Graphs

Irina Prosvirnina

- Trees
- Weighted graphs
- An algorithm which produces a minimal spanning tree (Prim's algorithm)
- Rooted trees

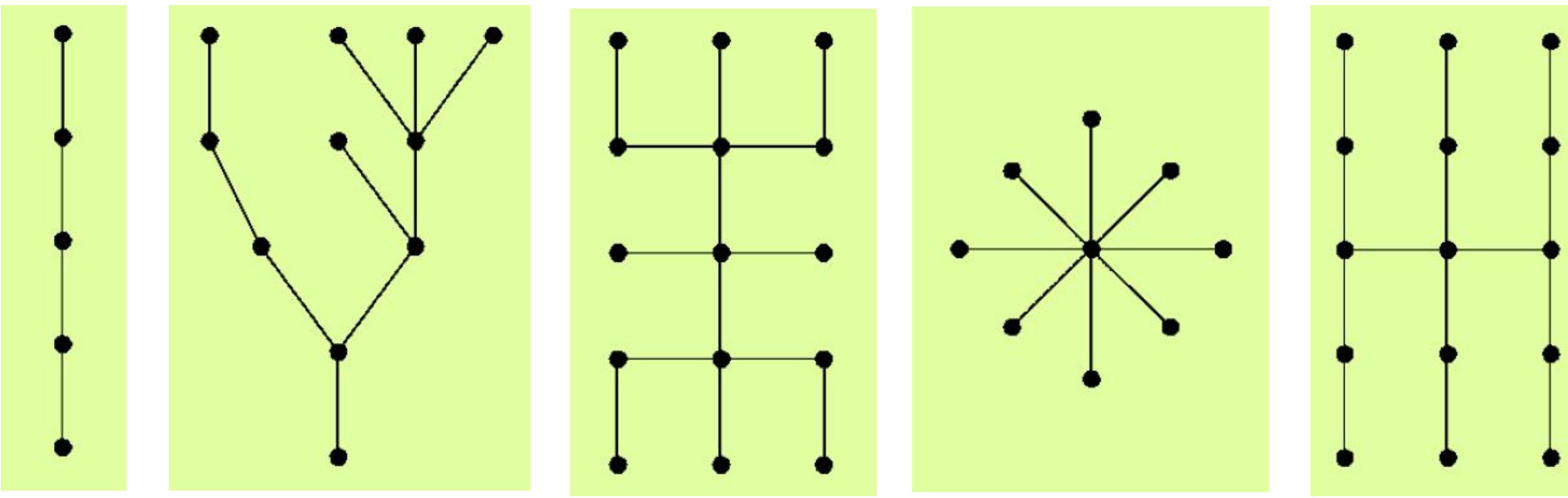
Trees

Definition 1

A **tree** is a connected graph which contains no cycles.

- It is immediately apparent from the definition that a tree has no loops or multiple edges.
- Any loop is a cycle by itself, and if edges e_i and e_j join the same pair of vertices then the sequence e_i, e_j is also a cycle.

Trees



Some examples of trees are given in the figure.

Trees

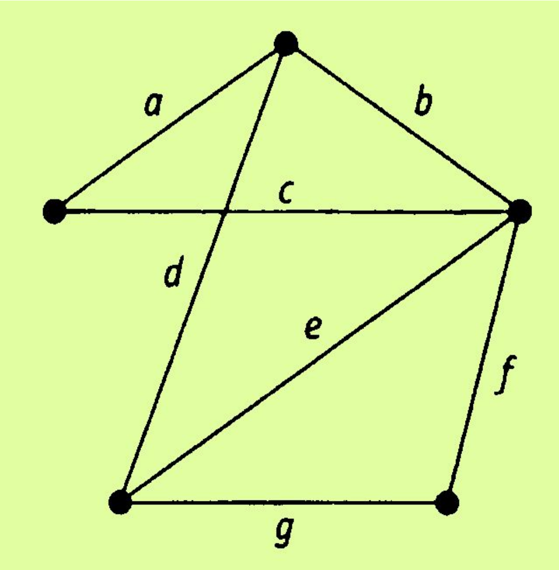
Definition 2

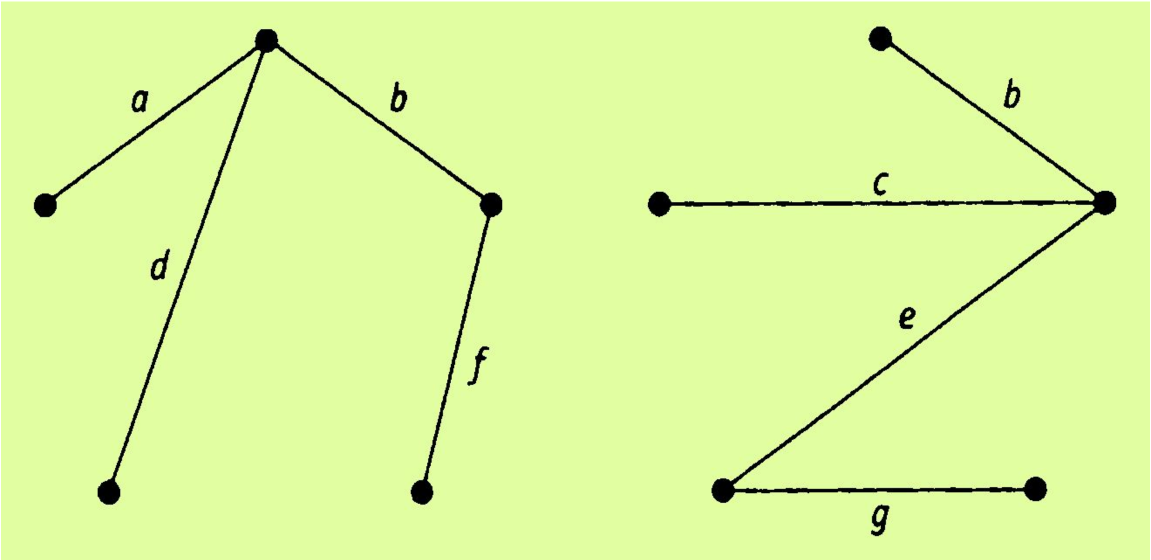
Let Γ be a connected graph with vertex set V.

A **spanning tree** in Γ is a subgraph which is a tree and has vertex set V.

Trees

Example 1





Graph Γ

Spanning trees for the graph Γ

Trees

Theorem 1

Every connected graph contains a spanning tree.

? Every connected graph contains a spanning tree.

Proof

Let Γ be a connected graph.

If Γ contains no cycle then there is nothing to prove as Γ is its own spanning tree.

Suppose, then, Γ contains a cycle. Removing any edge from the cycle gives a graph which is still connected.

If the new graph contains a cycle then again remove one edge of the cycle.

Continue this process until the resulting graph T contains no cycles.

? Every connected graph contains a spanning tree.

Proof

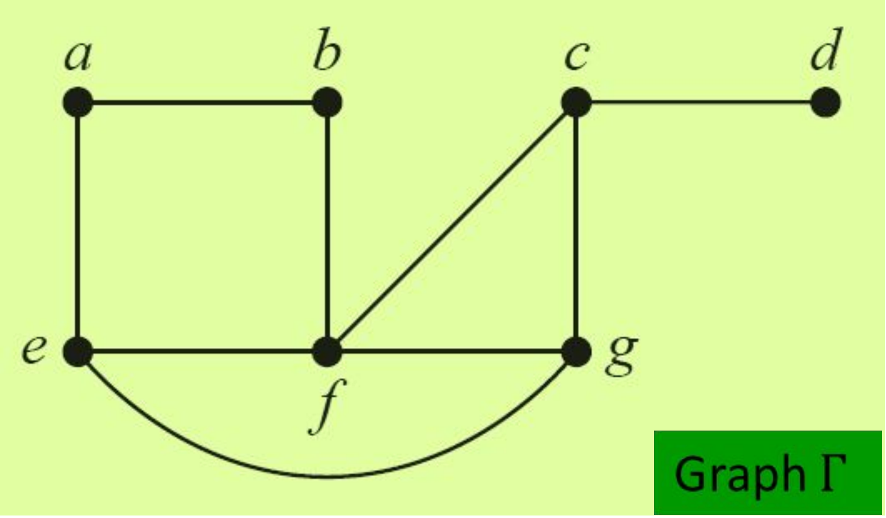
- We have not removed any vertices so T has the same vertex set as Γ, and at each stage of the above process we obtain a connected graph.

Therefore T itself is connected; it is a spanning tree for Γ. ■

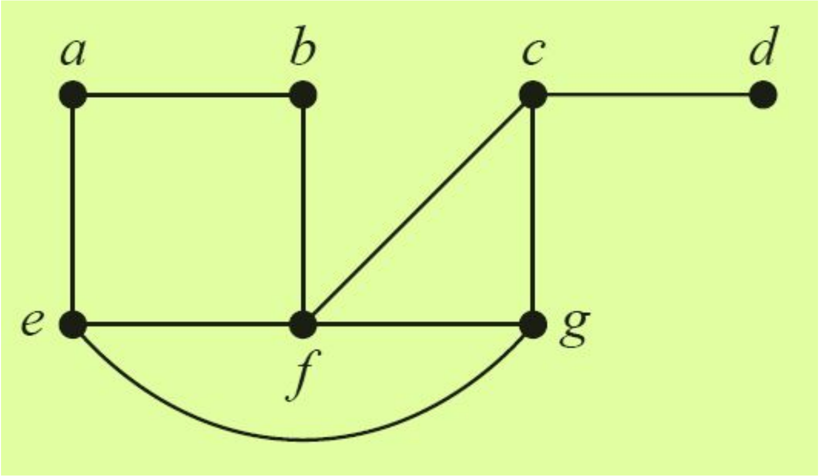
Trees

Example 2

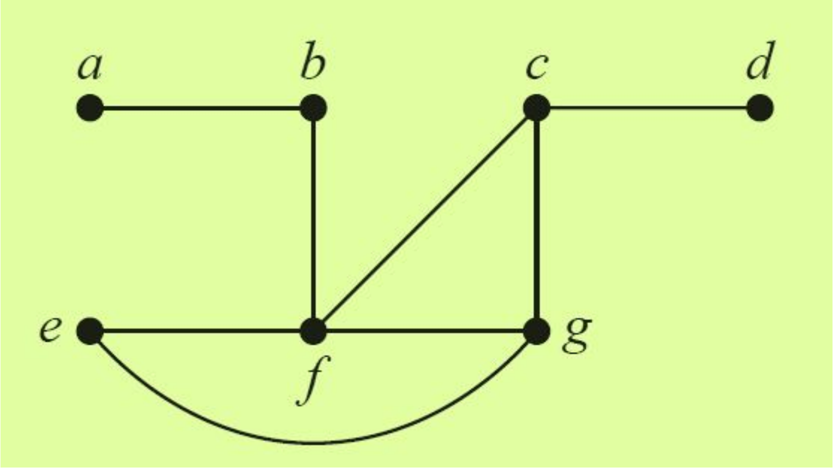
Find a spanning tree of the graph Γ shown in the figure.



Example 2
Find a spanning tree of the graph Γ shown in the figure.

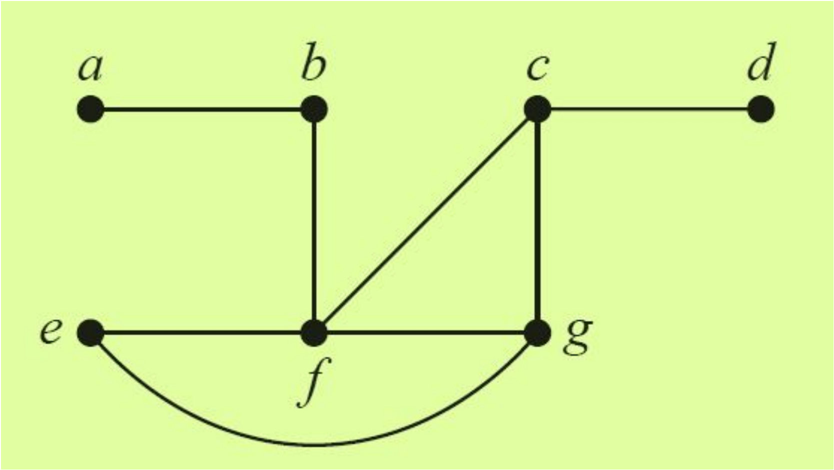


Edge removed: $\{a, e\}$

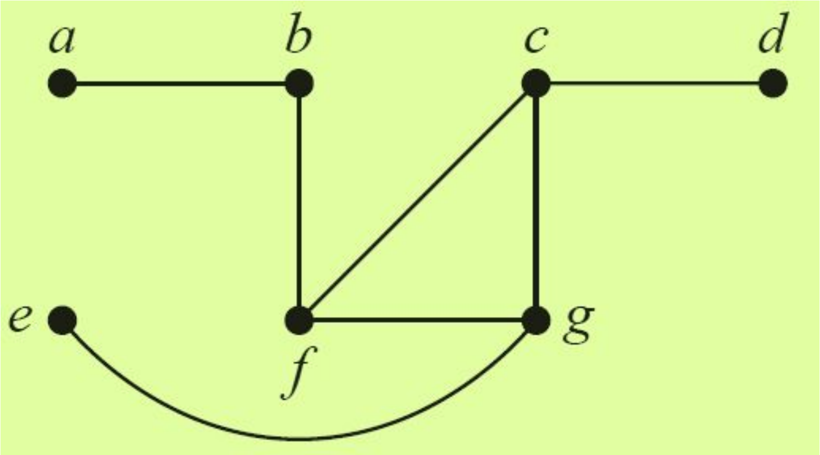


Example 2

Find a spanning tree of the graph Γ shown in the figure.

