1.10. Trees and Spanning Trees.

1.10.1. Definition: Tree.

A *tree* is a connected graph which has no non-trivial circuits.

Notes:

* In a graph, a path consisting of a single vertex is a trivial circuit.

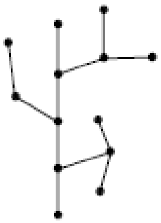
* A graph is a tree if and only if it is circuit free and connected.

* A tree can contain a trivial circuit, thus a single vertex is a tree.

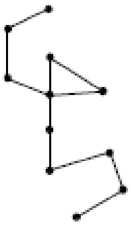
Exercise:

Which of these graphs are trees?

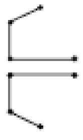
G_1:



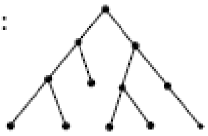
G_2:



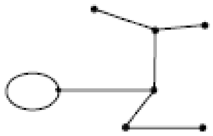
G_3:



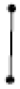
G_4:



G_5:



G_6:



Trees can be used in conjunction with connected graphs.

1.10.2. Definition: Spanning Tree.

A *spanning tree* for a graph *G* is a sub-graph of *G* which is a tree that includes every vertex of *G*.

Notes.

∗ A spanning tree of a graph *G* is a “maximal” tree contained in the graph *G*.

∗ When you have a spanning tree *T* for a graph *G*, you cannot add another edge of G to *T* without producing a circuit.

Exercises:

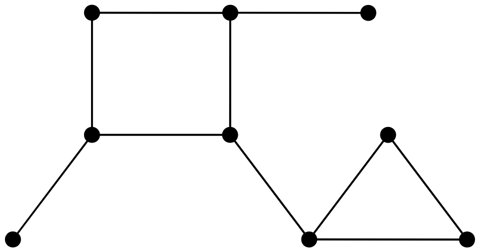
◆ Can you have a spanning tree for a graph that is not connected?

◆ Does every connected graph have a spanning tree?

◆ Are spanning trees unique in each graph?

- If not, do two spanning trees for the same graph have anything "in common"?

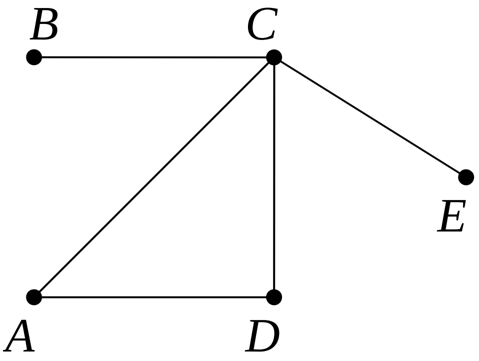
- Find all spanning trees for the following graph. (There are 12)



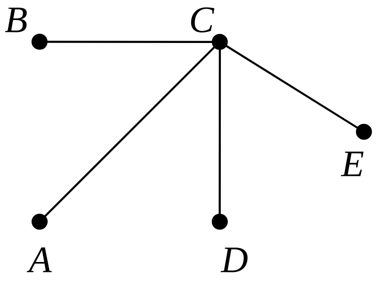
Example:

Consider the following graph, G, representing pairs of people (A, B, C, D and E) who are acquainted with each other.

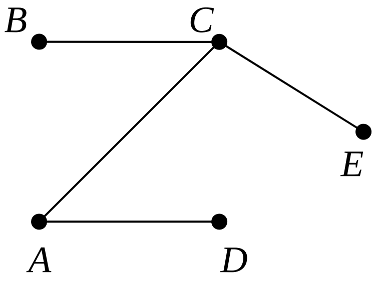
G:



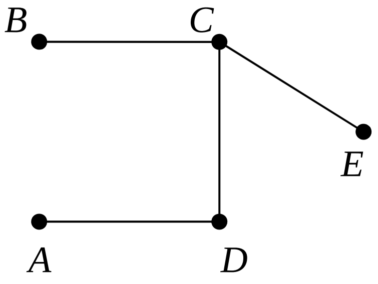
We wish to install the minimum number of phone lines so that communication between these people is maintained. As an adviser, you need to find a spanning tree T for G.



