DISCRETE MATHEMATICS

Chapter 8: Tree

Engineering Notes

8.1 Definition & Properties of Trees — Distance & Centre in a Tree

A tree is one of the most important special types of graph, and it forms the foundation for data structures, hierarchical models, and network design. A tree combines two simple ideas — connectedness and the absence of cycles — into a structure with a great many useful properties.

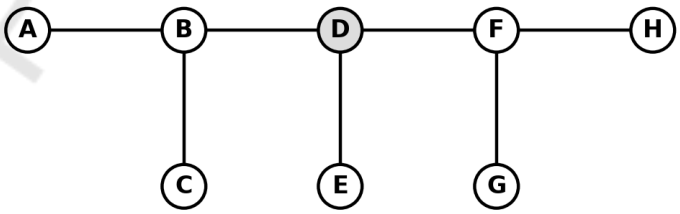
Definition of a Tree

A tree is a connected, undirected graph that contains no cycles (no closed loops). Equivalently, a tree is a connected acyclic graph. A graph consisting of a disjoint collection of trees (i.e., not necessarily connected as a whole) is called a forest.

Properties of Trees

If T is a tree with n vertices, the following properties always hold, and any one of them may be taken as an equivalent definition of a tree:

- A tree with n vertices has exactly n − 1 edges.
- A tree is connected, and removing any single edge disconnects it into exactly two components.
- A tree contains no cycles; adding any single new edge between two existing vertices creates exactly one cycle.
- There is exactly one (unique) path between every pair of vertices in a tree.
- A tree with two or more vertices has at least two vertices of degree 1 (called leaves or pendant vertices).
- A connected graph with n vertices is a tree if and only if it has exactly n − 1 edges.



A tree: connected, acyclic, n = 8 vertices, n − 1 = 7 edges.
Diameter (longest shortest-path) = 4, between A/C and G/H — so the radius is 2.
Shaded vertex D has eccentricity 2 (the smallest of all vertices) — D is the CENTRE of the tree.

Distance in a Tree

Since there is a unique path between any two vertices u and v of a tree, the distance d(u, v) is defined as the number of edges on this unique path. This is well-defined precisely because a tree has no alternative (cycle-creating) path to compare it with.

- The eccentricity of a vertex v, denoted e(v), is the maximum distance from v to any other vertex in the tree: e(v) = max{d(v, u) | u is a vertex of the tree}.
- The radius of a tree, r(T), is the minimum eccentricity among all its vertices: r(T) = min{e(v) | v is a vertex}.
- The diameter of a tree, diam(T), is the maximum eccentricity among all its vertices (equivalently, the length of the longest path in the tree): diam(T) = max{e(v) | v is a vertex}.

Centre of a Tree

The centre of a tree is the set of vertices whose eccentricity equals the radius of the tree, i.e., the "most central" vertex or vertices. A classical result states that every tree has either exactly one centre vertex or exactly two centre vertices, and if there are two, they must be adjacent.

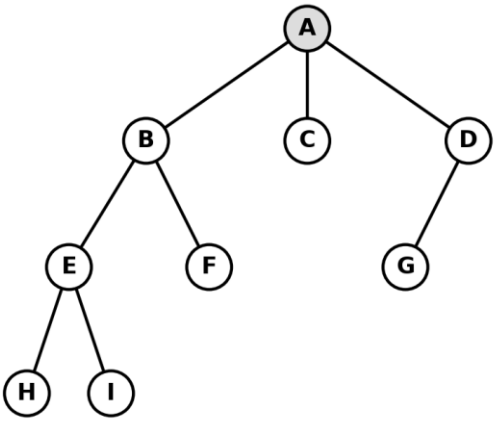
- The centre can be found by repeatedly removing all leaves (vertices of degree 1) from the tree, one layer at a time, until either a single vertex or a pair of adjacent vertices remains — that remainder is the centre.
- In the tree shown above, repeatedly stripping the leaves (A, C, E, G, H, then B, F) leaves only vertex D, so D alone is the centre, with radius 2 and diameter 4.

