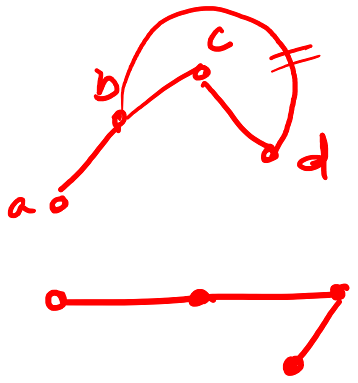
Tree

Tree

- A tree is a connected graph without any circuits.
- It has to be a simple graph.

Some Properties of Tree



THEOREM 1

There is one and only one path between every pair of vertices in a tree, *T*.

Proof: Since T is a connected graph, there must exist at least one path between every pair of vertices in *T*. Now suppose that between two vertices *a* and *b* of *T* there are two distinct paths. The union of these two paths will contain a circuit and *T* cannot be a tree. ■

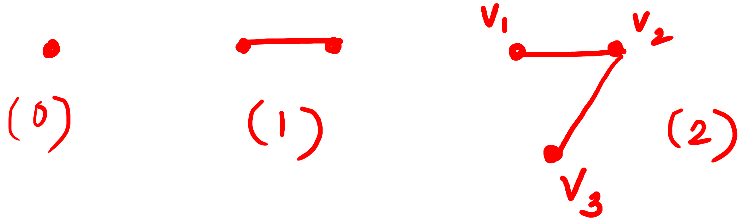
Some Properties of Tree

THEOREM 2

If in a graph G there is one and only one path between every pair of vertices, G is a tree.

Proof: Existence of a path between every pair of vertices assures that G is connected. A circuit in a graph (with two or more vertices) implies that there is at least one pair of vertices a, b such that there are two distinct paths between a and b. Since G has one and only one path between every pair of vertices, G can have no circuit. Therefore, G is a tree. ■

Some Properties of Tree



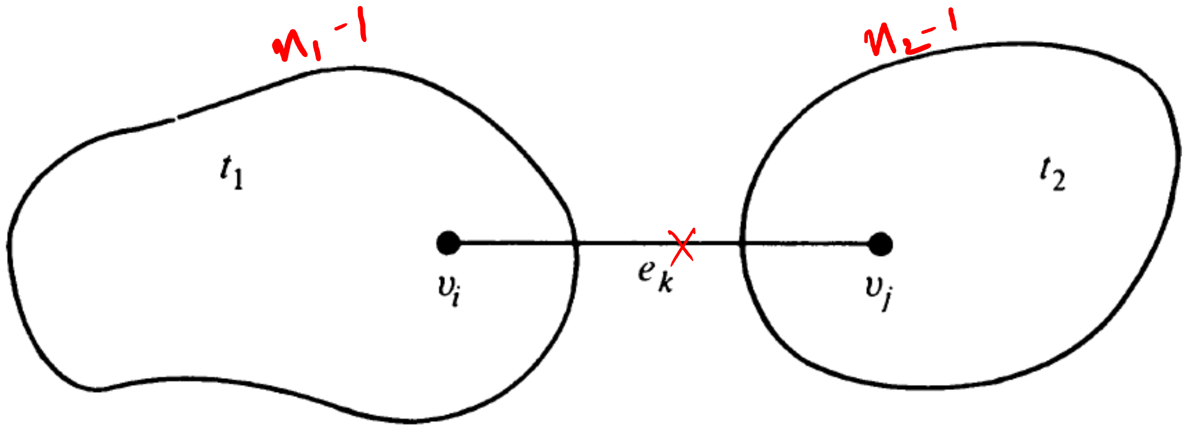
THEOREM 3

A tree with n vertices has $n - 1$ edges.

THEOREM 4

Any connected graph with n vertices and $n - 1$ edges is a tree.

$n = n_1 + n_2$



Some Properties of Tree

THEOREM 5

A graph is a tree if and only if it is minimally connected.

- A connected graph is said to be *minimally connected* if removal of any one edge from it disconnects the graph. A minimally connected graph cannot have a circuit; otherwise, we could remove one of the edges in the circuit and still leave the graph connected. Thus a minimally connected graph is a tree.
- Conversely, if a connected graph G is not minimally connected, there must exist an edge e_i in G such that $G - e_i$ is connected. Therefore, e_i is in some circuit, which implies that G is not a tree.

Some Properties of Tree

THEOREM 6

A graph G with n vertices, $n - 1$ edges, and no circuits is connected.

- ***Proof:***
  - Suppose there exists a circuitless graph G with n vertices and n − 1 edges which is disconnected. In that case G will consist of two or more circuitless components.
  - Without loss of generality, let G consist of two components, g1 and g2. Add an edge e between a vertex v1 in g1 and v2 in g2. Since there was no path between v1 and v2 in G, adding e did not create a circuit.
  - Thus, G ∪ e is a circuitless, connected graph (i.e., a tree) of n vertices and n edges, which is not possible, because of Theorem 3.

a graph G with n vertices is called a tree if

1. G is *connected* and is *circuitless*, or
2. G is *connected* and has $n - 1$ *edges*, or
3. G is *circuitless* and has $n - 1$ *edges*, or
4. There is *exactly one path* between every pair of vertices in G, or
5. G is a *minimally connected* graph.

Pendant vertices in a Tree

In any tree (with two or more vertices), there are at least two pendant vertices.

vertices - n $(n-1)$ edges

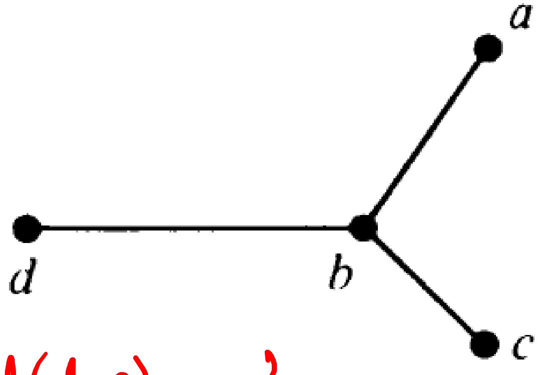
$$\sum 2e = 2(n-1)$$

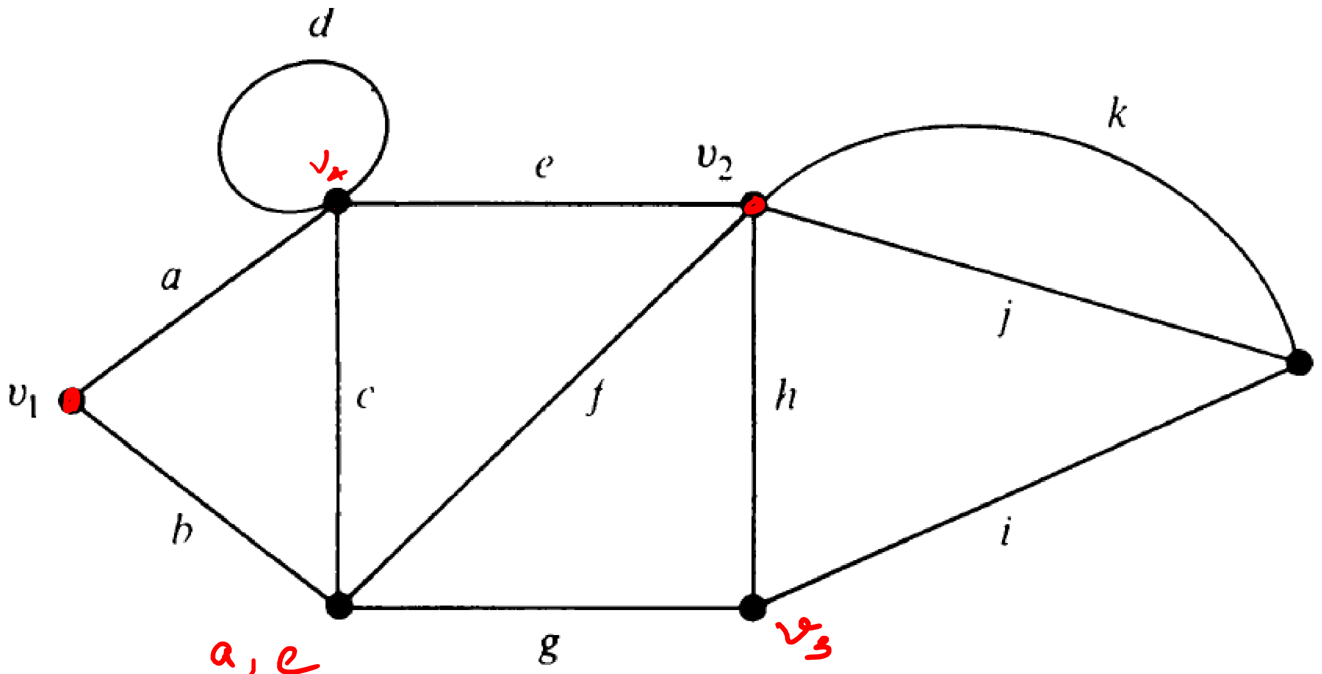
$$= \boxed{2n-2.}$$

$n \geq 2.$

Distance in a Tree

In a connected graph G, the *distance* $d(v_i, v_j)$ between two of its vertices v_i and v_j is the length of the shortest path (i.e., the number of edges in the shortest path) between them.





