

Using Prim's Algorithm as a Method to Dynamically Distribute Bandwidth

Mario Filino (13512055)¹

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia

¹mario.filino@students.itb.ac.id

Abstract—Graph as discrete structure has many implications in representing real life data and by using Prim's algorithm to find minimum spanning tree, many kinds of optimization can be made. One possible optimization is by making priority in the graph such that edges with larger weight value will be given higher priority. In this paper, writer will use that priority in distributing bandwidth speed in communication links depending on their traffic. This paper will focus on looking for priority in a graph by using Prim's algorithm to find maximum weighted tree thus being given higher priority. This method of searching for priority in a graph has many other implication besides the on discussed in this paper.

Index Terms—Minimum spanning tree, Prim's algorithm, Bandwidth, Graph

I. INTRODUCTION

Graph is a type of discrete structure firstly introduced by Leonhard Euler on his paper, *Seven Bridges of Königsberg*. Graph has many kind of application both in mathematics and computer science. One application of the graph theory is its implementation as a method to dynamically distribute bandwidth by implementing a Prim's algorithm to find the minimum spanning tree on a weighted graph which will be discussed in this paper.

Nowadays, data network can be found almost anywhere we go and it is part of our life to be connected to a network whether it is by an email or a social media. Growing network traffic is a common thing nowadays and to compensate that needs of larger bandwidth on a network, one solution might lie in optimizing data bandwidth distribution across the network.

Suppose for example we have a network connecting A to B and C connected to B, with the same static bandwidth distribution in the link connecting A to B and C to B, say 100 Mbps. In this example, assume that node B is the root bridge. When traffics are equally distributed across two links or that the traffic are below the network link capacity, it might not be a problem and wouldn't require any optimization. Now, consider a case where the traffic in the link connecting node A and B has a much higher traffic and that the traffic has exceed its link bandwidth capacity whereas link connecting node C and B has a

lower traffic. In this case, network congestion will occur in the link connecting node A and B which can result in poor performance of the network and the deterioration of the quality of service. In many cases, we might see the result of this congestion as a delay in data queuing, data loss or even the rejection of a new connection. We can see from the above example that an equally distributed bandwidth speed may not always be the best solution in a data network. Such problem can be a huge disadvantage when referred to a large scale network connection say in a data network based company or any company with large data traffic on the customer service area. Addressing the above case, optimization can to be made by dynamically distributing bandwidth speed across the network based on the traffic which can be regarded as priority level by using an algorithm called Prim's algorithm. So that a higher traffic will have a higher bandwidth speed whereas the lower traffic can have its link bandwidth speed lowered thus improving the efficiency of the limited bandwidth speed. Thus, this paper will discuss a way of implementing the minimum spanning tree using the Prim's algorithm to make priority on the bandwidth distribution of links connecting nodes in a network.

II. THEORIES

2.1 Definition of Graph

Graph is a discrete structure used to represent the connection of objects called vertices by links called edges. Graph has many implications in representing acquaintanceships between objects such as cities, power grid, computers in network, and even in biology using a specific structure of a graph called tree to represent a phylogenetic tree. Mathematically, graph is defined by a notation of $G = (V, E)$ with V being the set of vertices and E being the set of edges of a graph. In a graph, set V has to be a nonempty set whereas the set E can be an empty set. An empty graph is defined as a graph having an empty set of E . There are several types of graph depending on the existence of direction on the edges, whether vertices can be connected by multiple edges, whether edges can form a loop, and whether the edges have a value. Graph with the edges having direction is called directed graph

whereas the opposite is called undirected graph. Graph with multiple edges connecting two vertices is called nonsimple graph whereas the opposite is called simple graph. The type of graph that will be considered in this paper is a simple graph with undirected edges.

Consider the following example of a graph:

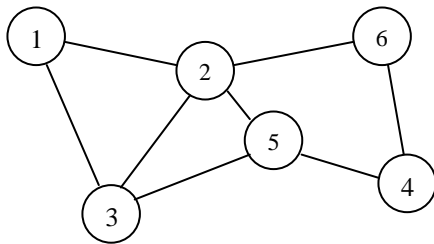


Figure 2.1 Graph

The above image is an example of a simple graph with undirected edges with six vertices and eight edges. Thus, we can write:

$$V = \{1, 2, 3, 4, 5, 6\}$$

$$E = \{(1, 2), (1, 3), (3, 5), (2, 3), (2, 5), (2, 6), (5, 4), (6, 4)\}$$

2.2 Weighted Graph

A weighted graph is simply a graph with the edges being given a number as a value or weight of the edges. These values can represent many things, it can be a number representing distance between vertices, cost going from one vertex to another vertex, etc. We usually call these values as the weight of the edges.

