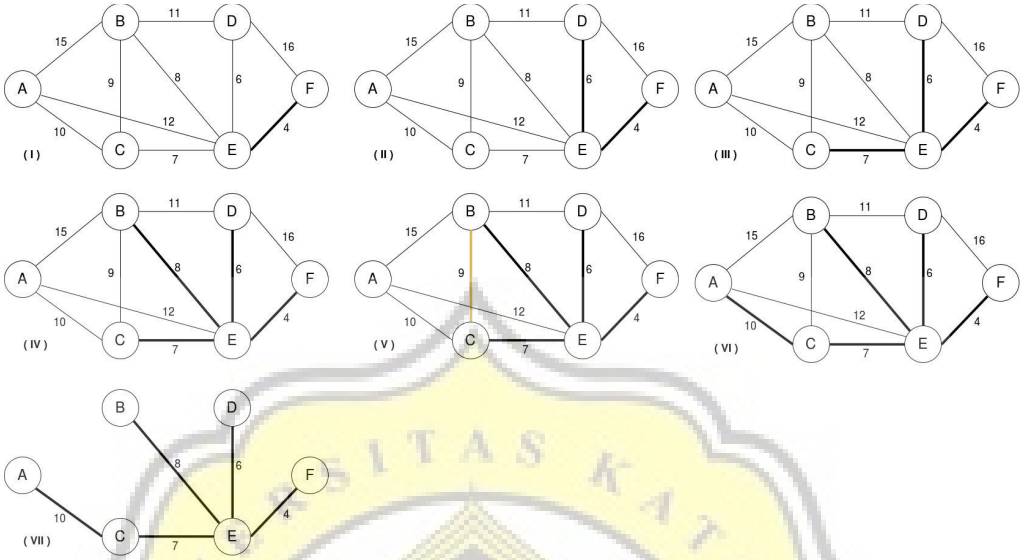

Illustration 5.16: MST Kruskal Form from Graph III

Illustration above shows MST forming step with Kruskal algorithm. After all the nodes is visited, the unselected edge the will be removed. The final form is shown on step (V). MST forming step with Prim algorithm can be seen on the illustration below.