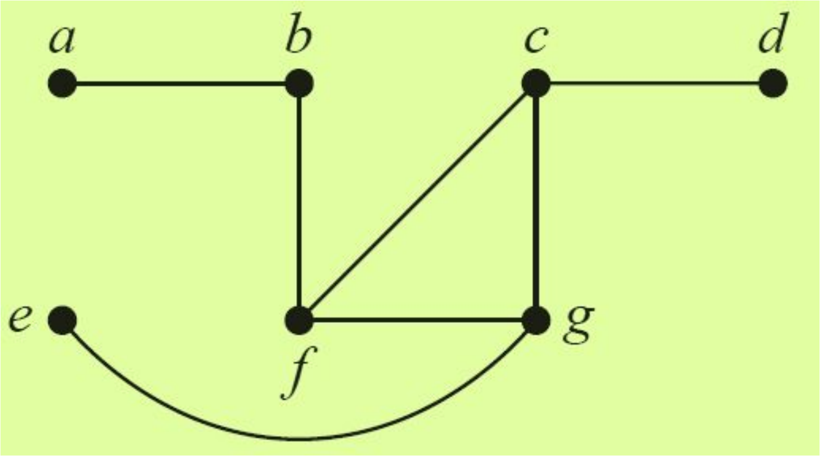


Edge removed: $\{e, f\}$

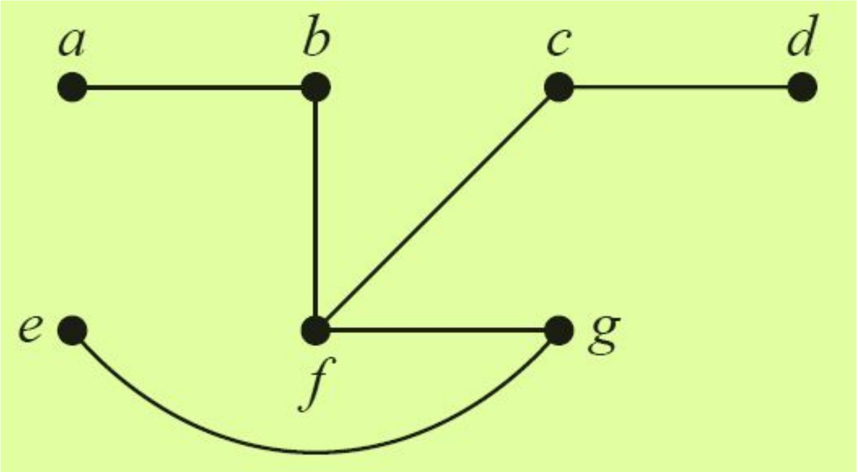


Example 2

Find a spanning tree of the graph Γ shown in the figure.

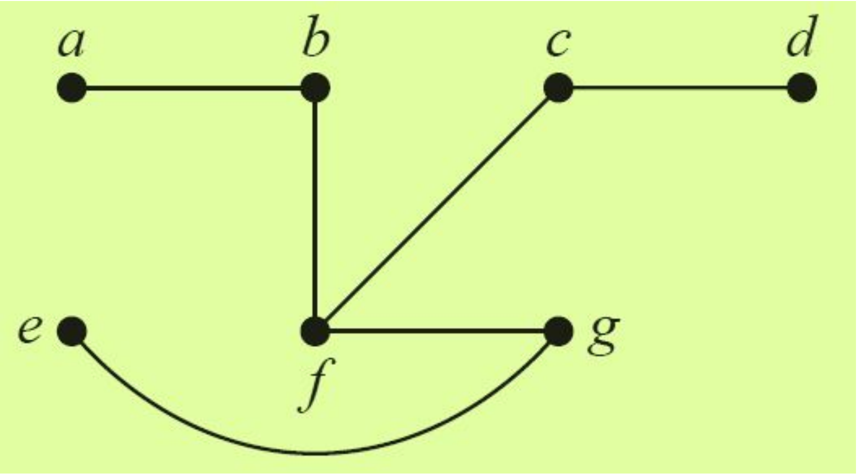


Edge removed: $\{c, g\}$



Example 2

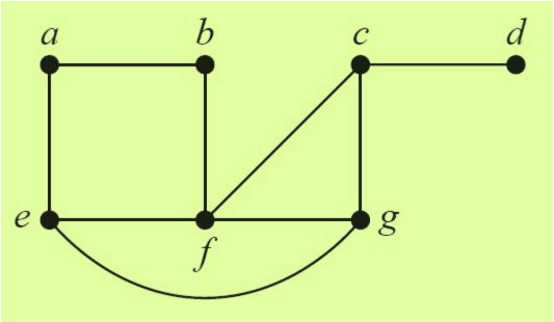
Find a spanning tree of the graph Γ shown in the figure.

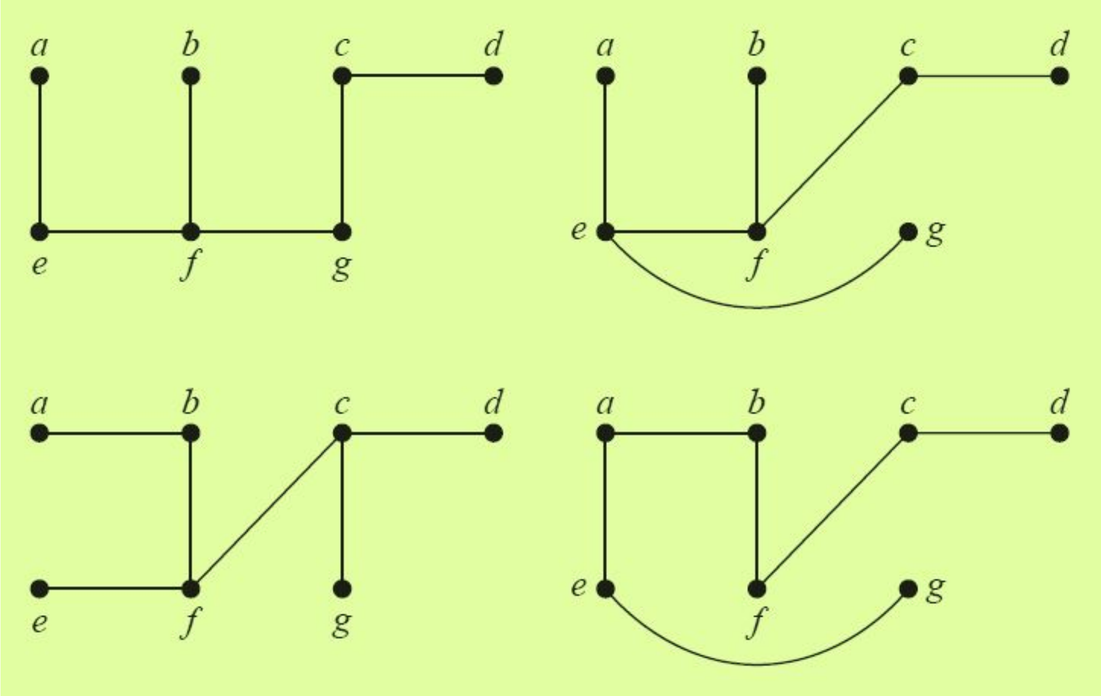
- 

The tree shown in the figure is not the only spanning tree of Γ.

Example 2

Spanning trees for the graph Γ.



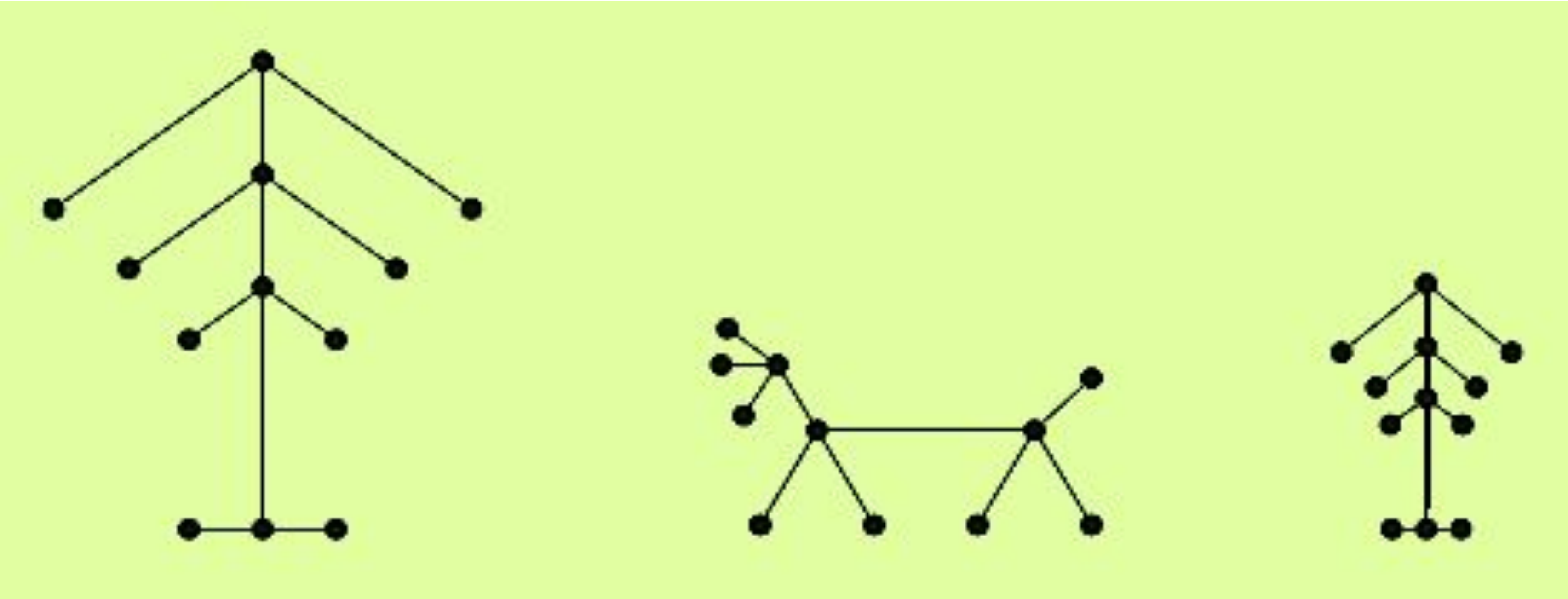


Trees

Definition 3

A **forest** is a (not necessarily connected) graph each of whose components is a tree.

Trees



This is a forest with three connected components.

Trees

Theorem 2

Let T be a tree with vertex set V and edge set E. Then:

a) for every pair of distinct vertices v and w there is a unique path in T connecting them;

b) deleting any edge from T produces a graph with two components each of which is a tree;

c) $|E|=|V|-1$.

Furthermore, a connected graph satisfying any one of these properties is a tree.

? Let $T = (V, E)$ be a tree. Then for every pair of distinct vertices v and w there is a unique path in T connecting them.

Proof

- Let v and w be any two disjoint vertices in T; since T is connected, there exists a path P_1: e_1, e_2, ..., e_n joining v to w.

Suppose that there is another path $P_2 : f_1, f_2, \ldots, f_m$ also joining v to w.

At some point the two paths must diverge; let v^* be the last vertex the two paths have in common before they diverge.

Since the two paths both end at w, they must also converge again; let w^* be the first vertex at which P_1 and P_2 converge.

? Let $T = (V, E)$ be a tree. Then for every pair of distinct vertices v and w there is a unique path in T connecting them.

Proof

- We need to take w^* to be the first vertex at which they converge because two paths may later diverge once more.

Define a path as follows: take those edges of P_1 joining v^* to w^* followed by those edges of P_2 (in reverse order) joining w^* to v^*.

This path joins v^* to itself and repeats no edge; it is a cycle in T.

This is a contradiction since T is a tree. Therefore there is a unique path connecting v to w. ■

? Let $T = (V, E)$ be a tree. Then deleting any edge from T produces a graph with two components each of which is a tree.

Proof

- Let e be any edge in T joining vertices v and w, and let Γ is the graph obtained by removing e from T.

Since e is itself the unique path in T joining v to w, there is no path in Γ connecting v and w; thus Γ is not connected.

? Let $T = (V, E)$ be a tree. Then deleting any edge from T produces a graph with two components each of which is a tree.

Proof

Let V_1 be the set of vertices of Γ which can be joined by a path (in Γ) to v, and let V_2 be the set of vertices of Γ which can be joined by a path to w.

Then $V_1 \cup V_2 = V$ and V_1 and V_2 define two connected subgraphs of Γ. (Exercise: prove this last statement.)

Each of these components of Γ must be a tree because any cycle in one of them would also be a cycle in T. ■

? Let Γ be a connected graph and for every pair of distinct vertices v and w there is a unique path in Γ connecting them. Then Γ is a tree.

