Data Structures and Algorithms (6)

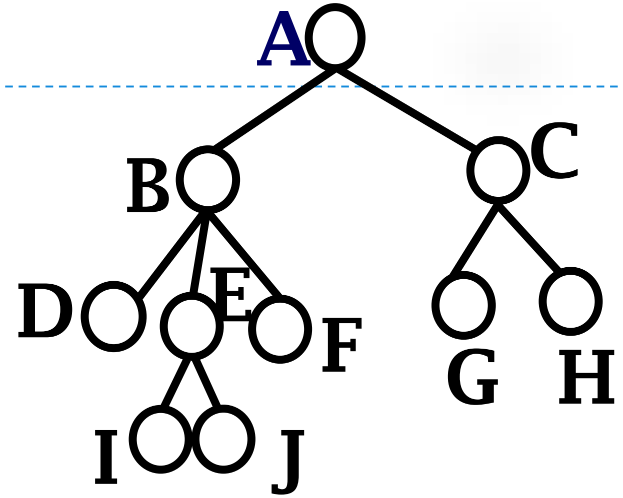
Instructor: Ming Zhang

Textbook Authors: Ming Zhang, Tengjiao Wang and Haiyan Zhao

Higher Education Press, 2008.6 (the "Eleventh Five-Year" national planning textbook)

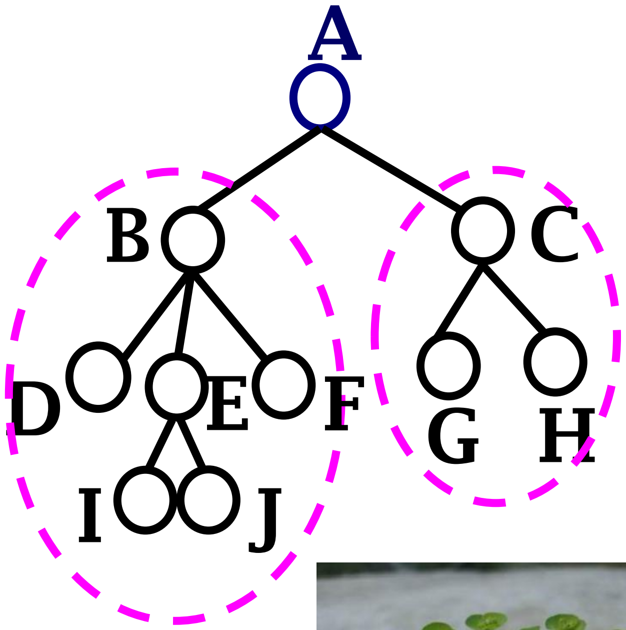
https://courses.edx.org/courses/PekingX/04830050x/2T2014/

Chapter 6 Trees



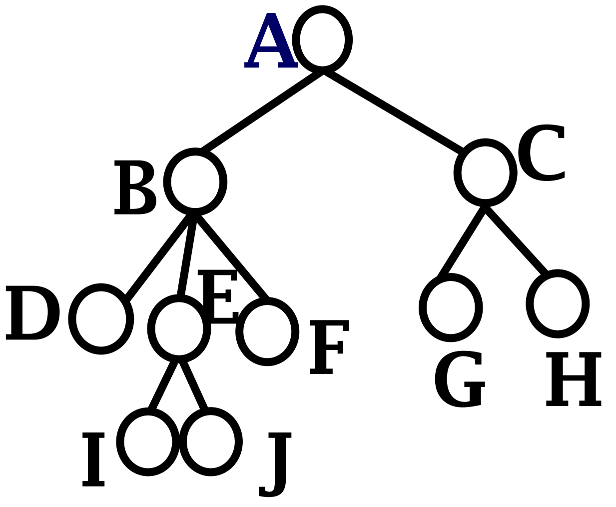
- General Definitions and Terminology of Tree
  - Trees and Forest
  - Equivalent Transformation between a Forest and a Binary Tree
  - Abstract Data Type of Tree
  - General Tree Traversals
- Linked Storage Structure of Tree
- Sequential Storage Structure of Tree
- K-ary Trees

Trees and Forest



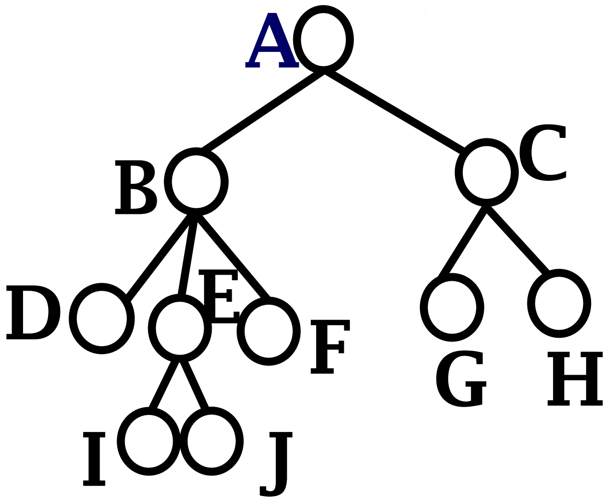
- A tree T is a finite set of one or more nodes :
  - there is one specific node R, called the root of T
  - If the set T-{R} is not empty, these nodes are partitioned into m > 0 disjoint finite subsets T_1 , T_2 , … , T_m, each of which is a tree. The subsets T_i are said to be subtrees of T.
  - Directed ordered trees: the relative order of subtrees is important
- An ordered tree with degree 2 is not a binary tree
  - After the first child node is deleted
  - The second child node will take the first child node’s place

