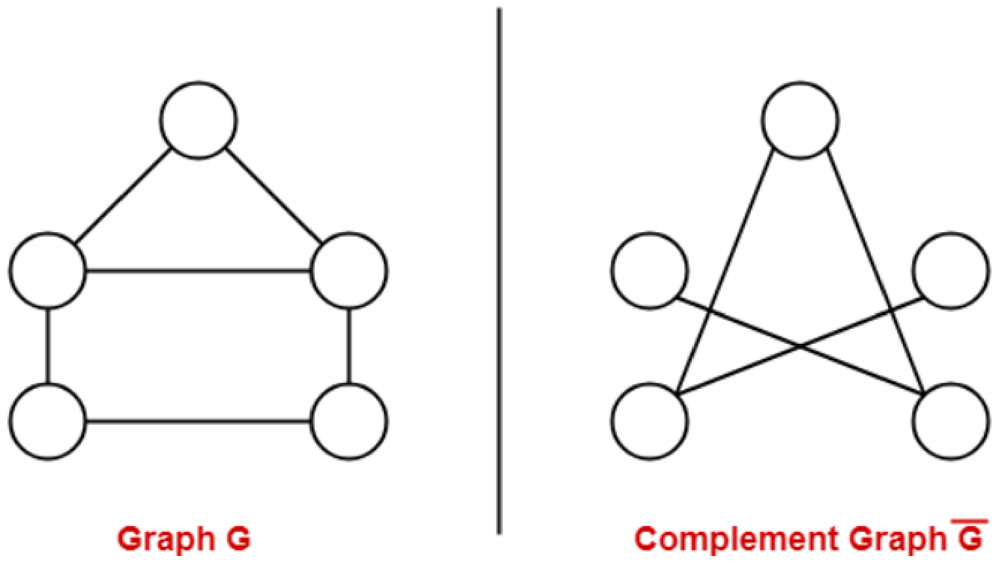
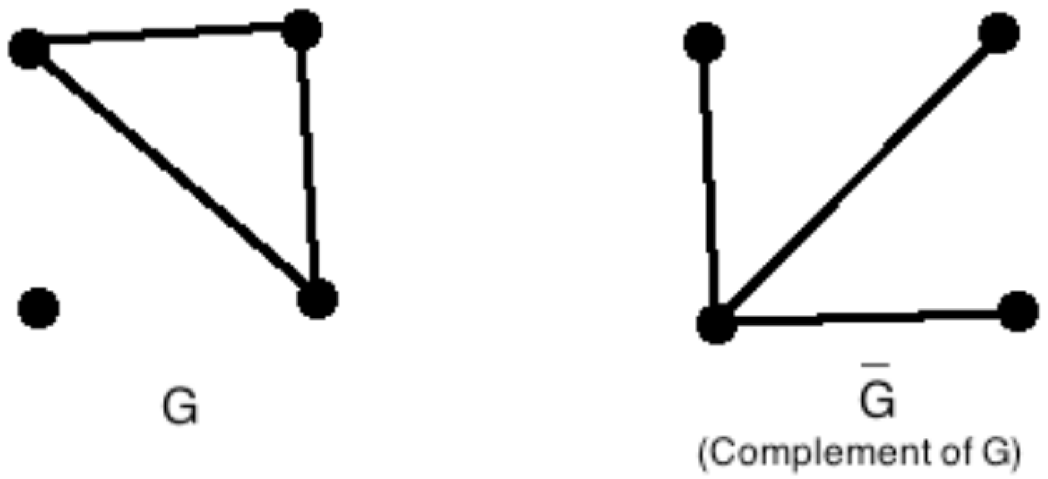
The complement of a graph G, sometimes called the edge-complement, is the graph G', some times denoted $\overline{G}$, with the same vertex set but whose edge set consists of the edges not present in G.

Complement of a simple graph G is a simple graph G' with all the vertices of G in which there is an edge between two vertices v and w if and only if there exist no edge between v and w in the original graph G.