$d(d,a) = 2$

$d(d,c) = 2$

$d(d,b) = 1$

$d(a,d) = 2$

$d(c,d) = 2$

$d(b,d) = 1$

a, e

$f(v_1,v_2) \leq f(v_1,v_3) + f(v_3,v_2)$

$\underline{2}$ $\qquad$ $2 + 1 = \underline{3}$

Distance in a Tree

A Metric: Before we can legitimately call a function $f(x, y)$ of two variables a "distance" between them, this function must satisfy certain requirements. These are

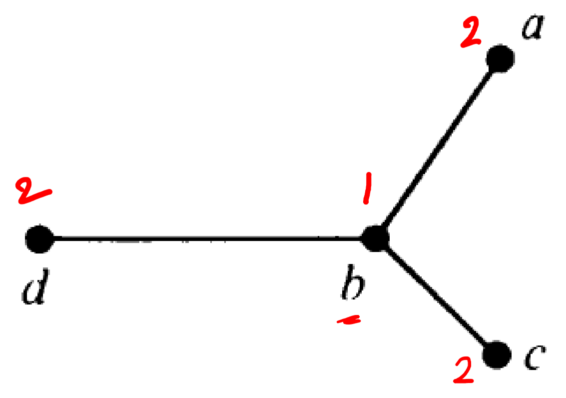
1. Nonnegativity: $f(x, y) \geq 0$, and $f(x, y) = 0$ if and only if $x = y$.
2. Symmetry: $f(x, y) = f(y, x)$.
3. Triangle inequality: $f(x, y) \leq f(x, z) + f(z, y)$ for any z.

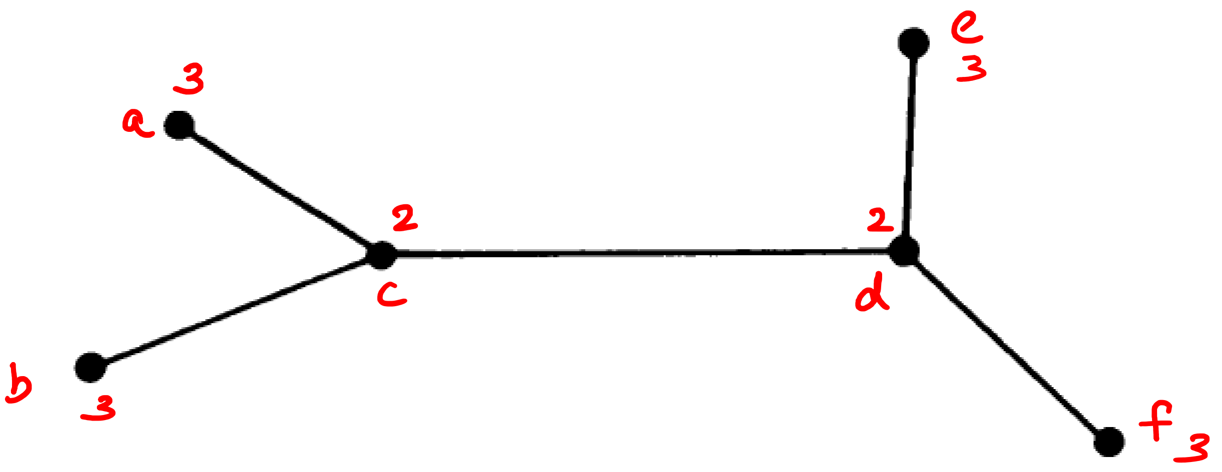
Centers in a Tree

The eccentricity $E(v)$ of a vertex v in a graph G is the distance from v to the vertex farthest from v in G; that is,

$$E(v) = \max_{v_i \in G} d(v, v_i).$$

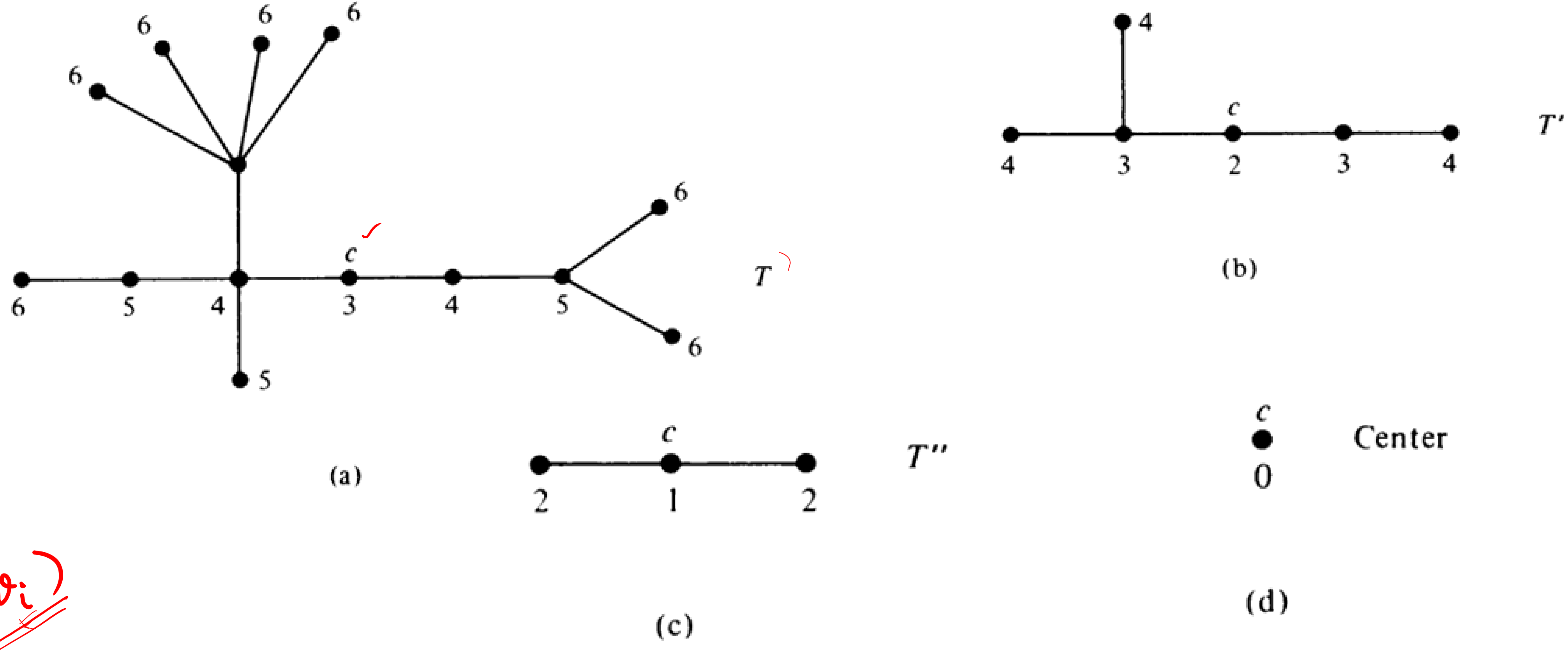
A vertex with minimum eccentricity in graph G is called a *center* of G.





Theorem

Every tree has either one or two centers.



$\max(v, v_i)$

Corollary

From the argument used in proving Theorem, we see that if a tree T has two centers, the two centers must be adjacent.

- Proof:
  - The maximum distance, max $d(v, v_i)$, from a given vertex v to any other vertex v_i occurs only when v_i is a pendant vertex.
  - With this observation, let us start with a tree T having more than two vertices. Tree T must have two or more pendant vertices (Theorem-7).
  - Delete all the pendant vertices from T. The resulting graph T′ is still a tree. The removal of all pendant vertices from T uniformly reduced the eccentricities of the remaining vertices (i.e., vertices in T′) by one. Therefore, all vertices that T had as centers will still remain centers in T′.
  - From T′ we can again remove all pendant vertices and get another tree T″. We continue this process until there is left either a vertex (which is the center of T) or an edge (whose end vertices are the two centers of T). Thus the theorem.

Radius and Diameter

The eccentricity of a center (which is the distance from the center of the tree to the farthest vertex) in a tree is defined as the *radius* of the tree.