8.2 Rooted Tree, Co-Tree — Definition & Example

Rooted Tree

A rooted tree is a tree in which one particular vertex has been designated as the root. Once a root is fixed, every other vertex acquires a natural direction "away from the root", and a family of terminology follows:

- Level (or depth) of a vertex: the number of edges on the path from the root to that vertex. The root itself is at level 0.
- Height of a rooted tree: the maximum level of any vertex in the tree.
- Parent and child: if vertex u is on the path from the root to vertex v, and u is directly connected to v with u one level higher, then u is the parent of v, and v is a child of u.
- Siblings: two vertices that share the same parent.
- Ancestor and descendant: u is an ancestor of v (and v a descendant of u) if u lies on the unique path from the root to v.
- Leaf (or terminal vertex): a vertex with no children. An internal vertex is any vertex that has at least one child.



A = root (level 0); B, C, D = children of A (level 1); E, F, G = level 2; H, I = leaves (level 3).
B is the parent of E and F; E and F are siblings; height of tree = 3.

Co-Tree

Given a connected graph G and a spanning tree T of G (a spanning tree is introduced formally in Section 8.4), the co-tree of T, denoted T′, is the subgraph of G consisting of all the vertices of G together with only those edges of G that are not part of T. In other words, if T uses some of the edges of G to connect all the vertices without forming a cycle, the co-tree collects exactly the leftover edges.

- If G has n vertices and m edges, then the spanning tree T has n − 1 edges, and the co-tree T′ has m − (n − 1) edges.
- The co-tree is generally not connected and may contain cycles when combined with parts of T; on its own (as just a set of edges), the co-tree edges are also called chords (see Section 8.4).
- Example: For a connected graph G with 6 vertices and 8 edges, if a spanning tree T uses 5 edges, then the co-tree T′ (with respect to T) consists of the remaining 8 − 5 = 3 edges of G.

8.3 Binary Trees

Definition and Properties

A binary tree is a rooted tree in which every vertex has at most two children, conventionally distinguished as the left child and the right child. A binary tree may be empty (no vertices at all), or it may consist of a root together with a left binary subtree and a right binary subtree, each of which is itself a binary tree.

- A full (or strictly) binary tree is one in which every vertex has either exactly 0 or exactly 2 children (never just 1).
- A complete binary tree of height h has every level completely filled except possibly the last, which is filled from left to right.
- Maximum number of vertices at level i (root at level 0) is 2^i.
- Maximum number of vertices in a binary tree of height h is 2^(h+1) − 1 (a full complete binary tree).
- A binary tree with n vertices has height at least $\lceil \log_2(n+1) \rceil - 1$, i.e., the height grows only logarithmically with the number of vertices in the best case.

- In any binary tree, if n2 is the number of vertices with exactly 2 children and n0 is the number of leaves (0 children), then n0 = n2 + 1.

Path Length

The internal path length of a binary tree is the sum of the depths (levels) of all internal vertices, measured from the root. The external path length is the sum of the depths of all the leaves (external/terminal vertices), often measured by extending the tree with imaginary "external nodes" wherever a child is missing. Path length is directly related to the efficiency of search and traversal operations performed on the tree — shorter path lengths generally mean faster access to data stored at the vertices.

Binary Tree Representation of General Trees

Any general (ordered) tree, where a vertex may have any number of children, can be converted into an equivalent binary tree using the left-child, right-sibling representation:

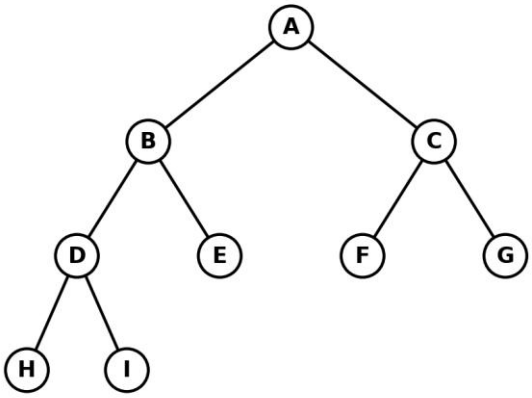
1. For each vertex v in the general tree, its leftmost child in the general tree becomes its left child in the binary tree.
2. The next sibling of v (to its right, at the same level in the general tree) becomes the right child of v in the binary tree.
3. This process is applied recursively to every vertex, producing a binary tree that preserves all the information of the original general tree using only left/right child pointers.

This representation is extremely useful in computer implementations, since a general tree with a variable, unbounded number of children per vertex can be stored using a fixed two-pointer (left child, right sibling) structure per vertex, exactly like a binary tree.