Here is a graph from figure 2.1 with values given to its edges as a weight.

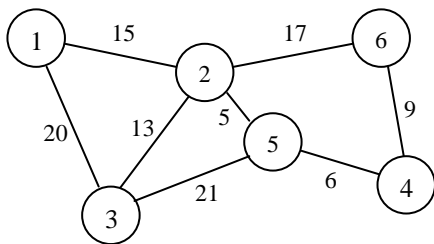


Figure 2.2 Weighted Graph

In a graph these edges, vertices and weight can be used to represent many things, that is why graph as a data structure has many implications in representing real life data.

2.3 Minimum Spanning Tree

Spanning tree is a subgraph which contains all vertex of the graph and does not create a cycle. For every connected graph there will be a spanning tree. From the above example of weighted graph in figure 2.2, many spanning tree can be generated from such graph.

Here are several example of spanning tree generated from the weighted graph in figure 2.2.

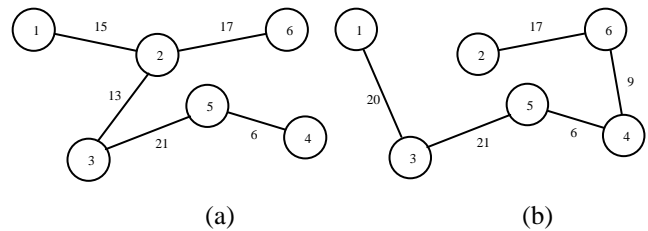


Figure 2.3 (a) Spanning Tree One (b) Spanning Tree Two

The two examples above shows different kind of spanning tree generated from the same graph in figure 2.2. The difference between both spanning tree above can be distinguished by the total cost obtained by adding all the total weight of a tree. Spanning tree (a) has a total cost of 72 whereas spanning tree (b) has a total cost of 73. Many other spanning trees can be generated from figure 2.2. From all of the spanning trees generated, the spanning tree with the least total cost is called the minimum spanning tree.

Here is the minimum spanning tree generated from the graph illustrated in figure 2.2.

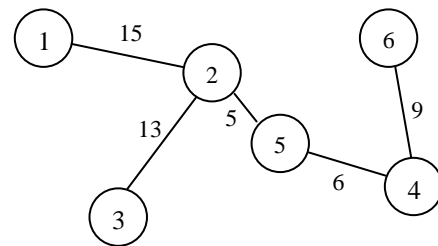


Figure 2.4 Minimum Spanning Tree

The above image is the minimum spanning tree generated using Prim's algorithm, it has a total cost 48 which is the lowest cost possible with all vertex connected.

2.4 Prim's Algorithm

There are several algorithms that can be used for generating minimum spanning tree from a graph: Borůvka, Prim, and Kruskal algorithm. All three algorithms are a greedy algorithm meaning that the choice made at one stage does not affect decision on future stages. In this paper the writer uses the Prim's algorithm to find the minimum spanning tree.

Prim's algorithm has several time complexity depending on the data structure being used: $O(V^2)$ by using adjacency matrix, $O(E \log(V))$ by using binary heap and adjacency list, and $O(E + V \log(V))$ by using fibonacci heap and adjacency list. The implementation using fibonacci heap has the lowest time complexity but with a much more complicated algorithm than with adjacency matrix or binary heap.