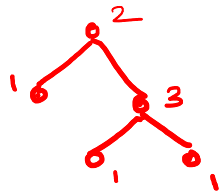
Diameter of a tree T, is defined as the length of the longest path in T.

Rooted Trees

A tree in which one vertex (called the *root*) is distinguished from all the others is called a *rooted tree*.

Generally, the term *tree* means trees without any root. However, for emphasis they are sometimes called *free trees* (or *nonrooted trees*) to differentiate them from the rooted kind.

Binary tree



A *binary tree* is defined as a tree in which there is exactly one vertex of degree two, and each of the remaining vertices is of degree one or three

Two properties:

1. The number of vertices n in a binary tree is always odd. This is because there is exactly one vertex of even degree, and the remaining $n - 1$ vertices are of odd degrees. Since from Theorem 1-1 the number of vertices of odd degrees is even, $n - 1$ is even. Hence n is odd.

2. Let p be the number of pendant vertices in a binary tree T. Then $n - p - 1$ is the number of vertices of degree three. Therefore, the number of edges in T equals

$$\frac{1}{2}[p + 3(n - p - 1) + 2] = n - 1;$$

hence $$p = \frac{n+1}{2}.$$

Spanning Trees

G (5) T' $(3) - 2$ T'' $(4) - 3$ T_5 $(5) - 4$

A tree T is said to be a *spanning tree* of a connected graph G if T is a subgraph of G and T contains all vertices of G.

Since the vertices of G are barely hanging together in a spanning tree, it is a sort of skeleton of the original graph G. This is why a spanning tree is sometimes referred to as a *skeleton* or *scaffolding* of G.

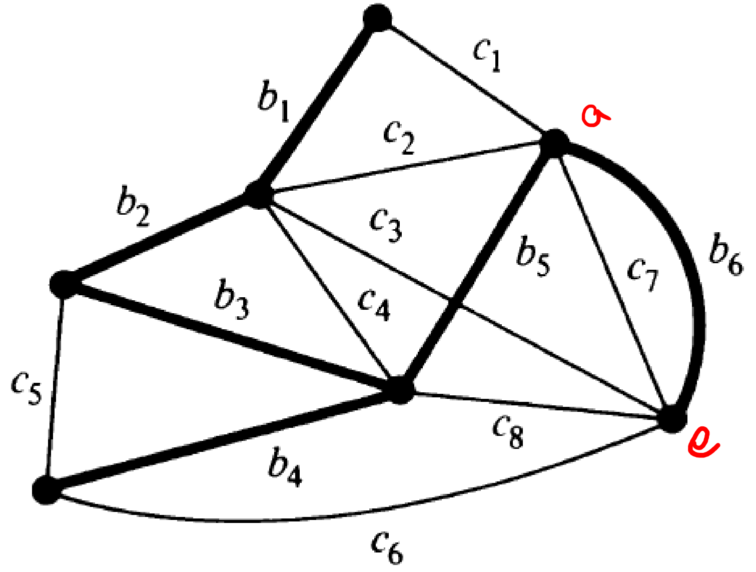
Since spanning trees are the largest trees among all trees in G, it is also quite appropriate to call a spanning tree a *maximal tree subgraph* or *maximal tree* of G.

a disconnected graph with k components has a *spanning forest* consisting of k spanning trees.

THEOREM

Every connected graph has at least one spanning tree.

$b_1 \cdot\cdot\cdot b_5\, c_7$ b_6



An edge in a spanning tree T is called a *branch* of T. An edge of G that is not in a given spanning tree T is called a *chord*.

$$T \cup \bar{T} = G,$$

where T is a spanning tree, and $\bar{T}$ is the complement of T in G. Since the subgraph $\bar{T}$ is the collection of chords, it is quite appropriately referred to as the *chord set* (or *tie set* or *cotree*) of T.

THEOREM

With respect to any of its spanning trees, a connected graph of n vertices and e edges has $n - 1$ tree branches and $e - n + 1$ chords.

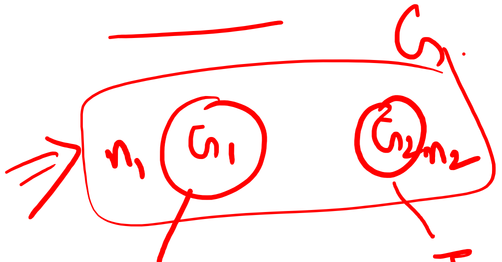
$G \Rightarrow |V| = n \qquad |E| = e$

$T \Rightarrow |V| = n \qquad |E_T| = n-1$

branch

$e - (n-1)$

$\boxed{e - n + 1}$



rank $\qquad r = n - k,$

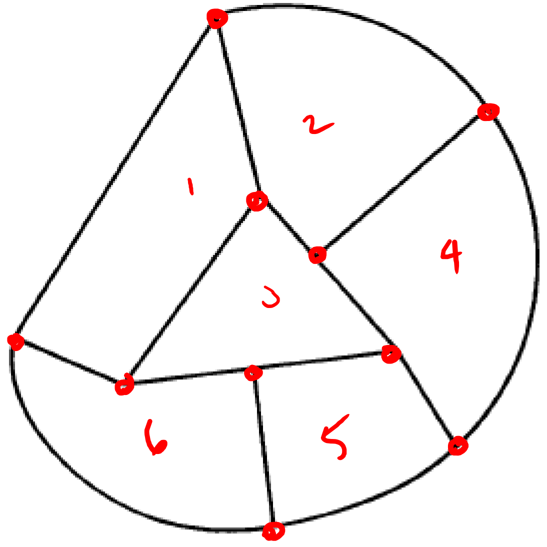
nullity $\qquad \mu = e - n + k$

T_1 $|V| = n_1$ $\qquad T_2$ $|V| = n_2$

$|E_1| = n_1 - 1 \qquad |E_2| = n_2 - 1$

$e - (n_1 + n_2 - 2) = \boxed{e - n + 2} \qquad n_1 - 1 + n_2 - 1 = n_1 + n_2 - 2$

- The nullity of a graph is also referred to as its cyclomatic number, or first Betti number farm consisting of six walled plots of land, as shown in Fig. and these plots are full of water, how many walls will have to be broken so that all the water can be drained out?



Here $n = 10$ and $e = 15$.

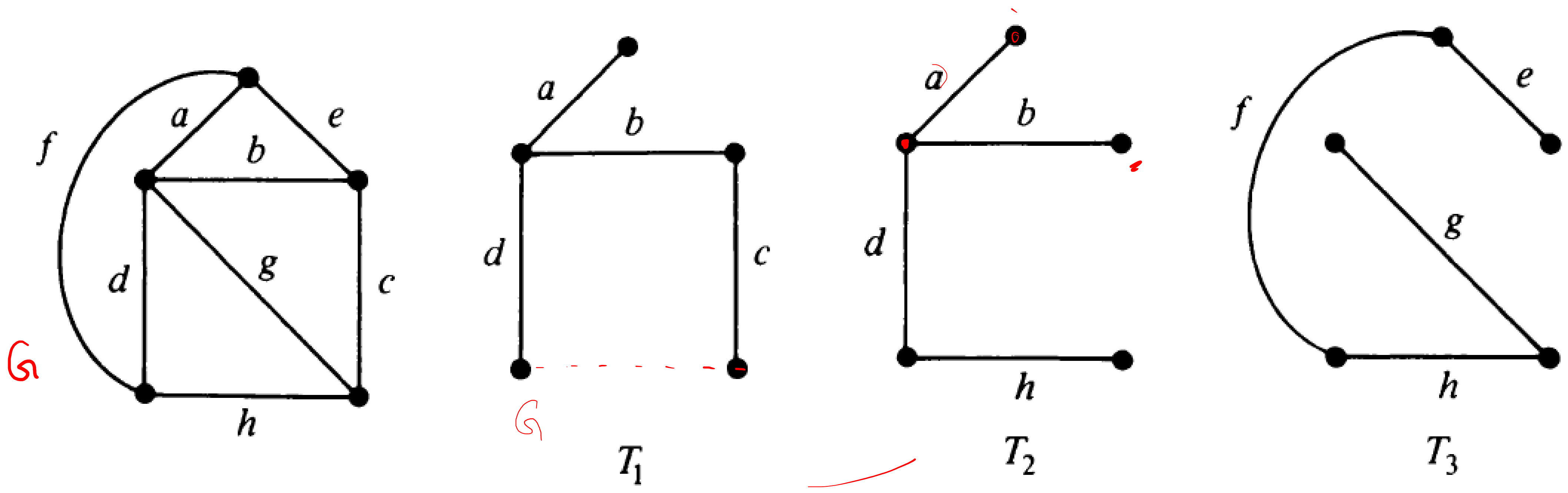
$n - 1 = 9$

$e - n + 1$

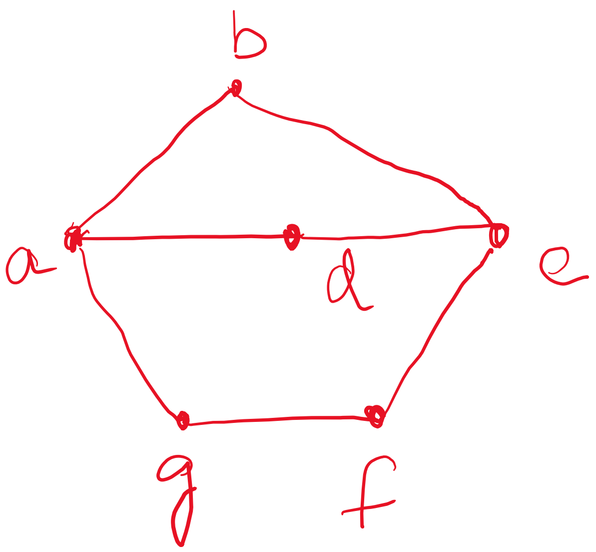
$15 - 10 + 1 = \underline{\underline{6}}$