Proof

If there is a cycle in Γ containing a pair of distinct vertices v and w then this cycle provides two distinct paths connecting v and w.

Since this contradicts $a)$, there is no such cycle.

? Let Γ be a connected graph and for every pair of distinct vertices v and w there is a unique path in Γ connecting them. Then Γ is a tree.

Proof

- There can also be no loops (cycles connecting only one vertex) in Γ.

If e is a loop at vertex v, and w is any other vertex, then there are two distinct paths connecting v and w: one path which begins with e and one which does not.

Therefore Γ contains no cycles at all and so is a tree. ■

? Let Γ be a connected graph and deleting any edge from T produces a graph with two components each of which is a tree. Then Γ is a tree.

Proof

- If Γ contains a cycle, then we could delete an edge of the cycle without disconnecting Γ, contradicting b). Therefore Γ must contain no cycles at all and so is a tree. ■

? Let $T = (V, E)$ be a tree. Then $|E| = |V| - 1$.

Proof

- The proof is by induction on the number of vertices of *T* and uses part b).

Exercise: prove the statement. ■

? Let Γ be a connected graph and $|E|=|V|-1$. Then Γ is a tree.

Proof

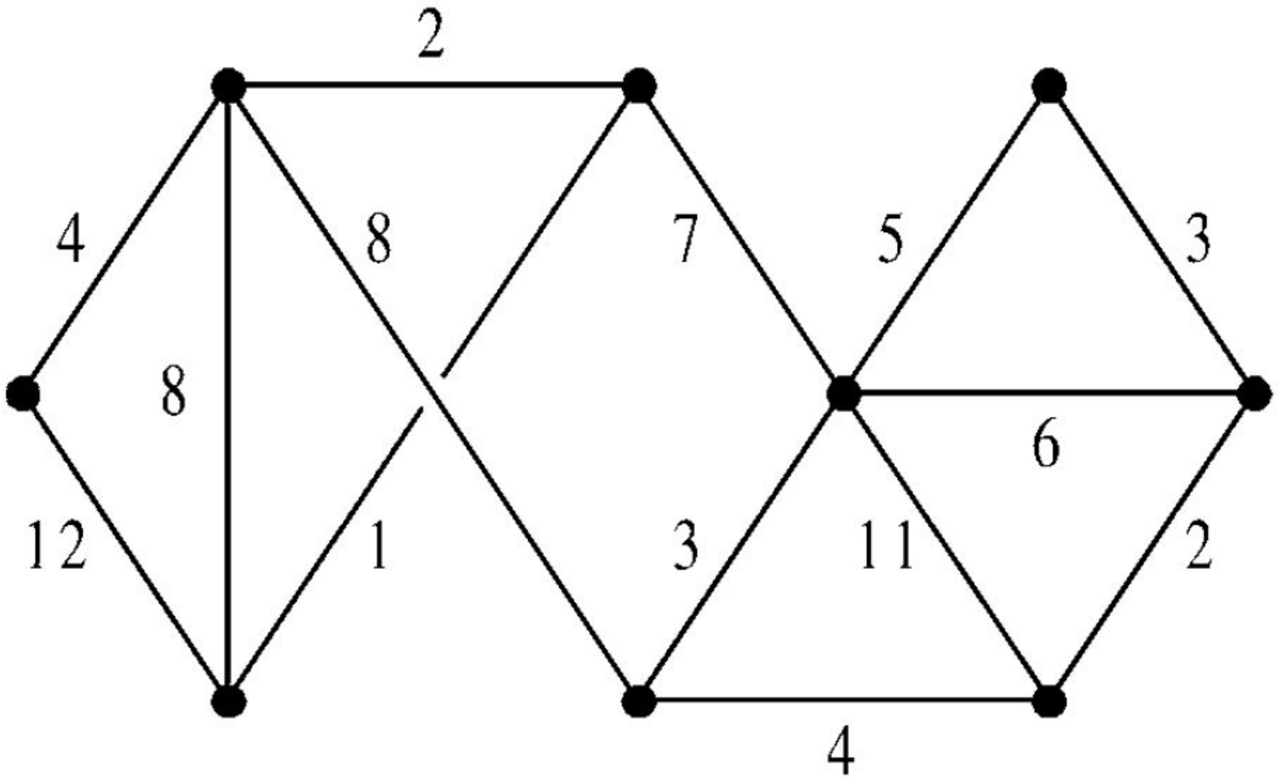
- Exercise: prove this statement. ■

Weighted graphs

-

Weighted graphs

For example, the figure is a diagram of a weighted graph.



Weighted graphs

•

Weighted graphs

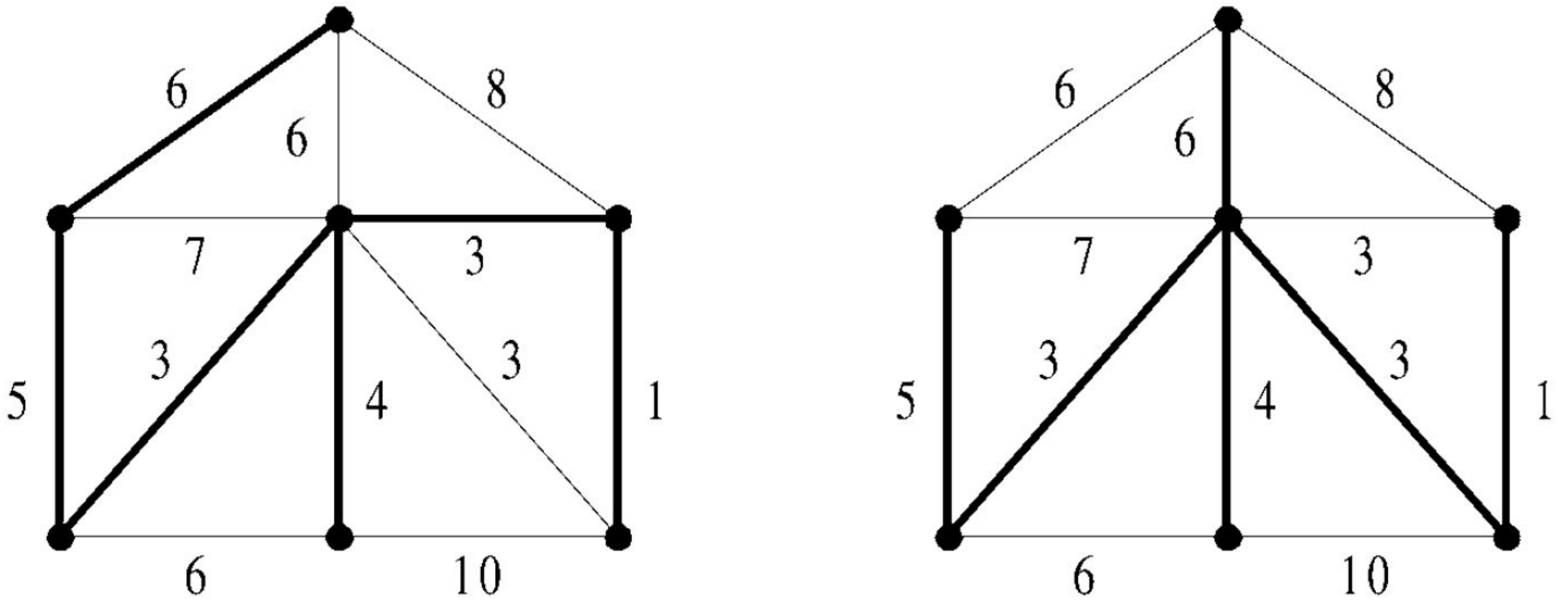
•

Weighted graphs

•

Weighted graphs

The figure shows a weighted graph with two minimal spanning trees, both of weight 22.

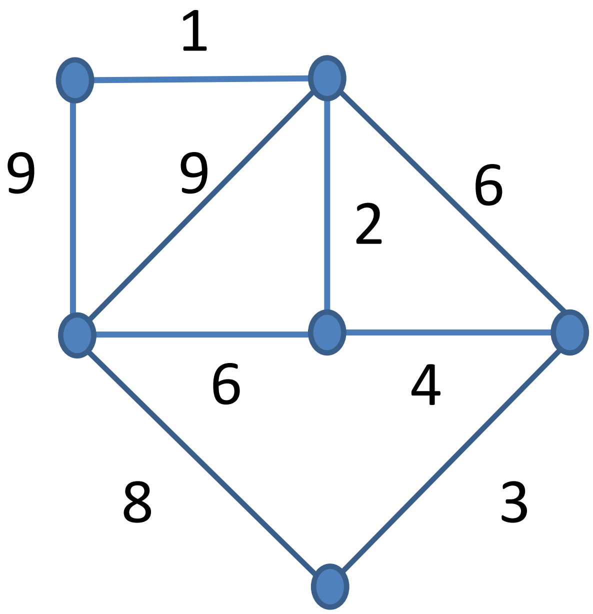


An algorithm which produces a minimal spanning tree (Prim's algorithm)

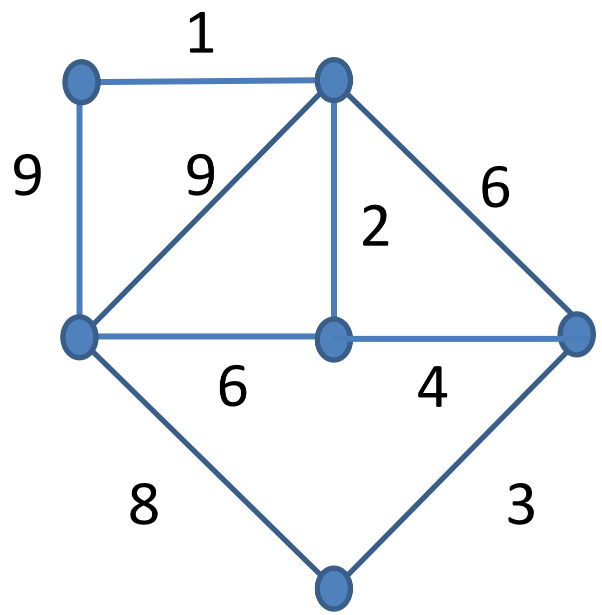
-

The construction of a minimal spanning tree using Prim's algorithm

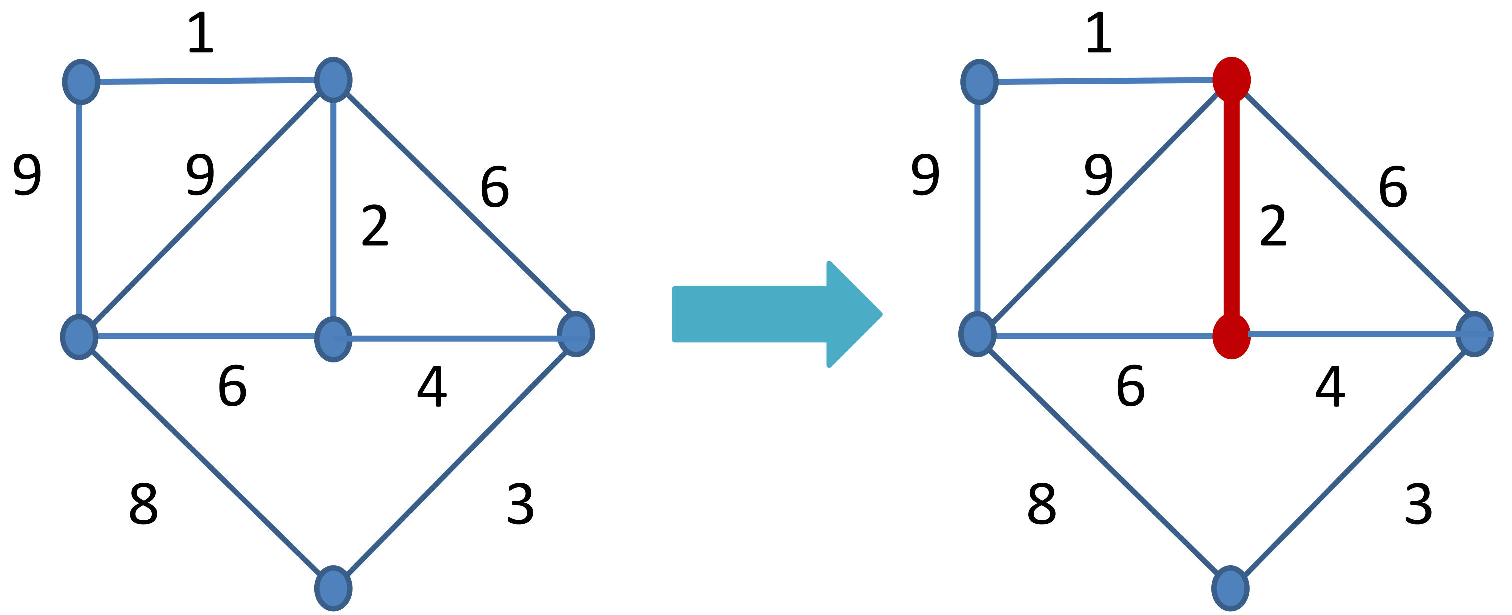
-



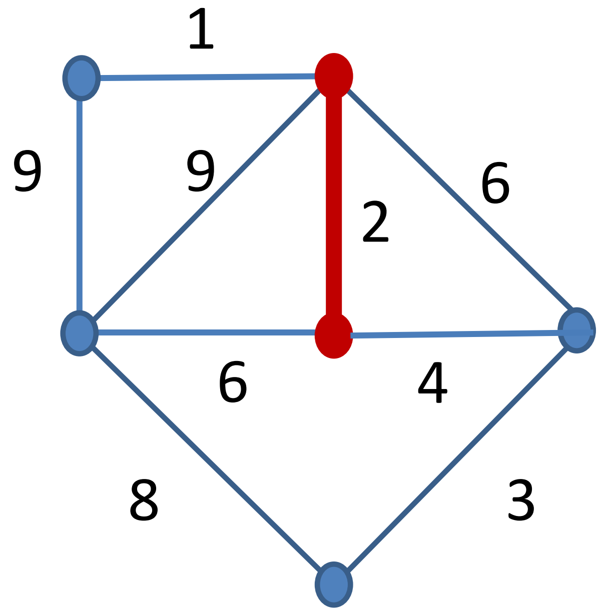
The construction of a minimal spanning tree using Prim's algorithm