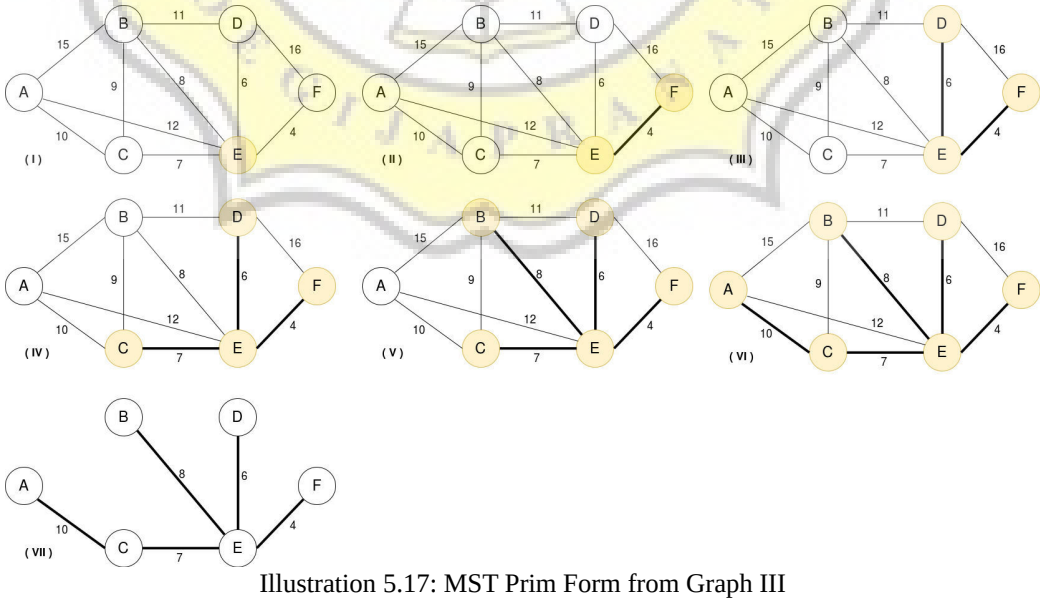

Illustration 5.17: MST Prim Form from Graph III

Sample IV

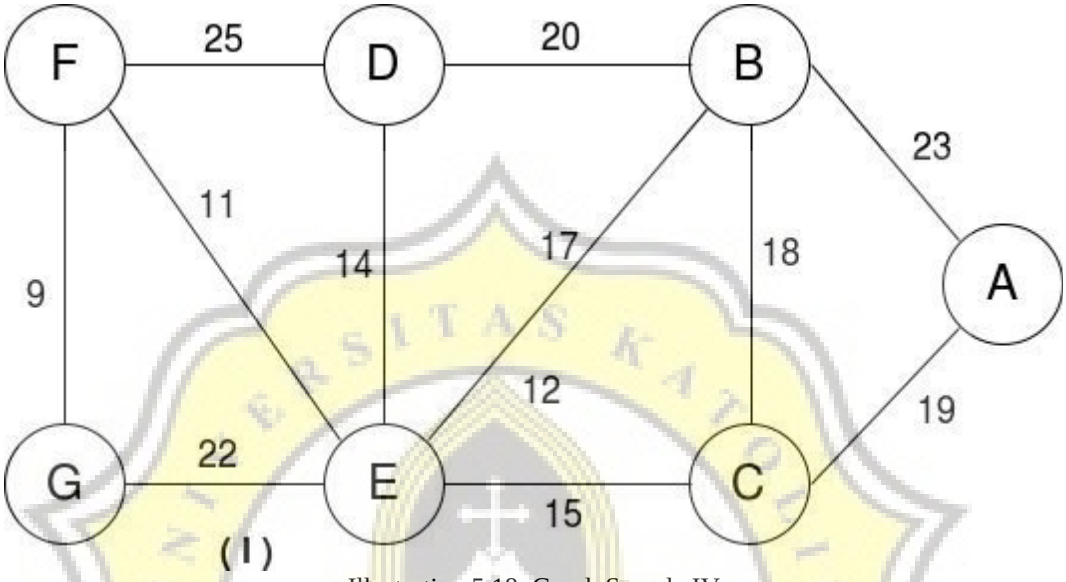

Illustration 5.18: Graph Sample IV

Table 5.8: Data Sample IV

Start Node	End Node	Cost
A	B	23
A	C	19
B	C	18
B	D	20
C	E	15
B	E	17
D	E	14
D	F	25
F	E	11
E	G	22
F	G	9

The results of MST using Kruskal and Prim algorithm is shown on figure below.