Example:



Essentially, if a graph G is on n-vertices then the complement $\bar{G}$ is the complete graph K_n with all of the edges in G deleted. We can see this very clearly in the following example showing both the graph G and its complement:



Since the vertex set for $\bar{G}$ is the same as G, the number of vertices in the complement of G is the same as that in G, that is:

V|G| = **V|G'|**

Number of vertices in G = Number of vertices in G'

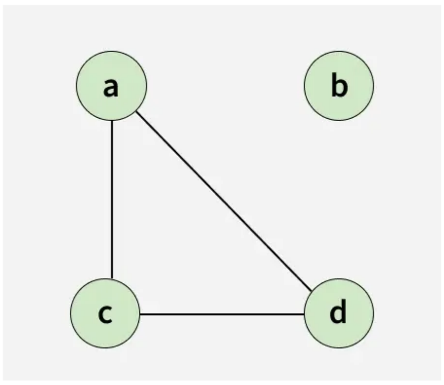
Thus, the complement of a graph G (V, E) is denoted as G' (V, E'), where:

- V is the set of vertices (remains unchanged),
- E' is the set of edges in the complement graph.

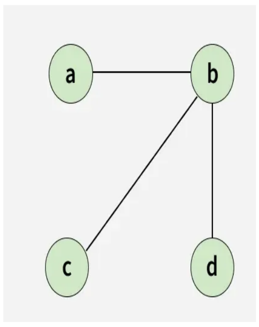
Key Point:

The number of vertices in the complement of a graph remains unchanged. Only the edges differ.

Example:

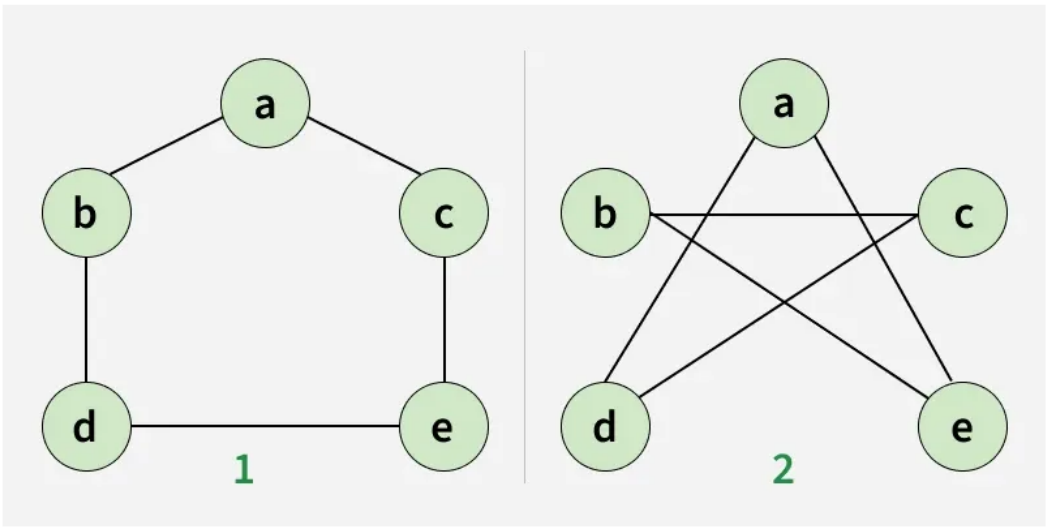


Complement of this graph :