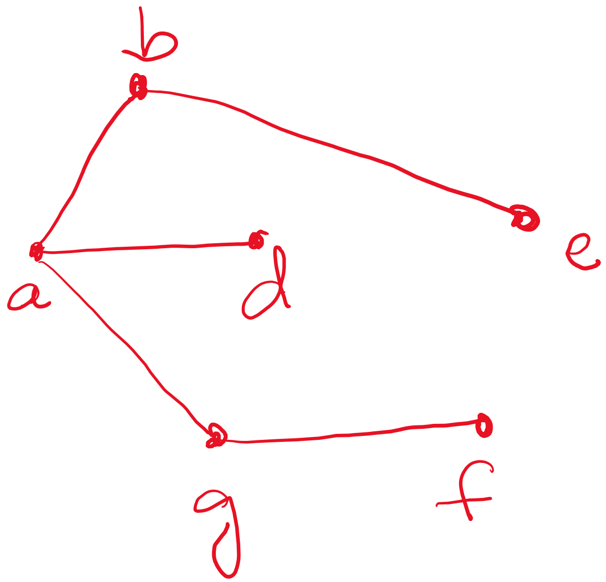
Finding all spanning trees of a graph

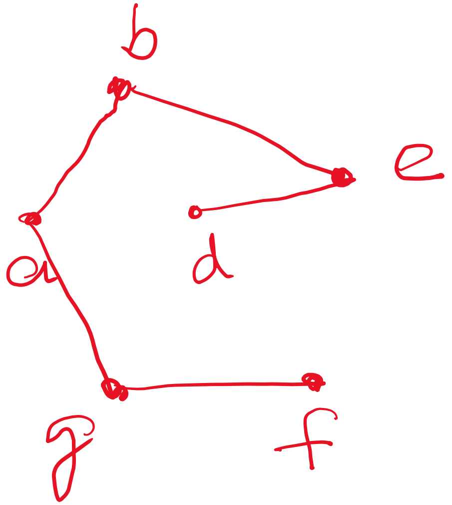
$T_2 \oplus T_3 = \{a, b, d, e, f, g\}$

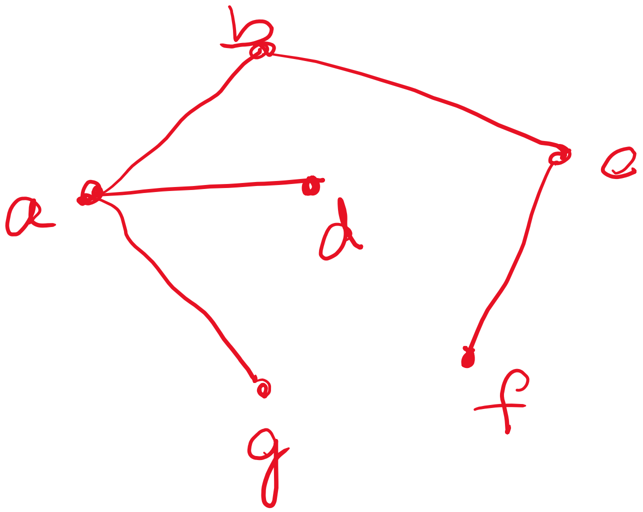


This generation of one spanning tree from another, through addition of a chord and deletion of an appropriate branch, is called a *cyclic interchange* or *elementary tree transformation*.


b
a
d
e
g
f


b
a
d
e
g
f


b
e
a
d
g
f


b
a
d
e
g
f

The distance between two spanning trees T_i and T_j of a graph G is defined as the number of edges of G present in one tree but not in the other. This distance may be written as $d(T_i, T_j)$.

Let $T_i \oplus T_j$ be the ring sum of two spanning trees T_i and T_j of G, Let $N(g)$ denote the number of edges in a graph g. Then, from definition,

$$d(T_i, T_j) = \frac{1}{2} N(T_i \oplus T_j)$$

The distance between the spanning trees of a graph is a *metric*. That is, it satisfies

$$d(T_i, T_j) \geq 0 \quad \text{and} \quad d(T_i, T_j) = 0 \text{ if and only if } T_i = T_j,$$

$$d(T_i, T_j) = d(T_j, T_i),$$

$$d(T_i, T_j) \leq d(T_i, T_k) + d(T_k, T_j).$$

Starting from any spanning tree of a graph G, we can obtain every spanning tree of G by successive cyclic exchanges.

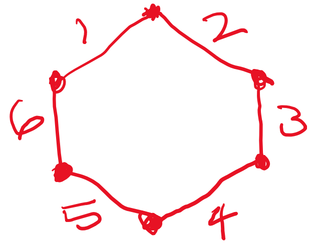
$n = r+1$ $\quad e = n-1 = r$

Since in a connected graph G of rank r (i.e., of $r + 1$ vertices) a spanning tree has r edges, we have the following results:

The maximum distance between any two spanning trees in G is

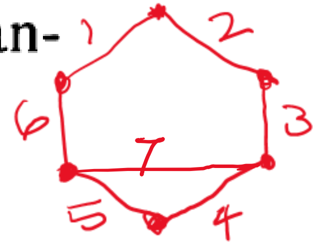
$$\max d(T_i, T_j) = \frac{1}{2} \max N(T_i \oplus T_j)$$

$$\leq r, \text{ the rank of } G.$$



$6C_5$

Also, if μ is the nullity of G, we know that no more than μ edges of a spanning tree T_i can be replaced to get another tree T_j.



Hence $$\max d(T_i, T_j) \leq \mu;$$

$7C_5 - 2$

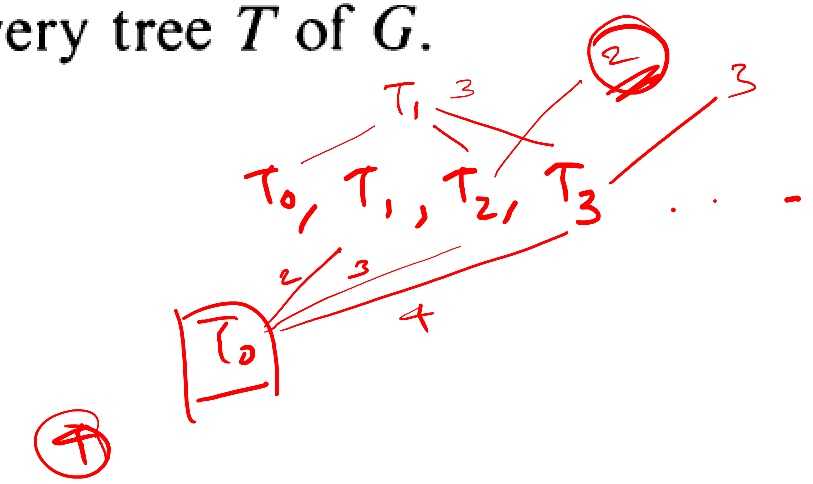
combining the two,

$$\max d(T_i, T_j) \leq \min(\mu, r),$$

Central Tree

Central Tree: For a spanning tree T_0 of a graph G, let $\max_i d(T_0, T_i)$ denote the maximal distance between T_0 and any other spanning tree of G. Then T_0 is called a *central tree* of G if

$$\max_i d(T_0, T_i) \leq \max_j d(T, T_j) \qquad \text{for every tree } T \text{ of } G.$$



Spanning Trees in a Weighted Graph

$1+1+1+1+1=5$

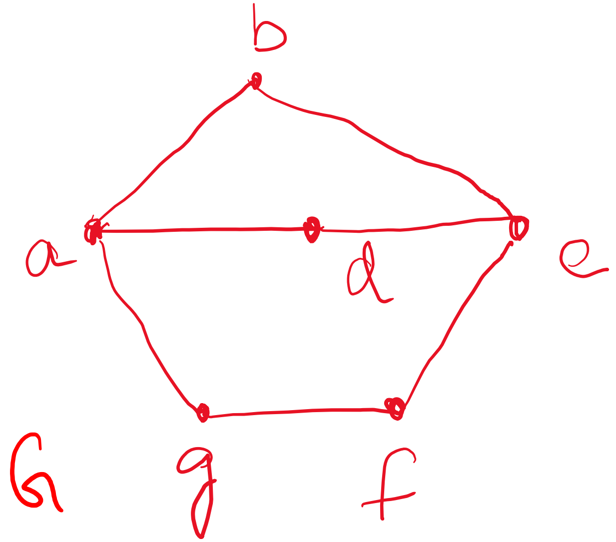
the *weight of a spanning tree* T of G is defined as the sum of the weights of all the branches in T. In general, different spanning trees of G will have different weights.