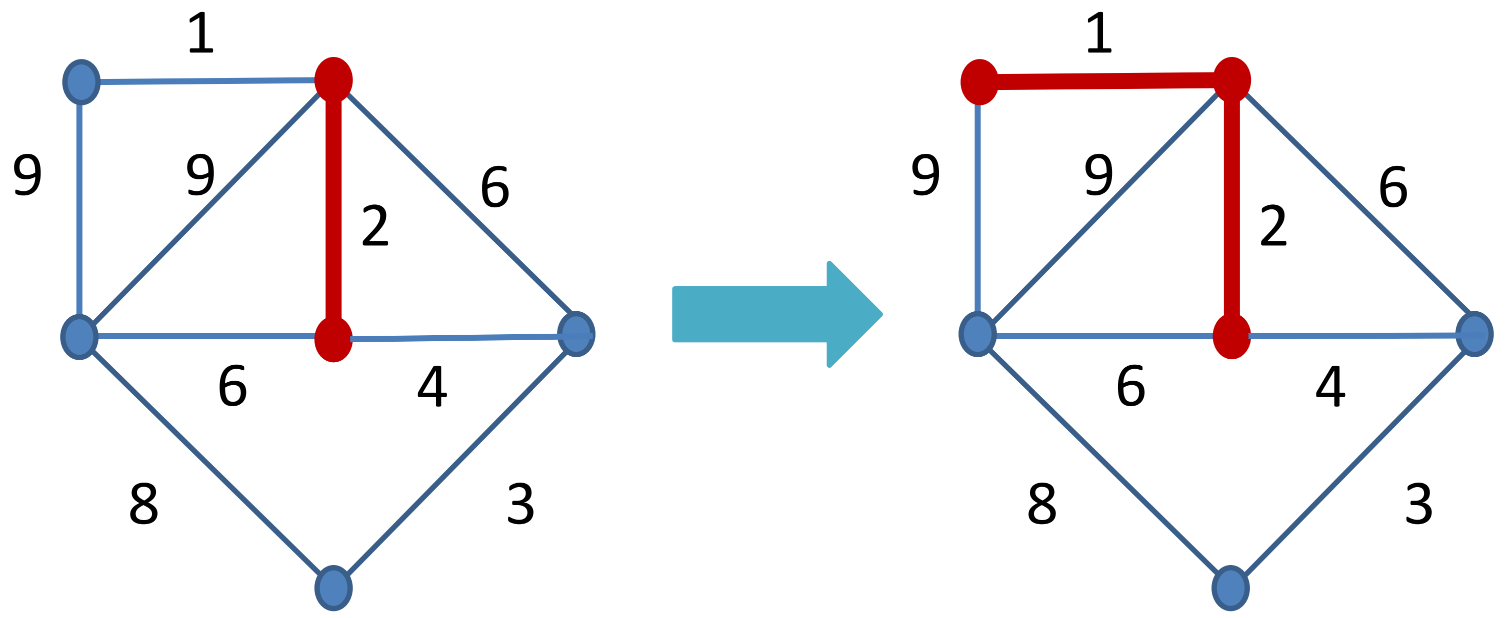
The construction of a minimal spanning tree using Prim's algorithm



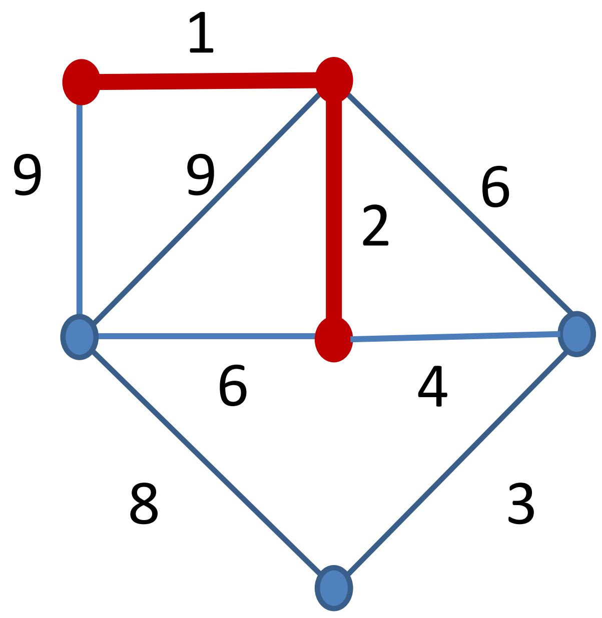
The construction of a minimal spanning tree using Prim's algorithm



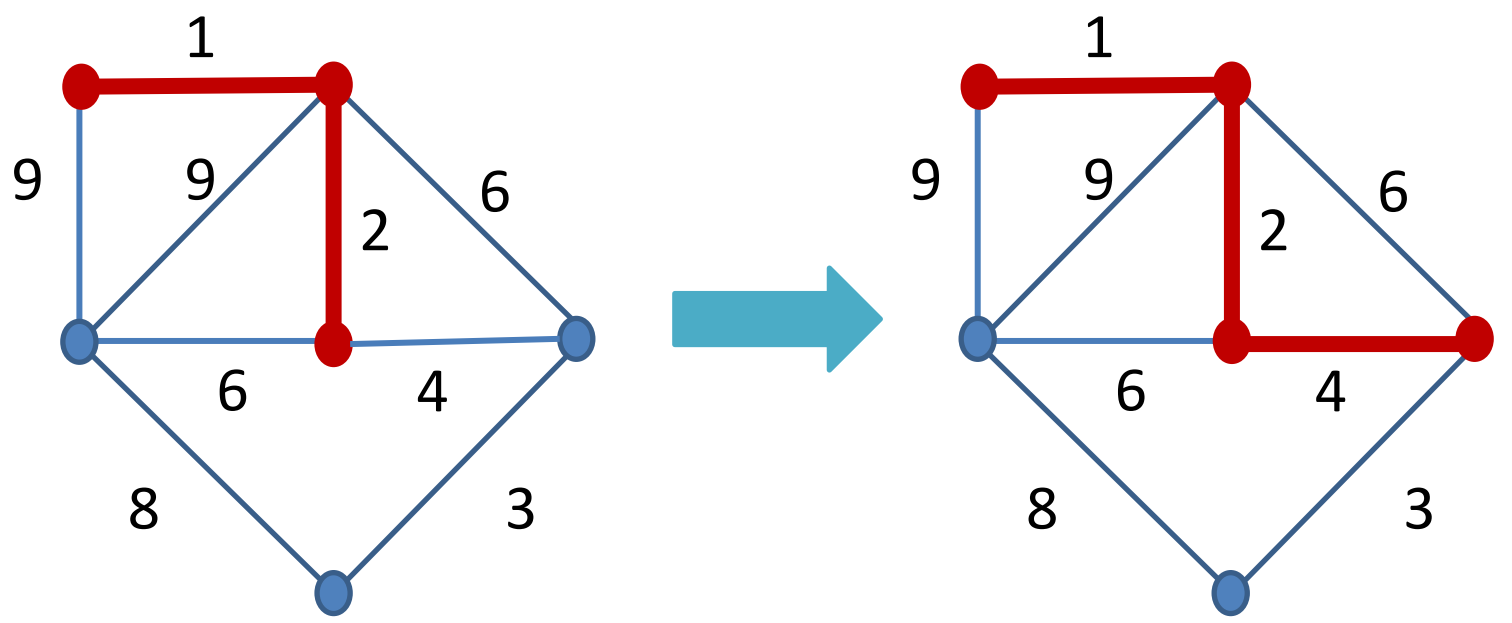
The construction of a minimal spanning tree using Prim's algorithm



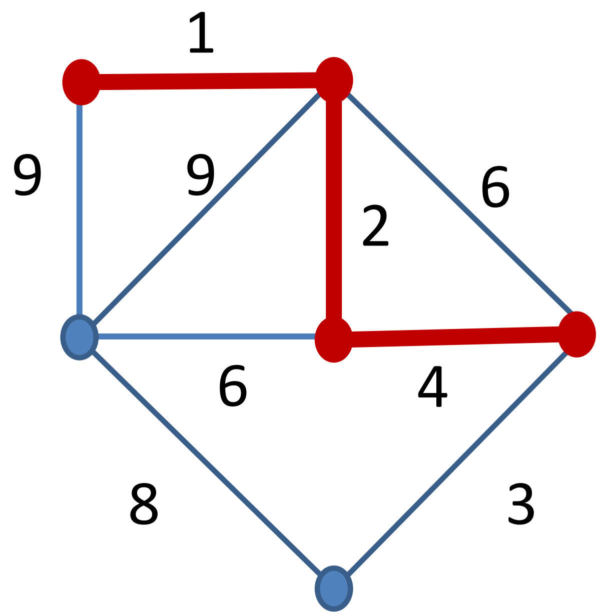
The construction of a minimal spanning tree using Prim's algorithm



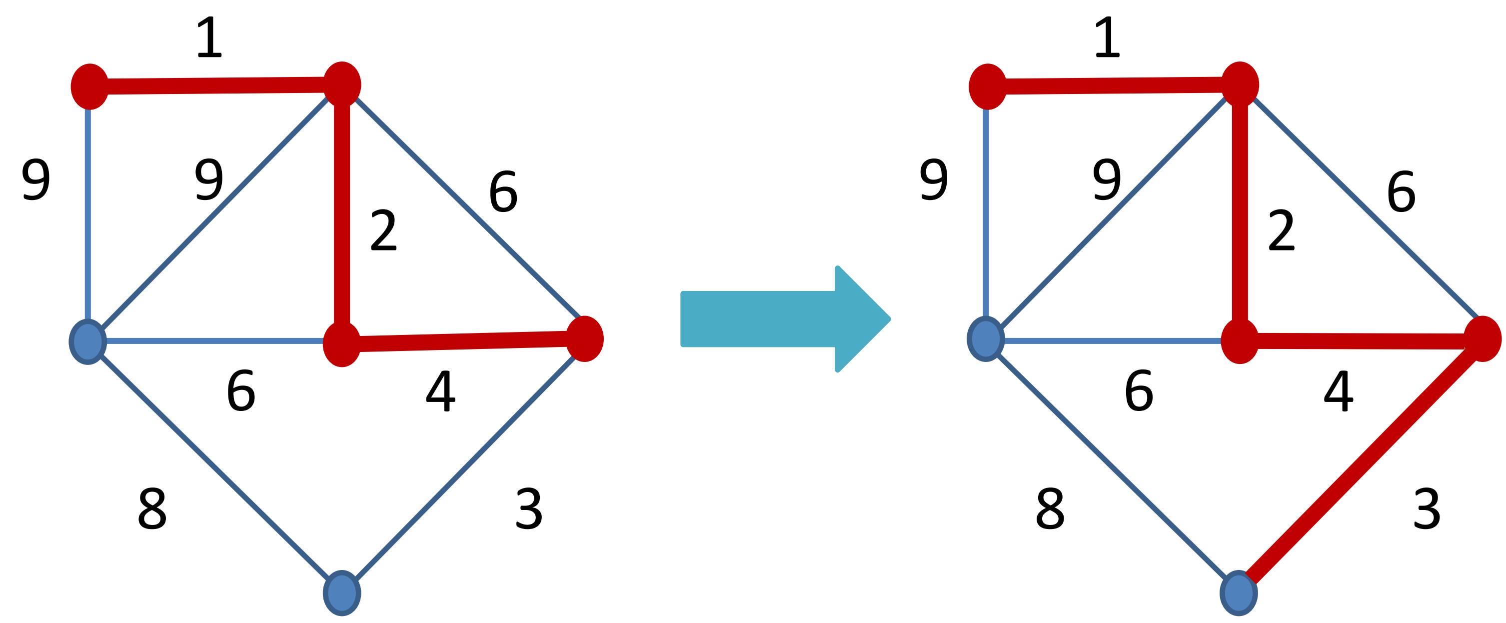
The construction of a minimal spanning tree using Prim's algorithm



