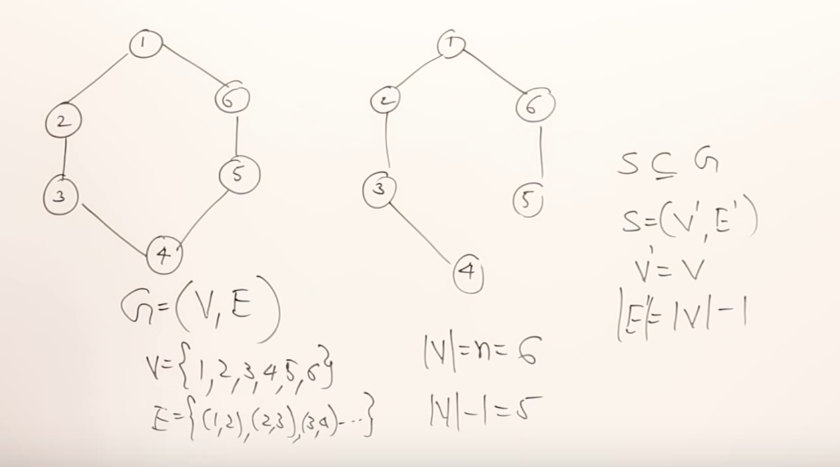
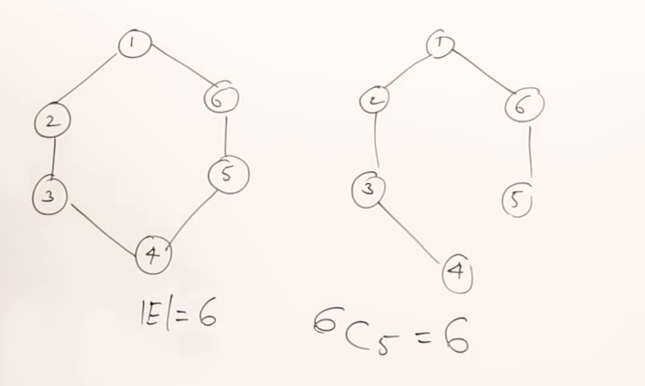
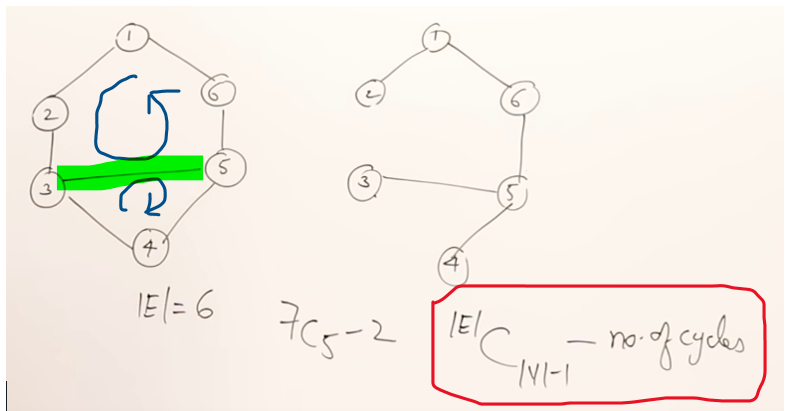
Spanning Tree  
Spanning tree is a sub-graph of a graph having all vertices but only (n-1) edges.

Graph 🡪 cycle  
Tree 🡪 Not a cycle

  
 Graph Spanning tree

For a given graph how many different possible spanning trees can be generated ??

  
from 6 edges of a graph, minimum spanning tree will have only 5 edges. [ # of edges(spanning tree) = (# of Vertices(graph) – 1) ]  
These 5 edges can be formed in 6 ways (6 C5 = 6)

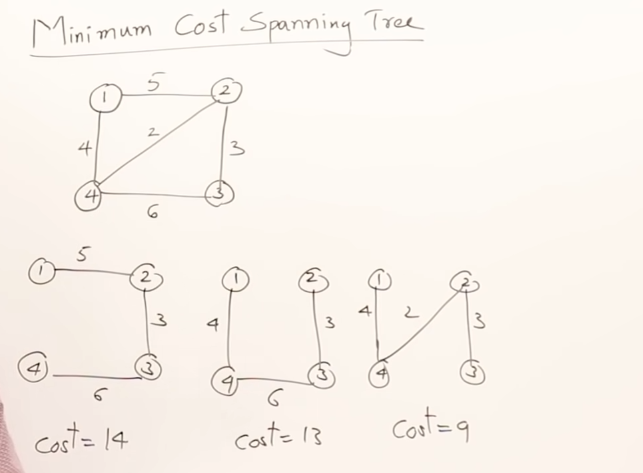


from 7 edges of a graph, minimum spanning tree will have only 5 edges.[ Number of edges in a spanning tree = (Vertices(graph) – 1) ].