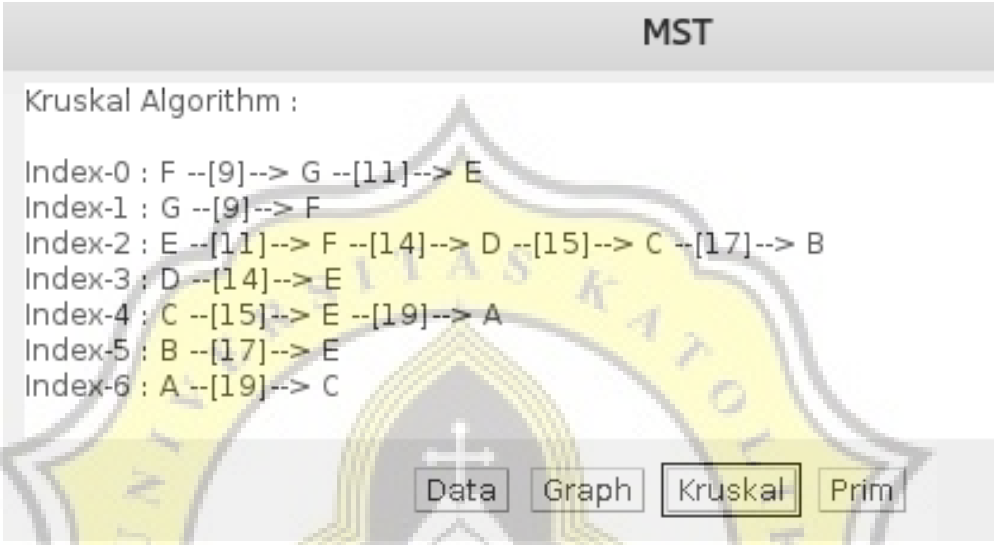

Illustration 5.19: MST Result Kruskal IV

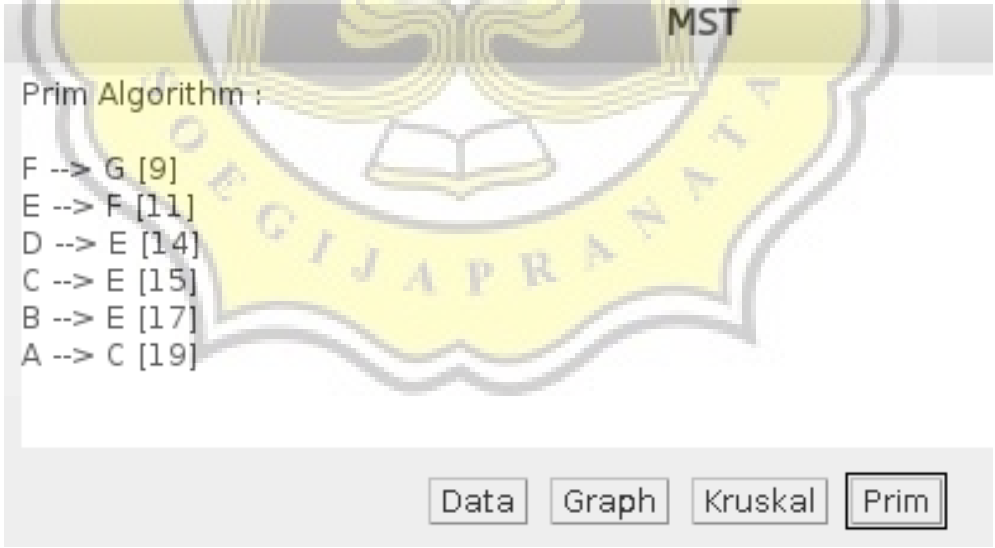

Illustration 5.20: MST Result Prim IV

The figures above shows MST result using Kruskal and Prim algorithm. Based on the program, both algorithm generate the same result. The MST result are F-G, E-F, D-E, C-E, B-E, and A-C.

If MST is done manually, the results is as follows.

Table 5.9: MST Forming Steps from Graph IV

Step	Kruskal Algorithm		Prim Algorithm	
	Selected Edge	**Inserted Node**	**Selected Node**	**Formed Edge**
0	F-G	F, G	F	-
1	E-F	E, F, G	G	F-G
2	D-E	D, E, F, G	E	E-F, F-G
3	C-E	C, D, E, F, G	D	D-E, E-F, F-G
4	B-E	B, C, D, E, F, G	C	C-E, D-E, E-F, F-G
5	B-C	*skipped*	B	B-E, C-E, D-E, E-F, F-G
6	A-C	A, B, C, D, E, F, G	A	A-C, B-E, C-E, D-E, E-F, F-G

The table above shows MST forming process step by step. On Kruskal algorithm, the first selected edge is E-F because it has the lowest cost. Node E and F then wil be inserted into tree. Edge B-C is skipped because it caused a cycle. The step is done at step 6 when all the nodes have been inserted into tree.

MST forming using Prim begins with selecting one of the node from the graph. On graph IV, the first selected node is F. There is no formed edge yet in the first step, because it has only one node. The next selected node is G, because edge F-G have the lowest cost. Edge F-G then will be inserted into tree. This step will be done until all the nodes have been visited.

Based on the MST results above, both Kruskal and Prim algorithm generate same result. The MST path are F-G, E-F, D-E, C-E, B-E, and A-C. Visual form of MST results from sample IV can be seen on figures below.

