THE IMPLEMENTATION OF KRUSKAL'S ALGORITHM FOR MINIMUM SPANNING TREE IN A GRAPH

Paryati[1], Krit Salahddine[2]

[1]UPN "Veteran" Yogyakarta, Country Indonesia, Email: yaya_upn_cute@yahoo.com
[2]Ibn Zohr University, Agadir, Country Marocco, Email: salahddine.krit@gmail.com

Abstract

Kruskal's Algorithm is an algorithm used to find the minimum spanning tree in graphical connectivity that provides the option to continue processing the least-weighted margins. In the Kruskal algorithm, ordering the weight of the ribs makes it easy to find the shortest path. This algorithm is independent in nature which will facilitate and improve path creation. Based on the results of the application system trials that have been carried out in testing and comparisons between the Kruskal algorithm and the Dijkstra algorithm, the following conclusions can be drawn: that a strength that is the existence of weight sorting will facilitate the search for the shortest path. And considering the characteristics of Kruskal's independent algorithm, it will facilitate and improve the formation of the path. The weakness of the Kruskal algorithm is that if the number of nodes is very large, it will be slower than Dijkstra's algorithm because it has to sort thousands of vertices first, then form a path.

Keywords: Kruskal's Algorithm, Spanning Tree

1. INTRODUCTION

Bakground

The theories about graphs began in 1736 and several important discoveries in graph theory were obtained in the 19th century, but not until the 1920s that the interest in graph theory bloomed. The first text about graph theory emerged in 1936. Undoubtedly, one of the reasons for interest in graph theory is due to its application in many fields, including computer science, chemistry, operation research, electrical engineering, linguistics, and economy.

The research will address the definition of graph and some terminologies along with a basic example of a graph. The shortest path algorithm which will be used is Kruskal's algorithm. It will show us how to find the shortest path between two given points.

Figure 1.1. shows a highway system that must be supervised by the investigator officer. Particularly, this investigator must make trips to all roads and make reports about their condition, the clarity of roads path, the condition of traffic signs, and so on. Because she lives in Yogyakarta, the most economic way in examining all roads must be started in Yogyakarta.

© The Authors, published by EDP Sciences. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (http://creativecommons.org/licenses/by/4.0/).

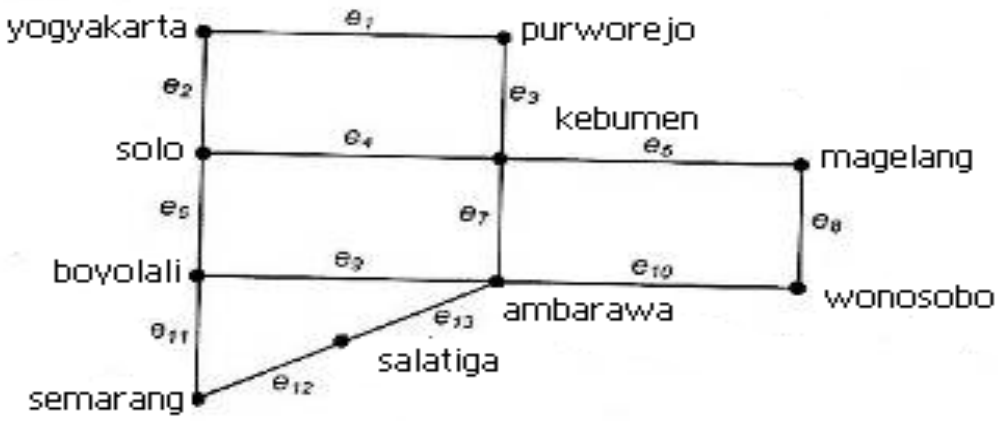


Figure 1. Shows a highway system that must be supervised by the investigator officer

2. LITERATURE REVIEW

2.1. The Definition of Graph and Digraph

A graph (or an undirected graph) *G* consists of a set *V* form vertices (nodes or points) and an *E* set from ribs (arcs) as such that every rib of $e \in E$ is associated with the unordered vertex pair. If there is an *e* rib that connects vertices *v* and *w*, it is written as $e = (v, w)$ or $e = (w, v)$. In this context, (v, w) suggest a rib between *v* and *w* is in an undirected graph and not an ordered pair.

