Finding a shortest path from a single source, to each an every vertices of the graph :- Dijkstra's Algorithm (Weighted Graph)

Other Algorithms:-

- Bellman Ford's Algorithm. (Negative Edges)
- Warshall's Algorithm (shortest distance between every pair of vertices in graph)

Not Included in CAT-II

Minimum Spanning Tree Problem. (undirected Weighted Graph)

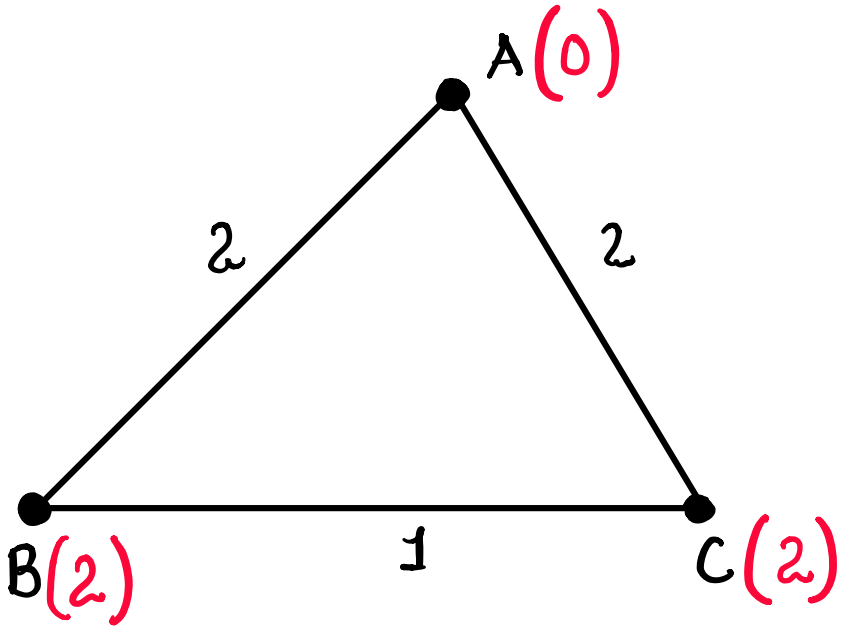
Two important problems.

1. Select a set of (n-1) edges in a graph such that there is a path from every pair of vertices.

2. Sum of weights of edges is to be minimized.

How it is different from Single Source Shortest Path Problem.

Consider the following Graph



Let's Apply Dijkstra's Algorithm with source vertex A.

Initialization :-

0	0	0	visited
A	B	C	
0	∞	∞	distance

(Iteration 1)

1	0	0	visited
A	B	C	
0	2	2	distance

(Iteration 2)

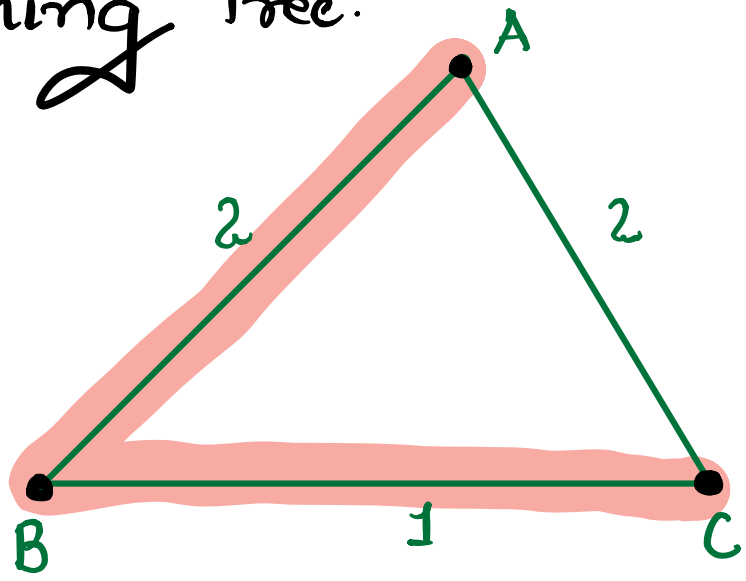
1	1	0	visited
A	B	C	
0	2	2	distance

(Iteration 3)

1	1	1	visited
A	B	C	
0	2	2	distance

Minimum Spanning Tree.

we select edges (A,B) and (B,C) because



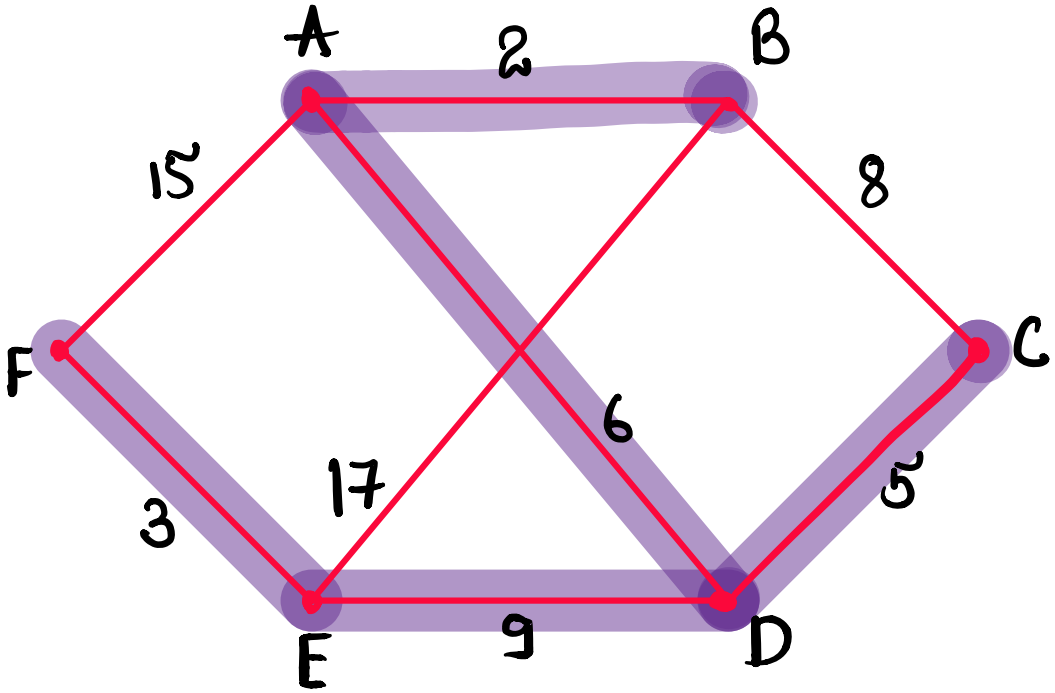
$W(A,B) + W(B,C) = 3$

$W(A,C) + W(B,C) = 3$

$W(A,B) + W(A,C) = 4$

→ You Can Choose either of these.