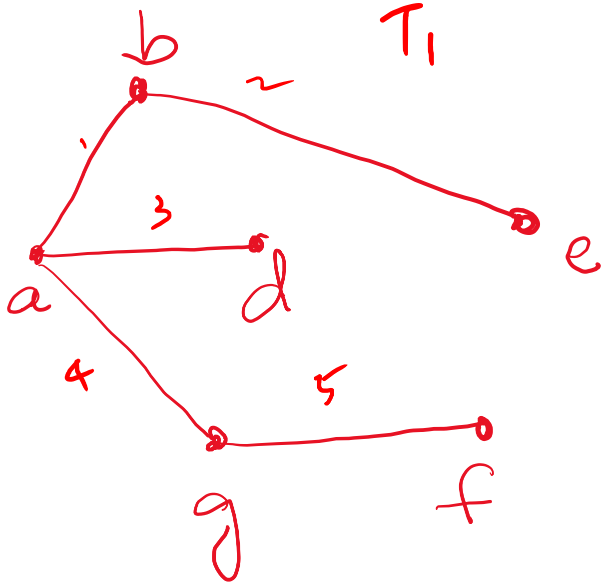
A spanning tree with the smallest weight in a weighted graph is called a *shortest spanning tree* or *shortest-distance spanning tree* or *minimal spanning tree*.

THEOREM

A spanning tree T (of a given weighted connected graph G) is a shortest spanning tree (of G) if and only if there exists no other spanning tree (of G) at a distance of one from T whose weight is smaller than that of T.

(5)

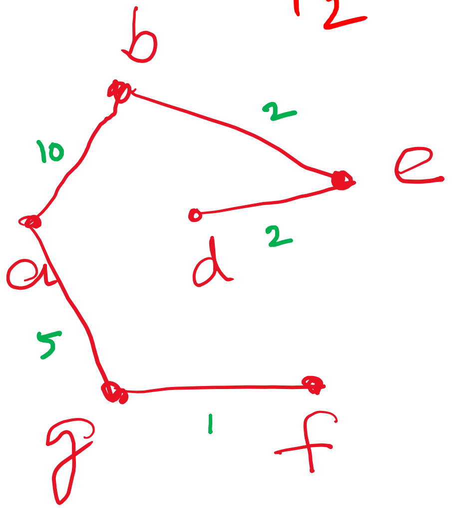


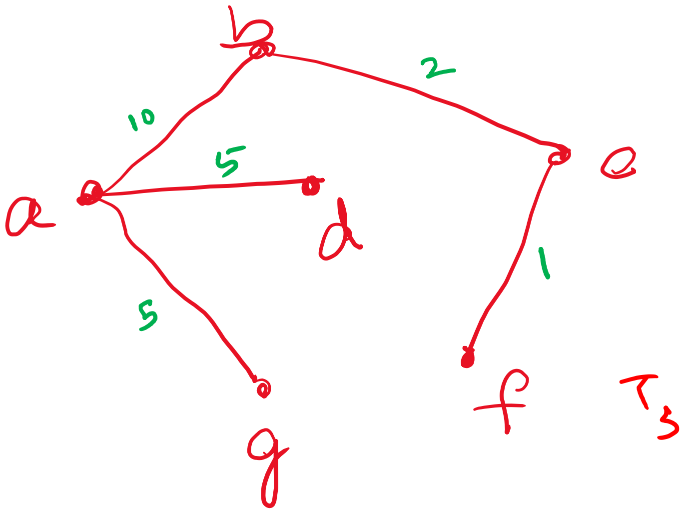


T_1

T_2 (5)

20





T_3 (5)

23

MST Methods

- Kruskal's
- Prim's

Select n-1 edges from a weighted graph of n vertices with minimum cost.

Greedy Strategy

- An optimal solution is constructed in stages
- At each stage, the best decision is made at this time
- Since this decision cannot be changed later,
  we make sure that the decision will result in a feasible solution
- Typically, the selection of an item at each stage is based on a least cost or a highest profit criterion

Kruskal’s Idea

- Build a minimum cost spanning tree T by adding edges to T one at a time
- Select the edges for inclusion in T in nondecreasing order of the cost
- An edge is added to T if it does not form a cycle
- Since G is connected and has $n > 0$ vertices, exactly $n-1$ edges will be selected