A directed graph or digraph G consists of a set V from vertices (nodes or points) and an *E* set from ribs (arcs) as such that every rib of $e \in E$ connects the ordered vertex pair. If there is a single *e* rib that connects the ordered pair (v, w) from vertices, it is written as $e = (v, w)$, which suggests a rib from *v* to *w*.
An *e* rib in a graph (directed or undirected) which connects the vertices pair *v* and *w* is said to be incidental on *v* and *w*, then *v* and *w* is said to be incidental on *e* and referred to as adjacent vertices.If *G* is a graph (directed or undirected) with vertices *v* and ribs *E*, then it is written as $G = (V, E)$. If it is not specifically mentioned, the set *E* and *V* are assumed as finite and *V* is assumed as not empty.

In the case in the figure 1.1. above after the model making it will appear as follows:

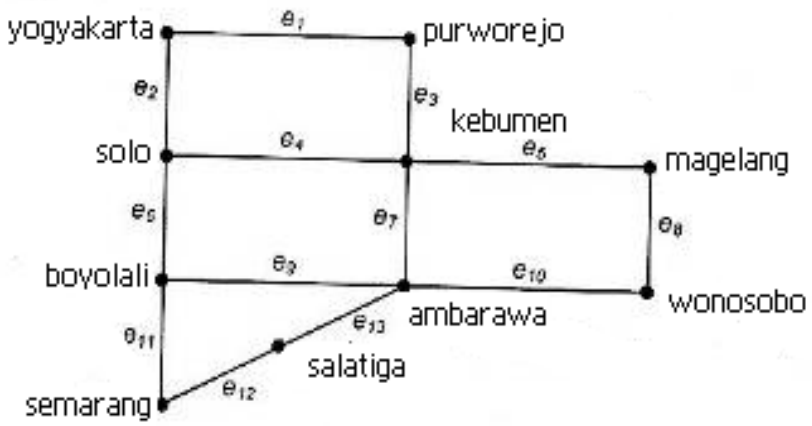


Figure 2. A Graph Model form a highway system shown in Figure 1.1.

The points in figure 1.1. are called vertices and the lines which connect those vertices are called ribs (edge). Every vertex is named by taking the first three letters of the appropriate city. The ribs are signed by e_1, …, e_{13}. In drawing a graph, the only important information are the imaginary vertices linked by the imaginary ribs as well. If we start from a vertex v_0, walk along a rib to vertex v_1, walk along another rib to vertex v_2 and so on, and finally arrive at vertex v_n, we refer to such complete trip as path from v_0 to v_n.

2.2. The Definition of Spanning Tree

A *T* tree is a Spanning Tree of graph *G* if *T* is a sub graph of *G* which contains all vertices of *G*. The characteristics of Spanning Tree are:

a. All graph *G* links in this tree must not form a circle.
b. A graph *G* is said to be a tree if and only if *G* is linked or form a path.
c. A linked graph G is said to be a tree if and only if its every edge forms a rib.
d. A linked graph G is said to be a tree if and only if it has an N peak and N-1 edge.

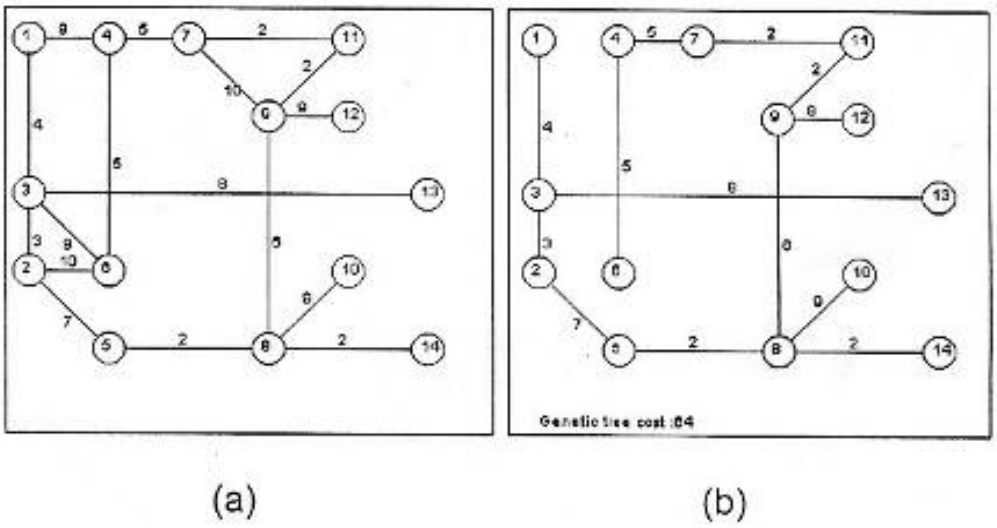


Figure 3. Tree

From the figure above, it can be stated that figure (a) is a wrong figure because it is not fit the characteristics of a tree as mentioned earlier. Figure (b) is a correct tree because it fulfills the characteristics of a tree.