Here is the pseudo code for Prim's algorithm.^[4]

```

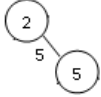
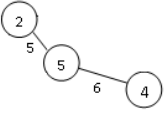
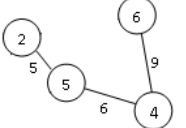
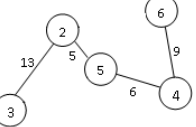
1. for each  $u \in V$ 
2.   do  $D[u] \leftarrow \infty$ 
3.  $D[r] \leftarrow 0$ 
4.  $PQ \leftarrow \text{make-heap}(D, V, \{\}) // \text{No edges}$ 
5.  $T \leftarrow \emptyset$ 
6.
7. while  $PQ \neq \emptyset$  do
8.    $(u, e) \leftarrow PQ.\text{extractMin}()$ 
9.   add  $(u, e)$  to  $T$ 
10.  for each  $v \in \text{Adjacent}(u)$ 
11.    do if  $v \in PQ \ \&\& \ w(u, v) < D[v]$ 
12.      then  $D[v] \leftarrow w(u, v)$ 
13.        PQ.decreasePriorityValue
           (  $D[v], v, (u, v)$  )
14. return  $T // T$  is a mst.

```

Figure 2.5 Prim's Algorithm Pseudo Code

The above pseudo code explains the steps of how Prim's algorithm generated a minimum spanning tree. Lines 1 – 5 initialize the priority queue PQ with all vertices of a graph. All the vertices are then set to infinity (line 2) except for vertex r which is set to zero (line 3). The vertex r then becomes the starting vertex of the tree T. The tree T is then initialize with Nil (line 5) indicating an empty tree. Lines 8 – 9 then adds the closest vertex and edge to tree T.^[4] Lines 10 – 13 works by getting all the adjacent vertices, then updates the D value for all adjacent vertex which has not been added to the tree, save the current minimum weight edge, then restore the heap property.^[4] The iteration continues until the priority queue PQ which initially contains all vertices of the graph becomes an empty set, the minimum spanning tree T is then returned by the function.

By using the above code to generate a spanning tree, here is the example of the steps in generating minimum spanning tree from graph in figure 2.2.

Steps	Edges	Weight	Spanning Tree
1	(2, 5)	5	
2	(5, 4)	6	
3	(4, 6)	9	
4	(3, 2)	13	

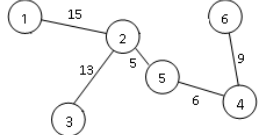
5	(1, 2)	15	
---	--------	----	---

Table 1 Steps of Generating Minimum Spanning Tree

From the above execution table, we can see that in every step we select a vertex which will contributes to giving the minimum edge weight. Going from step 1 to step 2 we can either choose vertex 1, 3, 6, or 4. By connecting vertex 5 and 4, we get the lowest edge weight of 6, thus we choose vertex 4 and edge (5, 4). The similar steps continue until the spanning tree contains all vertices from the graph. In the example above, we get a minimum spanning tree with total cost 48.

2.5 Maximum Spanning Tree

In this paper, the writer uses Prim's algorithm to generate a maximum spanning tree. This can be done in two ways: changing the Prim's algorithm to pick the maximum edges weight or by negating the value of the edges weight. The writer uses the second option to maintain Prim's algorithm as it is and still generating a minimum spanning tree. Thus by replacing the negated value with the original edges weight value after minimum spanning tree is generated; we generated the maximum spanning tree with the highest possible total cost.

2.6 Computer Network

Today, internet being a specific computer network has become the largest system created by mankind, with hundreds of millions of connected computer, communication links, and switches; with billions of users connected to it through their laptop, desktop, smartphones, tablets, and other gadget devices.^[3]

Simply, a computer network is telecommunication network connecting computers or end systems in general that allow them to exchange data. In this paper, nodes of a graph will represent the network's end system and the edges representing the network's communication link. The absolute value of the edges weight will represent communication links traffic or priority, with priority level proportional to its traffic; higher traffic means higher priority level.

III. ANALYSIS AND IMPLEMENTATION

In this paper, the writer uses a simple example of a network represented in the graph below and will use Prim's algorithm to increase the effectiveness of bandwidth distribution throughout the network.

