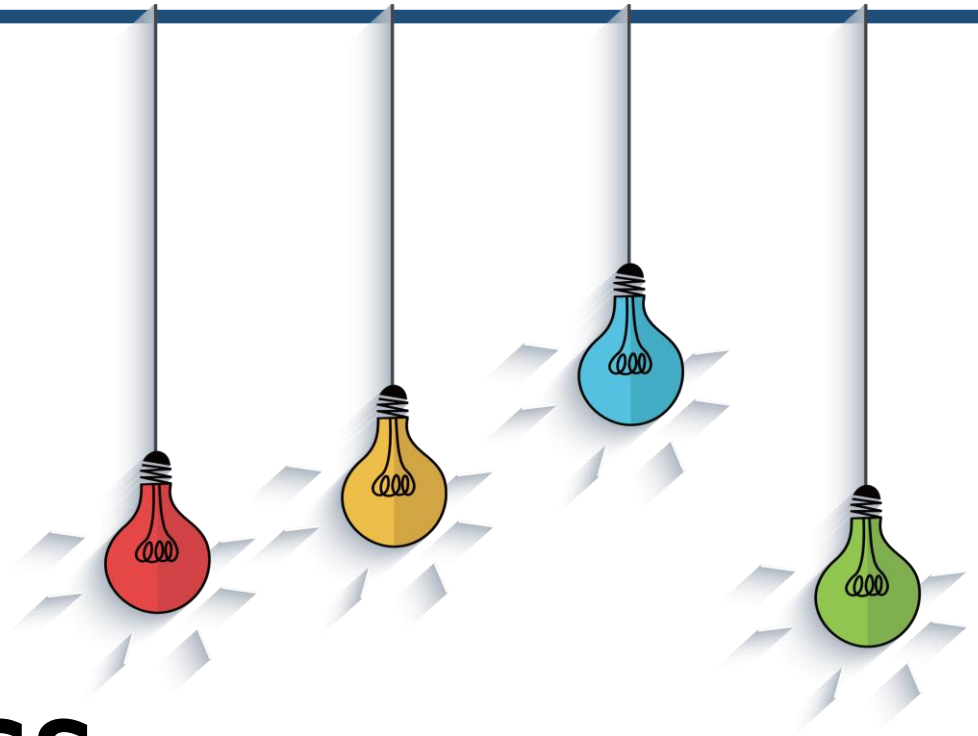
UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA



IF120
Discrete Mathematics

13 Trees 2

Angga Aditya Permana, Januar Wahjudi, Yaya Suryana, Meriske, Muhammad Fahrury Romdendine

REVIEW

- Trees' Terminologies
- Spanning Trees
- Binary Trees

OUTLINE

- Tree Traversals
- Decision Trees
- Trees Isomorphism
- Game Trees

Tree Traversals

- Breadth-first search and depth-first search provide ways to "walk" a tree, that is, to traverse a tree in a systematic way so that each vertex is visited exactly once.
- Now, we consider three additional tree-traversal methods, i.e.