T_1



T_2



T_3

How many edges does each tree have? 4

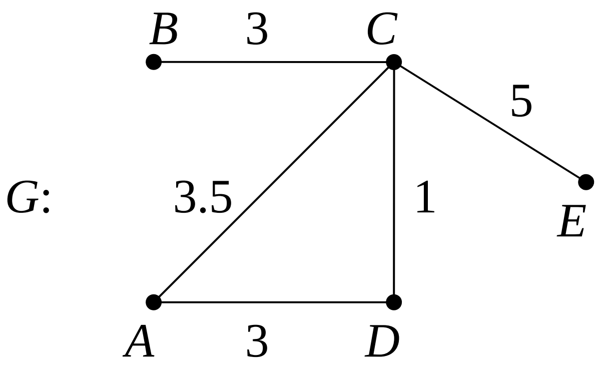
Note:

Any two spanning trees will have the same number of edges.

To make the problem more practical:

Suppose the cost involved in installing each line is not the same. How do you find the network having the least cost?

Begin by putting the cost (in $100) along each edge.
This results in a weighted graph.



Calculating the cost for each tree:

$T_1 = 3 + 5 + 3.5 + 1 = 12.5$

$T_2 = 3 + 5 + 3.5 + 3 = 14.5$

$T_3 = 3 + 1 + 3 + 5 = 12$

The tree with the lowest “cost" is called the minimum spanning tree.
In this case, the minimum spanning tree is T_3.

When the graph has a large number of vertices, it is not easy to find all spanning trees. In fact, we are not usually interested in finding all spanning trees for a graph. We would like an algorithm which finds the minimum spanning tree for a weighted connected graph. Kruskal's Algorithm is one such algorithm.

1.10.3. Definition: Weighted Graph

A *weighted graph* is a graph for which each edge has an associated real number, called the weight of the edge.

The sum of the weights of all of the edges is the total weight of the graph.

A minimum spanning tree for a weighted graph is a spanning tree that has the least possible total weight when compared with all other spanning trees for the graph.

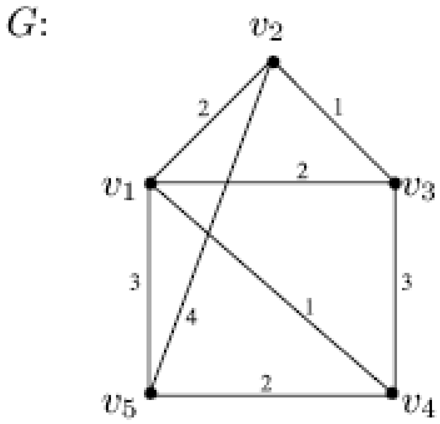
Note. If G is a weighted graph and e is an edge, then we will use $w(e)$ to represent the weight of the edge e and $w(G)$ to represent the total weight of the graph G.

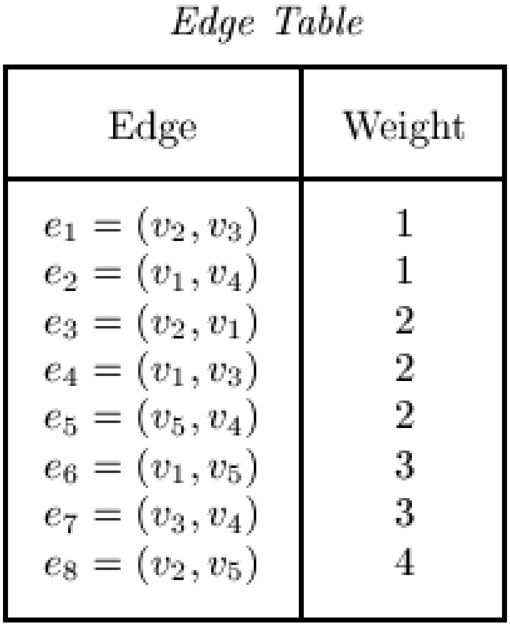
1.10.4. Kruskal's Algorithm

Example.

Find the minimal spanning tree for the following connected weighted graph *G*.

The starting point of Kruskal's Algorithm is to make an “edge” list, in which the edges are listed in order of increasing weights.