Prim's Algorithm



Edges Set = $\{(A,B)\}$ (step 1)

$\{(A,B),(A,D)\}$ (step 2)

$\{(A,B),(A,D),(D,C)\}$ (Step 3)

$\{(A,B),(A,D),(D,C),(D,E)\}$ (step 4)

$\{(A,B),(A,D),(D,C),(D,E),(E,F)\}$ (step 5)

PrimsAlgorithm(G)

$W(v_i, v_j)$ represents weight on the edge (v_i, v_j).

Choose the edge $(v_i, v_j) \in E$ such that $W(v_i, v_j)$ is minimum.

$STE = \{(v_i, v_j)\}$ // Edges in Tree

$STV = \{v_i, v_j\}$ // Vertices in the tree

Repeat $(n-2)$ times.

1. Choose the edge (v_k, v_l) such that $W(v_k, v_l)$ is minimum and $v_k \in STV$ and $v_l \notin STV$

2. $STV \leftarrow STV \cup \{v_l\}$
   $STE \leftarrow STE \cup \{(v_k, v_l)\}$

end Algorithm

Prim's Algorithm. (with structures)

Prim's

```
for i = 1 to n
        visited[i] = 0;
        dist[i] = ∞;
        neighbour[i] = -1
endfor.
```

STE ← []

visited[1] ← 1

You can choose any other vertex Also.

for each edge (1, j)

neighbour[j] = 1

dist[j] = Weight(1, j)

Repeat (n-1) times.

Choose j such that visited[j] != 1

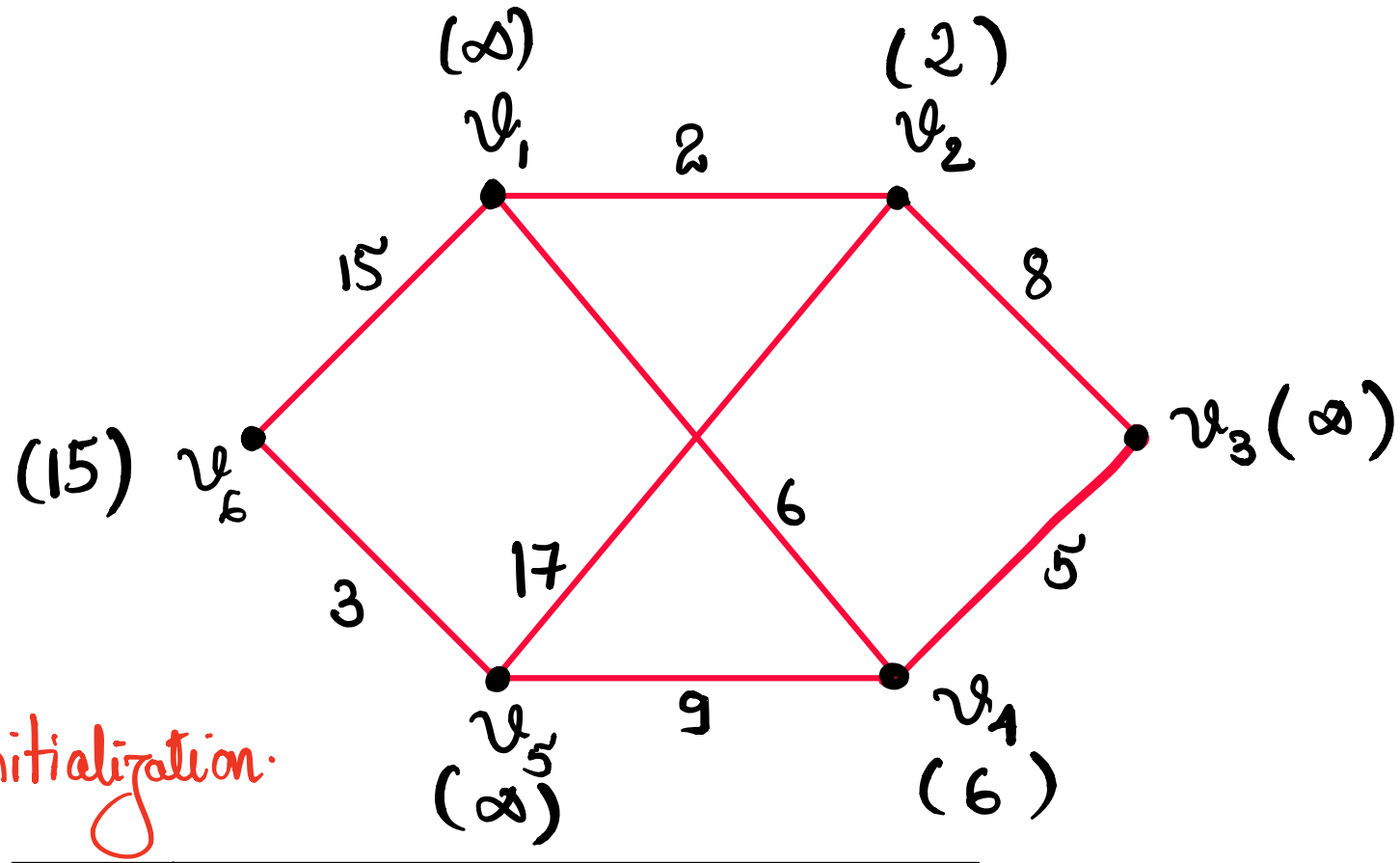
and dist[j] is minimum

visited[j] = 1

STE ← STE ∪ {(j, neighbour[j])}

```
for each edge(j, k) with
                    visited[k] != 1
    if dist[k] > W[j][k]
        dist[k] = W[j][k]
        neighbour[k] = j
    endif
endfor each
```

Let's apply the above algorithm in the following undirected weighted graph.