2.3. The Definition of Kruskal's Algorithm

Kruskal's algorithm is one to find minimum spanning tree in a graph connectivity which gives options of always processing the edge limit with the least weight.

This algorithm run by considering the graph's biggest edge limit when searching for track in node in a graph which has been put into a spanning tree. If an edge limit is considered will integrate (with one of the points not in the spanning tree), or the integration of a point in spanning tree (one of which is not in the spanning tree), then the edge limit and the final point is included in the spanning tree. Considering one of the edge limits, algorithm continues by considering the next larger edge limit weight. In this term, when the edge limit weight values are the same, then it is unnecessary to determine which weight must be chosen first. Algorithm will stop when every point have been included in the spanning tree.

Note that, when spanning tree was made, there was a possibility that the edge limit is not connected with the part of the main tree, although in the end they will integrate with the main three.

2.4. Source Code

```
public class Algorithms
{
public static Graph KruskalsAlgorithm (Graph g)
{
int n = g.getNumberOfVertices();
Graph result = new GraphAsLists (n);
for(int v = 0; v < n ; ++v )
result.addVertex (v);
```

```
PriorityQueue queue =
new BinaryHeap (g.getNumberOfEdges());
Enumeration p= g.getEdges();
while (p.hasMoreElements())
{
Edge edge = (Edge) p.nextElement();
Int weight= (Int) edge.getWeight();
queue.enqueue (new Assosiation (weight, edge));
}
Partition partition = new PartitionAsForest(n);
while(!queue.isEmpty() && partition.getCount() > 1)
{
Association assoc = (Association) queue.dequeueMin();
Edge edge = (Edge) assoc.getValue ();
int n0 = edge.getV0 ().getNumber ();
int n1 = edge.getV1 ().getNumber ();
Set s = partition.find (n0);
Set s = partition.find (n1);
If (s !=t)
{
partition.join (s,t);
result.addEdge (n0,n1);
}
}
return result;
}
}
```

3. RESEARCH METHODOLOGY

Kruskal's Algorithm

The first research conducted used the testing method on Kruskal's algorithm step-by-step. The algorithms are as follows:

$E_{(1)}$ is a set of ribs from Minimum Genetic Tree
$E_{(2)}$ is a set of the remaining ribs
V is a set of vertices
n is node or point
Steps:
$E_{(1)}=0, E_{(2)}=E$
WHILE $E_{(1)}$ consists less than $n-1$ ribs and $E_{(2)}=0$ DO

- Form the side of $E_{(2)}$ choose one with the lowest value→ $e_{(ij)}$
- $E_{(1)}=E_{(2)}-\{ e_{(ij)}\}$
- IF $V_{(i)}$, $V_{(j)}$ is not included in the same tree, THEN
- Unite tree from $V_{(i)}$ and $V_{(j)}$ become one single tree
- END of IF

- END of WHILE
- END of Algorithm

4. ANALYSIS

In conducting analysis for Kruskal's algorithm a test is employed by having implementation over all two algorithms with the case example as follows:

Kruskal's Algorithm

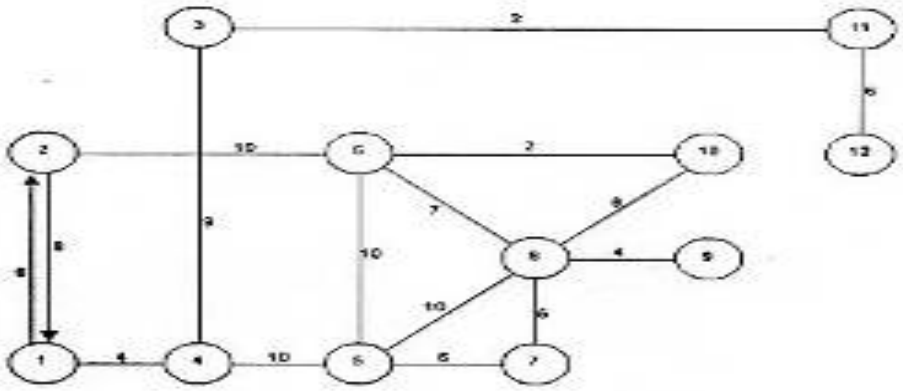


Figure 4. Implementation over all two algorithms

Initial step: if we look at the characteristics of a tree, then the above graph does not fulfill the characteristics of a graph, because the track is circular. Thus, we will include Kruskal's algorithm to solve the problem above. The first step to run Kruskal's algorithm is arranging *e* (rib weight) in the right order from the minimum value to the maximum. After the arrangement (Exhibit 4.1.a) then we make the track for both vertices with minimum weight.