Logical Structure of Tree



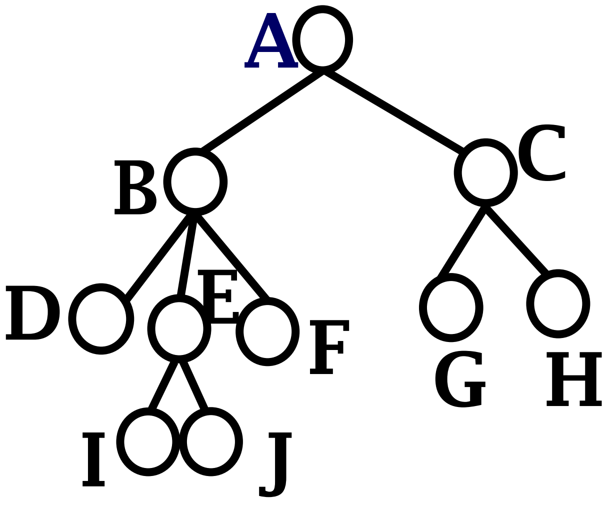
- A finite set K of n nodes, and a relation r satisfying the following conditions:
  - There is a unique node $k_0 \in K$, who has no predecessor in relation r.
    - Node k_0 is called the root of the tree.
  - Except k_0, all the other nodes in K has a unique predecessor in relation r
- An example as in the figure on the right
  - Node set K = { A, B, C, D, E, F, G, H, I, J }
  - The relation on K: r = { <A, B>, <A, C>, <B, D>, <B, E>, <B, F>, <C, G>, <C, H>, <E, I>, <E, J> }

Terminology of Tree



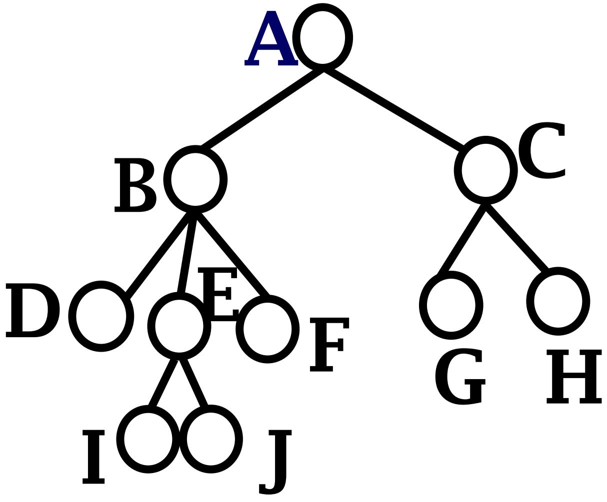
- **Node**
  - **Child node, parent node, the first child node**
    - If <k, k′> ∈ r, we call that k is the parent node of k′, and k′ is the child node of k
  - **Sibling node, previous/next sibling node**
    - If <k, k′> ∈ r and <k, k″>∈ r, we call k′ and k″ are sibling nodes
  - **Branch node, leaf node**
    - Nodes who have no subtrees are called leaf nodes
    - Other nodes are called branch nodes

Terminology of Tree



- **Edge**
  - The ordered pair of two nodes is called an edge
- **Path, path length**
  - Except the node k_0, for any other node $k \in K$, there exists a node sequence $k_0, k_1, \ldots, k_s$, s.t. k_0 is the root node, $k_s = k$, and $\langle k_{i-1}, k_i \rangle \in r$ $(1 \le i \le s)$.
  - This sequence is called a path from the root node to node k, and the path length (the total number of edges in the path) is s
- **Ancestor, descendant**
  - If there is a path from node k to node k_s, we call that k is an ancestor of k_s, and k_s is a descendant of k

Terminology of Tree

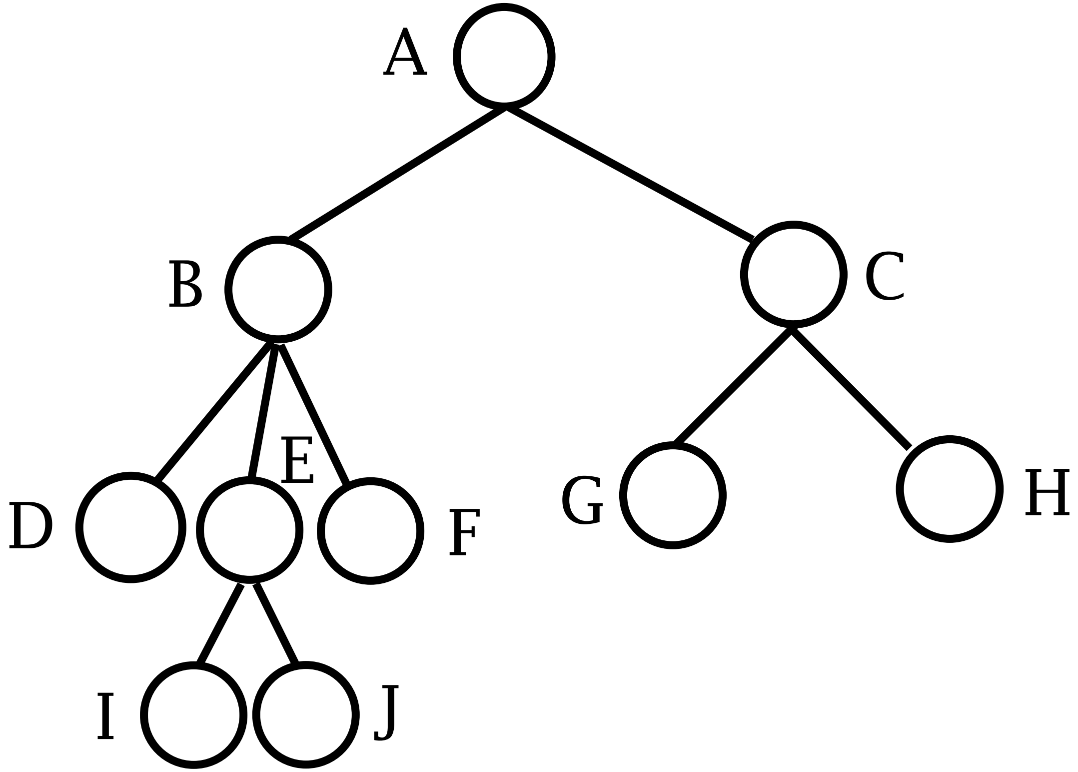


- **Degree**: The degree of a node is the number of children for that node.
- **Level**: The root node is at level 0
  - The level of any other node is the level of its parent node plus 1
- **Depth**: The depth of a node M in the tree is the path length from the root to M.
- **Height**: The height of a tree is the depth of the deepest node in the tree plus 1.