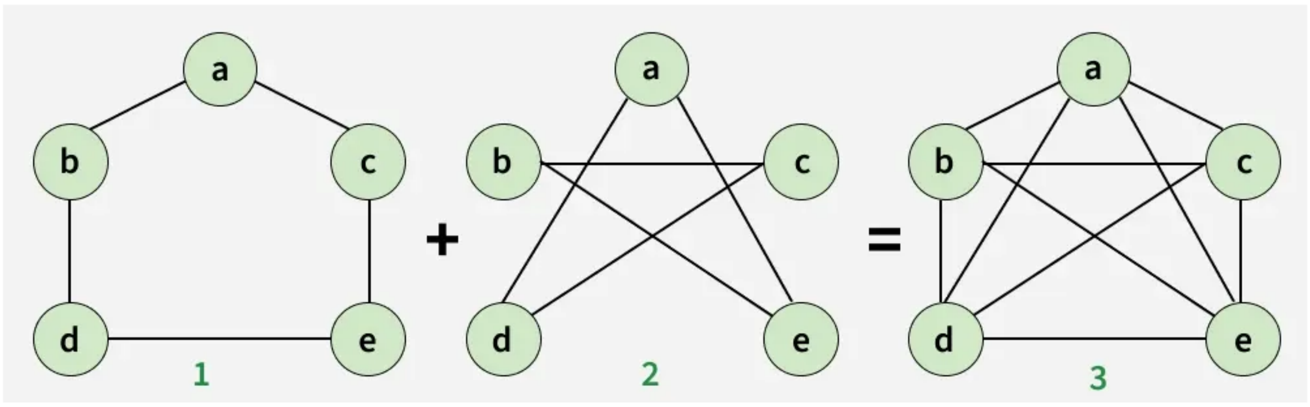


Properties of Complement of Graph

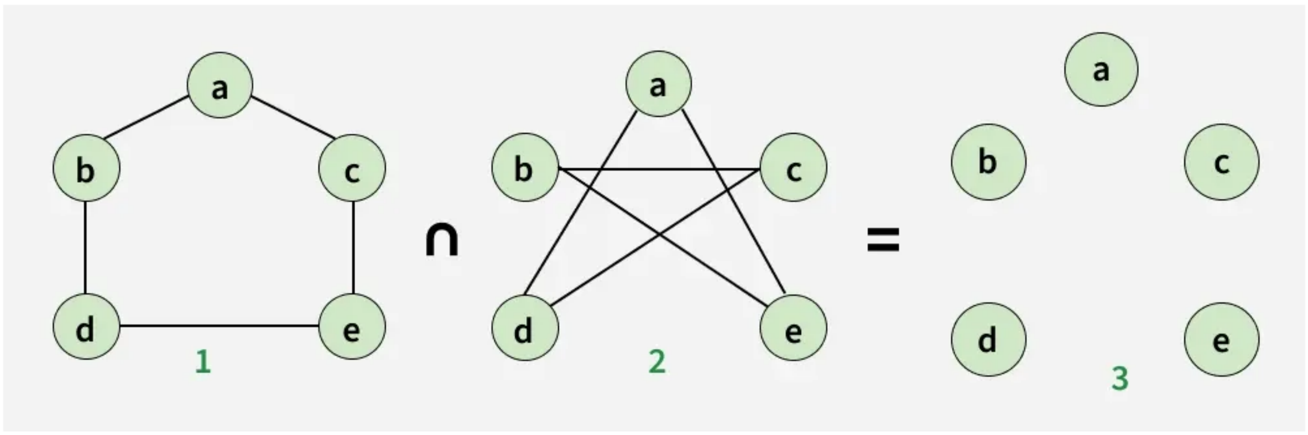
1. If E be the set of edges of graph G' then **E(G')={ (u, v) | (u, v) ∉ E(G) }**



2. Union of graph G and its complement G' will give a **complete graph(K_n).**



3. The intersection of two complement graphs has no edges, also known as **null graph**



Binary Search Trees

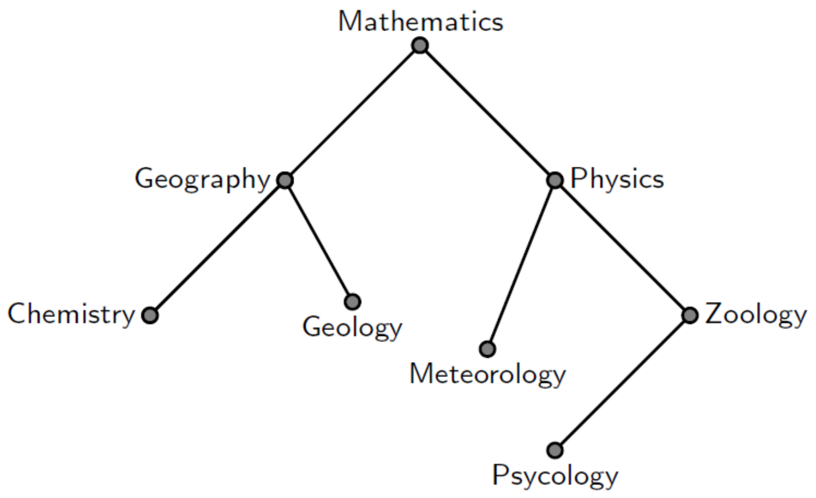
1. Searching for items in a list is one of the most important tasks that arises in computer science. Our primary goal is to implement a searching algorithm that finds items efficiently when the items are totally ordered. This can be accomplished through the use of a binary search tree.
2. A binary search tree is a binary tree in which each child of a vertex is designated as a right or left child, no vertex has more than one right child or left child, and each vertex is labeled with a key, which is one of the items. Furthermore, vertices are assigned keys so that the key of a vertex is both larger than the keys of all vertices in its left subtree and smaller than the keys of all vertices in its right subtree.