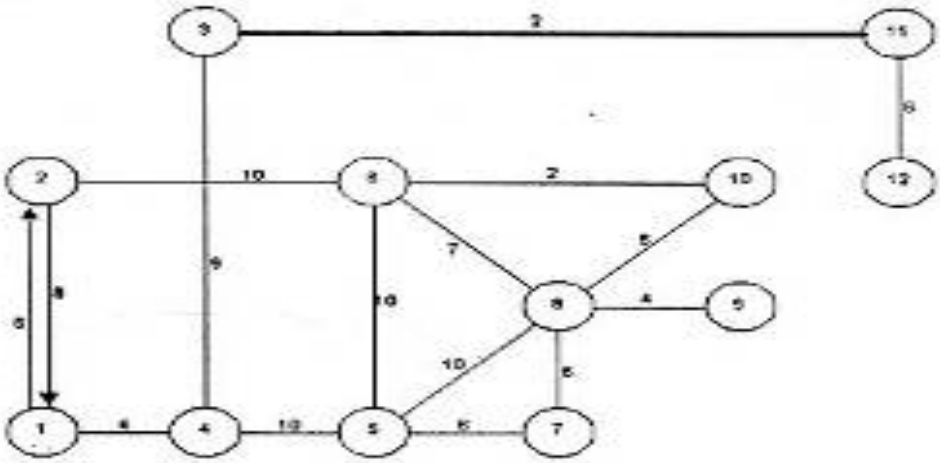


Figure 5. The first step to run Kruskal's algorithm

Step 1: in the line with the table order, then what to do first is making track $V_3 - V_{11}$ with the rib weight 2 (track with bold lines).

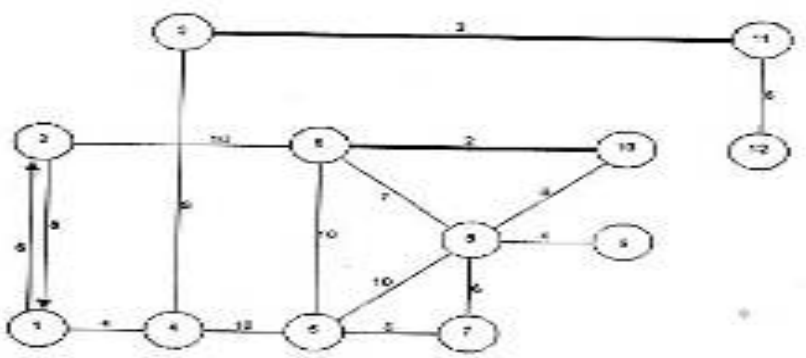


Figure 6. Step 1

Step 2: the next order is $V_6 - V_{10}$ with the rib weight 2. It appears that the track is separated from step 1. However, considering the characteristics of the independent Kruskal's algorithm which means that there is a possible edge limit (vertex) unconnected with the part of the main tree, although eventually will join the main tree, it can happen.

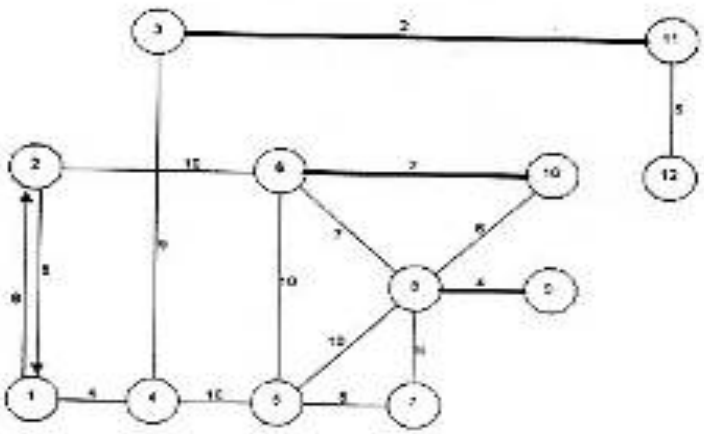


Figure 7. Step 2

Step 3: the next order is $V_8 - V_9$ with the rib weight 4.