Examples for Kruskal’s Algorithm

0 10 5

2 12 3

1 14 6

1 16 2

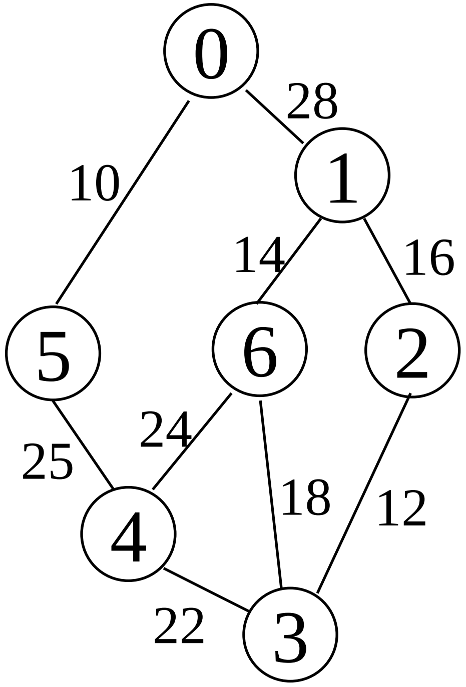
3 18 6

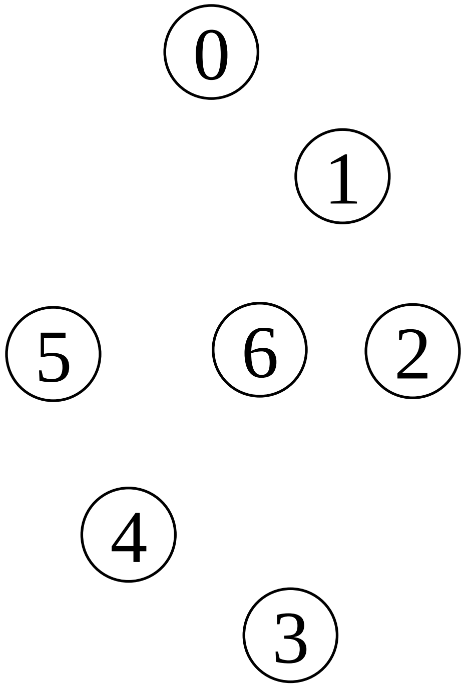
3 22 4

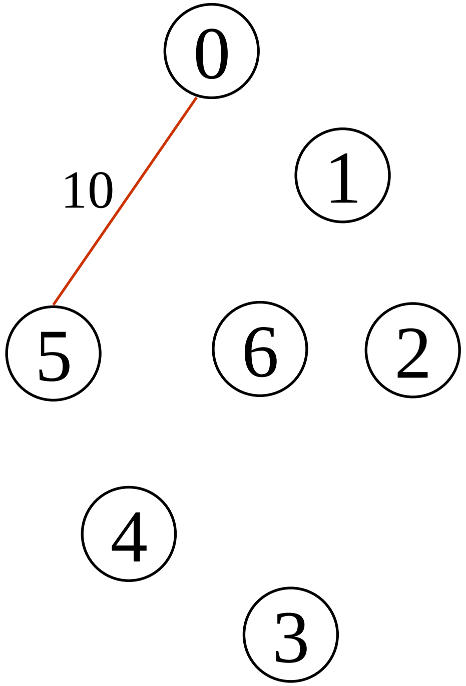
4 24 6

4 25 5

0 28 1







0 —10— 5

2 —12— 3

1 —14— 6

1 —16— 2

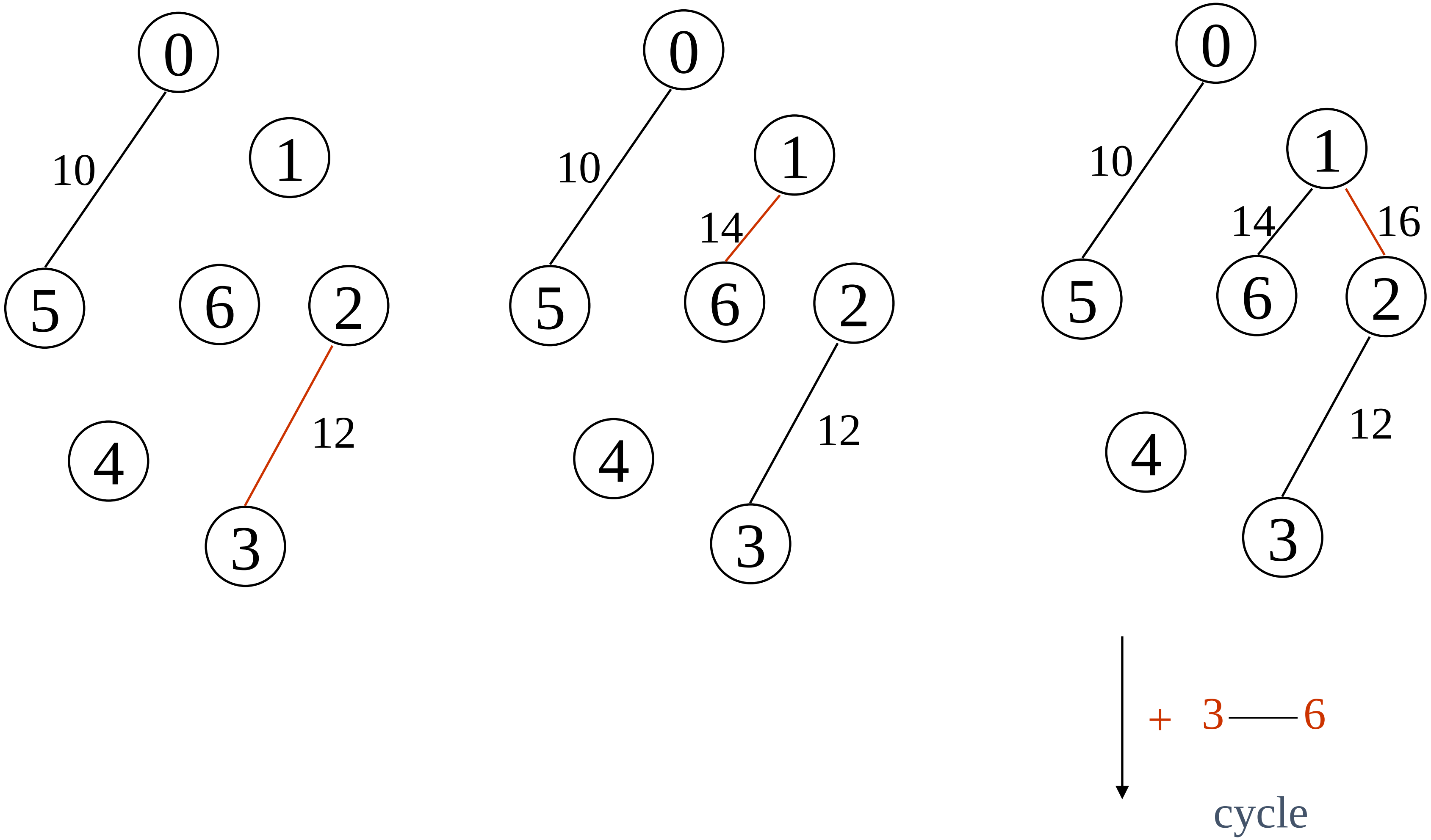
3 —18— 6

3 —22— 4

4 —24— 6

4 —25— 5

0 —28— 1



0 — 10 — 5

2 — 12 — 3

1 — 14 — 6

1 — 16 — 2

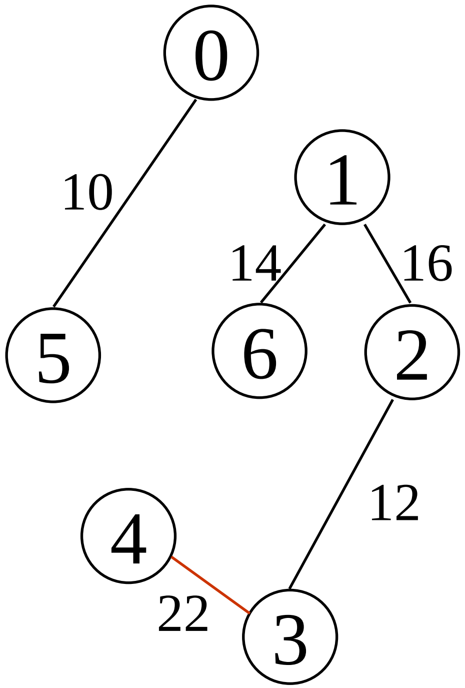
3 — 18 — 6

3 — 22 — 4

4 — 24 — 6

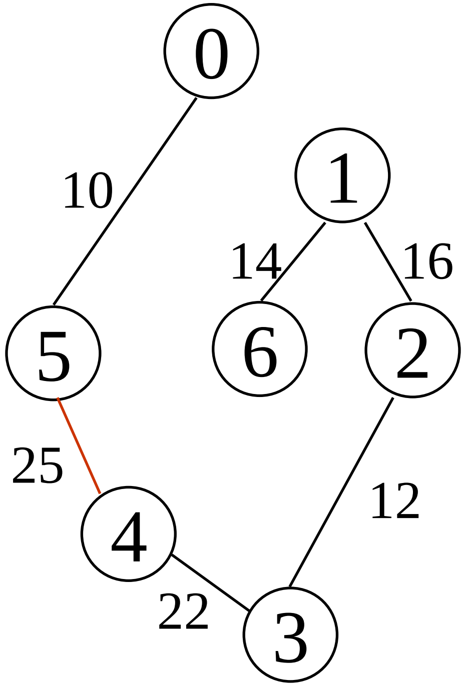
4 — 25 — 5

0 — 28 — 1



+ 4 —— 6

cycle

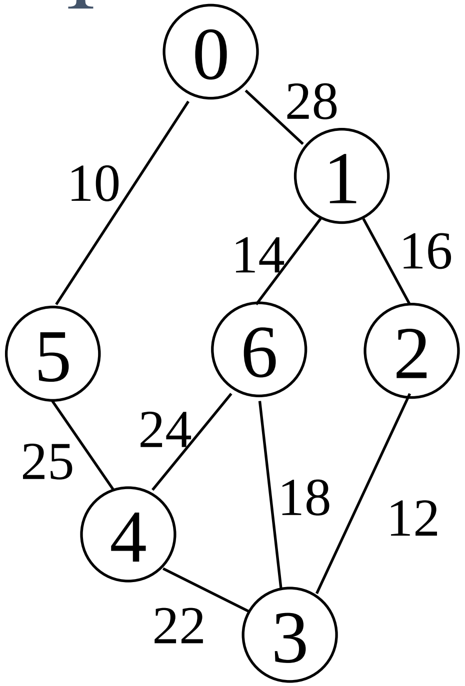


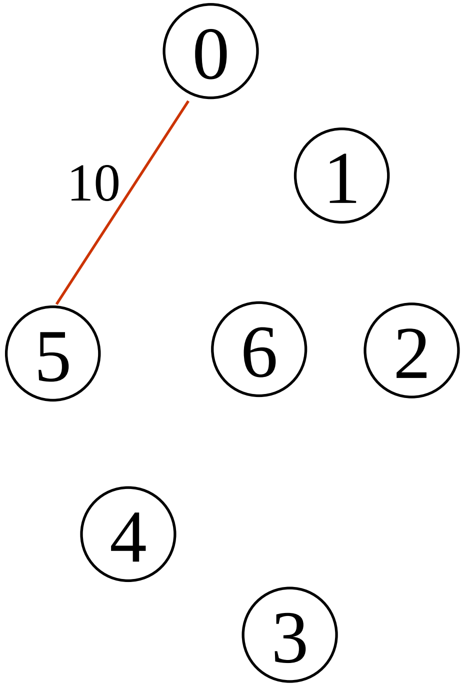
cost = 10 +25+22+12+16+14

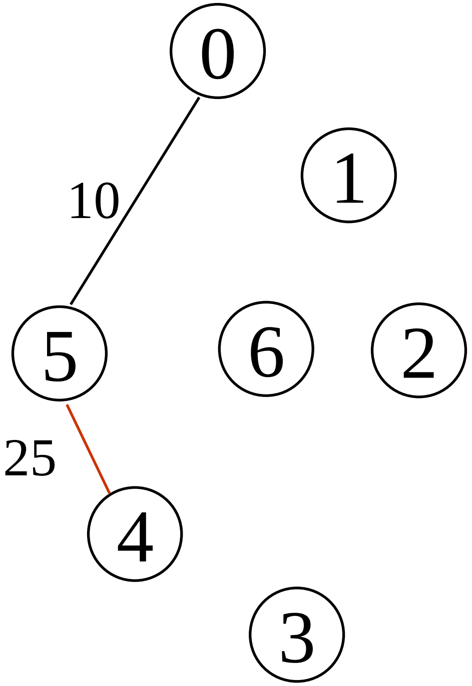
Kruskal's Algorithm

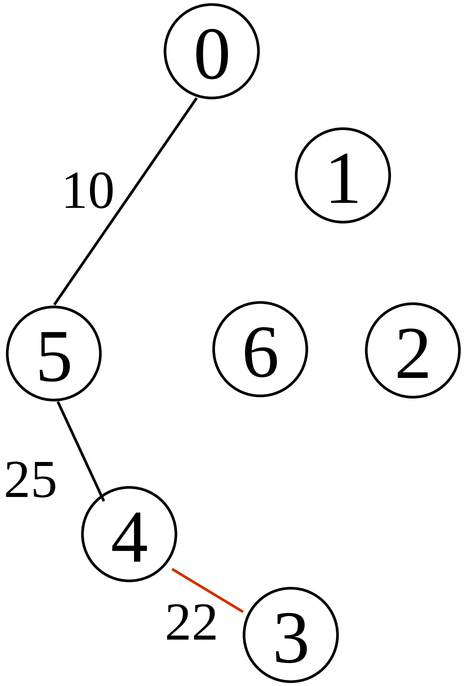
```
T= {};
while (T contains less than n-1 edges && E is not empty)