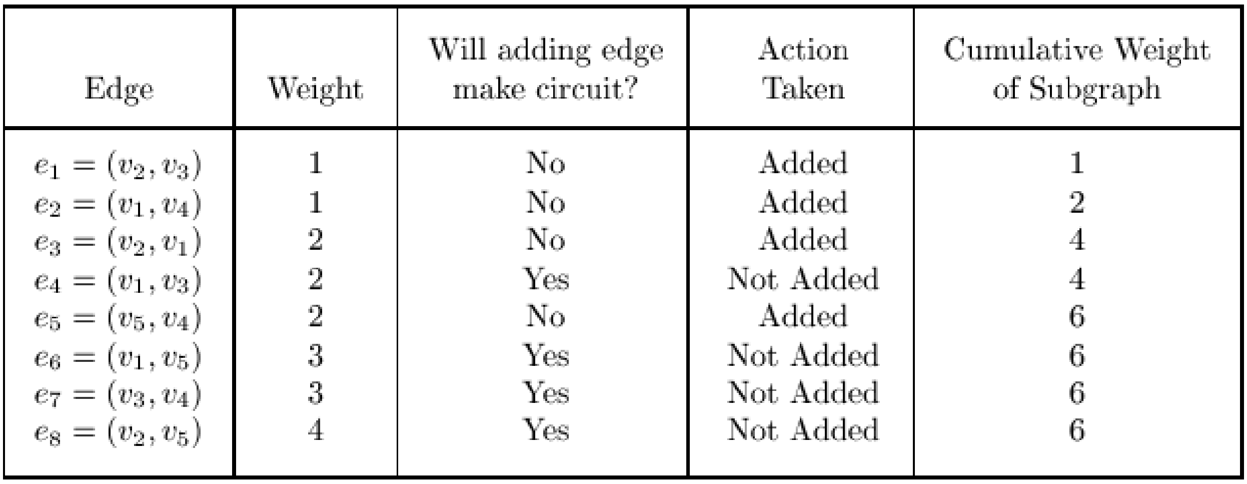




Edge Table

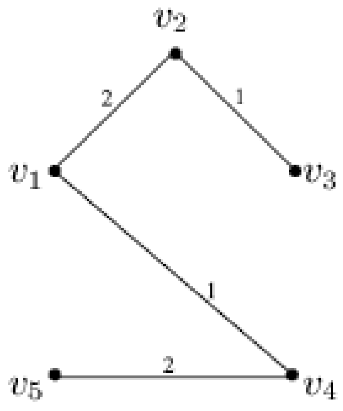
Edge	Weight
$e_1 = (v_2, v_3)$	1
$e_2 = (v_1, v_4)$	1
$e_3 = (v_2, v_1)$	2
$e_4 = (v_1, v_3)$	2
$e_5 = (v_5, v_4)$	2
$e_6 = (v_1, v_5)$	3
$e_7 = (v_3, v_4)$	3
$e_8 = (v_2, v_5)$	4

Kruskal's Algorithm then includes the 3 additional columns as follows:



Edge	Weight	Will adding edge make circuit?	Action Taken	Cumulative Weight of Subgraph
$e_1 = (v_2, v_3)$	1	No	Added	1
$e_2 = (v_1, v_4)$	1	No	Added	2
$e_3 = (v_2, v_1)$	2	No	Added	4
$e_4 = (v_1, v_3)$	2	Yes	Not Added	4
$e_5 = (v_5, v_4)$	2	No	Added	6
$e_6 = (v_1, v_5)$	3	Yes	Not Added	6
$e_7 = (v_3, v_4)$	3	Yes	Not Added	6
$e_8 = (v_2, v_5)$	4	Yes	Not Added	6

The minimum spanning tree is drawn below.



Note that there are 4 edges in the minimum spanning tree. If the weighted connected graph G has n vertices, then the minimum spanning tree will always have $n-1$ edges. Knowing this fact, we need not have checked whether the edges e_6, e_7, e_8 in the table included in the example created a circuit.

Kruskal's algorithm will always produce a minimum spanning tree for any connected weighted graph *G*.