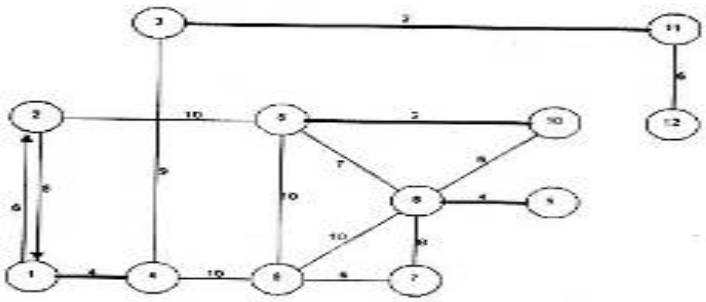


Figure 8. Step 3

Step 4: the next order is $V_1 - V_4$ with the rib weight 4.

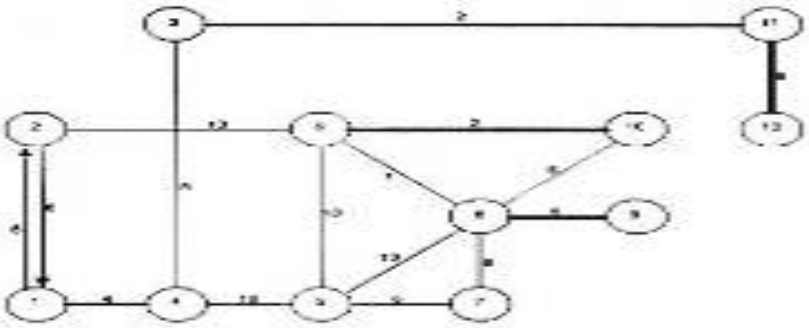


Figure 9. Step 4

Step 5: the next order is $V_{11} - V_{12}$ with the rib weight 6.

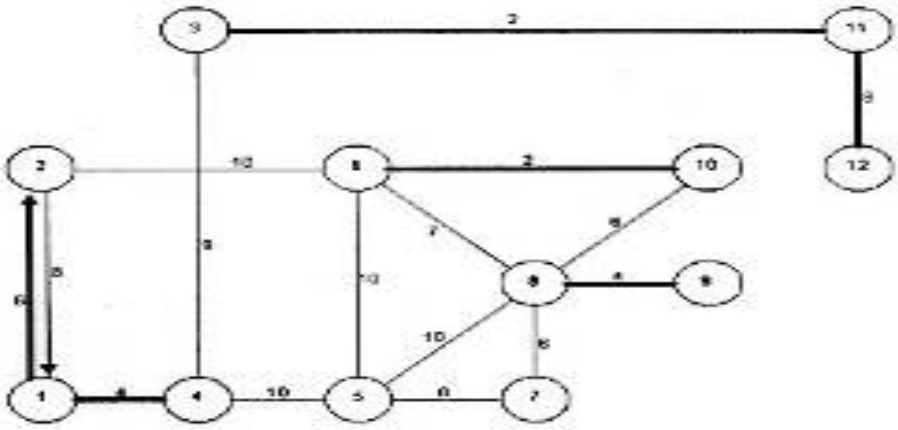


Figure 10. Step 5

Step 6: the next order is $V_1 - V_2$ with the rib weight 6.

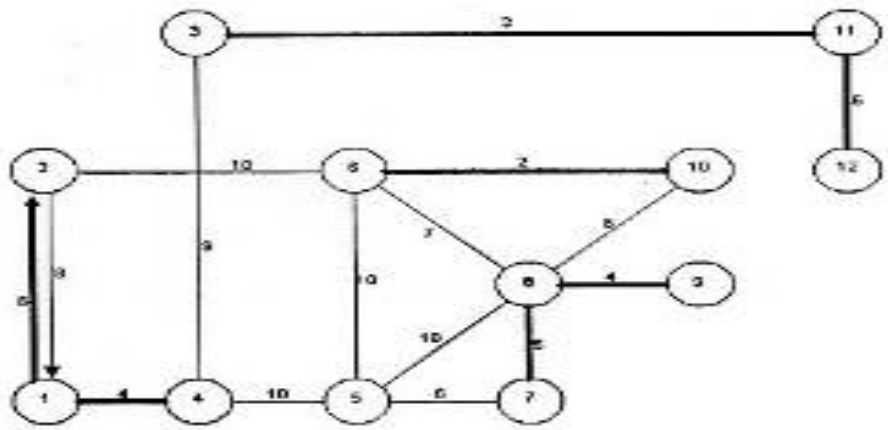


Figure 11. Step 6

Step 7: the next order is $V_8 - V_7$ with the rib weight 6.