Initialization:

1	0	0	0	0	0	visited
∞	2	∞	6	∞	15	dist
-1	1	-1	1	-1	1	neighbour

STE = []

Iteration 01

1	1	0	0	0	0

visited

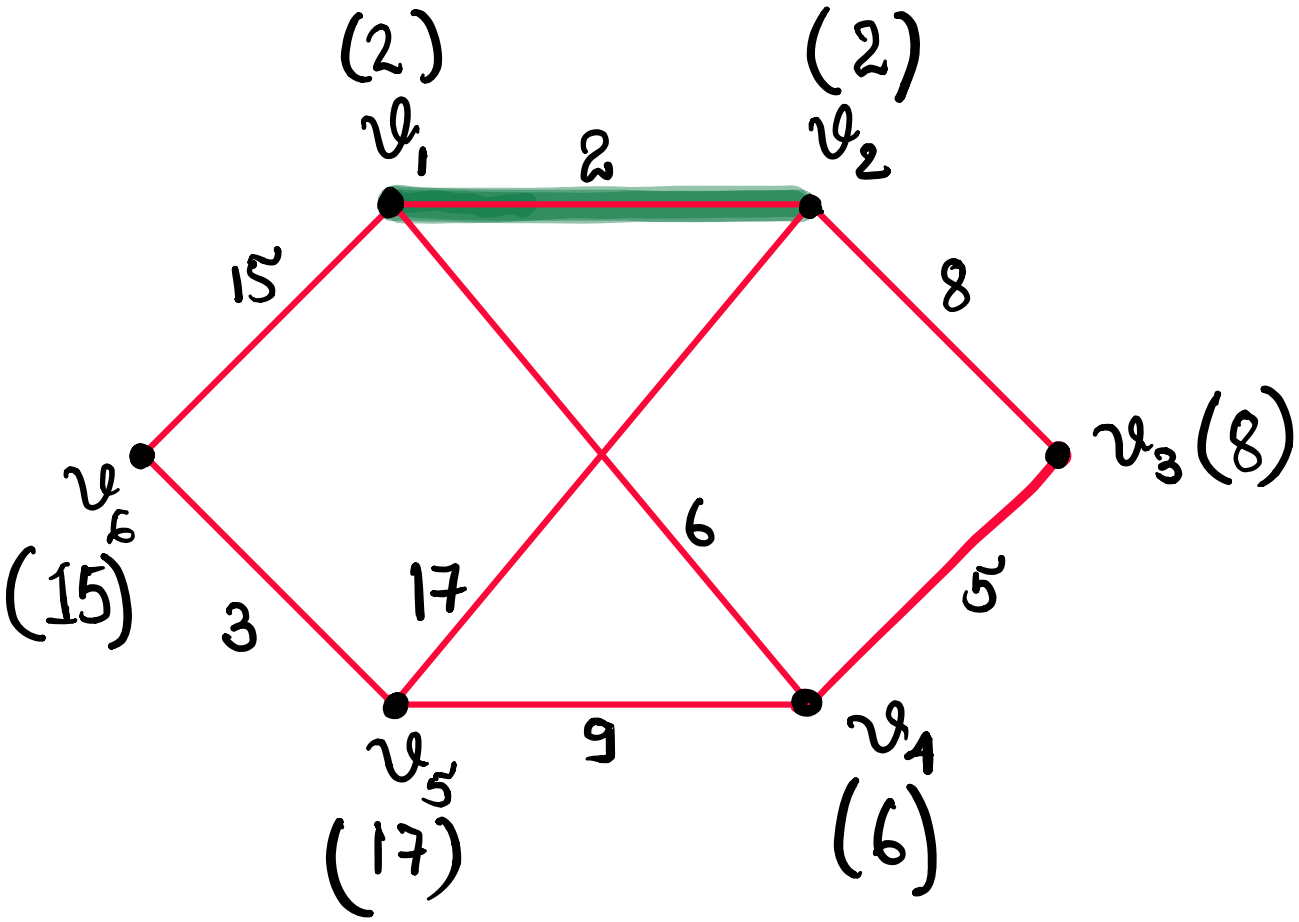
2	2	8	6	17	15

dist

2	1	2	1	2	1

neighbour

$STE = [(2, 1)]$



Iteration 02

1	1	0	1	0	0

visited

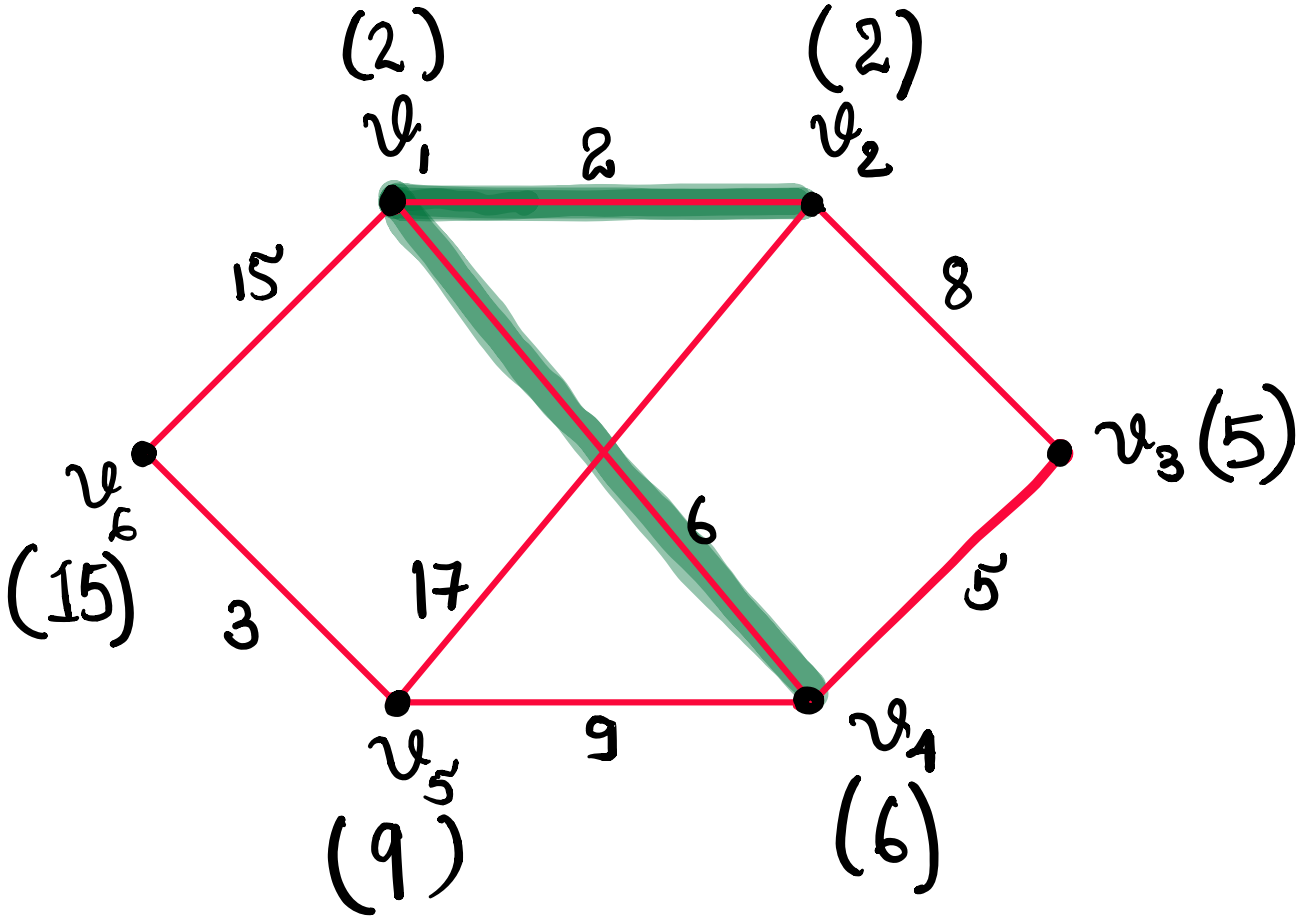
2	2	5	6	9	15

dist

2	1	4	1	4	1

neighbour

$$STE = [(2,1),(4,1)]$$



Iteration 03

1	1	1	1	0	0

visited

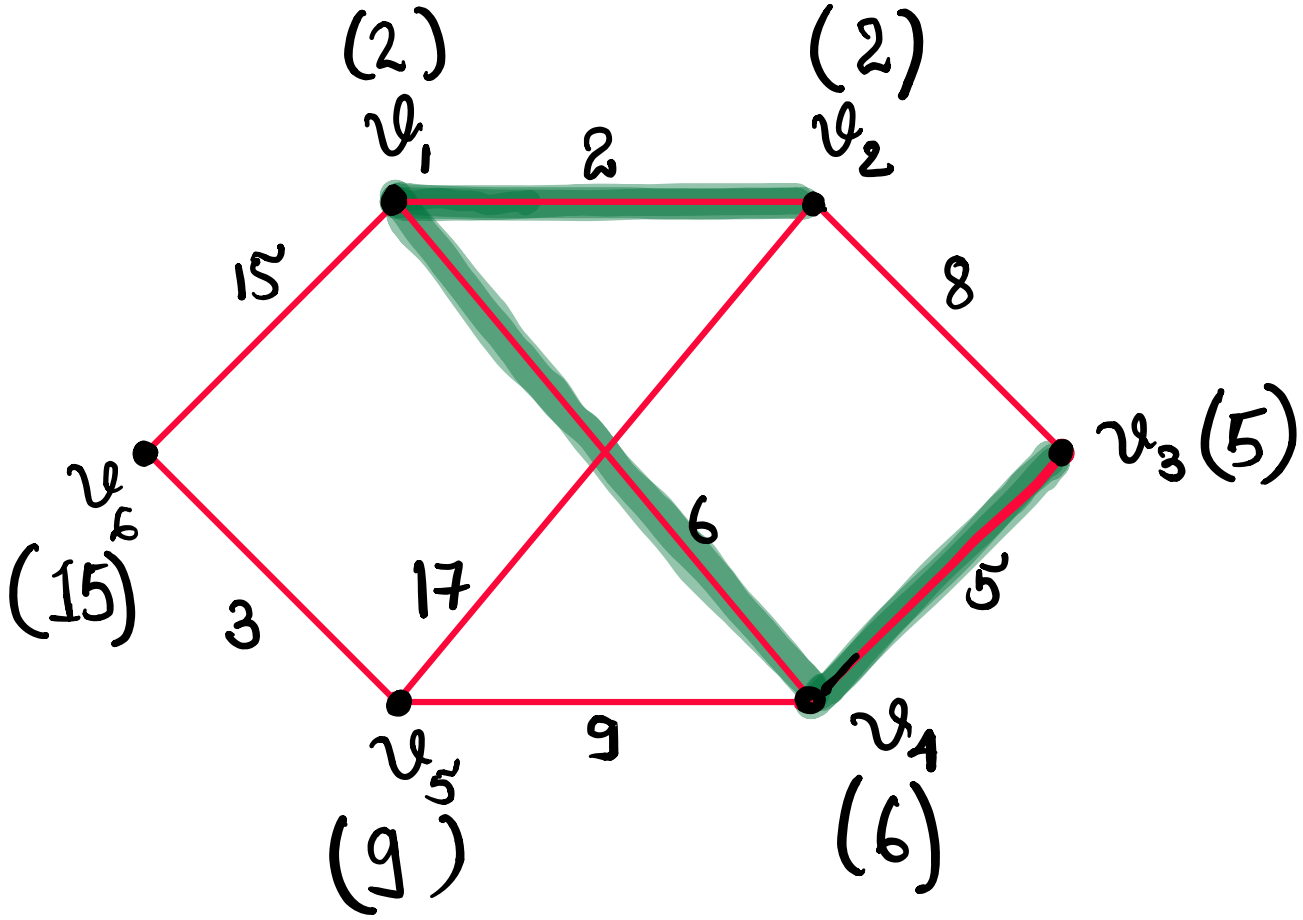
2	2	5	6	9	15

dist

2	1	4	1	4	1

neighbour

$$STE = [(2,1),(4,1),(3,4)]$$



Iteration 04

1	1	1	1	1	0

visited

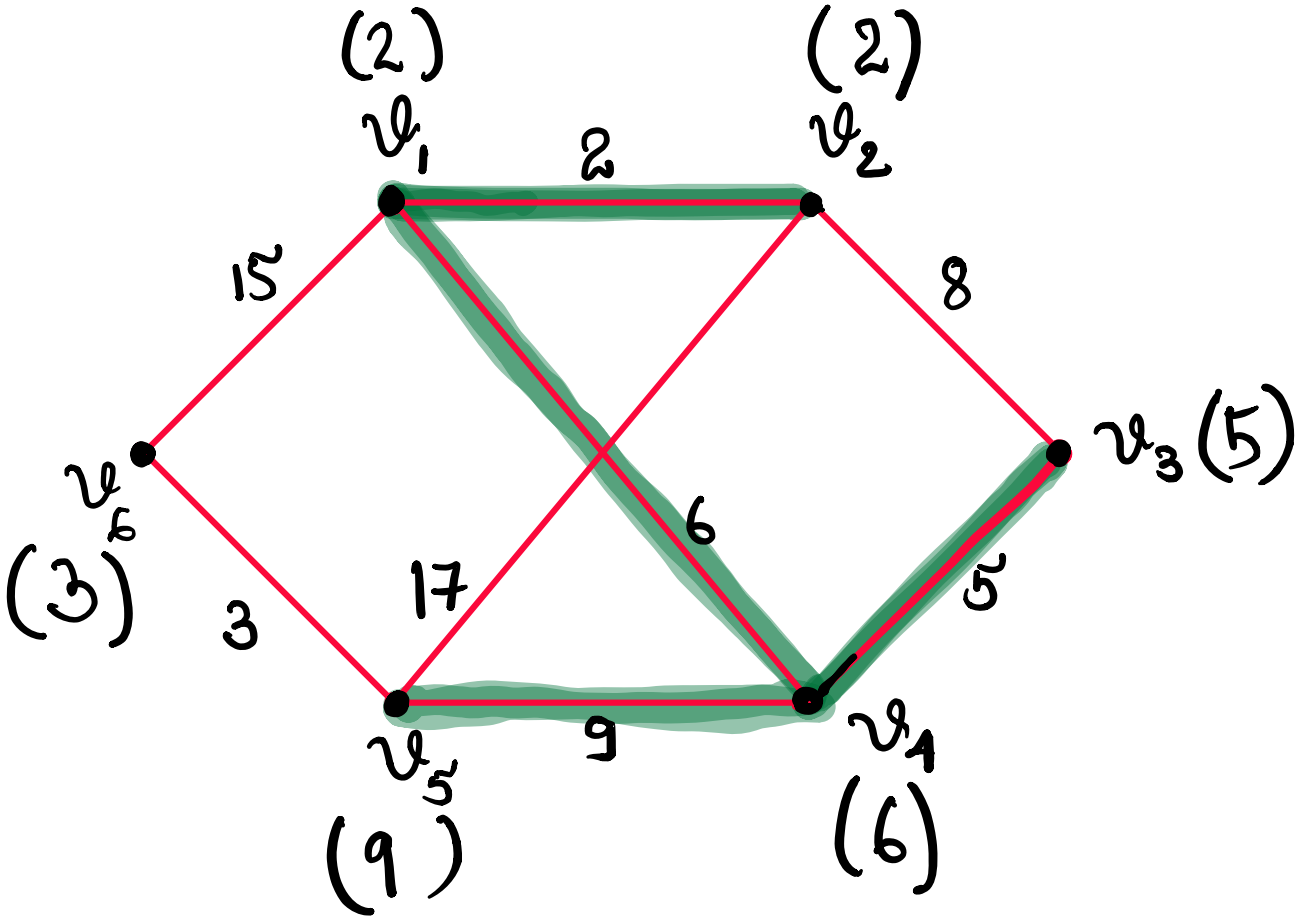
2	2	5	6	9	3

dist

2	1	4	1	4	5

neighbour

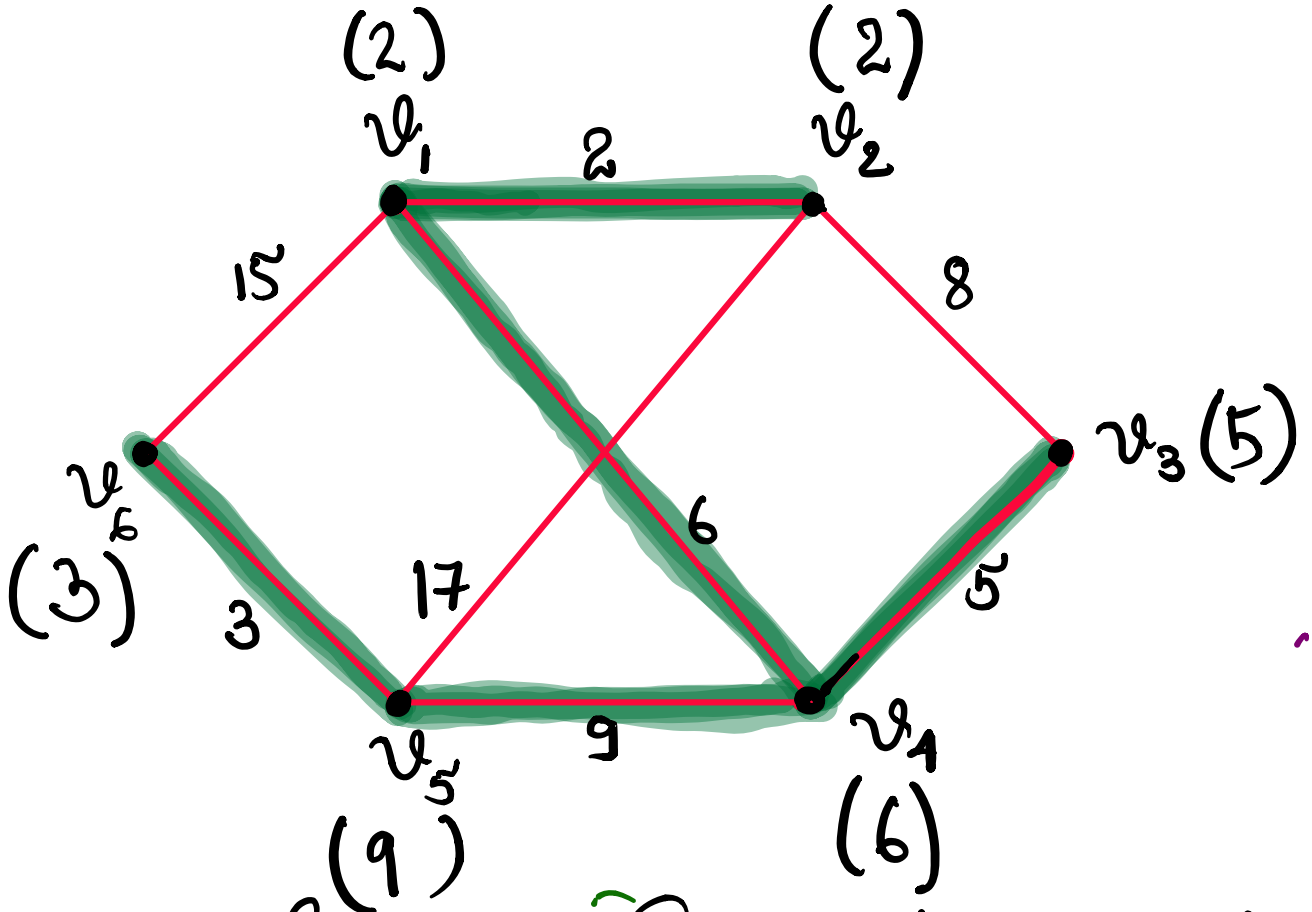