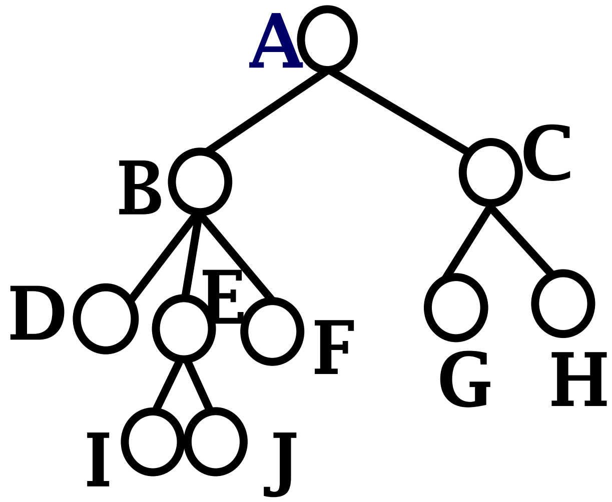
Different Representations of Trees

- Classic node-link representation
- Formal (set theory) representation
- Venn diagram representation
- Outline representation
- Nested parenthesis representation

Node-Link Representation



Formal Representation



The logical structure of a Tree is:

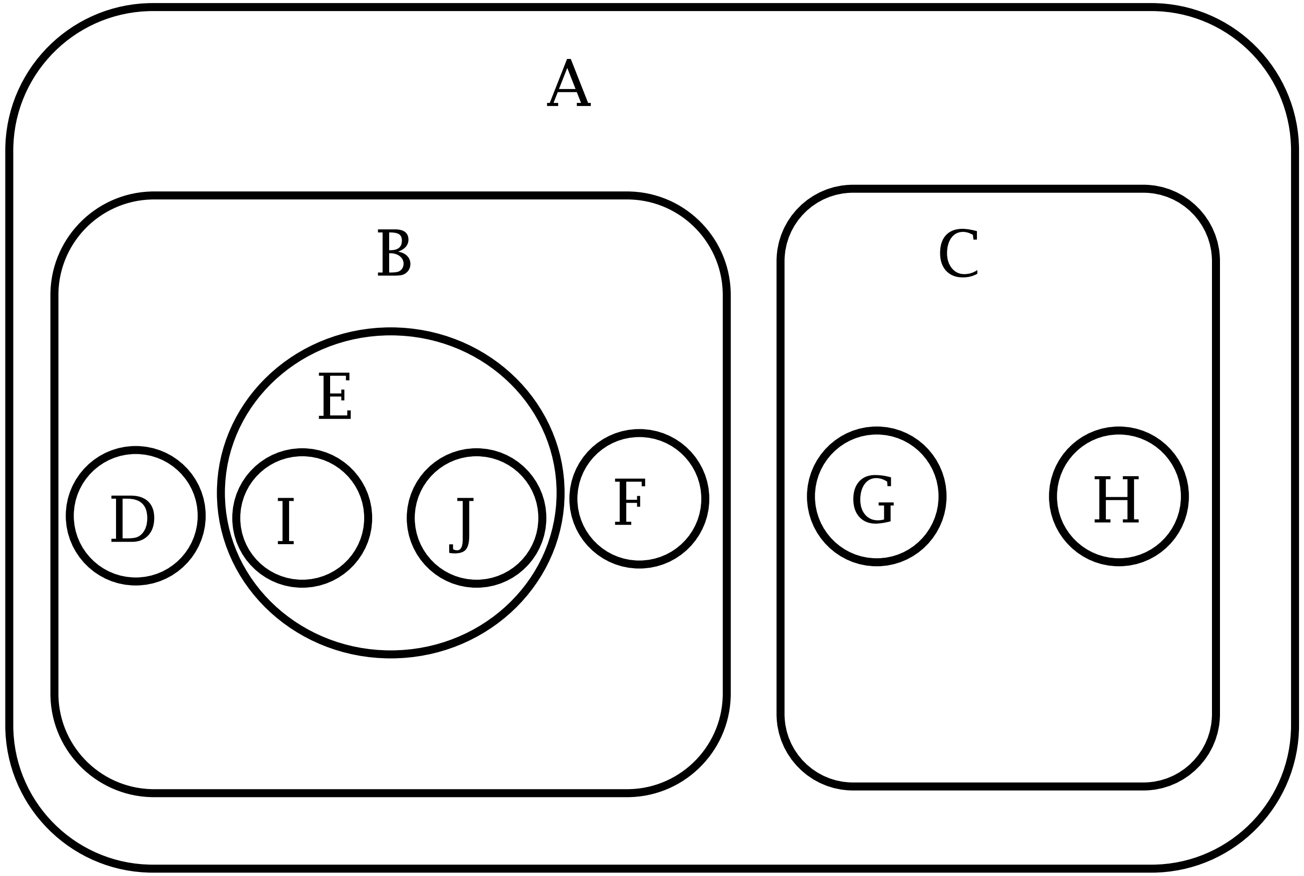
Node set:

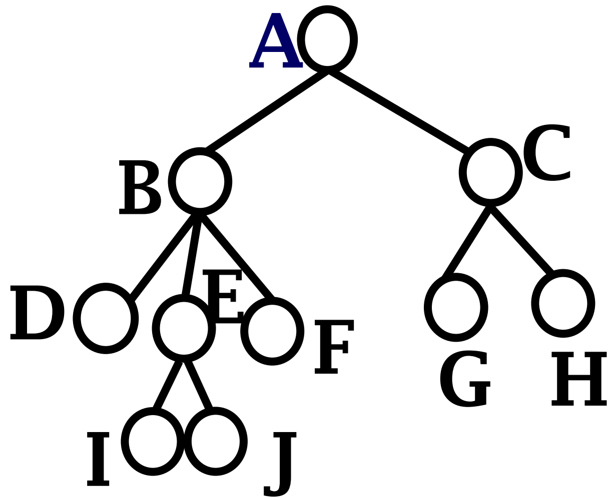
K = {A, B, C, D, E, F, G, H, I, J}

The relation on K:

