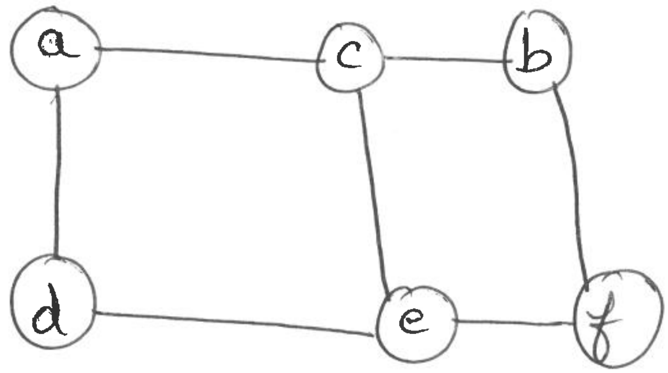
GRAPH REPRESENTATION (continued...)



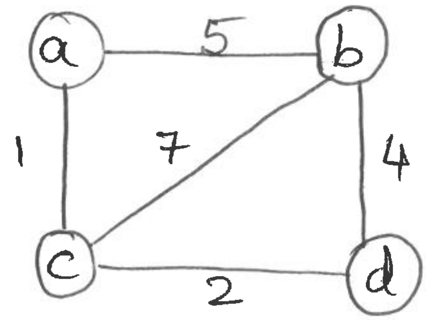
bi-directional edge

① Adjacency Matrix

② Adjacency Lists

a	→ c → d
b	→ c → f
c	→ a → b → e
d	→ a → e
e	→ c → d → f
f	→ b → e

Weighted Graphs



Adjacency Matrix

$$\begin{array}{c|cccc} & a & b & c & d \\ \hline a & \infty & 5 & 1 & \infty \\ b & 5 & \infty & 7 & 4 \\ c & 1 & 7 & \infty & 2 \\ d & \infty & 4 & 2 & \infty \end{array} \Rightarrow \text{Cost matrix}$$

Adjacency Lists

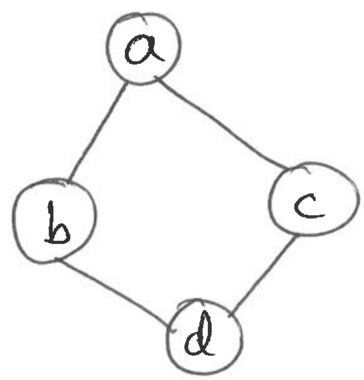
a → b, 5 → c, 1

b → a, 5 → c, 7 → d, 4

c → a, 1 → b, 7 → d, 2

d → b, 4 → c, 2

Cycle

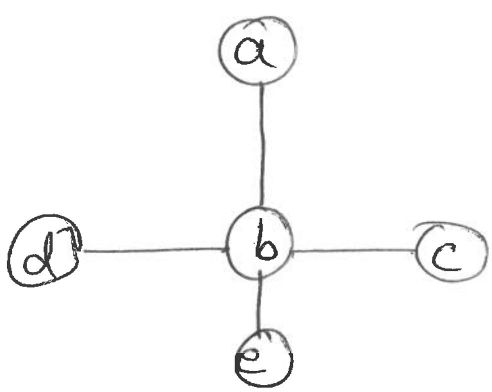


A path of positive length that starts and ends at the same vertex and does not traverse the same edge more than once.

a – c – d – b – a
(or)
a – b – d – c – a

represent the same cycle.

Acyclic Graph



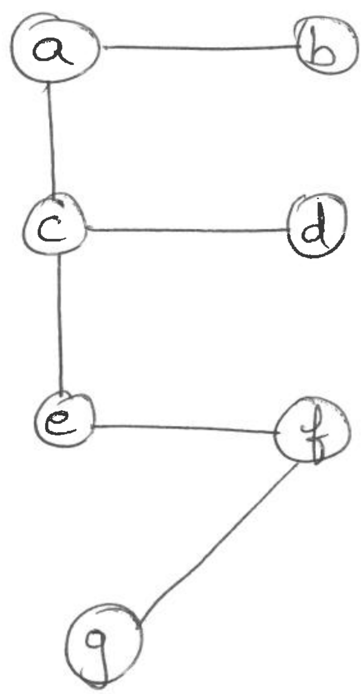
A graph with no cycles.