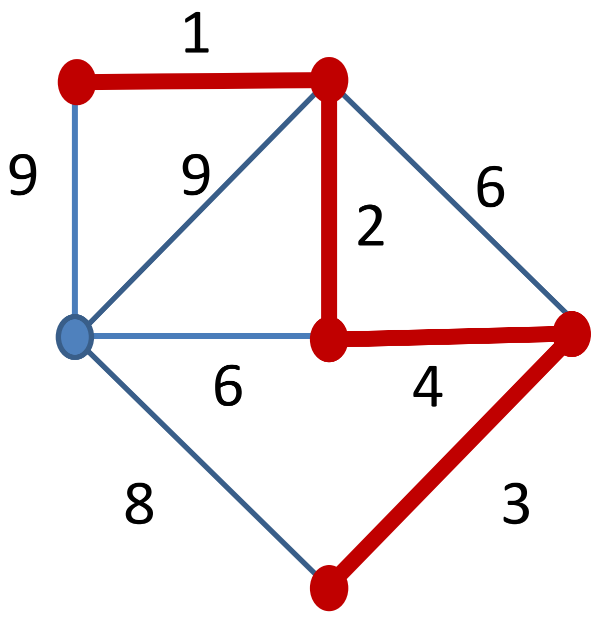
The construction of a minimal spanning tree using Prim's algorithm



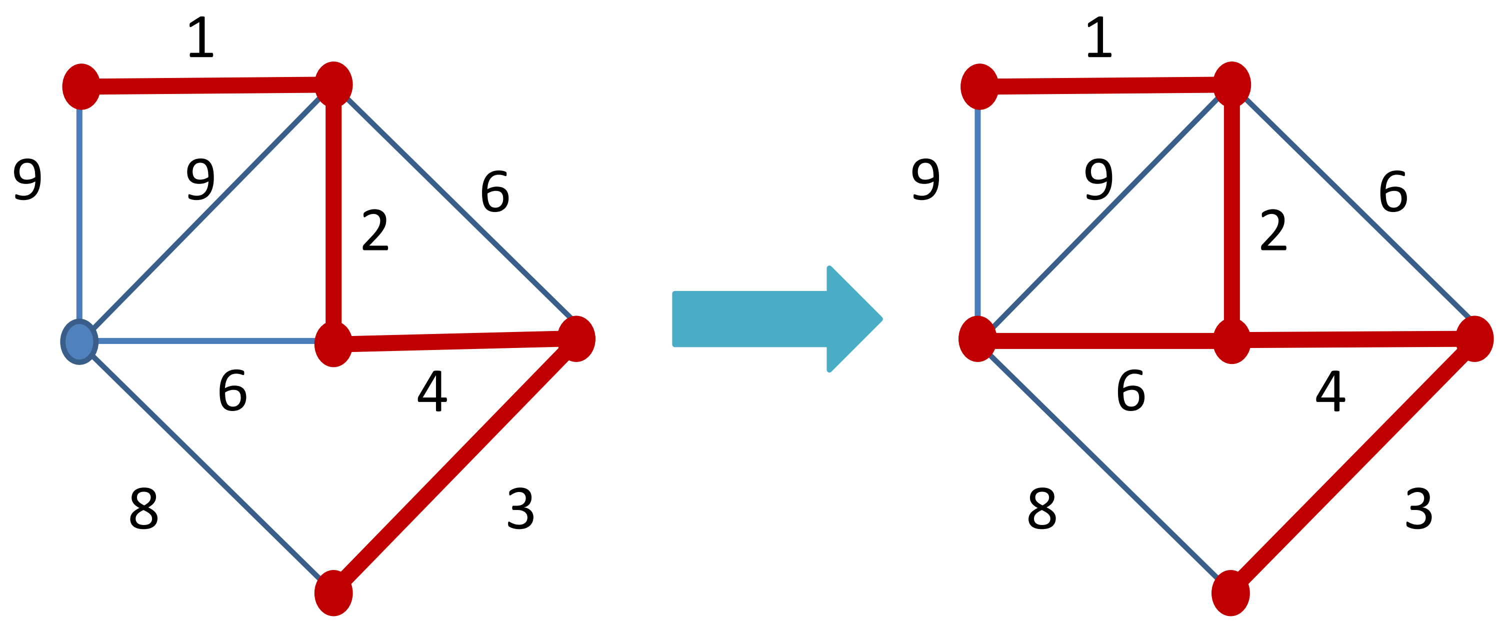
The construction of a minimal spanning tree using Prim's algorithm



The construction of a minimal spanning tree using Prim's algorithm

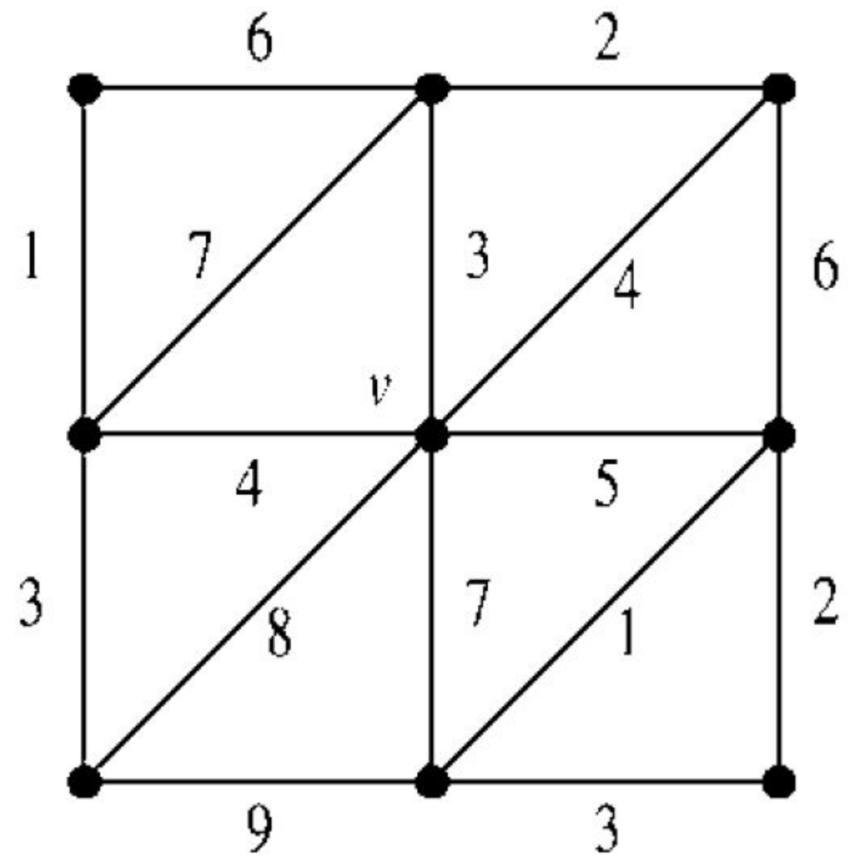


The construction of a minimal spanning tree using Prim's algorithm



Prim’s algorithm

•



Rooted trees

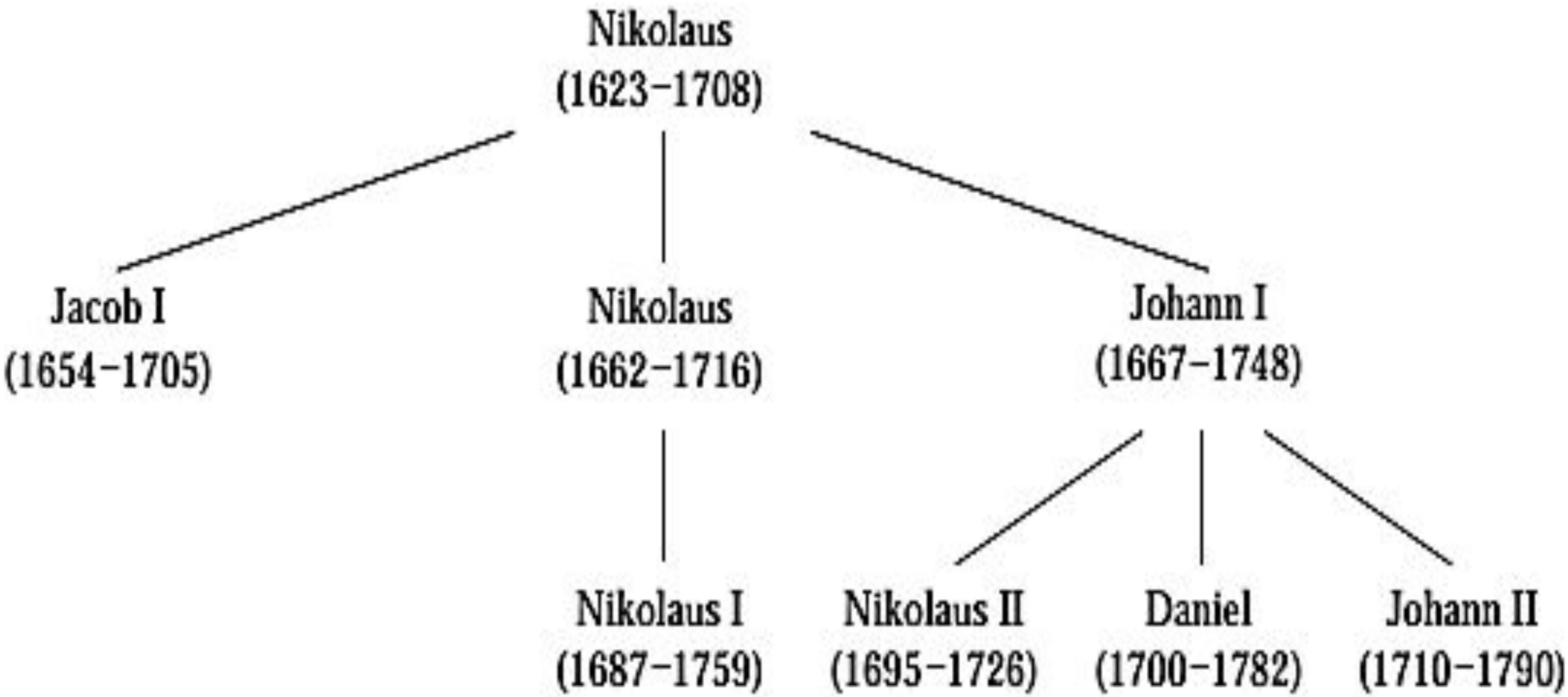
Many of the applications of graph theory, particularly in computing, use a certain kind of tree, called a ‘rooted tree’.

This is simply a tree where a particular vertex has been distinguished or singled out from the rest.

Rooted trees

For example, family tree are graphs that represent genealogical charts.

The family tree of the male members of the Bernoulli family of Swiss mathematicians is shown in the figure.



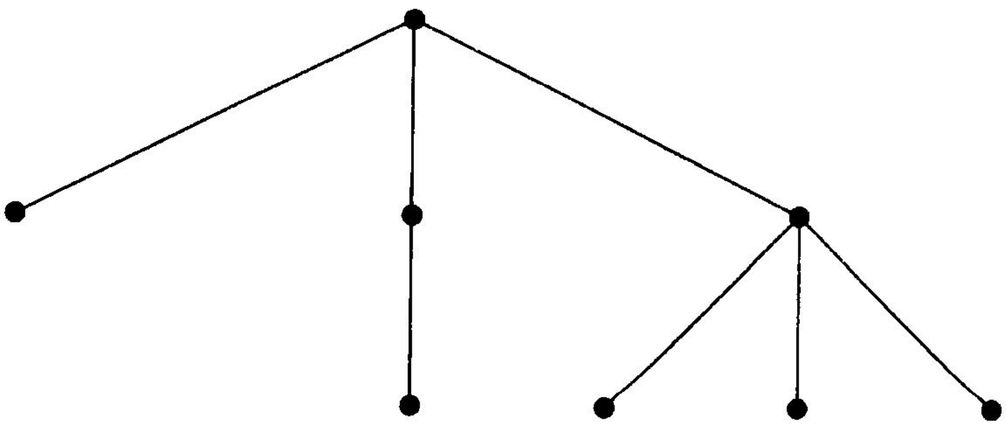
Rooted trees

Family trees use vertices to represent the members of a family and edges to represent parent – child relationships.

The undirected graph representing a family tree (restricted to people of just one gender and with no inbreeding) is an example of a rooted tree.