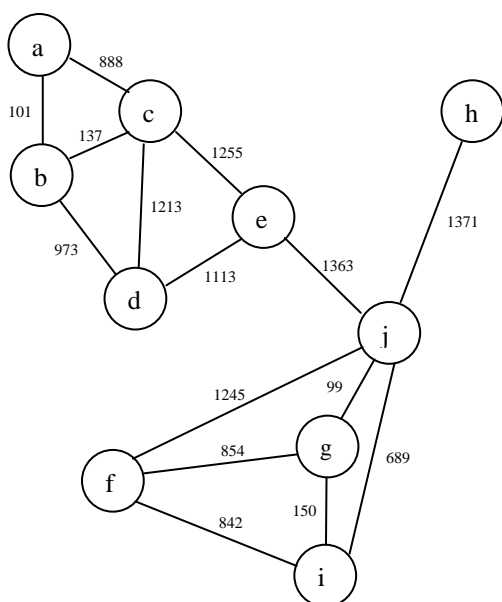


Figure 3.1 Simple Network Diagram

From the above graph, let nodes a, b, c, and d represent regional offices of a certain company with e being the head office. Nodes j and h being the local and global isp respectively. The rest of the nodes represent home networks connected to the local isp. Now assume that bandwidth is equally distributed throughout the whole network with traffic capacity (representing bandwidth capacity) of 1000, meaning that each communication link can handle up to 1000 packet of data.

In the above case it can be seen that network congestion will occur on link (c, e), (c, d), (d, e), (e, j), (f, j), and (f, i). This congestion can result in data lost and poor network connection which has to be avoided. For this case, the writer suggest implementing Prim's algorithm in order to find which communication link has to be given a higher bandwidth speed and which can have a lower bandwidth speed. Therefore, bandwidth does not have to be distributed equally but instead allocating it according to priority or traffic. In that way, communication links with higher traffic or priority will have higher bandwidth speed whereas lower priority communication links will have lower bandwidth speed.

The method of finding the solution for this problem is straightforward due to the network diagram already represented as a graph. Therefore any kind of transformation into a graph is unnecessary. The edges weight in the given graph above are in positive value, therefore we can't use Prim's algorithm directly without any modification as it will generate spanning tree with the lowest traffic instead of the highest traffic. Thus, it is necessary to negate these values first.

In this paper, the writer will find the solution for the above problem in three levels of bandwidth distribution: highest, second highest, and lowest bandwidth distribution. This is to differentiate bandwidth distribution in communication links with very low traffic such in

communication link between node g and i with communication links with high traffic but below bandwidth capacity such as communication link connecting node i and j. By using three level of bandwidth distribution, each communication link will have bandwidth speed closest to their traffic and therefore more efficient.

To find these three levels of bandwidth distribution, finding several minimum spanning trees is needed with different types of modification to the edges weight value. To find the first minimum spanning tree, values representing traffic have to be negated thus generating spanning tree with highest absolute value of the total cost. If values of exceeding traffic capacity still exist in remaining edges of the graph, the second minimum spanning tree from the same graph will have to be generated. To find spanning tree with the lowest traffic, original values of the graph edges will be used.

Here is the network graph with the negated edges value to generate spanning tree with highest traffic, all edges have the same color which is black indicating same bandwidth distribution throughout the whole network.

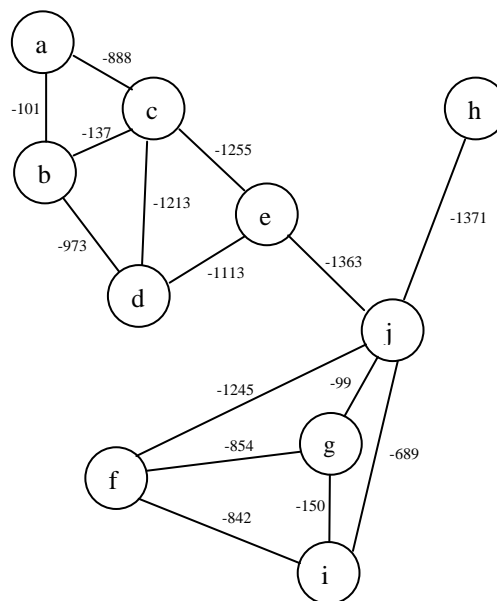


Figure 3.2 Network Graph with Negated Values

To find the minimum spanning tree, start by selecting an edge with the lowest value. Thus, edge connecting node h and j is selected with its value -1371. In node j, node e, f, g, and i can be chosen. Edge (e, j) is selected due to having the lowest value of -1363. Then we continue this process by choosing edge (c, e), edge (f, j), edge (c, d), edge (a, c), edge (b, d), edge (f, g), and edge (f, i). All edge with the minimum possible values without creating any cycle. Thus we have generated the minimum spanning tree from the above graph.

Here is the result of the minimum spanning tree generated from the above graph indicated by red colored edges with the steps given in the table below.