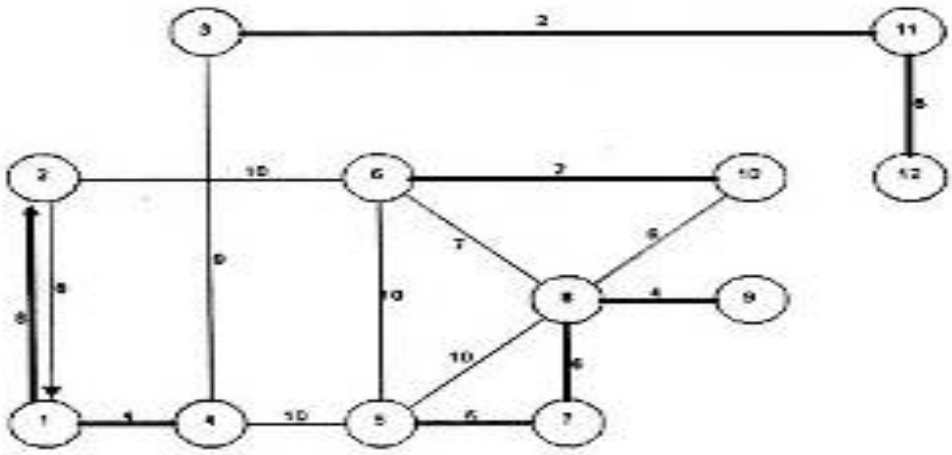


Figure 12. Step 7

Step 8: the next order is $V_5 - V_7$ with the rib weight 6.

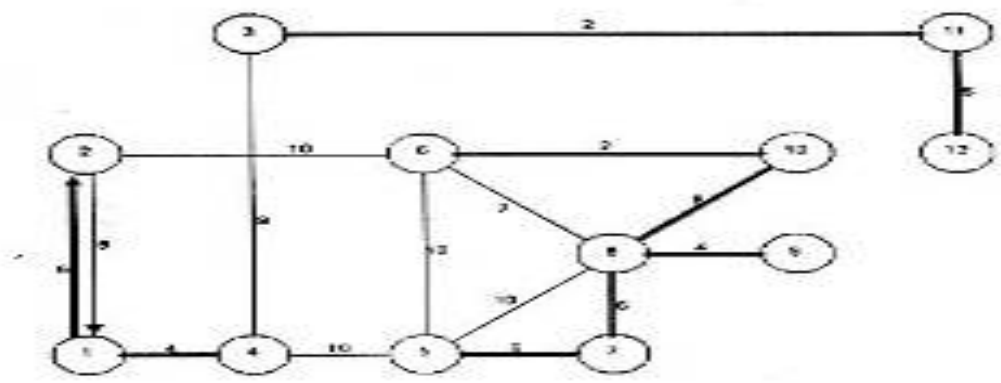


Figure 13. Step 8

Step 9: the next order is $V_8 - V_{10}$ with the rib weight 6.
Step 10: the next order is $V_6 - V_8$ with the rib weight 7. This path is not allowed because it will form circle in path $V_6 - V_8 - V_{10}$.
Step 11: the next order is $V_2 - V_1$ with the rib weight 8. This path is not allowed because vertex $V_2 - V_1$ has been passed. In fact, the requirement of a tree only allows it to be passed once.

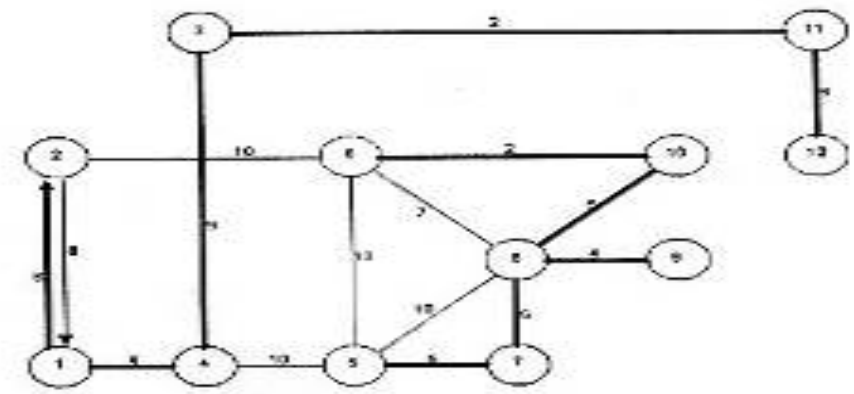


Figure 14. Step 11

Step 12: the next order is $V_3 - V_4$ with the rib weight 9.