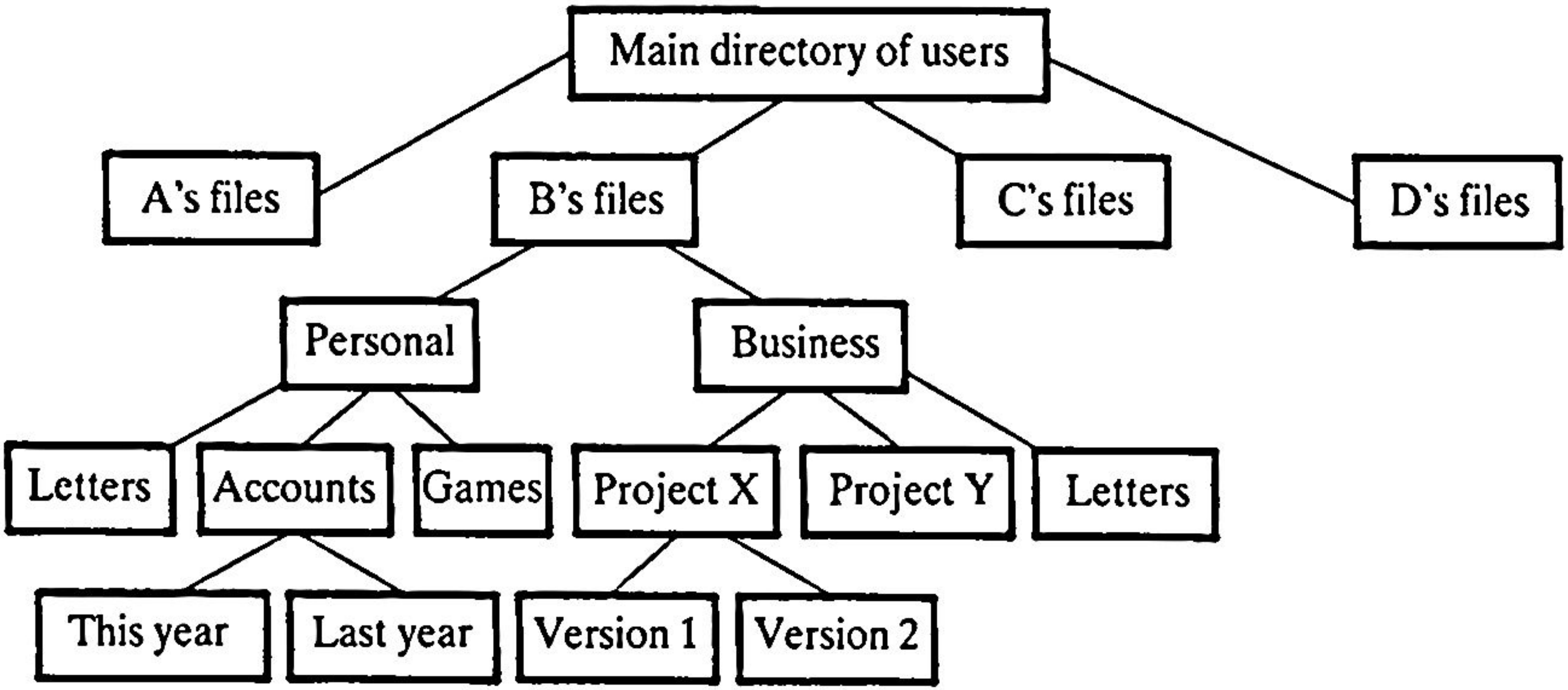
Rooted trees

The undirected graph representing the family tree of the male members of the Bernoulli family of Swiss mathematicians is an example of a rooted tree.



Rooted trees

Rooted trees are perhaps most familiar in computing as models for the structure of file directories.



Rooted trees

Some of the other important uses of rooted trees in computing include the representation of data and the representation of algebraic expressions.

Rooted trees

Definition 1

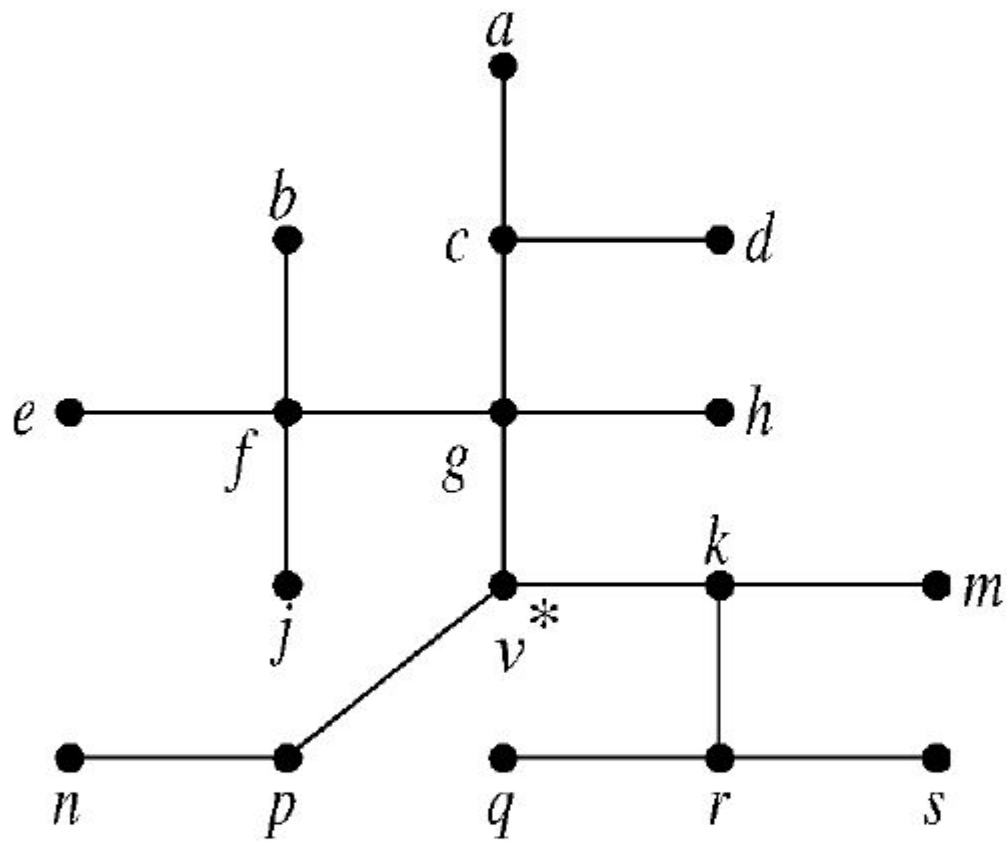
A **rooted tree** is a pair (T, v^*) where T is a tree and $v^* \in V_T$.

The distinguished vertex v^* is called the **root** of the tree.

A **leaf** in a rooted tree is a vertex which has degree 1 which is not equal to the root; an **internal vertex** is a vertex which is neither the root nor a leaf.

Rooted trees

- The figure shows a tree T with root v^*.



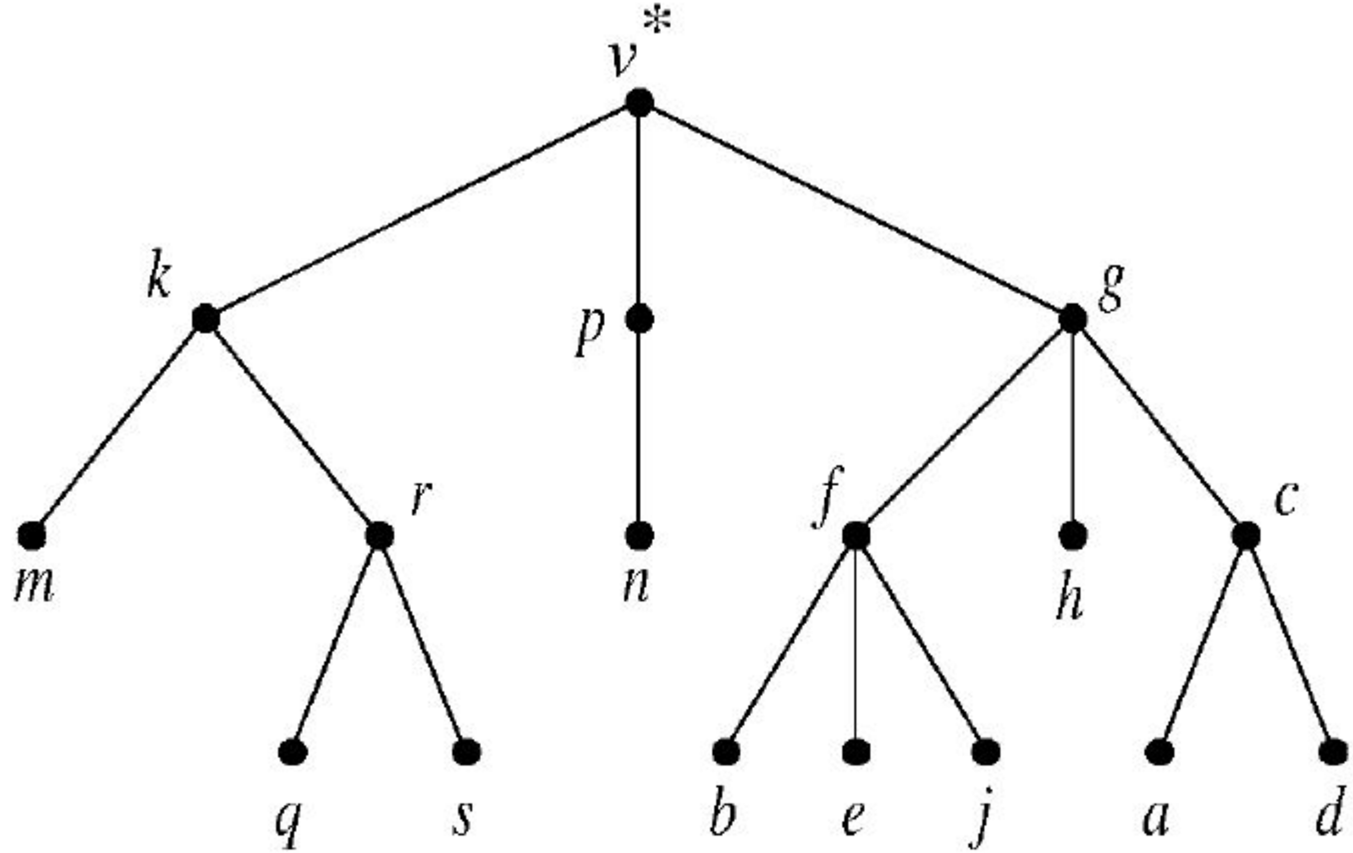
Rooted trees

- Unlike in nature, it is usual to draw the diagram of a rooted tree so that it 'grows downwards' with the root at the top of the diagram.

The diagram of T is redrawn in the following figure growing downwards with the root at the top.

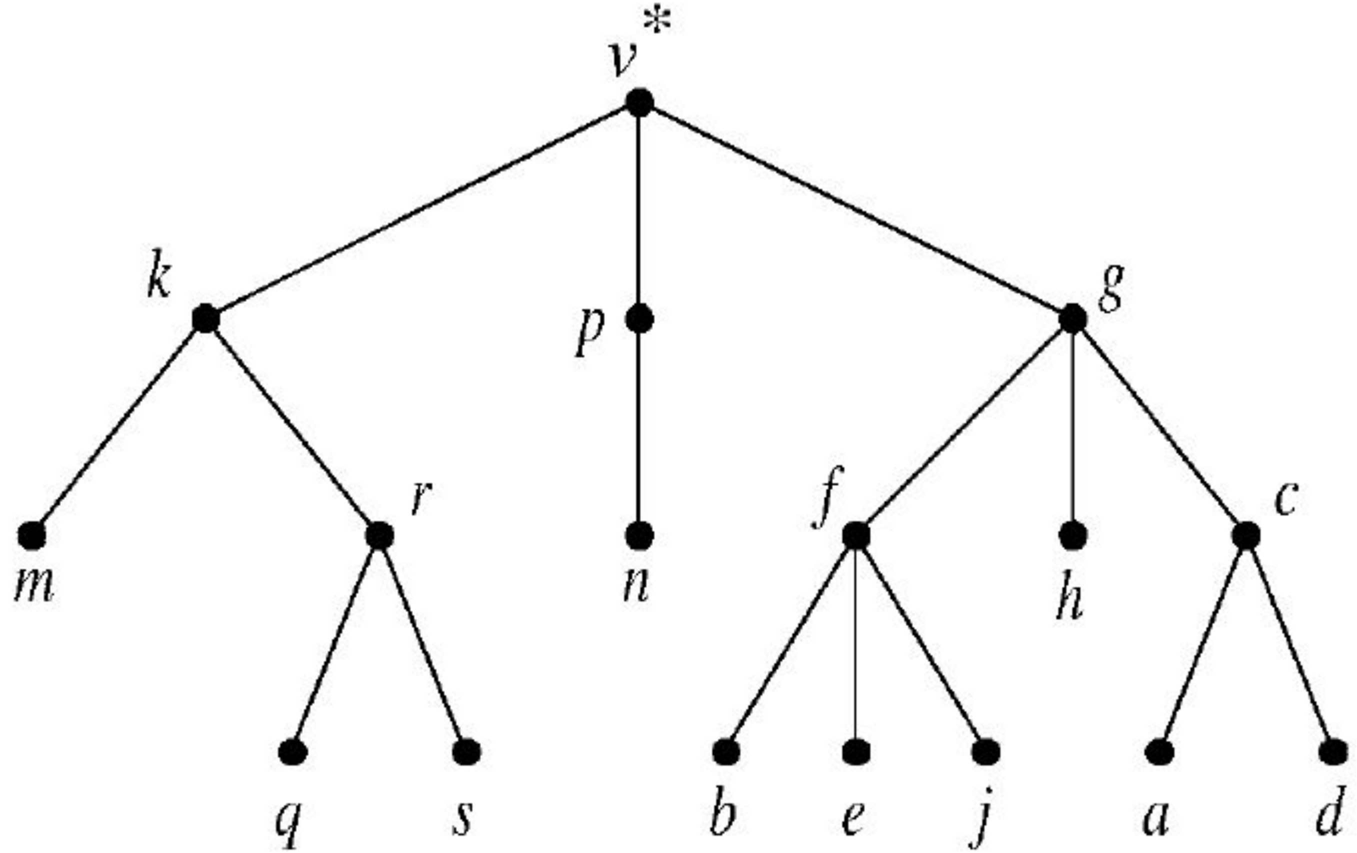
Rooted trees

- The figure shows a tree T with root v^* at the top of the diagram.