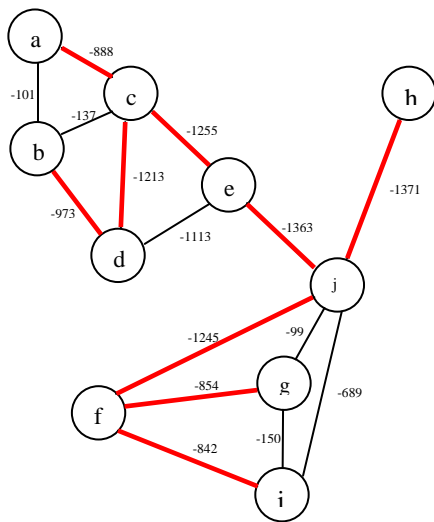


Figure 3.3 Links with Highest Traffic

Steps	Edges	Weight
1	(j, h)	-1371
2	(e, j)	-1363
3	(c, e)	-1255
4	(f, j)	-1245
5	(c, d)	-1213
6	(a, c)	-999
7	(b, d)	-973
8	(f, g)	-854
9	(f, i)	-842

Table 2 Execution of Prim's Algorithm

The above tree generated shows communication links with the highest amount of traffic and therefore have to be given highest priority in bandwidth distribution. But in above example of network, congestion will still occur on link connecting node d and e. Therefore edge (e, d) has to be included in the above graph with red colored edge. To do so, the algorithm to find the second minimum spanning tree can be used.

Here is the algorithm to find the second minimum spanning tree.^[7]

Set $\Delta|T| = \infty$

Set $E_{new} = 0$ and $E_{old} = 0$

For every edge e not in tree, do:

Add edge e to the tree, thus creating a cycle

Find the maximum weight edge k in the cycle, $k \neq e$

Remove k from the tree

Calculate the difference in tree weight δ

If $\delta < \Delta|T|$ then

Set $\Delta|T| = \delta$

Set $E_{new} = e$

Set $E_{old} = k$

Replace edge k with edge e in the graph

This algorithm works by trying all edge replacement and find the edge replacement with the least cost

difference with the first minimum spanning tree. First and second minimum spanning tree will differ by exactly one edge and that new edge will indicate links with network congestion, in this case it should be edge (d, e). This process of searching second minimum spanning tree continues until no link with congestion exists in the graph.

By using above algorithm, here is the table showing execution process on finding second minimum spanning tree of graph in figure 3.3.

Steps	e	k	δ	$\Delta T $	E_{new}	E_{old}
1	(a, b)	(b, d)	872	872	(a, b)	(b, d)
2	(b, c)	(b, d)	836	836	(b, c)	(b, d)
3	(d, e)	(c, d)	100	100	(d, e)	(c, d)
4	(g, j)	(f, g)	755	100	(d, e)	(c, d)
5	(g, i)	(f, i)	692	100	(d, e)	(c, d)
6	(i, j)	(f, i)	153	100	(d, e)	(c, d)

Table 3 Steps in finding second minimum spanning tree

As the iteration stops, edge (c, d) will be removed from the tree replacing it with edge (d, e) thus creating the second minimum spanning tree with total cost differ by 100 with first minimum spanning tree. By overlapping the first and second minimum spanning tree in the graph, we will get edge (d, e) included in partition of links with highest traffic. This process continues until all edges with traffic exceeding traffic capacity are included in this partition, in this case finding the third and fourth minimum spanning tree is unnecessary.

Here is the network graph with first and second minimum spanning tree overlapped.

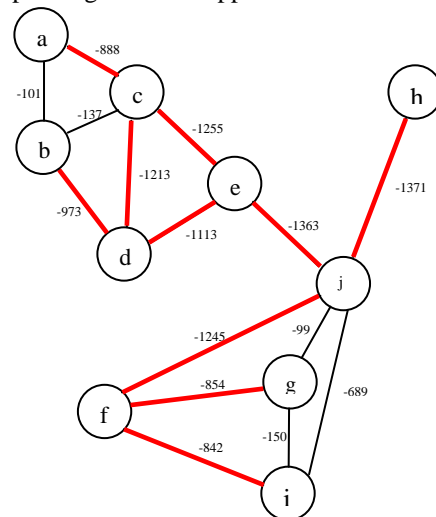


Figure 3.4 First and Second MST Overlapped

From the above diagram, we can see that communication links (a, c), (b, c), (g, j), and (g, i) differ by significant amount of traffic with link (j, i), therefore it is necessary to find first minimum spanning tree from the graph with unnegated weight to highlight edges (a, b), (b, c), (g, i), and (g, j). By using Prim's algorithm to find minimum spanning tree from graph in figure 3.1, we will get minimum spanning tree with edges (j, h), (j, e), (d, e),