Binary Tree Traversal

Traversal means visiting every vertex of a binary tree exactly once, in a systematic order. The three classical traversal orders are defined recursively as follows:

- Preorder (Root, Left, Right): visit the root first, then recursively traverse the left subtree, then recursively traverse the right subtree.
- Inorder (Left, Root, Right): recursively traverse the left subtree, then visit the root, then recursively traverse the right subtree. (For binary search trees, inorder traversal visits the vertices in sorted order.)
- Postorder (Left, Right, Root): recursively traverse the left subtree, then the right subtree, then visit the root last.



Binary tree (each node has at most 2 children).
Preorder (Root-L-R): A B D H I E C F G Inorder (L-Root-R): H D I B E A F C G Postorder (L-R-Root): H I D E B F G C A

Problems

Problem 1: A binary tree has 15 vertices and is a full complete binary tree. Find its height. Solution: For a full complete binary tree of height h, the number of vertices is 2^(h+1) − 1. Setting 2^(h+1) − 1 = 15 gives 2^(h+1) = 16 = 2^4, so h + 1 = 4, i.e., h = 3.

Problem 2: In a binary tree, there are 8 leaves. If every internal vertex has exactly 2 children, find the number of internal vertices. Solution: Using n0 = n2 + 1 with n0 = 8, we get n2 = 7. Hence there are 7 internal vertices, and the tree has n0 + n2 = 15 vertices in total.

8.4 Spanning Tree — Branch, Chord — Definition & Properties

Spanning Tree

Let G be a connected graph. A spanning tree of G is a subgraph T of G that (i) includes every vertex of G, and (ii) is itself a tree (connected and acyclic). In other words, a spanning tree connects all the vertices of G together using the minimum possible number of edges, with no cycles.

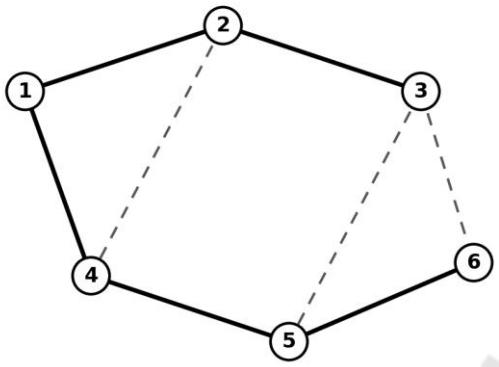
- Every connected graph has at least one spanning tree (found, for example, by removing edges one at a time from any cycle until no cycles remain).
- If G has n vertices, every spanning tree of G has exactly n − 1 edges, regardless of how many edges G itself has.
- A connected graph may have many different spanning trees. A complete graph on n vertices, for instance, has n^(n−2) distinct spanning trees (Cayley's formula).

Branch and Chord

Once a particular spanning tree T has been chosen for a connected graph G with n vertices and m edges:

- Each edge of G that belongs to the spanning tree T is called a branch (or tree edge) of T. There are always exactly n − 1 branches.
- Each edge of G that does not belong to T is called a chord (or link, or tree-edge complement) with respect to T. There are always exactly m − (n − 1) chords. The set of all chords, together with the vertices of G, forms the co-tree of T introduced in Section 8.2.

- Adding any single chord back to the spanning tree T creates exactly one cycle, called the fundamental cycle of that chord.



Graph G with a spanning tree T highlighted.
Bold solid edges = branches of T (5 edges, n − 1 = 5); dashed grey edges = chords (co-tree of T, 3 edges).

Spanning Tree in a Weighted Graph

In a weighted graph, every edge carries a numerical weight (representing cost, distance, time, etc.). For such a graph, a spanning tree can be assigned a total weight equal to the sum of the weights of its branches. Since a connected weighted graph generally has many different spanning trees, a natural and important question is to find the spanning tree with the smallest possible total weight — this is called a minimum spanning tree (MST), and it is the central topic of Section 8.5.