$$STE = [(2,1),(6,1),(3,4),(5,4)]$$



Iteration 05

1	1	1	1	1	1	visited
2	2	5	6	9	3	dist
2	1	4	1	4	5	neighbour

$$STE = [(2,1),(4,1),(3,4),(5,4),(6,5)]$$



Minimum Spanning Tree is highlighted in Green. Edges in MST is given by STE

Kruskal's Algorithm

Algorithm :-

1. Sort the edges of the graph according to their weights

   Let $E_s = [e_1, e_2, e_3, \cdots, e_m]$

   be the sequence of sorted edges of the given graph

   $$\text{So, } W(e_i) \leq W(e_j) \ (\forall\, i < j)$$

where,

$W(e_i)$: weight anociated with edge e_i

2. STE = [ ] // Spanning tree edges.
3. $i \leftarrow 0$
4. Repeat till $|STE| == n-1$ → {no. of edges in spanning tree becomes exactly (n-1)

   if (STE $\cup$ {$E_s[i]$} does not form a cycle among the edges already in STE)

$STE \leftarrow STE \cup \{E_s[i]\}$

endif

$i = i + 1$;

Notice unlike Prim's algorithm, STE is not a tree at any ith iteration of Kruskal's algorithm

Idea of Correctness Proof of Kruskal's Algorithm.

1. During initialization of STE is empty. Hence, the graph can be perceived as the 'n' vertices w/o edges. (n distinct components)

2. Including an edge (first edge) in STE, decreases the no. of distinct component by 1.

3. Introducing an edge between vertices of same connected components forms a cycle and hence it must be avoided.

4. Each time an edge (i,j) is introduced between

v_i and v_j, $v_i \in$ component$[v_i]$ and $v_j \in$ component$[v_j]$.