The idea of the Kruskal's algorithm for finding a minimum spanning tree *T* of *G* is:

To examine the edges of *G* one by one in order of increasing weight. At each step, the edge being examined is added to the sub-graph *S* (which will be made up of the edges of *T*), if the addition of this edge to *S* does not create a circuit.

After $n-1$ edges have been added to *S* (where *n* is the number of vertices), then the edges of *S* together with the vertices of the graph *G* produce the required tree *T*.

Kruskal's Algorithm for finding minimum spanning trees for weighted graphs (Epp's version) is then:

```
Input: $G$ a connected weighted graph with $n$ vertices.
Algorithm Body:
(Build a sub-graph $T$ of $G$ to consist of all of the vertices of $G$ with edges added in order of increasing weight. At each stage, let $m$ be the number of edges of $T$.)
    1.  Initialise $T$ to have all of the vertices of $G$ and no edges.
    2.  Let $E$ be the set of all edges of $G$ and let $m = 0$.
        (pre-condition: $G$ is connected.)
    3.  While ($m < n - 1$)
            a.  Find an edge $e$ in $E$ of least weight.
            b.  Delete $e$ from $E$.
            c.  If addition of $e$ to the edge set of $T$ does not produce a circuit then add $e$ to the edge set of $T$ and set $m = m + 1$
End While (post-condition: $T$ is a minimum spanning tree for $G$.)
Output: $T$ (a graph)
End Algorithm
```

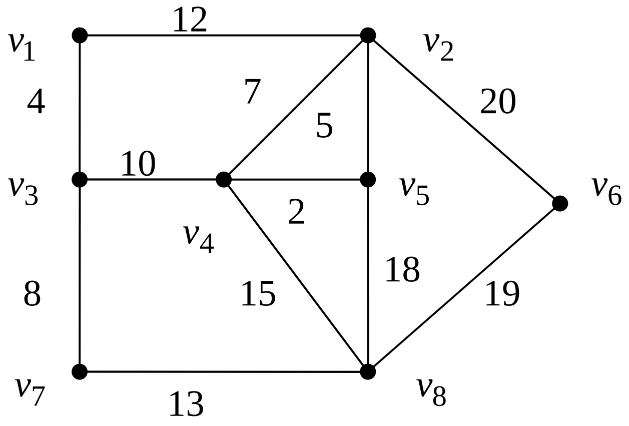
1.10.5. Theorem: Kruskal’s Output.

When a connected weighted graph is input into Kruskal's algorithm, the output is a minimum spanning tree.

The need to construct an ordered edge table before the algorithm begins and the need to check whether a circuit has been formed are two detriments to the computer implementation of Kruskal's algorithm.

Exercises:

- Use Kruskal's algorithm to find a minimum spanning tree for the following graph. Draw the minimum spanning tree.



Edge	Weight	Will adding edge make circuit?	Action taken	Cumulative weight of sub-graph

The minimum spanning tree is:

- Let G be a connected graph. What can you say about an edge e of G if:
  - e is included in every spanning tree for G?
  - e is not included in any spanning tree for G?

1.11. Planar Graphs.

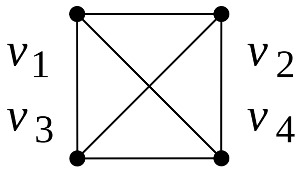
Given a graph which is drawn with incidental crossings of edges, is it possible to determine whether an isomorphic graph could be drawn without any crossings of the edges?

Exercises:

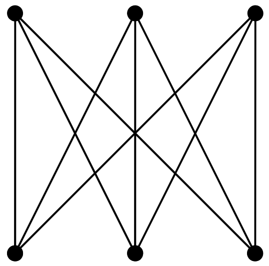
Can either of these graphs be drawn with no incidental crossings?

◆

K_4 :



◆



$K_{3,3}$

1.11.1. Definition: Plane Graph.

A graph *G* is said to be a plane graph if it is drawn in the plane in such a way that there are no incidental crossings of edges.

Notes:

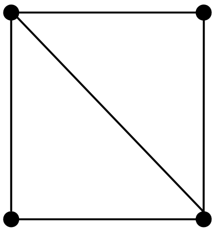
∗ A plane graph divides the plane into distinct “regions” called faces.

∗ The “outside” of a plane graph is always considered a face.