8.5 Algorithms for Constructing Spanning Trees

Graph-Theoretic Algorithms

A spanning tree of any connected graph (unweighted) can be constructed systematically using standard graph-traversal algorithms, which visit every vertex exactly once and record only the edges used to reach a new, previously unvisited vertex:

- Breadth-First Search (BFS) spanning tree: starting from a chosen root vertex, visit all its neighbours first, then their unvisited neighbours, level by level, recording each newly used edge. The result is a BFS spanning tree.
- Depth-First Search (DFS) spanning tree: starting from a chosen root vertex, repeatedly move to an unvisited adjacent vertex as deep as possible before backtracking, recording each newly used edge. The result is a DFS spanning tree.
- Both methods produce a valid spanning tree with exactly n − 1 edges, but the shape (and hence properties like height) of the resulting tree can differ considerably between the two methods.

Minimal (Minimum) Spanning Tree Algorithm

For a connected weighted graph, a minimum spanning tree (MST) is a spanning tree whose total edge weight is the smallest among all possible spanning trees of that graph. Two classical greedy algorithms are commonly used to construct an MST: Kruskal's Algorithm and Prim's Algorithm. This section focuses on Kruskal's Algorithm.

Kruskal's Algorithm

Kruskal's algorithm builds a minimum spanning tree by repeatedly adding the cheapest available edge that does not create a cycle, until all vertices are connected. The steps are:

1. List all the edges of the graph and sort them in non-decreasing (ascending) order of their weights.
2. Initialize the spanning tree T as empty (no edges selected yet).
3. Examine the edges one by one, in the sorted order. For each edge, add it to T only if doing so does not create a cycle with the edges already selected in T; otherwise, discard (reject) that edge and move to the next one.
4. Repeat step 3 until exactly n − 1 edges have been added to T (i.e., until all n vertices are connected). At this point T is the required minimum spanning tree.

- A cycle check is usually performed efficiently using the Union-Find (Disjoint Set) data structure: two vertices are in the same "set" if they are already connected by edges chosen so far; adding an edge between two vertices already in the same set would create a cycle and is rejected.

Worked Problem

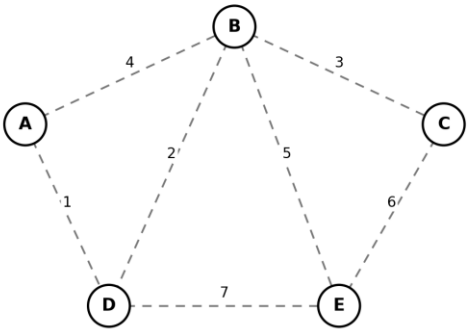
Problem: Apply Kruskal's algorithm to find the minimum spanning tree of the weighted graph with vertices {A, B, C, D, E} and edges: A–B (4), A–D (1), B–C (3), B–D (2), B–E (5), C–E (6), D–E (7).

Solution: Sorting the edges by weight gives: A–D (1), B–D (2), B–C (3), A–B (4), B–E (5), C–E (6), D–E (7). Now process them in order:

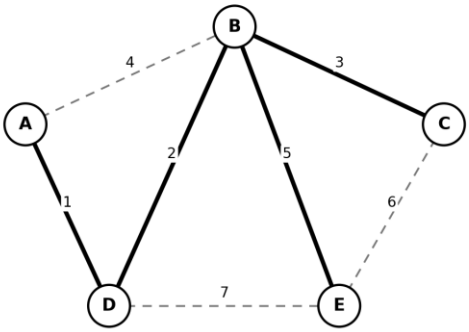
1. A–D (weight 1): connects A and D, no cycle — ACCEPT. Tree so far: {A–D}.
2. B–D (weight 2): connects B to the {A, D} component, no cycle — ACCEPT. Tree so far: {A–D, B–D}.
3. B–C (weight 3): connects C to the {A, B, D} component, no cycle — ACCEPT. Tree so far: {A–D, B–D, B–C}.
4. A–B (weight 4): both A and B are already in the same component — REJECT (would create a cycle).
5. B–E (weight 5): connects E to the {A, B, C, D} component, no cycle — ACCEPT. Tree so far: {A–D, B–D, B–C, B–E}.

At this point, 4 edges have been selected (n − 1 = 5 − 1 = 4), so all 5 vertices are connected and the algorithm stops (the remaining edges C–E and D–E are not examined). The minimum spanning tree consists of the edges A–D, B–D, B–C, B–E, with total weight 1 + 2 + 3 + 5 = 11.