{
  choose a least cost edge (v,w) from E;
  delete (v,w) from E;
  if ((v,w) does not create a cycle in T)
    add (v,w) to T
  else discard (v,w);
}
if (T contains fewer than n-1 edges)
  printf("No spanning tree\n");
```

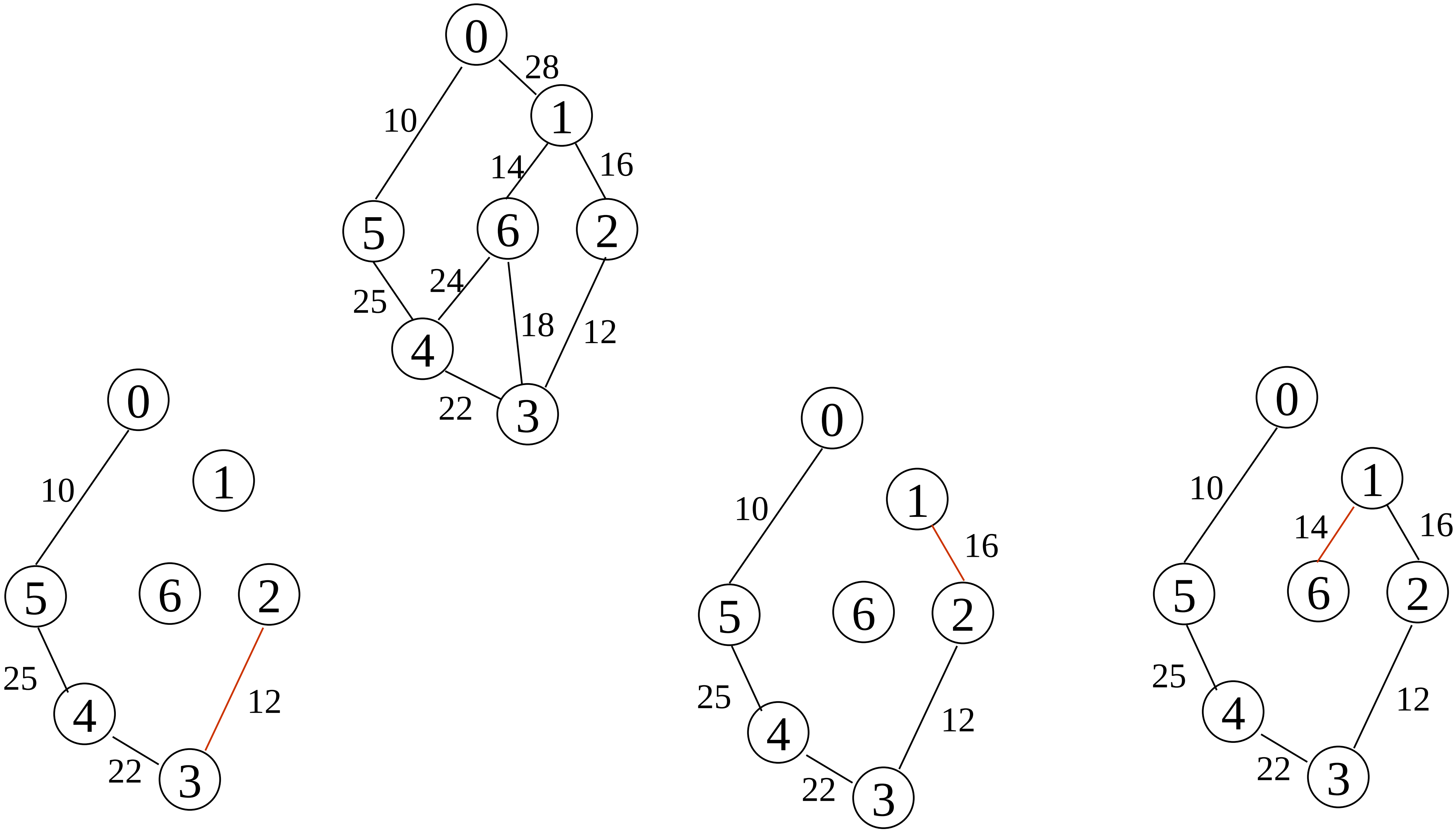
Examples for Prim’s Algorithm







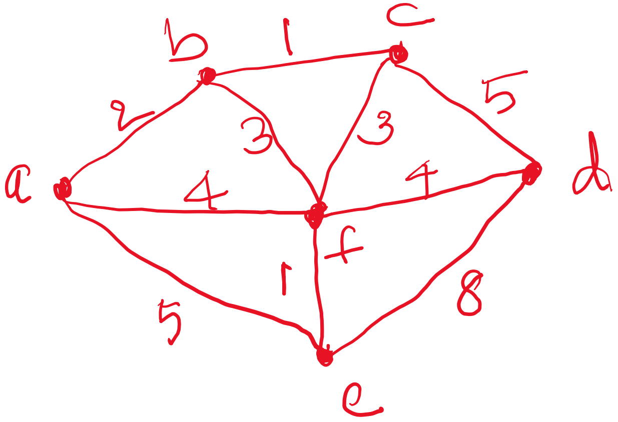



0
1
2
3
4
5
6
28
10
14
16
24
25
18
12
22
0
1
5
6
2
4
3
10
25
12
22
0
1
5
6
2
4
3
10
16
25
12
22
0
1
5
6
2
4
3
10
14
16
25
12
22

Prim's Algorithm

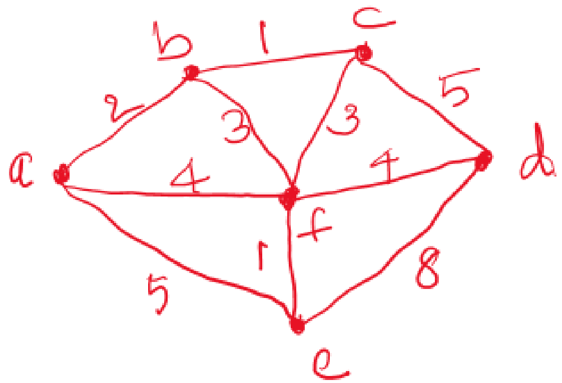
```
T={};
TV={0};
while (T contains fewer than n-1 edges)
{
  let (u,v) be a least cost edge such
    that u ∈ TV and v ∉ TV
  if (there is no such edge ) break;
  add v to TV;
  add (u,v) to T;
}
if (T contains fewer than n-1 edges)
  printf("No spanning tree\n");
```