Example

Form a binary search tree for the words: mathematics, Physics, Geography, Zoology, Meteorology, Geology, Psychology, and Chemistry (using alphabetical order).

Example

Form a binary search tree for the words: mathematics, Physics, Geography, Zoology, Meteorology, Geology, Psychology, and Chemistry (using alphabetical order).

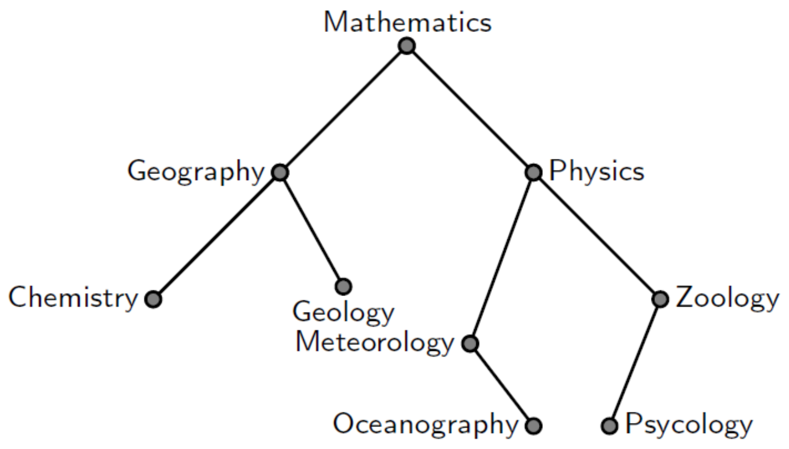


Example

Insert the word Oceanography into the binary search tree in the previous example.

Example

Insert the word Oceanography into the binary search tree in the previous example.
mathematics $<$ oceanography, physics $>$ oceanography, meteorology $<$ oceanography.

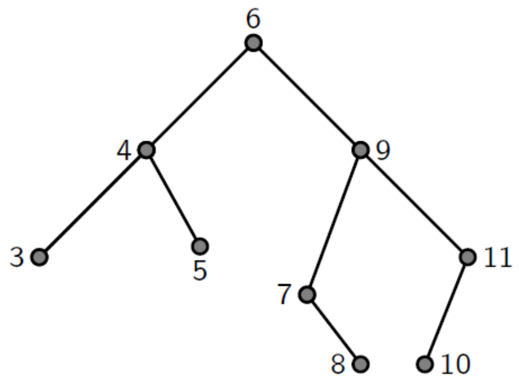


Example

Form a binary search tree for the numbers: 6, 9, 4, 11, 7, 5, 10, 3 and 8 (using the order on $\mathbb{N}$).

Example

Form a binary search tree for the numbers: 6, 9, 4, 11, 7, 5, 10, 3 and 8 (using the order on $\mathbb{N}$).



Tree Traversal

1. Ordered rooted trees are often used to store information.
2. We need procedures for visiting each vertex of an ordered rooted tree to access data.
3. We will describe several important algorithms for visiting all the vertices of an ordered rooted tree.

Traversal Algorithms

Procedures for systematically visiting every vertex of an ordered rooted tree are called traversal algorithms. We will describe three of the most commonly used such algorithms.

1. **preorder traversal**,**R**oot, **L**eft, **R**ight.
2. **inorder traversal**, **L**eft, **R**oot, **R**ight.
3. **postorder traversal**, **L**eft, **R**ight, **R**oot.

Preorder Traversal

Definition

Let T be an ordered rooted tree with root r. If T consists only of r, then r is the preorder traversal of T. Otherwise, suppose that $T_1, T_2, \cdots, T_n$ are the subtrees at r from left to right in T. The preorder traversal begins by visiting r. It continues by traversing T_1 in preorder, then T_2 in preorder, and so on, until T_n is traversed in preorder.