where component$[v_i] \subseteq V$ which contains all vertices connected to v_i.

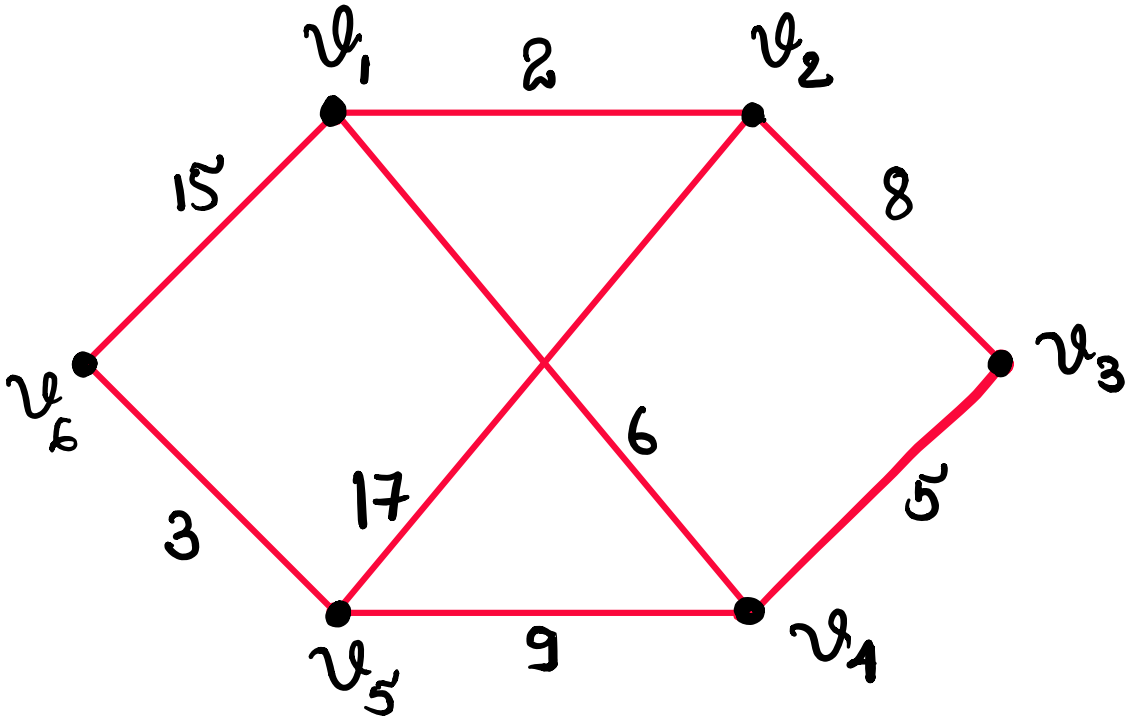
5. Cut property of MST can be applied to its correctness.

Kruskal's Algorithm

$E_S = [e_1, e_2, e_3, \ldots, e_m]$ // Sorted edges according to weights.

```
for (i = 1 to n)
        component[i] = i
end for

STE = []
i ← 1.
```

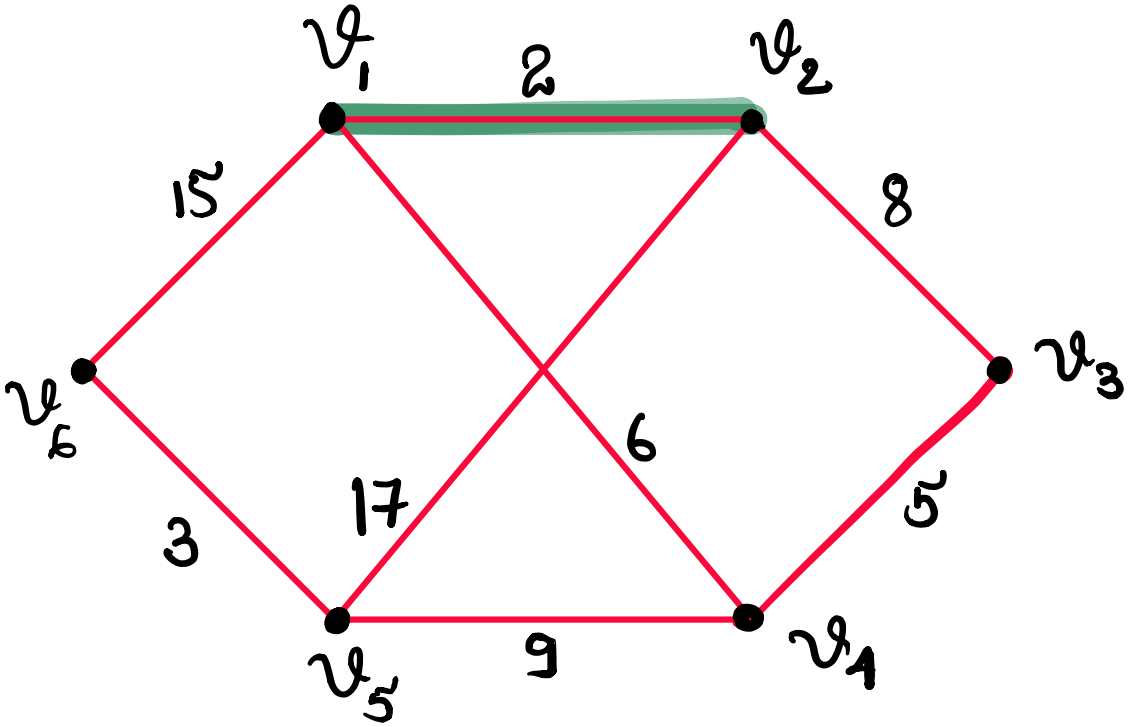
```
while |STE| < n-1

    Let E[i] = (l, m)

    if component[l] != component[m]

        STE ← STE U {E[i]}
        i ← i+1;
        for (j = 1 to n)
            if component[j] == component[m]
                component[j] = component[l]
            endif
        endfor

    endif

end while.
```