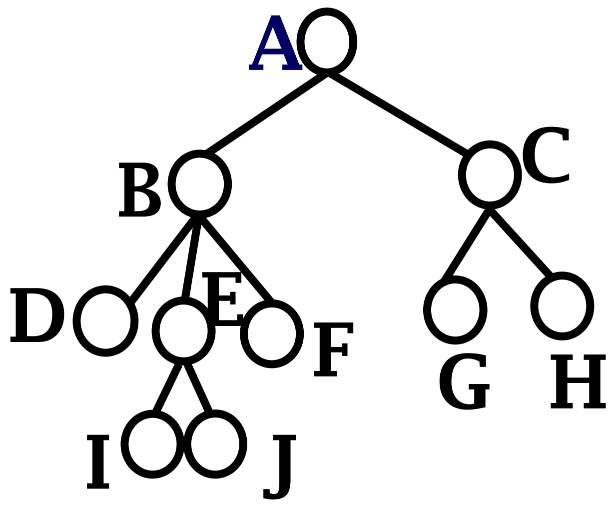
N = {<A, B>, <A, C>, <B, D>, <B, E>, <B, F>, <C, G>, <C, H>, <E, I>, <E, J>}

Venn Diagram Representation



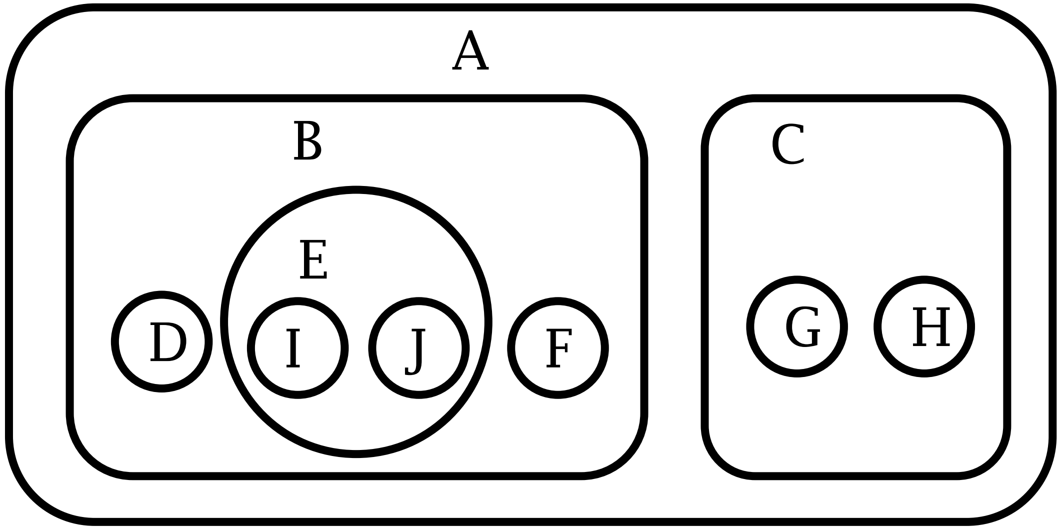


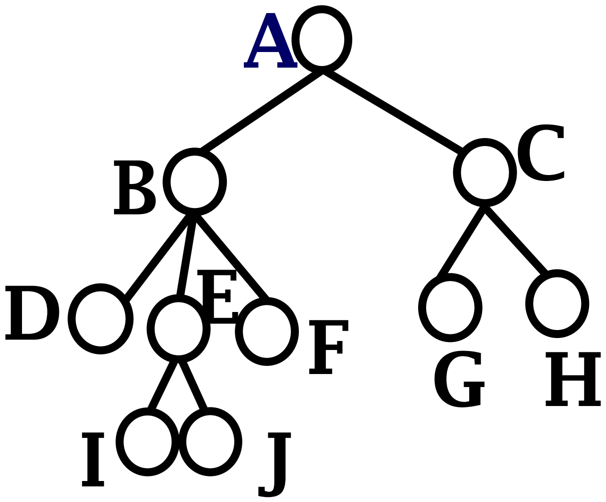
Nested Parenthesis Representation



(A(B(D)(E(I)(J))(F))(C(G)(H)))

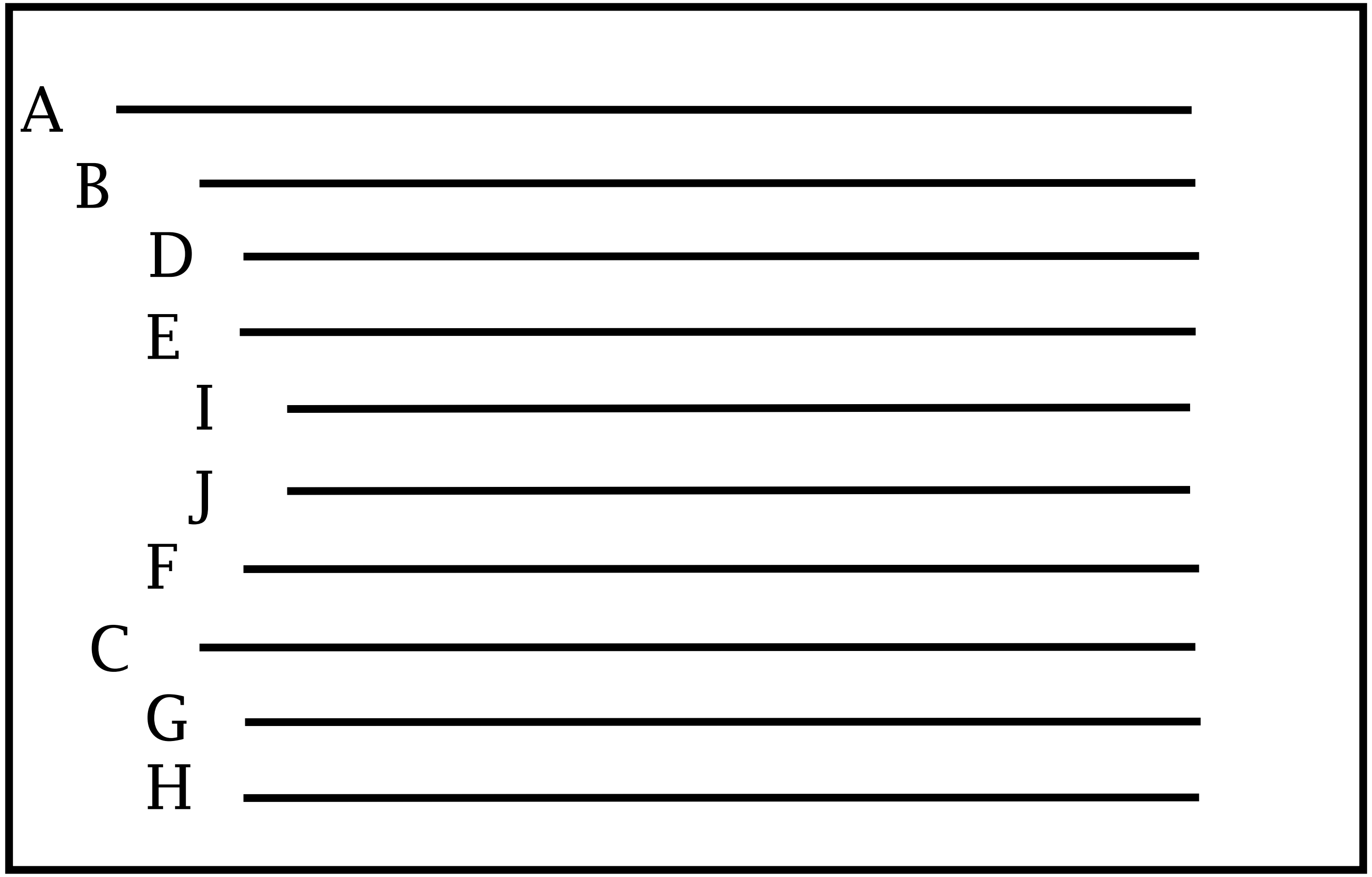
The conversion from Venn diagram to nested parenthesis