Example

In which order does a preorder traversal visit the vertices in the ordered rooted tree T shown in Figure (??) below

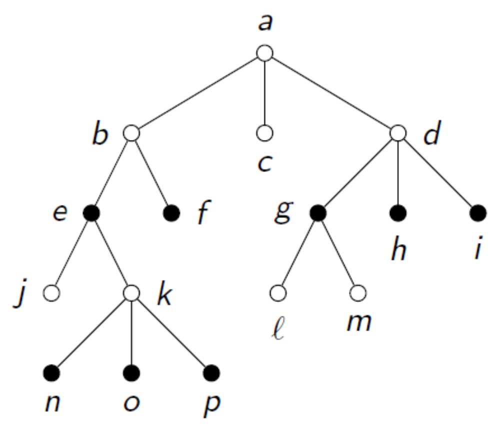


figure 2: The Ordered Rooted Tree T.

[illegible]
ordered rooted tree T shown in Figure (??) below

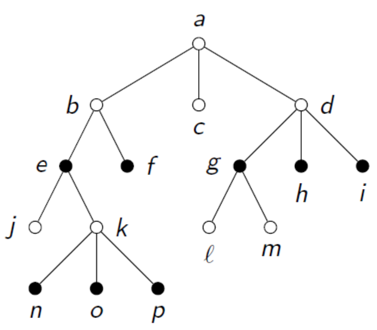


figure 2: The Ordered Rooted Tree T.

a b e j k n o p f c d g ℓ m h i

● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●

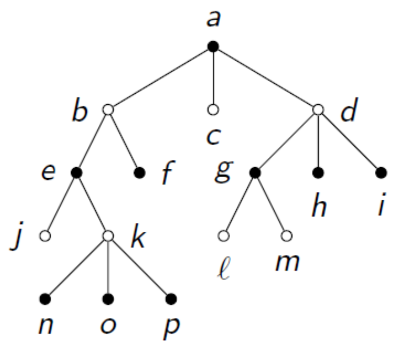
Inorder Traversal

Definition

Let T be an ordered rooted tree with root r. If T consists only of r, then r is the inorder traversal of T . Otherwise, suppose that $T_1, T_2, \cdots , T_n$ are the subtrees at r from left to right. The inorder traversal begins by traversing T_1 in inorder, then visiting r. It continues by traversing T_2 in inorder, then T_3 in inorder, . . . , and finally T_n in inorder.

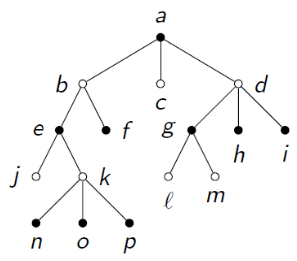
Example

In which order does an inorder traversal visit the vertices of the ordered rooted tree T below?



Example

In which order does an inorder traversal visit the vertices of the ordered rooted tree T below?



$$j \quad e \quad n \quad k \quad o \quad p \quad b \quad f \quad a \quad c \quad \ell \quad g \quad m \quad d \quad h \quad i$$

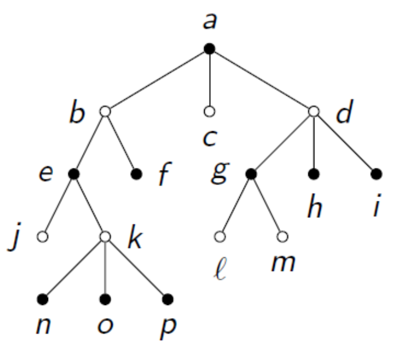
Postorder Traversal

Definition

Let T be an ordered rooted tree with root r. If T consists only of r, then r is the postorder traversal of T . Otherwise, suppose that $T_1, T_2, \cdots, T_n$ are the subtrees at r from left to right. The postorder traversal begins by traversing T_1 in postorder, then T_2 in postorder, $\cdots$, then T_n in postorder, and ends by visiting r.

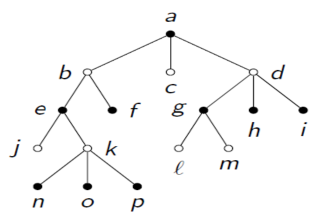
Example

In which order does a postorder traversal visit the vertices of the ordered rooted tree T shown below?



Example

In which order does a postorder traversal visit the vertices of the ordered rooted tree T shown below?



$j \quad n \quad o \quad p \quad k \quad e \quad f \quad b \quad c \quad \ell \quad m \quad g \quad h \quad i \quad d \quad a$

● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●