Connected Graph

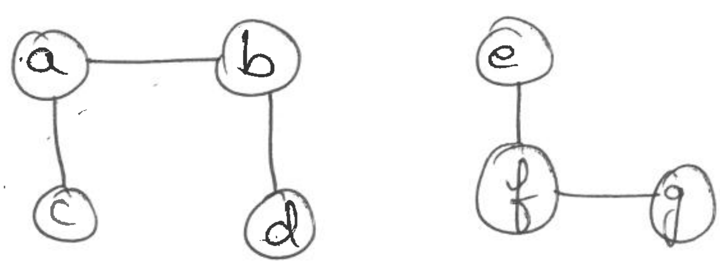
A graph where every pair of its vertices 'u' and 'v' there is a path from u to v.

Tree / Free Tree

A connected acyclic graph



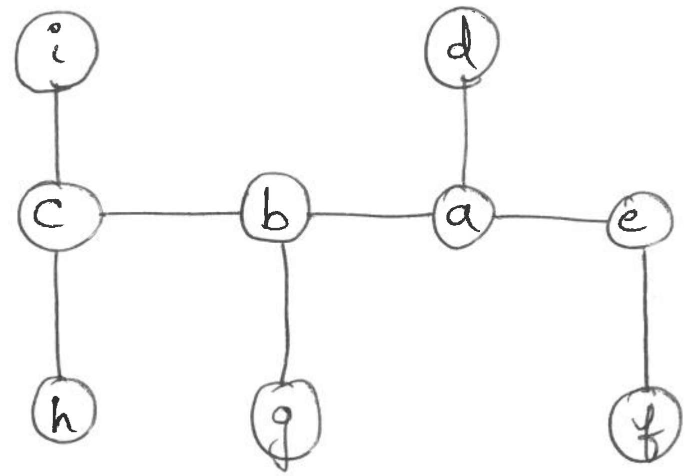
Forest



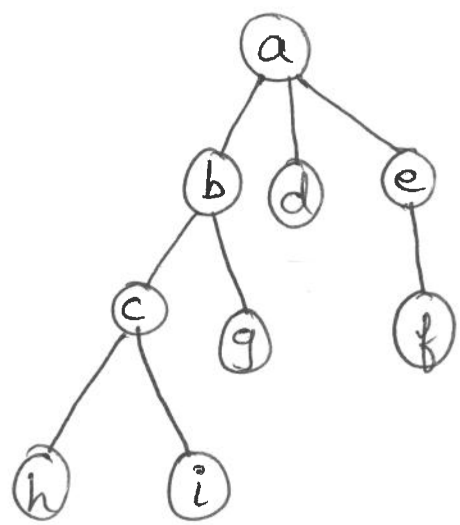
A graph with no cycles but not necessarily connected.

Example - 1

Free Tree



Rooted Tree → A tree with a root.



'a' ⟶ root node

b, d, e → siblings

c → descendant of 'a'

a → ancester of 'c'

b → child of 'a'

Ordered Tree

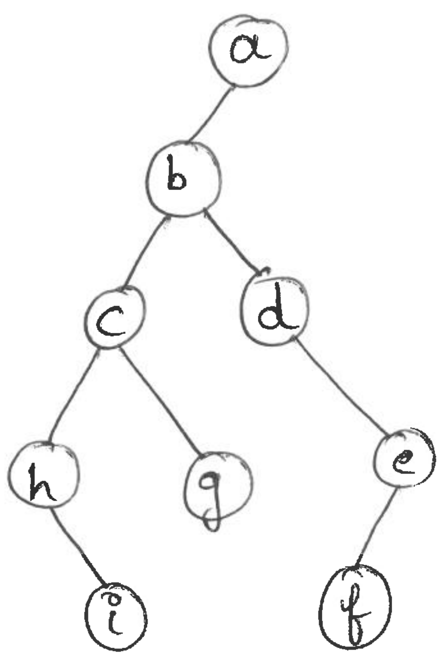
A rooted tree in which all the children are ordered.