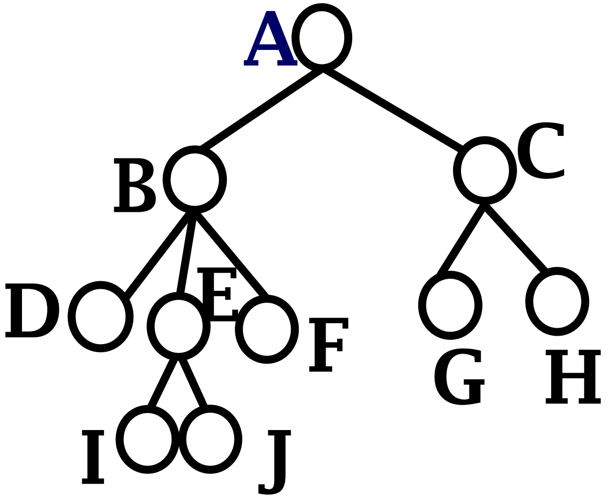




(A(B(D)(E(I)(J))(F))(C(G)(H)))

Outline Representation



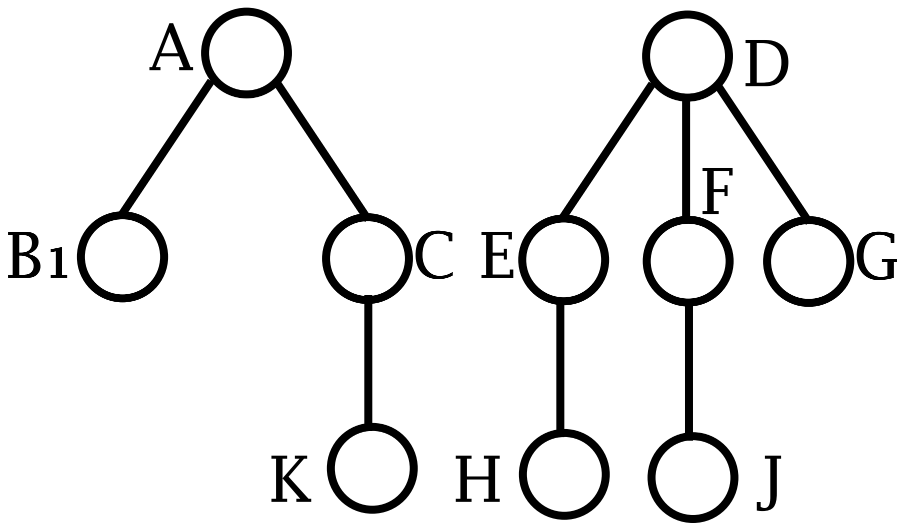


Book catalogue, Dewey representation

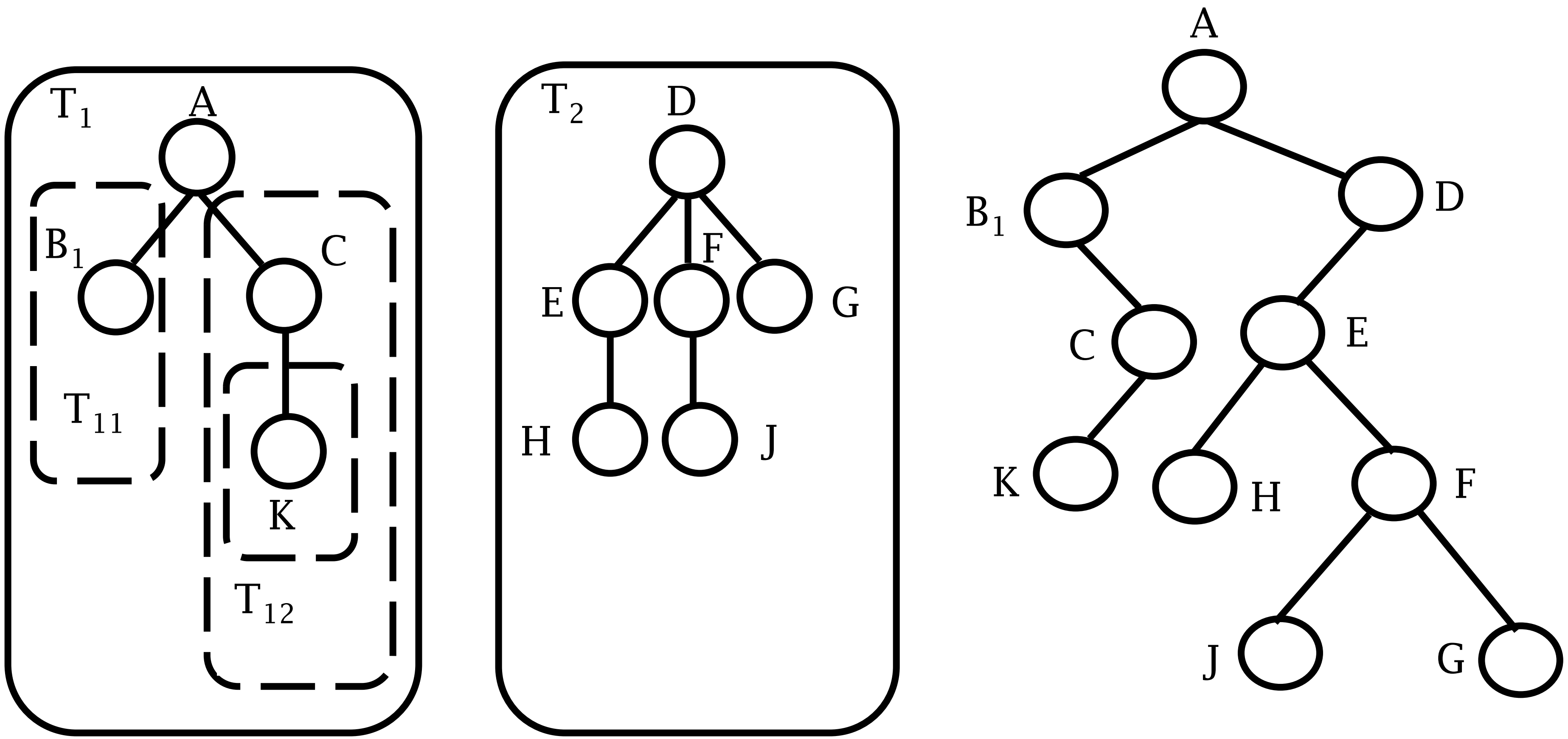
6 Trees
- 6.1 General Definitions and Terminology of Tree
  - 6.1.1 Tree and Forest
  - 6.1.2 Equivalence Transformation between a Forest and a Binary Tree
  - 6.1.3 Abstract Data Type of the Tree
  - 6.1.4 General Tree Traversals
- 6.2 Linked Storage Structure of Tree
  - 6.2.1 List of Children
  - 6.2.2 Static Left-Child/Right-Sibling representation
  - 6.2.3 Dynamic representation
  - 6.2.4 Dynamic Left-Child/Right-Sibling representation
  - 6.2.5 Parent Pointer representation and its Application in Union-Find Sets
- 6.3 Sequential Storage Structure of Tree
  - 6.3.1 Preorder Sequence with rlink representation
  - 6.3.2 Double-tagging Preorder Sequence representation
  - 6.3.3 Postorder Sequence with Degree representation
  - 6.3.4 Double-tagging Levelorder Sequence representation
- 6.4 K-ary Trees
- 6.5 Knowledge Conclusion of Tree