Weighted graph G (edges labelled with weight)



Minimum Spanning Tree by Kruskal's Algorithm
(A-D,B-D,B-C,B-E chosen; total weight = 1+2+3+5 = 11)



Multiple Choice Questions (MCQs)

1. A tree is defined as a graph that is:

a) Connected and contains a cycle
b) Disconnected and acyclic
c) Connected and acyclic
d) Complete and directed

2. A tree with 10 vertices has exactly how many edges?

a) 10
b) 9
c) 11
d) 20

3. Between any two vertices of a tree, the number of paths is:

a) Zero
b) Exactly one
c) Exactly two
d) Infinite

4. The centre of a tree consists of:

a) All leaves of the tree
b) The vertex/vertices with minimum eccentricity
c) The vertex with maximum degree
d) The root only

5. In a rooted tree, the number of edges from the root to a vertex v is called the:

a) Degree of v
b) Level (depth) of v
c) Weight of v
d) Eccentricity of v

6. With respect to a spanning tree T of a graph G, the edges of G not included in T form the:

a) Branch set
b) Co-tree (chords)
c) Root set
d) Complete graph

7. In a binary tree, each vertex can have at most:

a) 1 child
b) 2 children
c) 3 children
d) Any number of children

8. The maximum number of vertices in a binary tree of height h is:

a) 2h
b) h^2
c) 2^(h+1) − 1
d) 2^h − 1

9. In the inorder traversal of a binary tree, the visiting order is:

a) Root, Left, Right
b) Left, Root, Right

c) Left, Right, Root
d) Right, Root, Left

10. The left-child right-sibling method is used to:

a) Delete a binary tree
b) Represent a general tree as a binary tree
c) Find the centre of a tree
d) Sort the vertices of a graph

11. If a connected graph G has n vertices and m edges, every spanning tree of G has:

a) m edges
b) n edges
c) n − 1 edges
d) m − 1 edges

12. Adding a chord back to a spanning tree T always creates:

a) A disconnected graph
b) Exactly one cycle
c) Another spanning tree with fewer edges
d) No change

13. Kruskal's algorithm builds a minimum spanning tree by:

a) Always picking the most expensive edge first
b) Picking the cheapest edge that does not form a cycle
c) Visiting vertices in alphabetical order only
d) Randomly selecting edges

14. In Kruskal's algorithm, an edge is rejected when:

a) It has the smallest weight
b) It connects two vertices in different components
c) Adding it would create a cycle
d) It is the last edge in the sorted list

15. A minimum spanning tree of a connected weighted graph with n vertices always has:

a) n edges
b) n − 1 edges with the minimum possible total weight
c) The maximum possible total weight
d) No edges

Answer Key

1-c, 2-b, 3-b, 4-b, 5-b, 6-b, 7-b, 8-c, 9-b, 10-b, 11-c, 12-b, 13-b, 14-c, 15-b

Broad / Long Answer Questions

1. Define a tree. State and explain any four important properties that characterize a tree among all graphs.

2. Define the distance, eccentricity, radius, diameter and centre of a tree. Find the centre of a tree of your choice with at least 7 vertices, showing your working.

3. Define a rooted tree and explain the terms root, parent, child, sibling, level, height and leaf with a suitable labelled diagram.

4. Define the co-tree of a spanning tree. If a connected graph has 7 vertices and 10 edges, find the number of branches and chords for any spanning tree of the graph.

5. Define a binary tree. State the maximum number of vertices possible at level i and in a binary tree of height h, with proof or justification.

6. Explain, with an example, how a general tree (with vertices having any number of children) can be represented as a binary tree using the left-child right-sibling method.

7. Define preorder, inorder, and postorder traversal of a binary tree. For a binary tree of your choice with at least 7 vertices, write out all three traversal sequences.

8. Define a spanning tree of a connected graph. Explain the terms branch and chord with respect to a spanning tree, with a suitable example.

9. Explain the difference between a spanning tree obtained by Breadth-First Search (BFS) and one obtained by Depth-First Search (DFS), and describe how a minimum spanning tree differs from an ordinary spanning tree in a weighted graph.

10. State the steps of Kruskal's algorithm. Using Kruskal's algorithm, find the minimum spanning tree for a connected weighted graph with 6 vertices and at least 8 weighted edges of your choice, showing all steps and the total weight of the resulting tree.