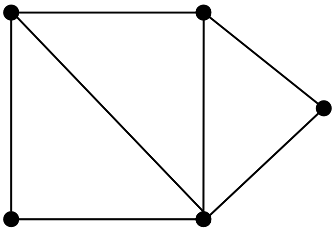
Exercises:

Number the faces in these graphs.

◆



◆



1.11.2. Definition: Planar Graph.

A graph *G* is said to be a planar graph if it can be drawn in the plane without incidental crossings of edges.

Notes:

* A plane graph is a connected planar graph.

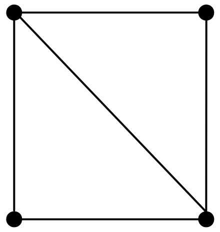
1.11.3. Theorem: Euler's Plane Graph Theorem.

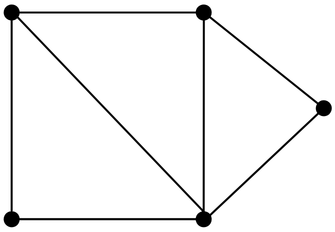
If $G = \{V, E\}$ is a plane graph and if *v* equals the number of vertices, *e* equals the number of edges, and *f* equals the number of faces, then $v - e + f = 2$.

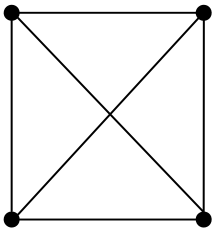
Exercises:

Check Euler's Plane Graph Theorem for these graphs

◆



- 

- 

Re-draw so that there are no edges crossing:

The problem with applying this Theorem is that the graph has to be drawn as a plane graph before it can be applied.

It does not help us decide if $K_{3,3}$ is a planar graph.

The following theorem includes a condition which can be checked in the original graph.

1.11.4. Theorem: Conditions of a Plane Graph.

If $G = \{V, E\}$ is a plane graph, without loops or parallel edges, $v = n(V)$, $e = n(E)$ and $v \geq 3$, then

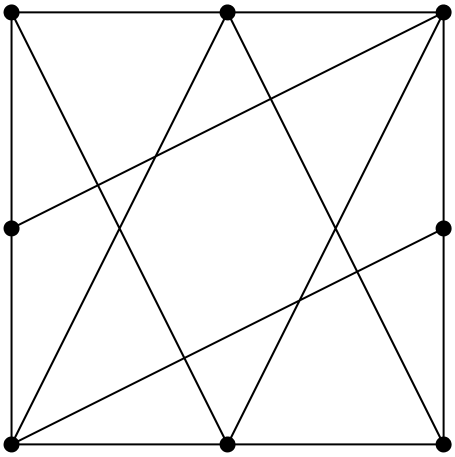
1. $e \leq 3v - 6$, AND
2. if every circuit has at least 4 edges then $e \leq 2v - 4$.

If the conditions of this theorem are also satisfied in any simple graph, *G*, then *G* may or may not be a planar graph.

However, if they are not satisfied, then *G* is not a planar graph.

Exercises:

- Check the conditions for a plane graph for the following graph.

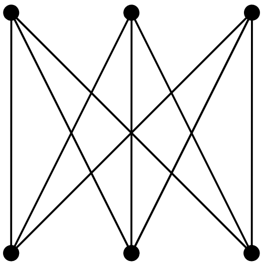


1.

2.

- Decide whether these graphs are planar.

  o

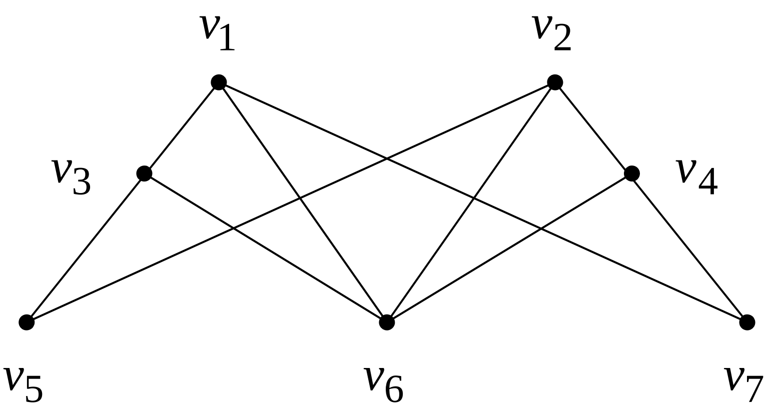


$K_{3,3}$

1.