Equivalent Transformation between a Forest and a Binary Tree

- **Forest**: A forest is a collection of one or more disjoint trees. (usually ordered)
- The correspondence between trees and a forests
  - Removing the root node from a tree, its subtrees become a forest.
  - Adding an extra node as the root of the trees in a forest, the forest becomes a tree.
- There is a one-to-one mapping between forests and binary trees
  - So that all the operations on forests can be transformed to the operations on binary trees

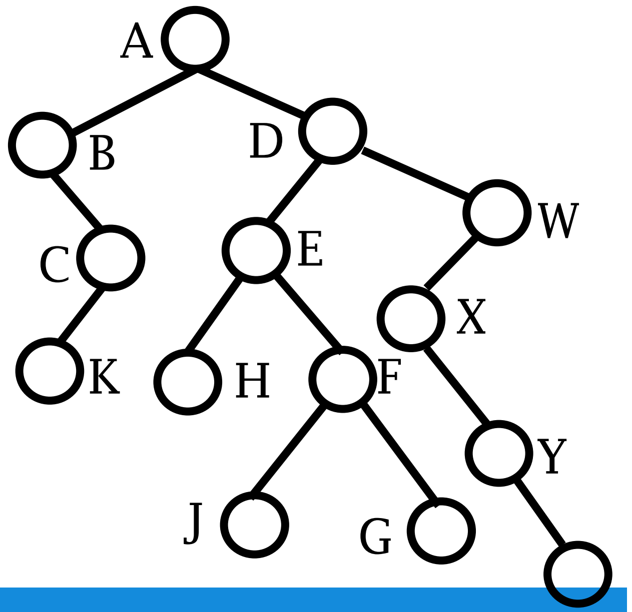


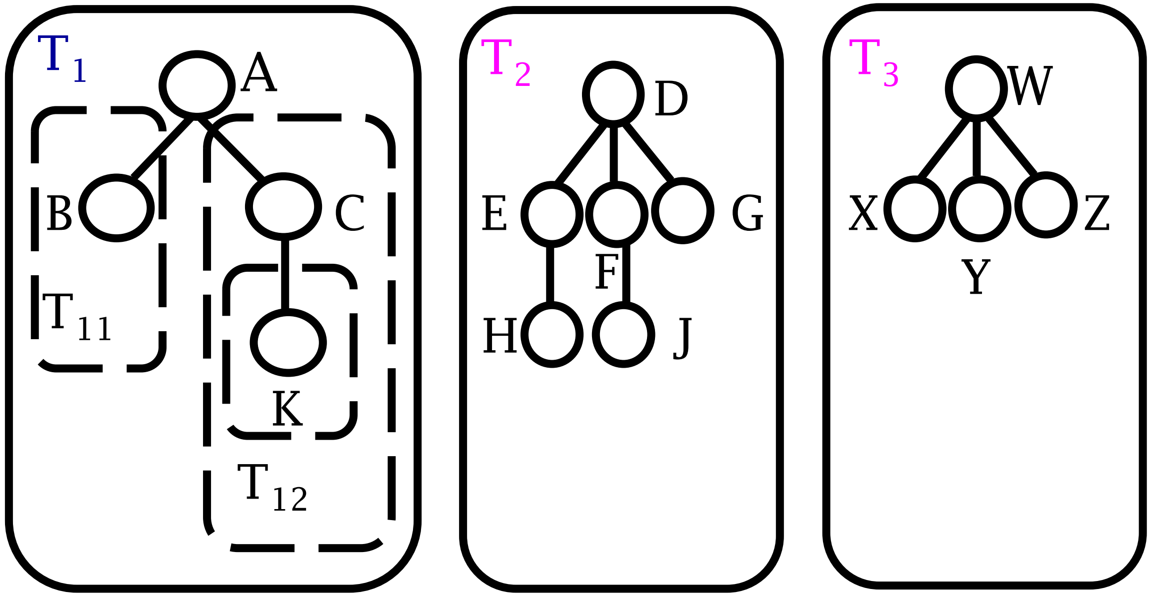
How to map a forest to a binary tree?