Binary Tree

A rooted tree in which each vertex has no more than 2 children.

(i) Convert the rooted tree in (eg 1) to a binary tree.

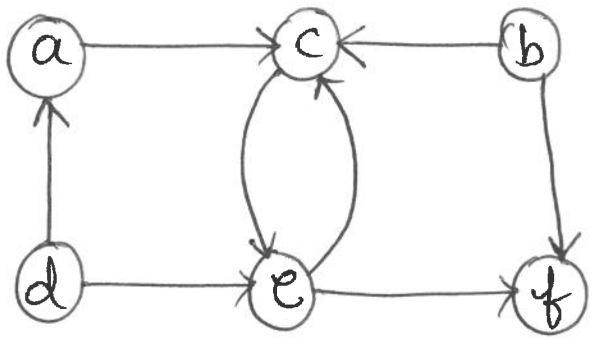
Use the first-child-next-sibling strategy.



BINARY TREE

Free Tree $\longrightarrow$ Rooted Tree $\longrightarrow$ Binary Tree

Digraph



Adjacency Matrix

$$
\begin{array}{c|cccccc|}
 & a & b & c & d & e & f \\
\hline
a & 0 & 0 & 1 & 0 & 0 & 0 \\
b & 0 & 0 & 1 & 0 & 0 & 1 \\
c & 0 & 0 & 0 & 0 & 1 & 0 \\
d & 1 & 0 & 0 & 0 & 1 & 0 \\
e & 0 & 0 & 1 & 0 & 0 & 1 \\
f & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
\end{array}
$$

Adjacency Lists

a → c
b → c → f
c → e
d → a → e
e → c → f
f

CHAPTER-2

The Analysis Framework

① Time Efficiency (or) Time Complexity

- how fast does the algorithm run?

② Space Efficiency

- the amount of memory units required by the algorithm in addition to space needed for its input and output.

Measuring Input size

Array $\rightarrow$ n

Matrix $\rightarrow$ $(n \times m)$ dimensions $\Rightarrow$ N total # of elements

Spell-check algorithm $\rightarrow$ # of characters

$\rightarrow$ # of words

Units for Measuring Time :

Basic Operation

→ the most important operation of the algorithm

→ typically will be found in the inner most loop.

→ Contributes the most to the total running time.

c_{op} → execution time of the algorithm's basic operation (on a particular compute

$C(n)$ → number of times this operation needs to be executed for an algorithm.

Time Efficiency, $\boxed{T(n) \approx c_{op}\, C(n)}$

eg1 If speed of the device increases by 10 times, then $T(n)$ also increases by 10 times

eg2 If input size is doubled,

let $C(n) = \frac{1}{2} n(n-1)$

$= \frac{1}{2} n^2 - \frac{1}{2} n$

$C(n) \approx \frac{1}{2} n^2$

$$\frac{T(2n)}{T(n)} = \frac{C_{op}\, C(2n)}{C_{op}\, C(n)}$$

$$= \frac{C_{op}\, \frac{1}{2}(2n)^2}{C_{op}\, \frac{1}{2} n^2}$$

$$= \frac{\cancel{C_{op}}\left(\cancel{\frac{1}{2}} \times 4\, \cancel{n^2}\right)}{\cancel{C_{op}}\left(\cancel{\frac{1}{2}}\, \cancel{n^2}\right)}$$

$$\frac{T(2n)}{T(n)} = 4$$

When input size is doubled, $T(n)$ increases by 4 times