2.

o

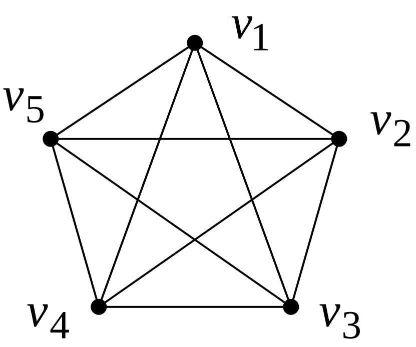


1.

2.

3.

o



This is the complete graph K_5

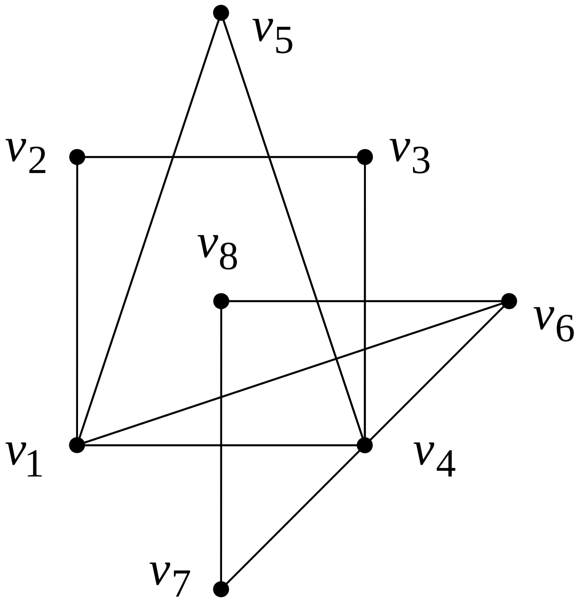
1.

1.11.5. Theorem: Smallest Non-Planar Graphs

$K_{3,3}$ and K_5 are the smallest possible non-planar graphs.

Further, a graph is non-planar if and only if it contains $K_{3,3}$ or K_5 as a sub-graph.

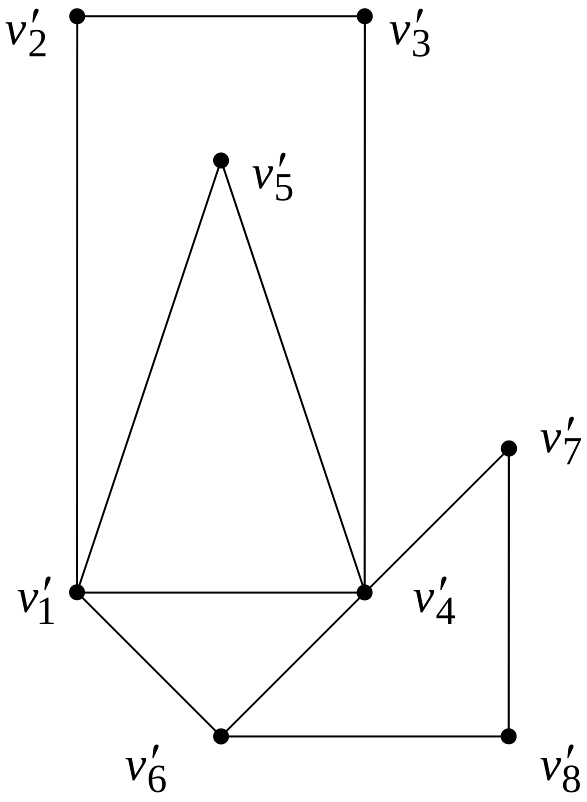
Example:



The graph does not have 6 vertices of degree 3 and so cannot contain $K_{3,3}$.

The graph does not have 5 vertices of degree 4 and so cannot contain K_5.

Redrawing the graph without crossings:



$v = 8,\ e = 11,\ f = 5$

$v - e + f = 8 - 11 + 5 = 2$