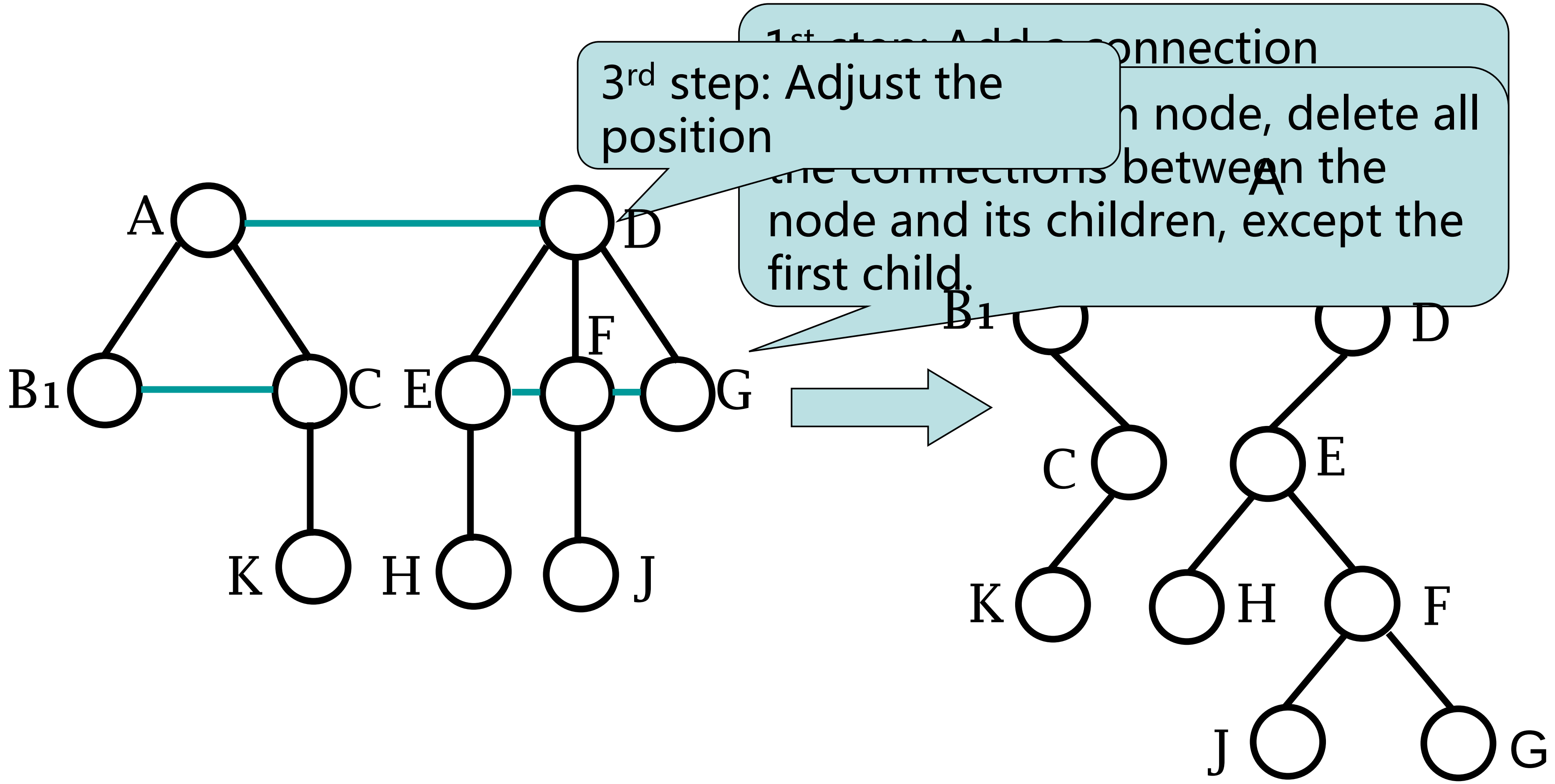
The transformation from a forest to a binary tree

- Ordered set F = {T_1, T_2, …, T_n} is a forest with trees T_1, T_2, …, T_n. We transform it to a binary tree B(F) recursively:
  - If F is empty (i.e., n=0), B(F) is an empty binary tree.
  - If F is not empty (i.e., n≠0), the root of B(F) is the root W_1 of the first tree T_1 in F;
  - the left subtree of B(F) is the binary tree B(F_{W1}), where F_{W1} is a forest consisting of W_1's subtrees in T_1;
  - the right subtree of B(F) is the binary tree B(F'), where F' = {T_2, …, T_n}.