component STE

1	2	3	4	5	6

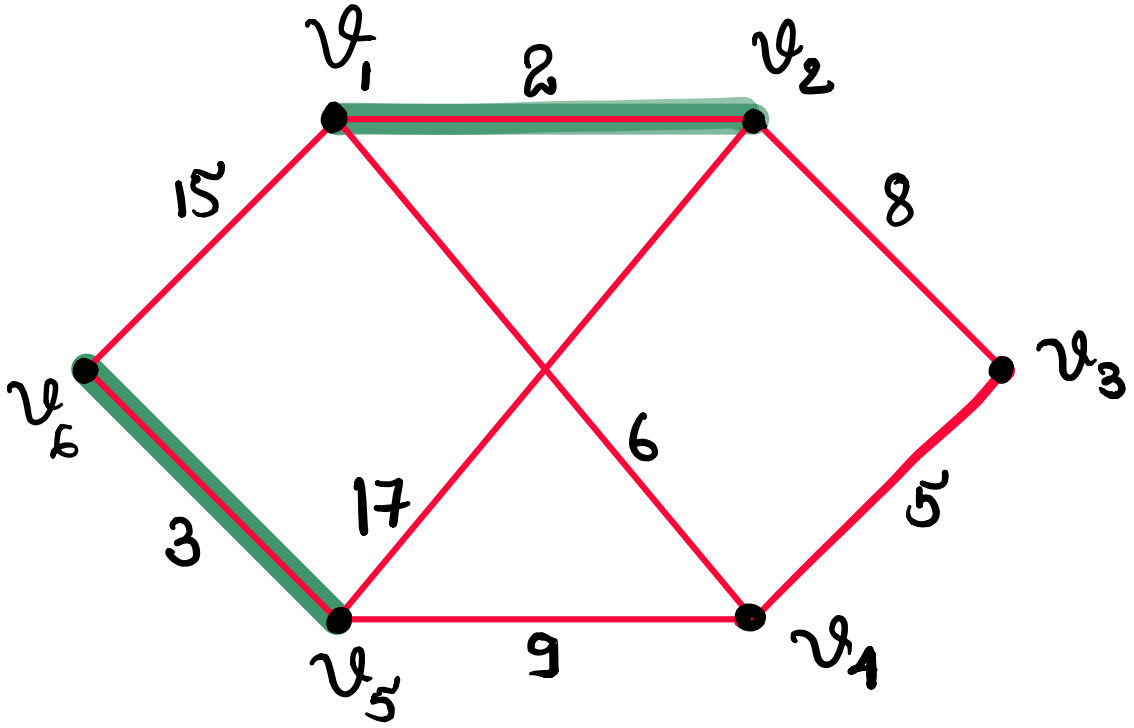
{ }



$l=1$, $m=2$

Component	1	1	3	4	5	6

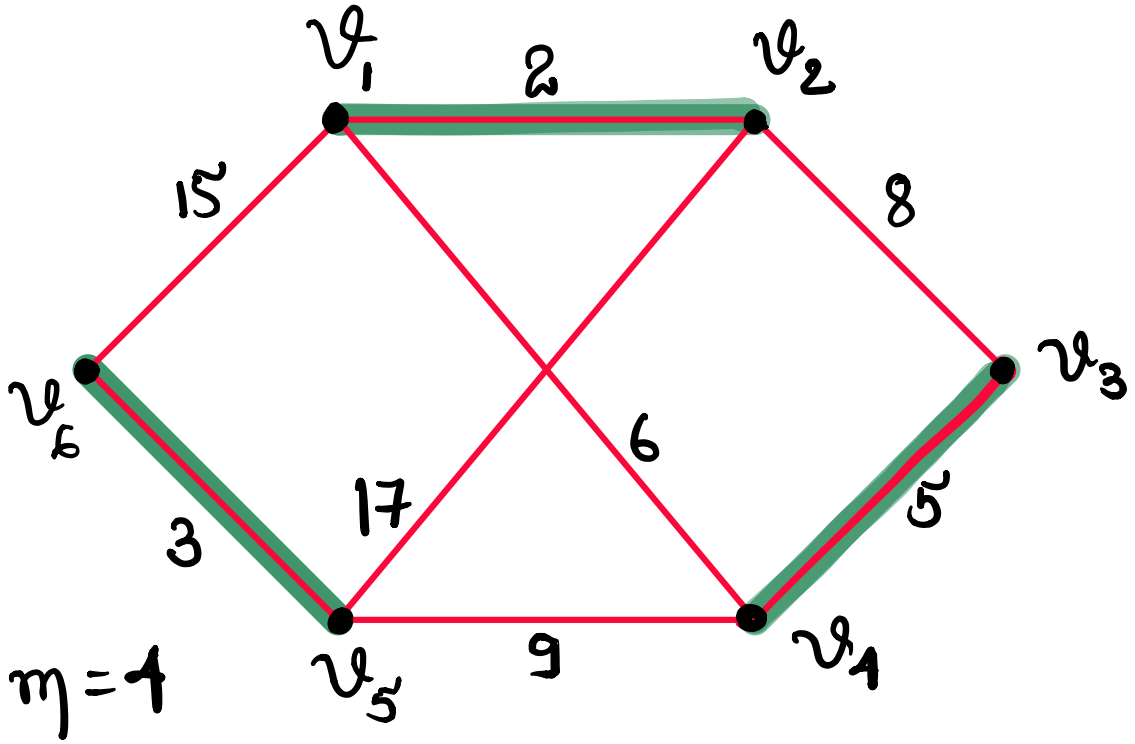
STE $\{(1,2)\}$



$l=5$, $l=6$

Component	1	1	3	4	5	5

STE $\{(1,2),(5,6)\}$



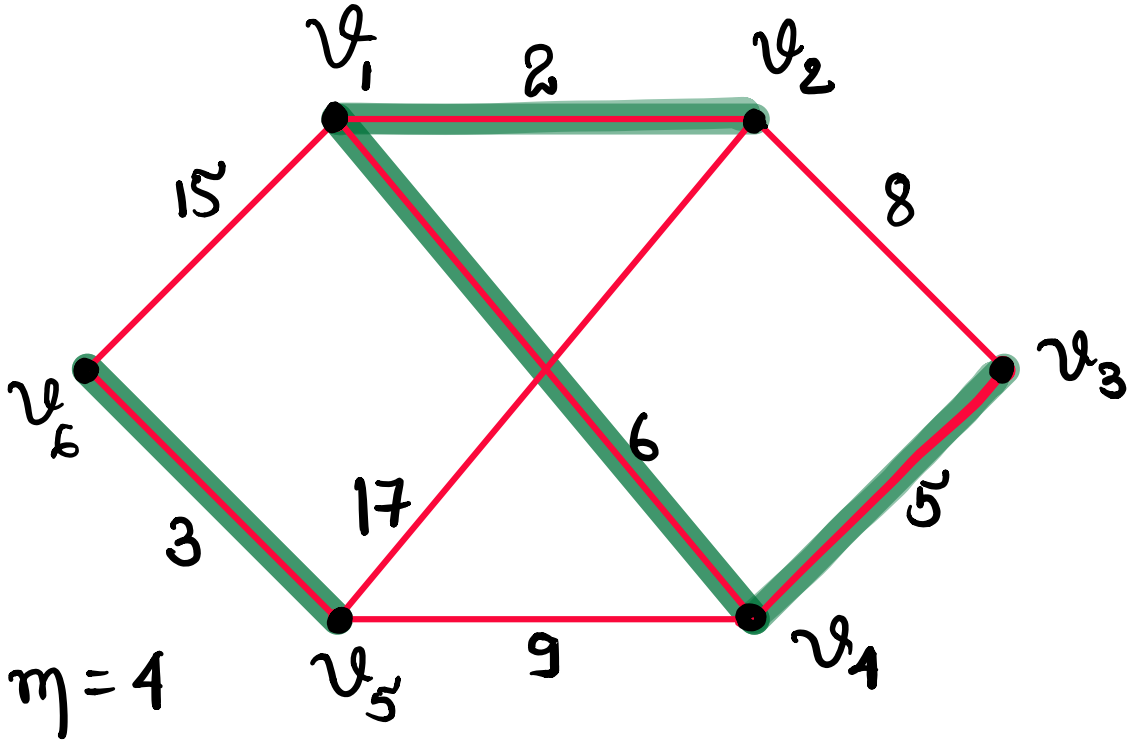
$l = 3, m = 4$

Component

1	1	3	3	5	5

STE