(b, d), (b, c), (a, b), (g, j), (g, i), and (g, f). We now have two sets of edges: edges highlighted in red in figure 3.4 (E_1) and edges of minimum spanning tree we just generated (E_2).

$$E_1 = \{(a, c), (b, d), (c, d), (c, e), (d, e), (e, j), (h, j), (f, j), (f, g), (f, i)\}$$

$$E_2 = \{(j, h), (j, e), (d, e), (b, d), (b, c), (a, b), (g, j), (g, i), (g, f)\}$$

By using set operation $E_2 - E_1$, we will get edges that are in E_2 but not in E_1 .

$$E_2 - E_1 = \{(b, c), (a, b), (g, j), (g, i)\}$$

We now have set $E_2 - E_1$ which is a set of edges with lowest traffic. Remaining edge (i, j) can be obtained by subtracting set of all edges in the graph with set of highlighted edges and set $E_2 - E_1$.

Here is the network graph with edges in set $E_2 - E_1$ highlighted in green and edge (i, j) highlighted in yellow.

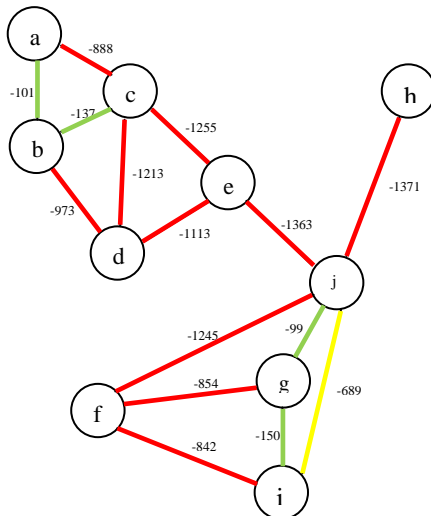


Figure 3.5 Completed Priority-Based Bandwidth distribution

The graph above shows the network communication links highlighted in different color based on traffic. Links with highest traffic are highlighted in red, links with the lowest traffic highlighted in green, and the rest highlighted in yellow. These colors indicated a priority in bandwidth distribution; links in red have the highest priority in bandwidth distribution, links in yellow have the second highest priority, and links in green have the least priority.

V. CONCLUSION

Prim's algorithm as a method to find minimum spanning tree has many implication in real life such as one discussed in this paper in distributing bandwidth speed based on priority. This similar approach can as well be

implemented in situation requiring priority such as in emergency situation, edge can be given a very high weight value.

VII. ACKNOWLEDGMENT

First of all, I would like to thank God for His guidance in writing this paper. I wish to express my sincere thanks to Dr. Ir. Rinaldi Munir and Mrs. Harlili for teaching us. I would also like to thank my parents for their support and courage.

REFERENCES

- [1] Rinaldi Munir, *Diktat Kuliah Matematika Diskrit*, Bandung: Program Studi Teknik Informatika ITB, 2006
- [2] K. H. Rosen, *Discrete Mathematics and Its Applications 7th*, New York: McGraw-Hill, 2012
- [3] James F. Kurose and Keith W. Ross, *Computer Networking: A Top-Down Approach 6th edition*, New York: Pearson, 2013
- [4] www.cs.helsinki.fi/u/ejunttil/opetus/tiraharjoitus/prim.ppt retrieved on 13 December 2013
- [5] <http://www2.hawaii.edu/~suthers/courses/ics311f13/Notes/Topic-17.html> retrieved on 13 December 2013
- [6] <http://www.cs.auckland.ac.nz/software/AlgAnim/prim.html> retrieved on 14 December 2013
- [7] <http://web.mit.edu/6.263/www/quiz1-f05-sol.pdf> retrieved on 15 December 2013

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 16 Desember 2013

Mario Filino
13512055