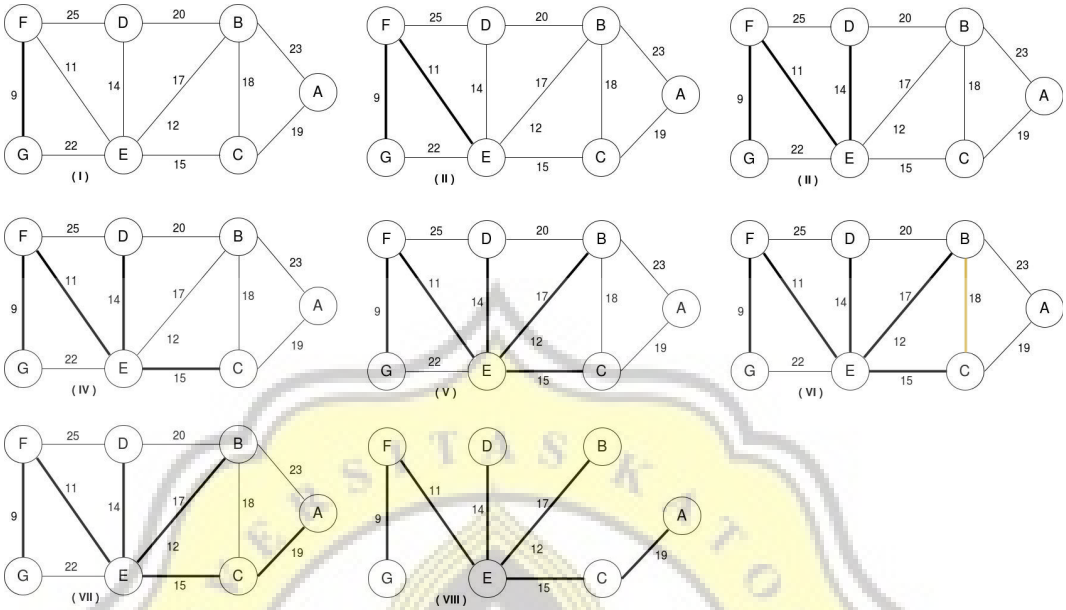

Illustration 5.21: MST Kruskal Form from Graph IV

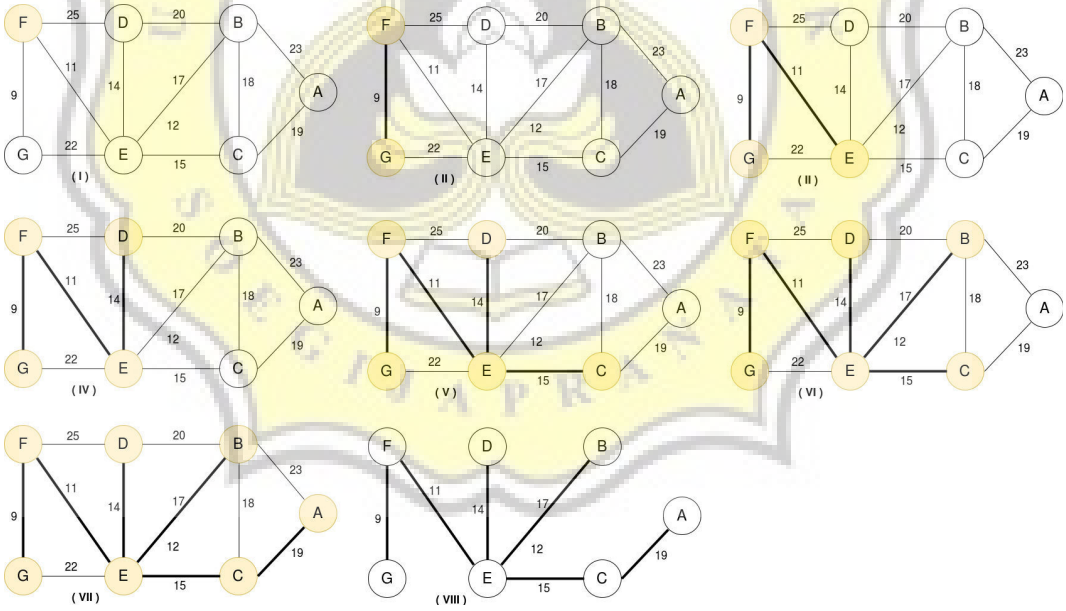

Illustration 5.22: MST Prim Form from Graph IV

The illustrations above are the MST forming step. On Kruskal, thick line means selected edge and colored line means skipped edge. While on Prim, colored node means selected node and thick line means inserted edge.