Actually these 5 edges can be formed in 21 [7 C5] ways. But the newly added edge forms 2 cycles in the graph.  
# of edges(spanning tree) = [ # of Vertices(graph) – 1 ]

Number of ways the spanning tree can be formed with a cycle in a graph =  
[ # of Edges(graph) C (# of vertices(graph)-1) ] - # of cycles(graph) ]

Minimum Cost Spanning Tree



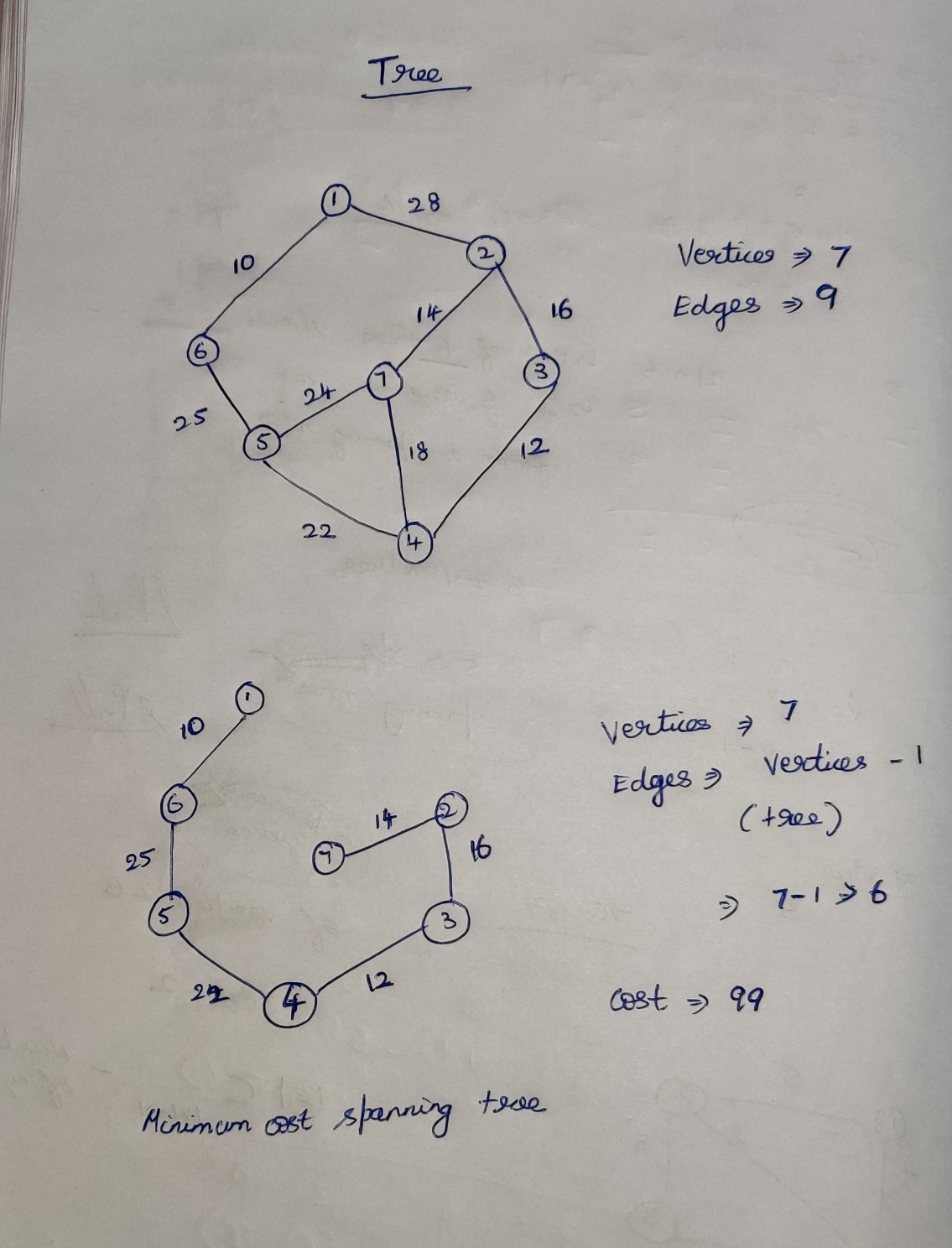
We can try to put manually all possible spanning trees , and can find the cost by adding all the weighs, from which we can find the minimum cost spanning tree.

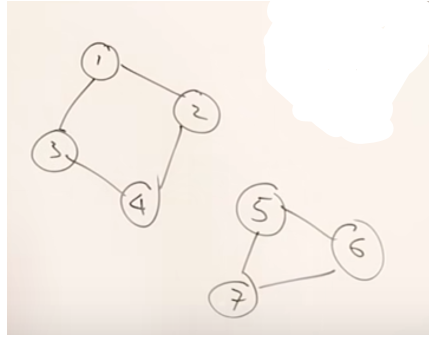
2 methods to find the minimum cost spanning tree  
1) Prim’s algorithm  
2) Kruskal’s algorithm

**Prim’s algorithm**

1. Select the minimum cost edge from the entire graph. [ 1 (10) 6 ]
2. Always select a minimum cost edge, but make sure that it should be connected to already selected vertices.

Next minimum is [4 (12) 3], but that is not connected to 1 and 6, So don’t choose that. To maintain a tree always the minimum edge which is connected.  
Take only those which are connected to 1 and 6 (i.e) [1 (28) 2] or   
[6 (25) 5].



For non-connected graph we cannot find the spanning trees at-all. 

Kruskal’s algorithm

1. Always select the minimum edge.
2. If it forms a cycle, discard it and move to next edge.