Kruskal's algorithm

- It is used to determine the minimum spanning tree..
- The objective is similar to Prim's but the approach is different.
- The difference is it ~~needs not to~~ chooses the lowest cost edge and while adding it to 'T', it checks for a cycle.

Algorithm :- $G = (V, E)$.

$E_T \leftarrow \phi$, count $\leftarrow 0$.

$E_{sorted} \leftarrow$ Sort E in the ascending order of cost.

while (count $< |V| - 1$)

select an edge e_k from E_{sorted}

if $T \cup e_k$ is acyclic.

add e_k to T.

count ++

else.

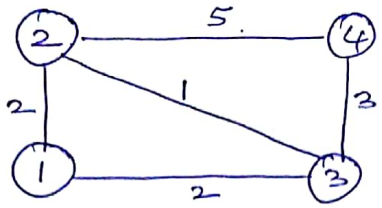
discard e_k.

$k \leftarrow k + 1$

~~count ++~~

end while.

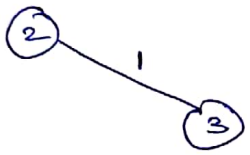
Example :-



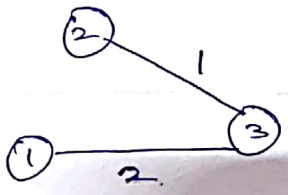
1) E = 1 2 2 3 5

{2,3} {1,3} {1,2} {4,3} {2,4}

count.

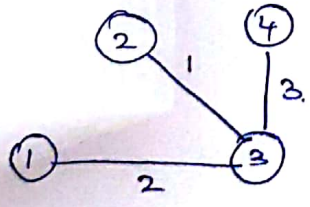
2) 

count = 1.

3) 

count = 2.

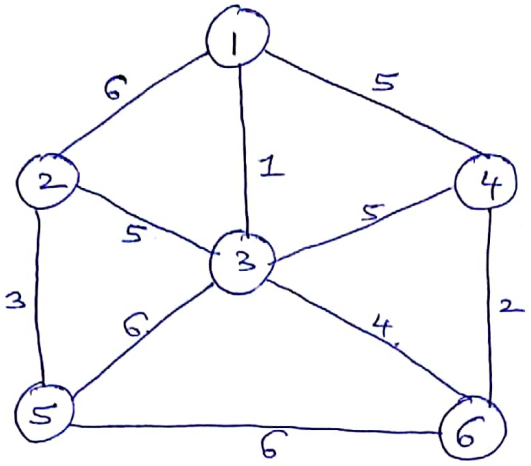
4) {1,2} form a cycle. ignore.

5) 

count = 3 < 4.

6) {2,4} form a cycle, Ignore

Example 2:-

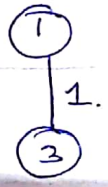


1) Start the edges according to weights.

k →	0	1	2	3	4	5	6	7	8	9
E = {	1 ✓	2 ✓	3 ✓	4 ✓	5	5	5	6	6	6 }
					{2,3}	{3,4}	{1,4}	{1,2}	{3,5}	{5,6}.

2) count = 0, k = 0.

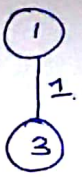
$e_k = 1$.

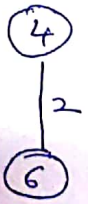


Increment

count = 1. k = 1

3) $e_k = 2$.

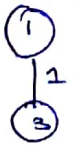


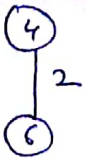


count = 2

k = 2

4) $e_k = 3$

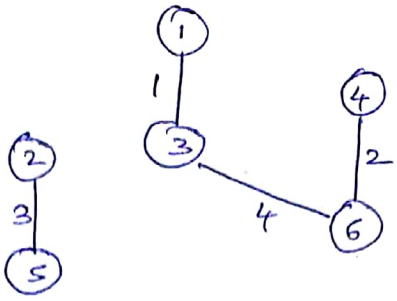




count = 3

k = 3.

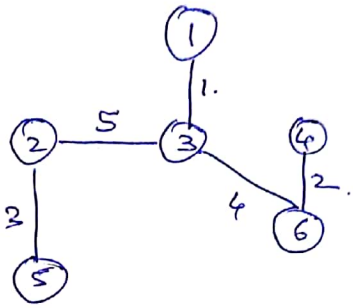
5) $e_k = 4$



Count = 4

k = ~~4~~

6) $e_k = 5$



→ Solution.

Count = 5 , k = 5

7) $e_k = 5$ {3, 4} forms a cycle Ignore

~~Count = 6~~ → k = 6.

8) $e_k = 5$ {1, 4} forms a cycle Ignore.

k = 7.

9) $e_k = 6$ {1, 2} forms a cycle Ignore.

k = 8.

10) $e_k = 6$ {3, 5} forms a cycle Ignore

k = 9

11) $e_k = 6$ {5, 6} forms a cycle Ignore.

k = 10 //