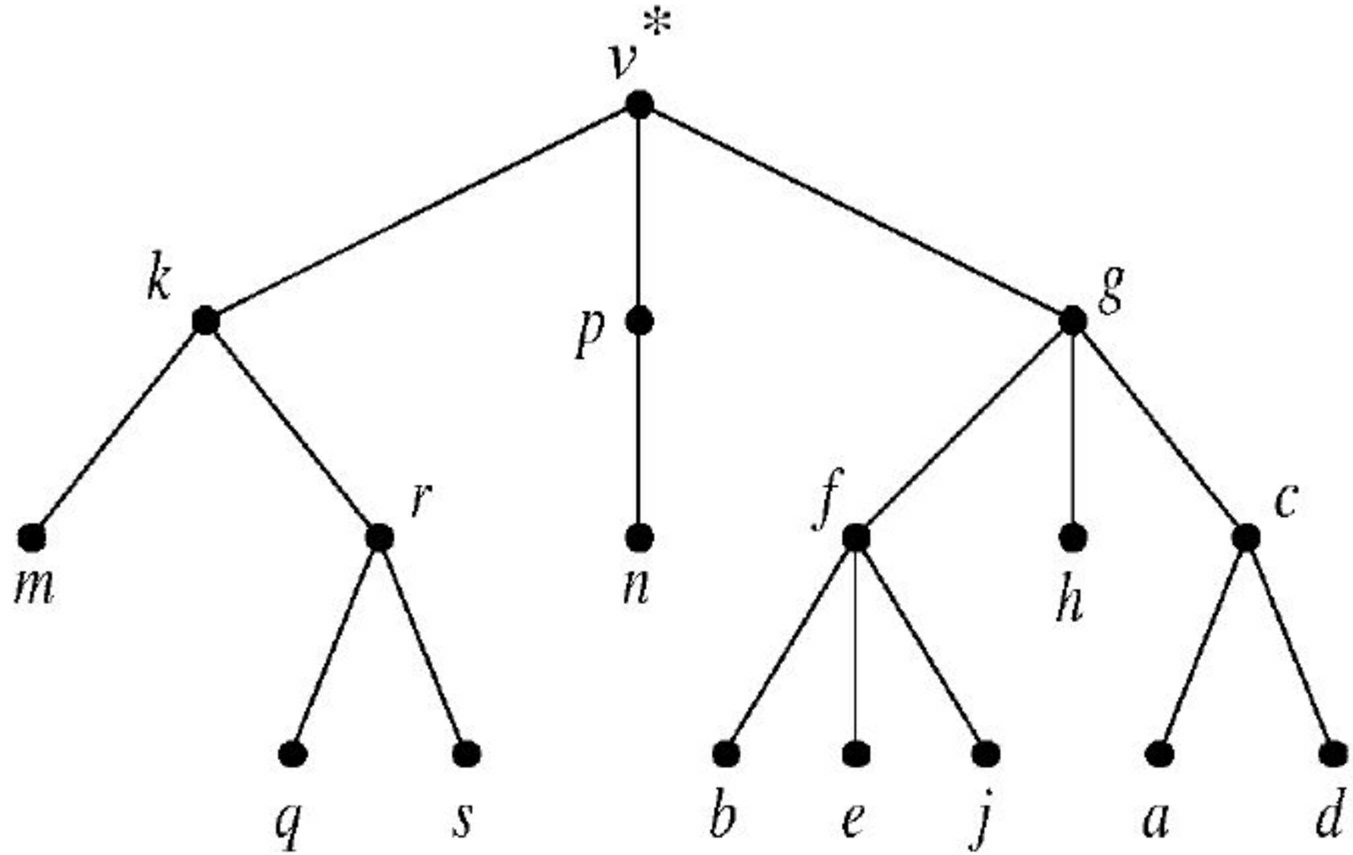
Rooted trees

- The choice of the root in this example is arbitrary; we could equally well have chosen the vertex m as the root.



Rooted trees

- The leaf vertices are $a, b, d, e, h, j, m, n, q$ and s, and the internal vertices are c, f, g, k, p and r.



Rooted trees

Definition 2

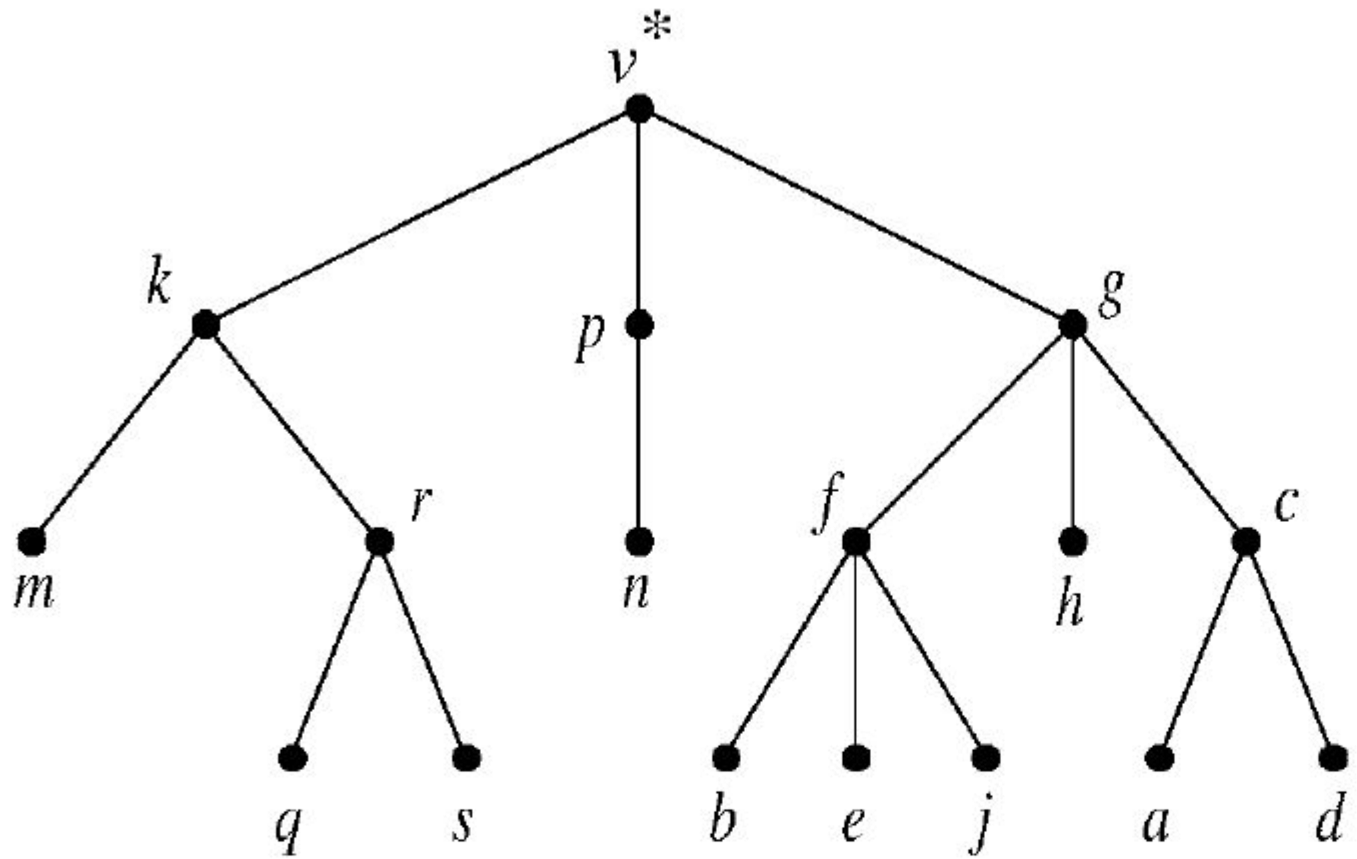
Let (T, v^*) be a rooted tree.

The **level** of a vertex w of T is the length of the (unique) path in T from v^* to w.

The **height** of T is the maximum of the levels of its vertices.

Rooted trees

Determine the level of each vertex of the following rooted tree.



Rooted trees

Definition 3

Let (T, v^*) be a rooted tree and let p be a vertex of level $k > 0$.

The (unique) vertex q of level $k-1$ which is adjacent to p is called the **parent** of p.

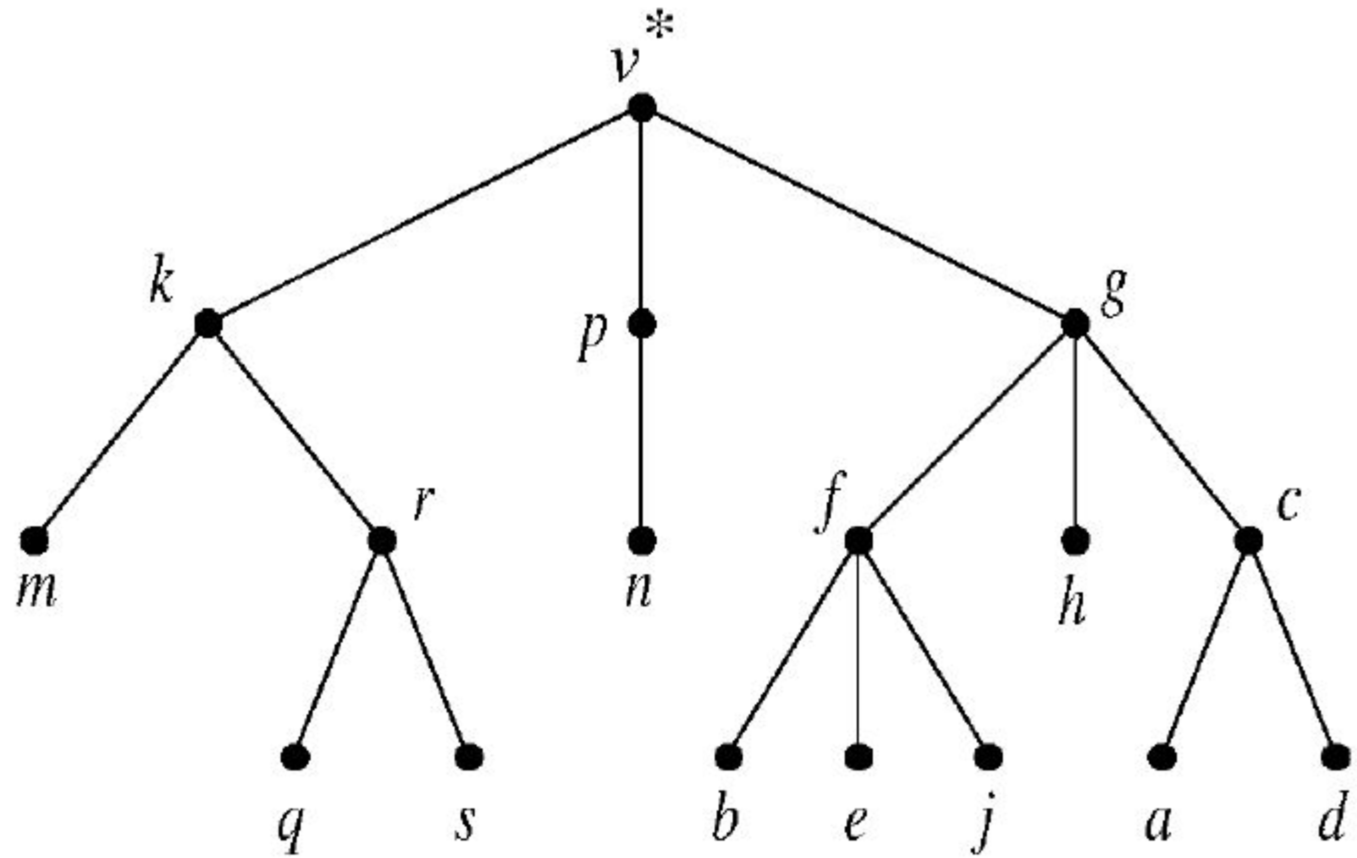
Similarly, p is the **child** of q, and any vertex of level k which is also adjacent to q is called a **sibling** of p.

Rooted trees

It is clearly possible to define further terms such as **grandparent**, **grandchild**, **ancestor**, **descendant**, etc.