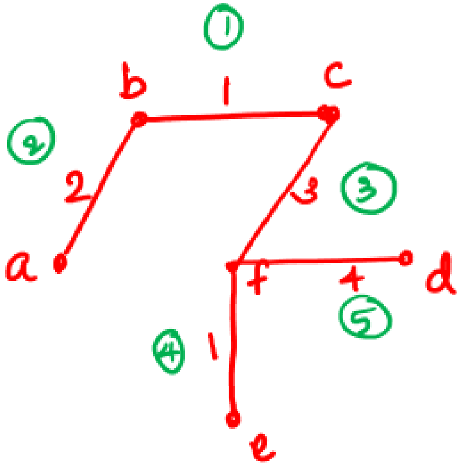
Prim's Methods

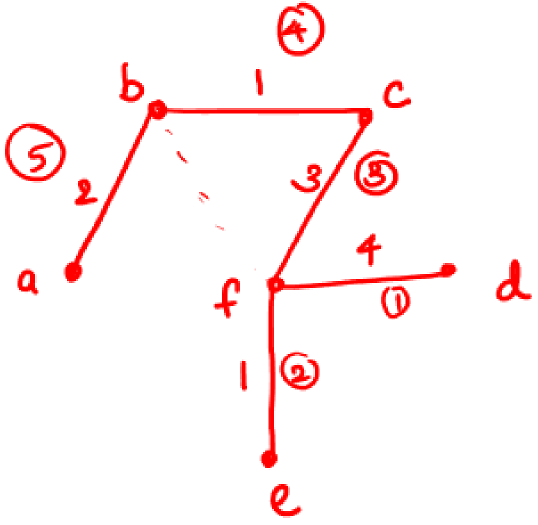


	a	b	c	d	e	f
a	–	2	∞	∞	5	4
b	2	–	1	∞	∞	3
c	∞	1	–	5	∞	3
d	∞	∞	5	–	8	4
e	5	∞	∞	8	–	1
f	4	3	3	4	1	–

Prim's Methods





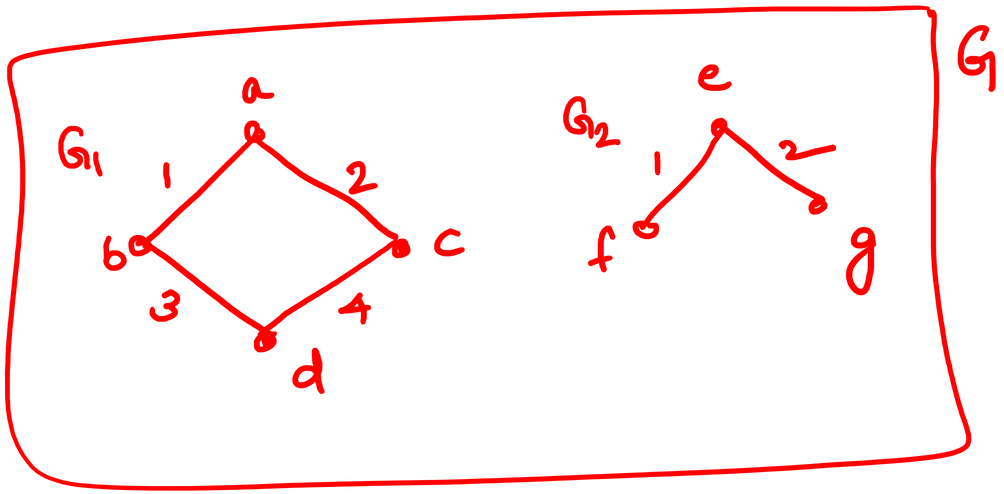


	a	b	c	d	e	f
→ a	–	2	∞	∞	5	4
→ b	2	–	1	∞	∞	3
→ c	∞	1	–	5	∞	3
→ d	∞	∞	5	–	8	4
→ e	5	∞	∞	8	–	1
→ f	4	3	3	4	1	–

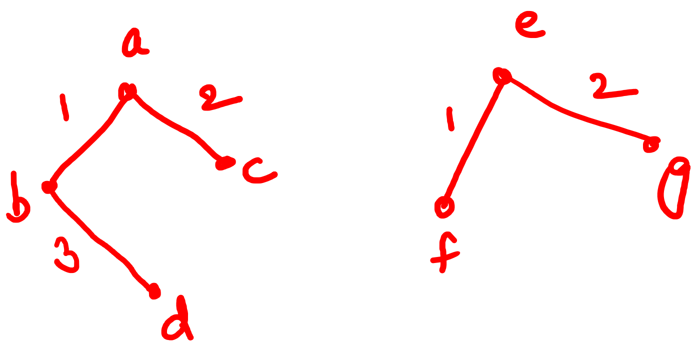
cost $= 2+1+3+1+4$

$= \underline{\underline{11}}$

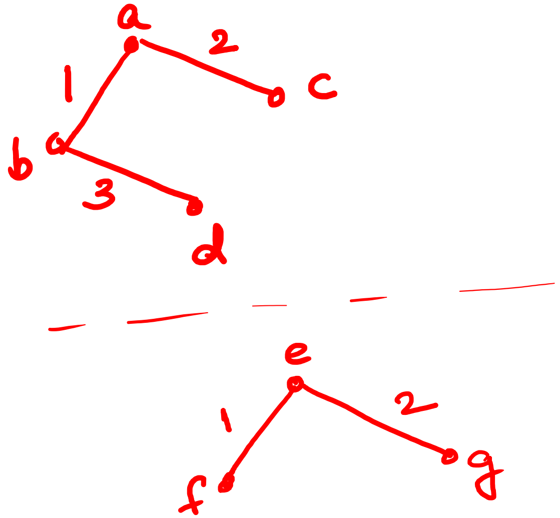
MST in Disconnected Graph

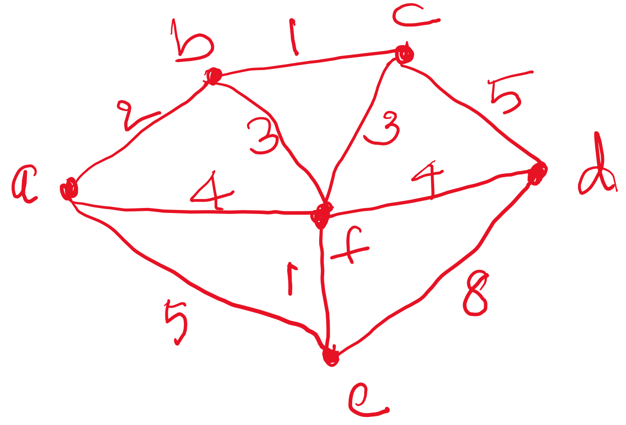


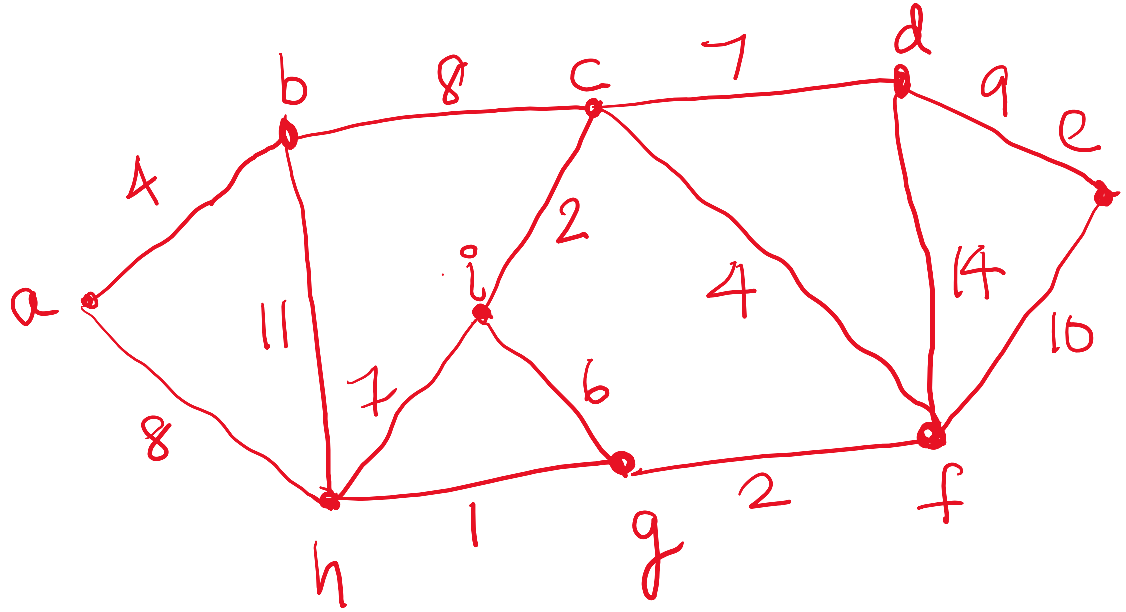
Kruskal's



Prim's




a
b
c
d
e
f
1
2
3
3
5
4
4
5
1
8


a
b
c
d
e
f
g
h
i
4
8
8
7
9
11
7
2
4
14
10
6
1
2