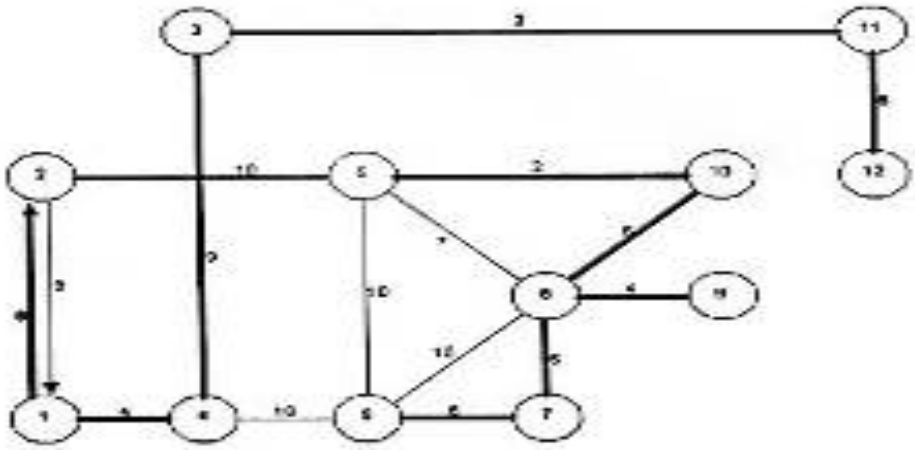


Figure 15. Step 12

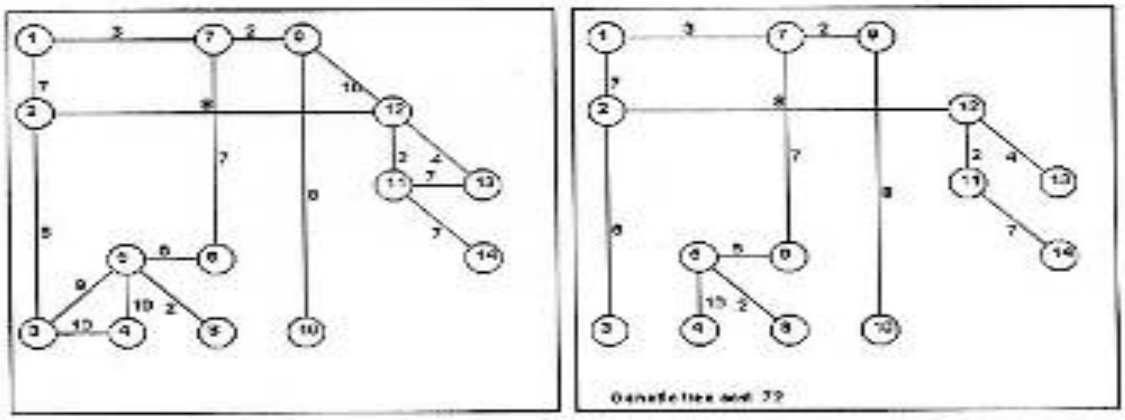
Step 13: the next order is $V_2 - V_6$ with the rib weight 10.
Step 14-16: this order is unnecessary to be done because it forms a circle.