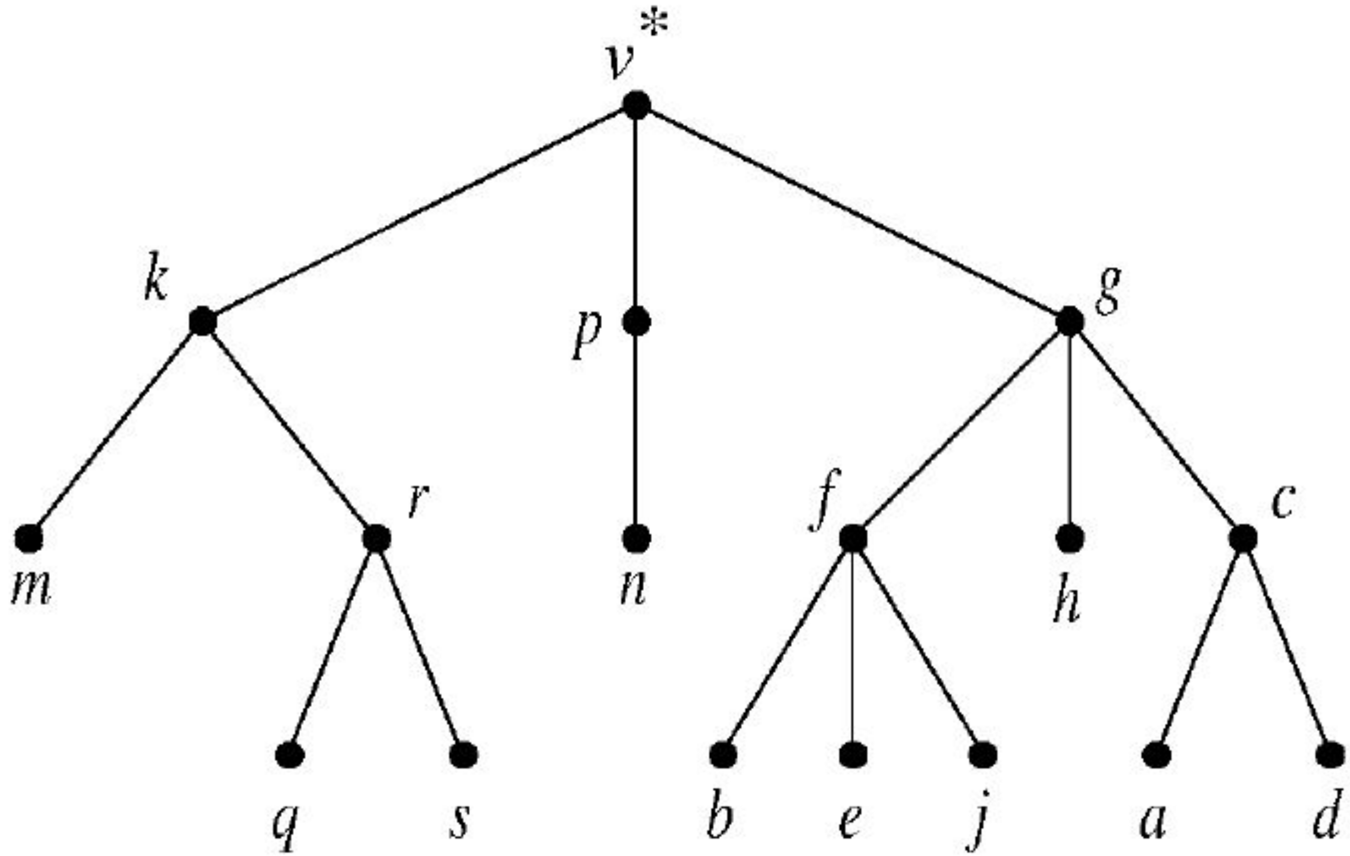
Rooted trees

- The rooted tree shown in the figure has height 3. Also, g is the parent of c, f and h; r is the parent of q and s, and so on.



Rooted trees

- Similarly b, e and j are siblings, a, b, d, e and j are all grandchildren of g, etc.



Rooted trees

Rooted trees can be used as models in such diverse areas as

- computer science,
- biology,
- management.

Rooted trees

Of particular importance in applications in computing are **binary rooted trees**.

A binary rooted tree has a property that each vertex has at most two children.

In a binary rooted tree the two subtrees of a vertex are referred to as the **left-subtree** and the **right-subtree** of the vertex.

If a vertex has no left child its left-subtree is said to be the **null tree** (i.e. a tree with no vertices).

Similarly, if a vertex has no right child its right-subtree is said to be the **null tree**.

Rooted trees

- These ideas can be put to use giving the following recursive definition of a binary tree.

Definition 4

A **binary tree** comprises a triple of sets (L, R, S) where L and R are binary trees (or are empty) and S is a singleton set.

The single element of S is the **root**, and L and R are called, respectively, the **left** and **right subtrees** of the root.

Rooted trees

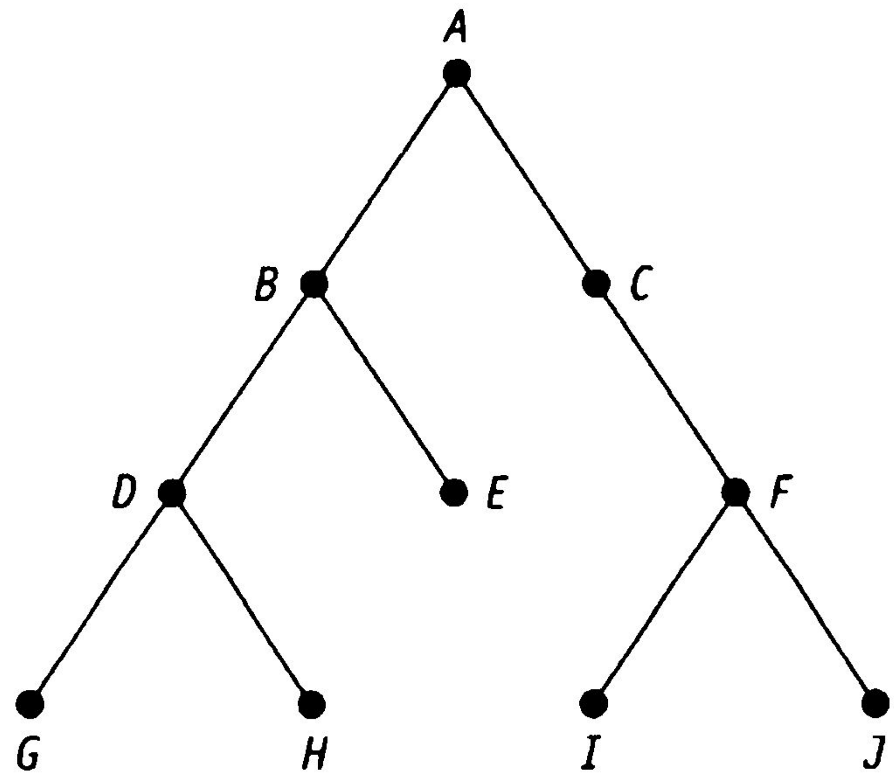
- This definition is recursive because it defines a binary tree in terms of the 'components' L, S and R, two of which are themselves binary trees.

Thus L and R, if non-empty, are both defined as triples of the form (L', S', R') and soon.

This way of defining binary trees is extremely useful for their computer representation.

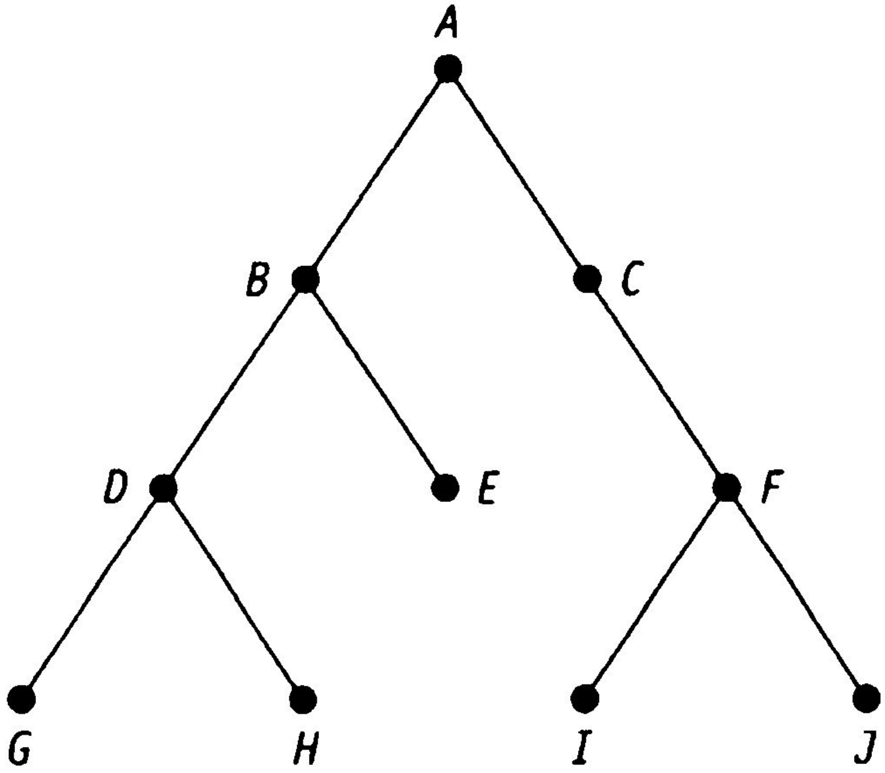
Rooted trees

T is a binary rooted tree.



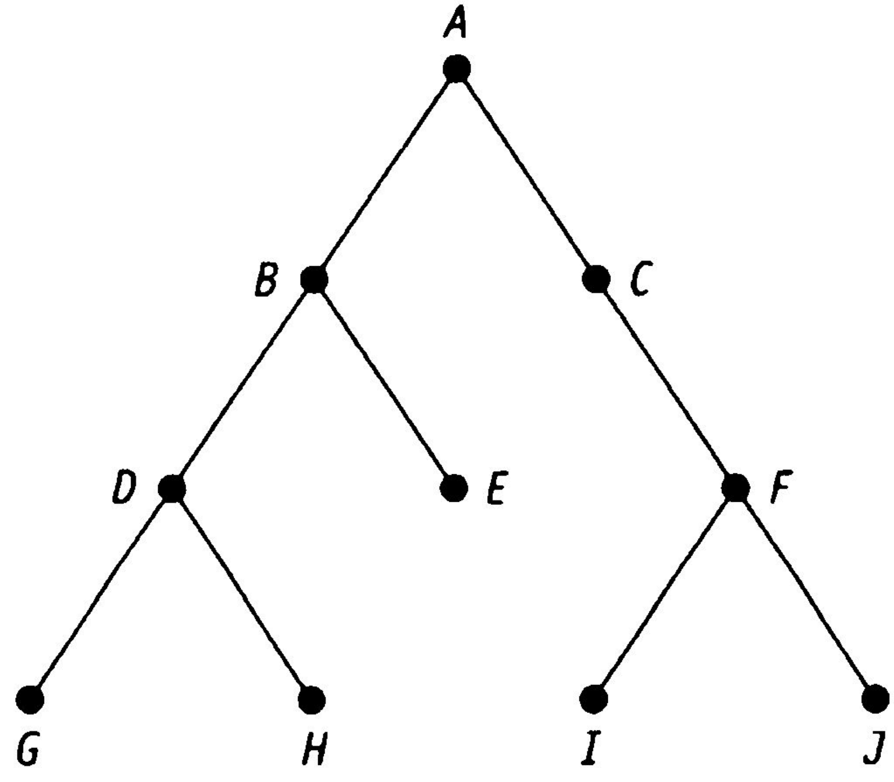
Rooted trees

- Find the root of T.



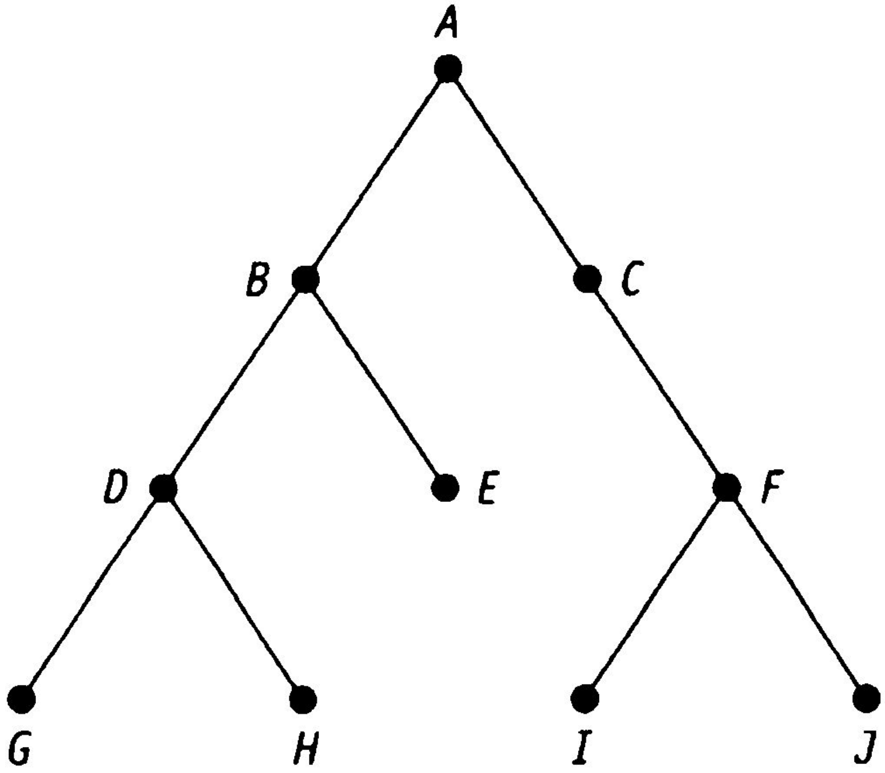
Rooted trees

- Find the root of the left-subtree of vertex B.



Rooted trees

- Find the leaves of T.



Rooted trees

- Find the children of vertex C.