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



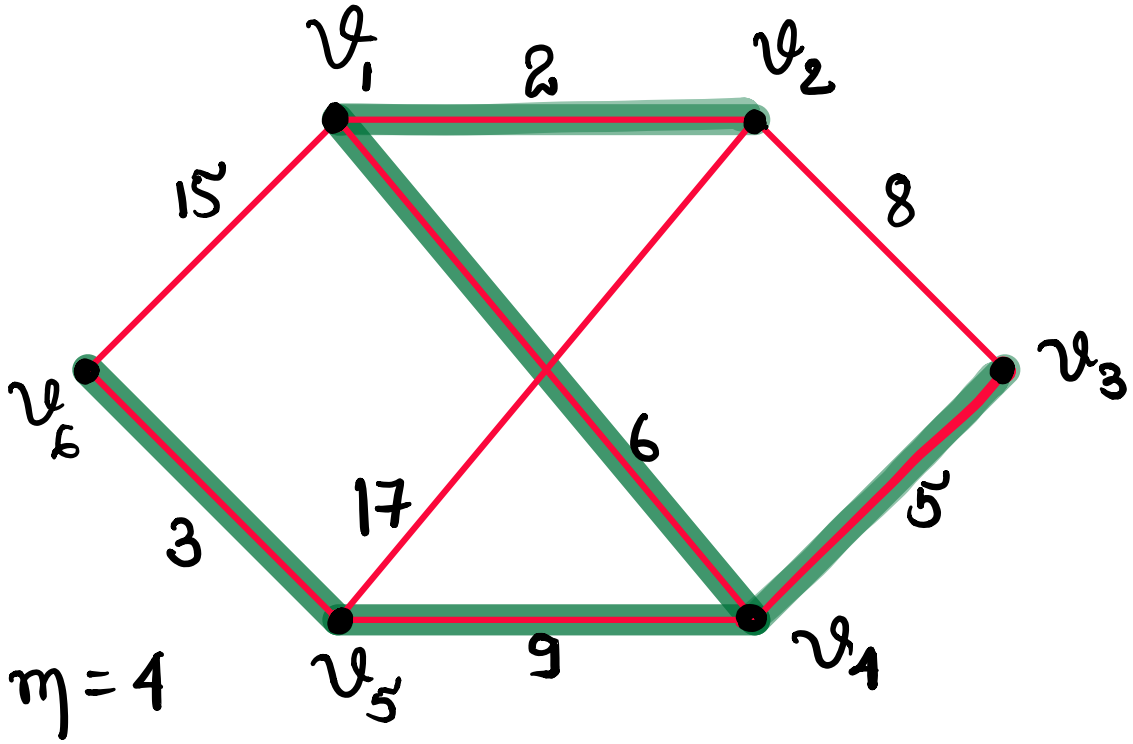
$l = 1, m = 4$

Component

1	1	1	1	5	5

STE

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



$l = 1, m = 4$

Component

1	1	1	1	1	1

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

Complexity

Discuss the Complexity of Prim's and Kruskal's Algorithm. (Homework)