Table 1. Step of path determination in Kruskal's algorithm

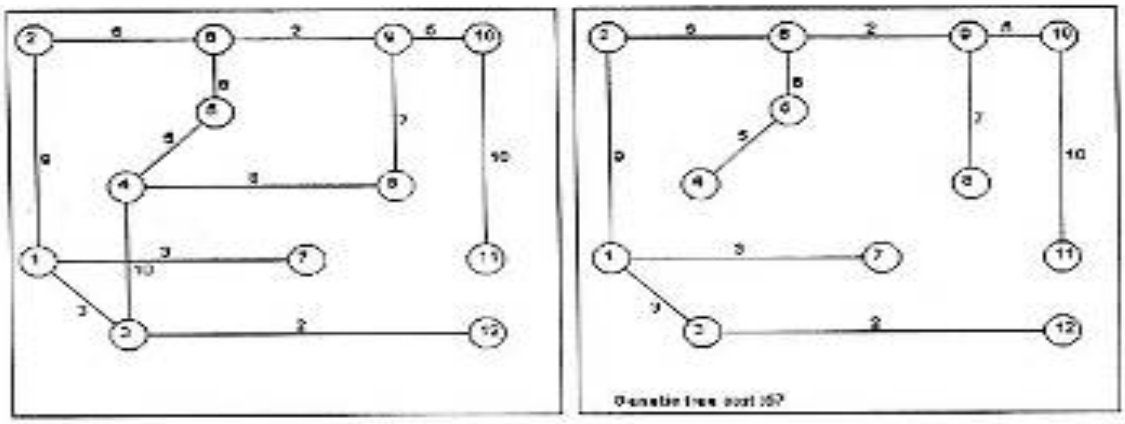
Step	v	e	
1	3-11	2	add to tree
2	6-10	2	add to tree
3	8-9	4	add to tree
4	1-4	4	add to tree
5	11-12	6	add to tree
6	1-2	6	add to tree
7	8-7	6	add to tree
8	5-7	6	add to tree
9	8-10	6	add to tree
10	6-8	7	reject because it forms a circuit
11	2-1	8	reject because it forms is turned back
12	3-4	9	add to tree
13	2-6	10	add to tree
14	6-5	10	reject because it forms a circuit
15	4-5	10	reject because it forms a circuit
16	5-8	10	reject because it forms a circuit

Total bobot-------> 61

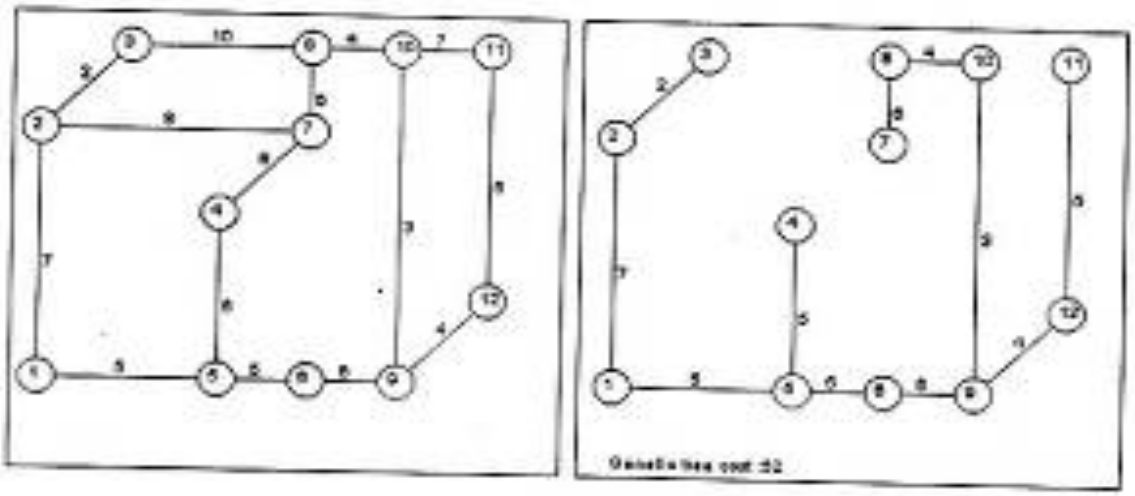
Other examples of Kruskal's Algorithm are as follows :



(a) Case (b) Solution
Figure 16. Step 14



(a) Case (b) Solution
(b) Figure 17. Step 15



(a) Case (b) Solution
Figure 18. Step 16

5. CONCLUSION

After test and comparison between Kruskal's algorithm, then conclusion can be drawn as follows:

a. Strength
   - The existence of weight sorting will ease the searching of the shortest path.
   - Considering the characteristics of Kruskal's independent algorithm, it will ease and enhance the formation of path track.

b. Weaknesses
   If the number of vertices is very large, it will slower than the Dijkstra's algorithm because it must sort thousands of vertices first, then forming the path.

REFERENCES

AKL, Selim G, 2019, The Desien and analysis of krus kall Algorithms, New Jersey Prentis Hall lnc
Budd Timothy, 2019, Object Oriented Programming Reading, Addison Wiley Publisher co lnc

Finkel,R.A and j.L.Bentley, 2019, Quadtrees A Data Structures For Retrieval on Composite Key, A cta Informatika Vol 4. Rlo.1

Richard Johnsonbaugh, 2019, Discrete Mathematics 4th ed.,Prentice-Hall, New Jersey
Quinn, Michael, 1994, Kruskal Computing Theory and Practic Secand edition Singapore Mc Graw Hill