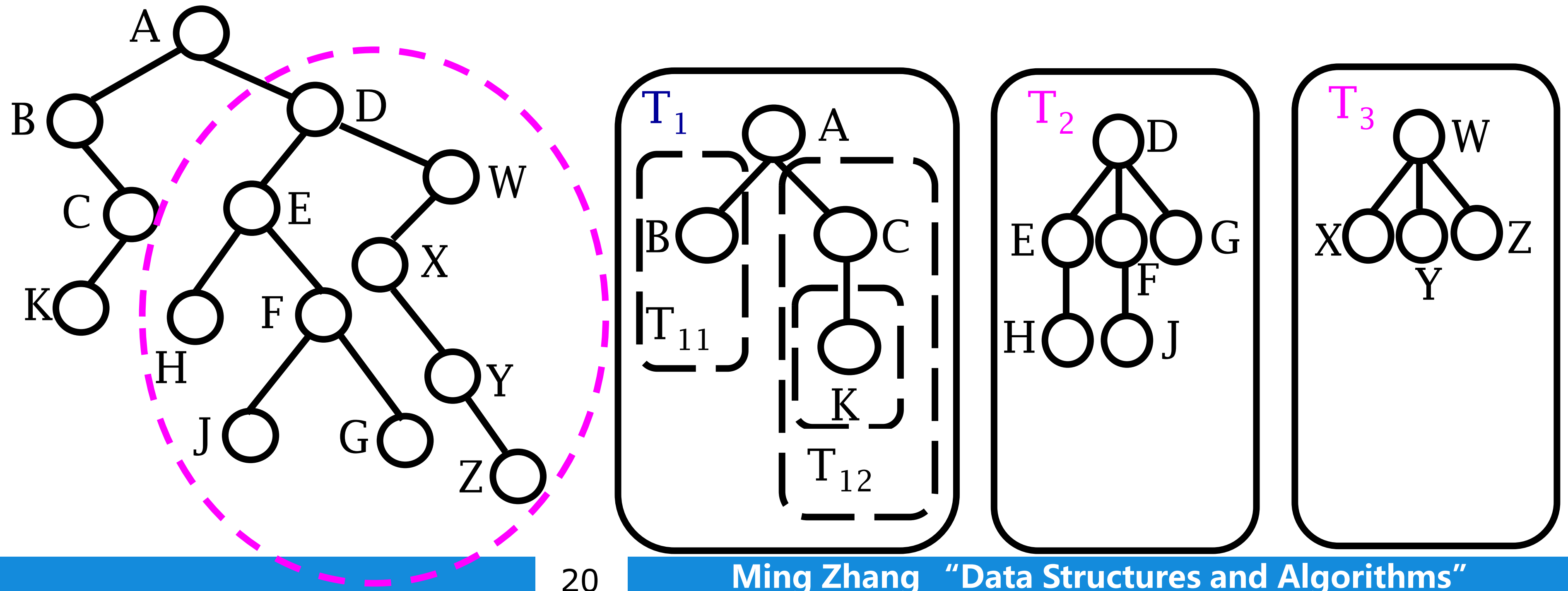


Convert a forest to a binary tree

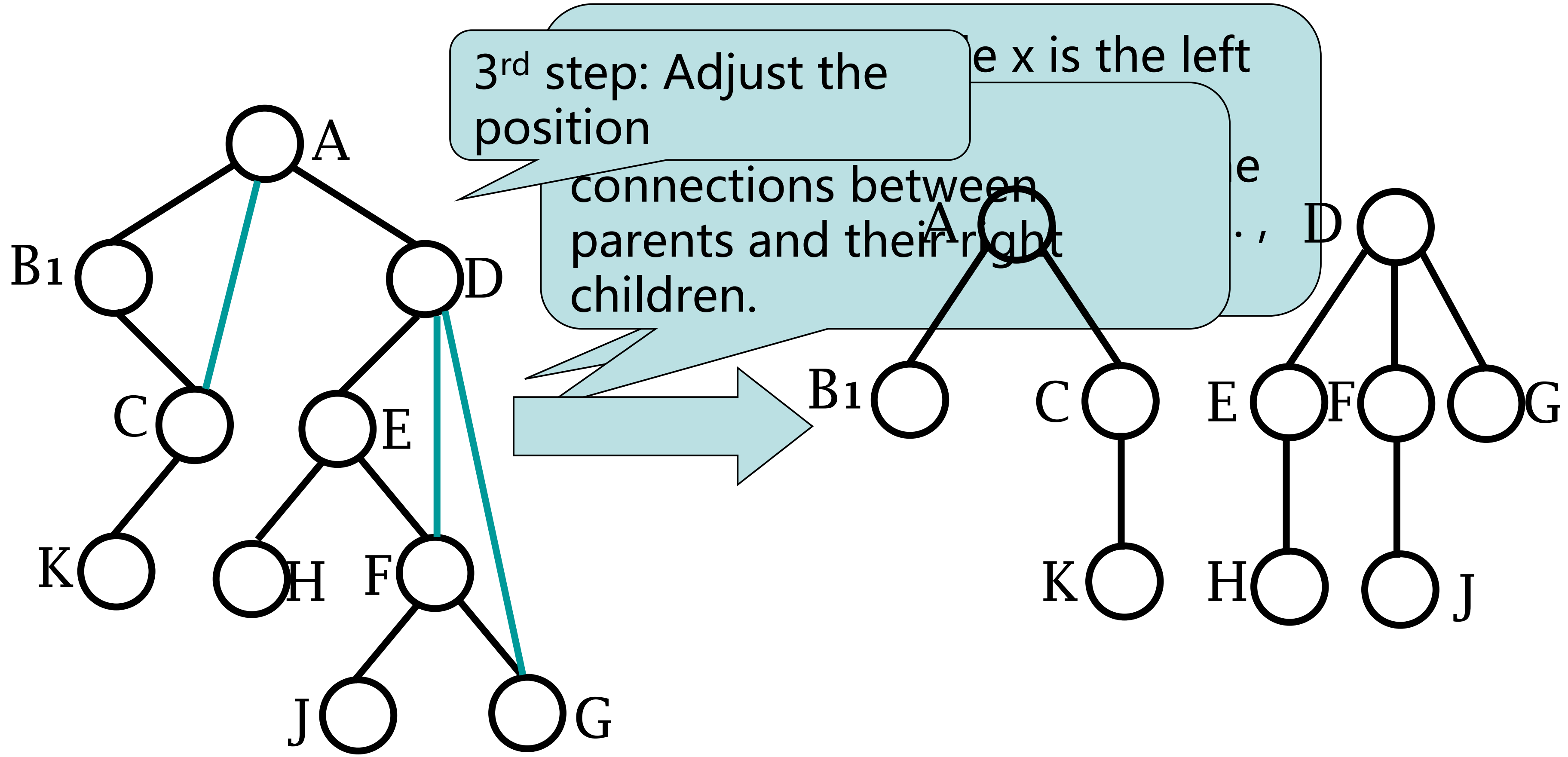


The transformation from a binary tree to a forest

- Assume B is a binary tree, r is the root of B, B_L is the left sub-tree of r, B_R is the right sub-tree of r. We can transform B to a corresponding forest F(B) as follows,
  - If B is empty, F(B) is an empty forest.
  - If B is not empty, F(B) consists of trees $\{T_1\} \cup F(B_R)$, where the root of T_1 is r, the subtrees of r are $F(B_L)$



Convert a binary tree to a forest



Questions

1. Is a tree also a forest?

1. Why do we establish the one-to-one mapping between binary trees and forests?

Data Structures and Algorithms

Thanks

the National Elaborate Course (Only available for IPs in China)
http://www.jpk.pku.edu.cn/pkujpk/course/sjjg/

Ming Zhang, Tengjiao Wang and Haiyan Zhao

Higher Education Press, 2008.6 (awarded as the "Eleventh Five-Year" national planning textbook)