1. **Preorder Traversal**
2. **Inorder Traversal**
3. **Postorder Traversal**

Preorder Traversals

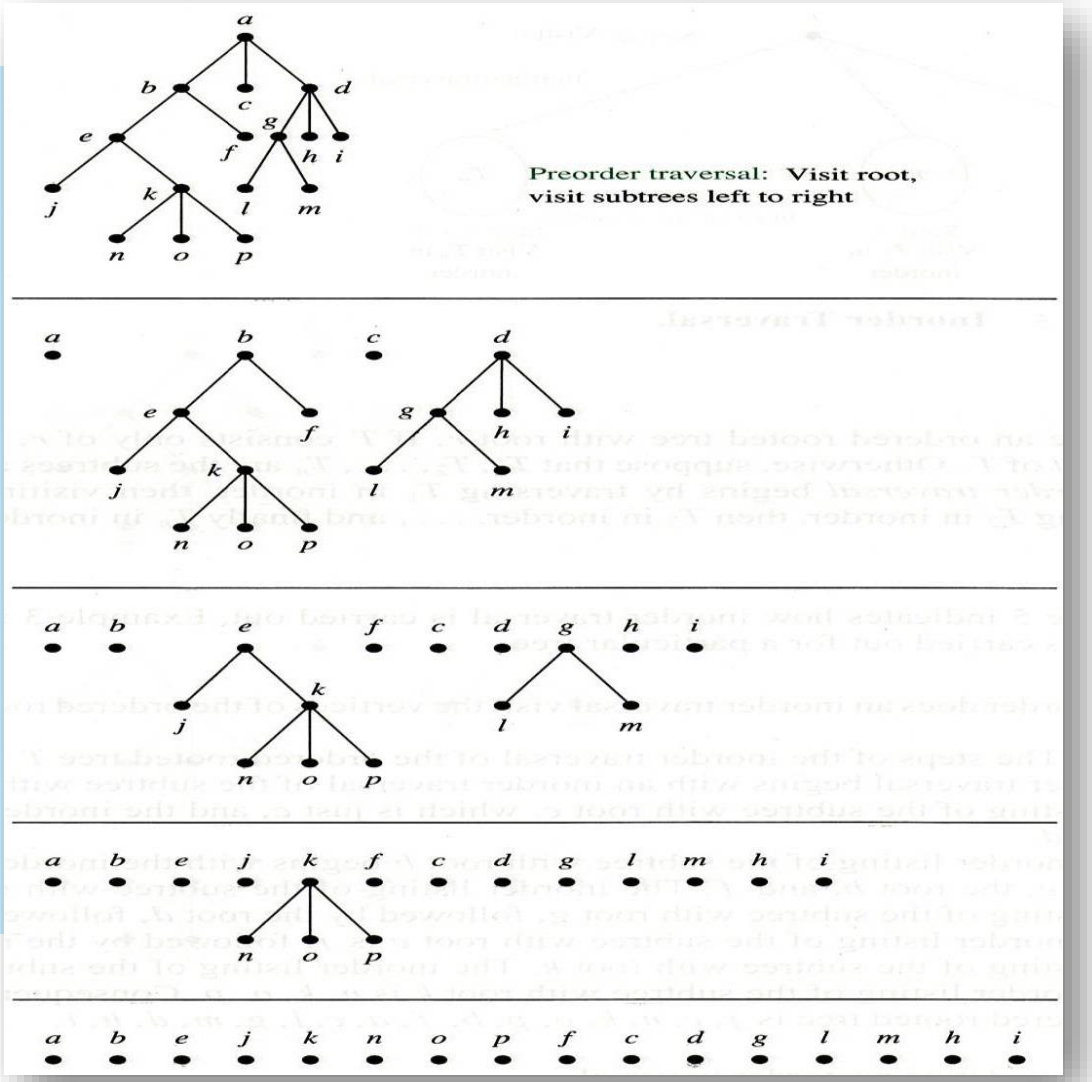
Preorder Traversal

This recursive algorithm processes the vertices of a binary tree using preorder traversal.

Input: *PT*, the root of a binary tree, or the special value *null* to indicate that no tree is input

Output: Dependent on how "process" is interpreted in line 3

```
    preorder(PT) {
1.      if (PT == null)
2.          return
3.      process PT
4.      l = left child of PT
5.      preorder(l)
6.      r = right child of PT
7.      preorder(r)
    }
```



Inorder Traversals

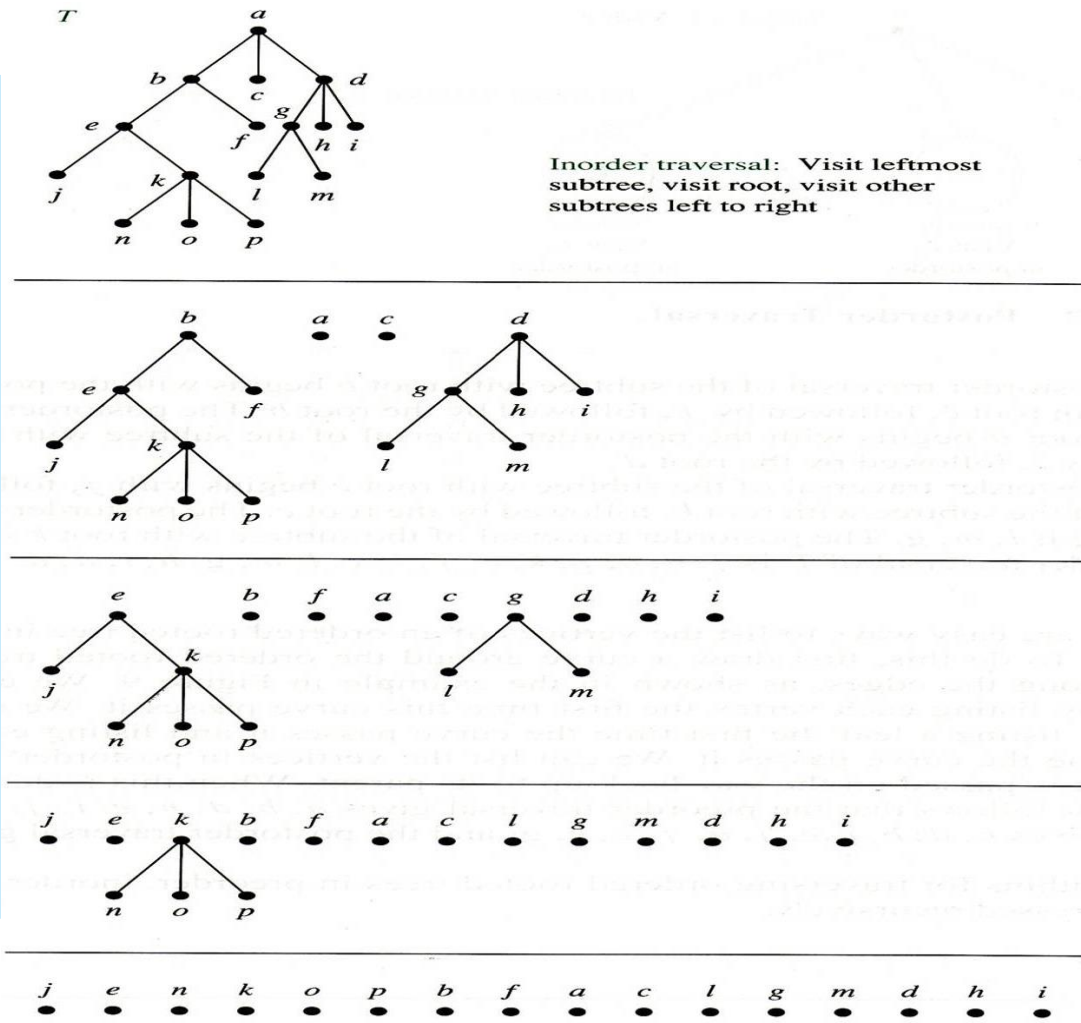
Inorder Traversal

This recursive algorithm processes the vertices of a binary tree using inorder traversal.

Input: *PT*, the root of a binary tree, or the special value *null* to indicate that no tree is input

Output: Dependent on how "process" is interpreted in line 5

```
inorder(PT) {
1.    if (PT == null)
2.       return
3.    l = left child of PT
4.    inorder(l)
5.    process PT
6.    r = right child of PT
7.    inorder(r)
}
```



Postorder Traversals

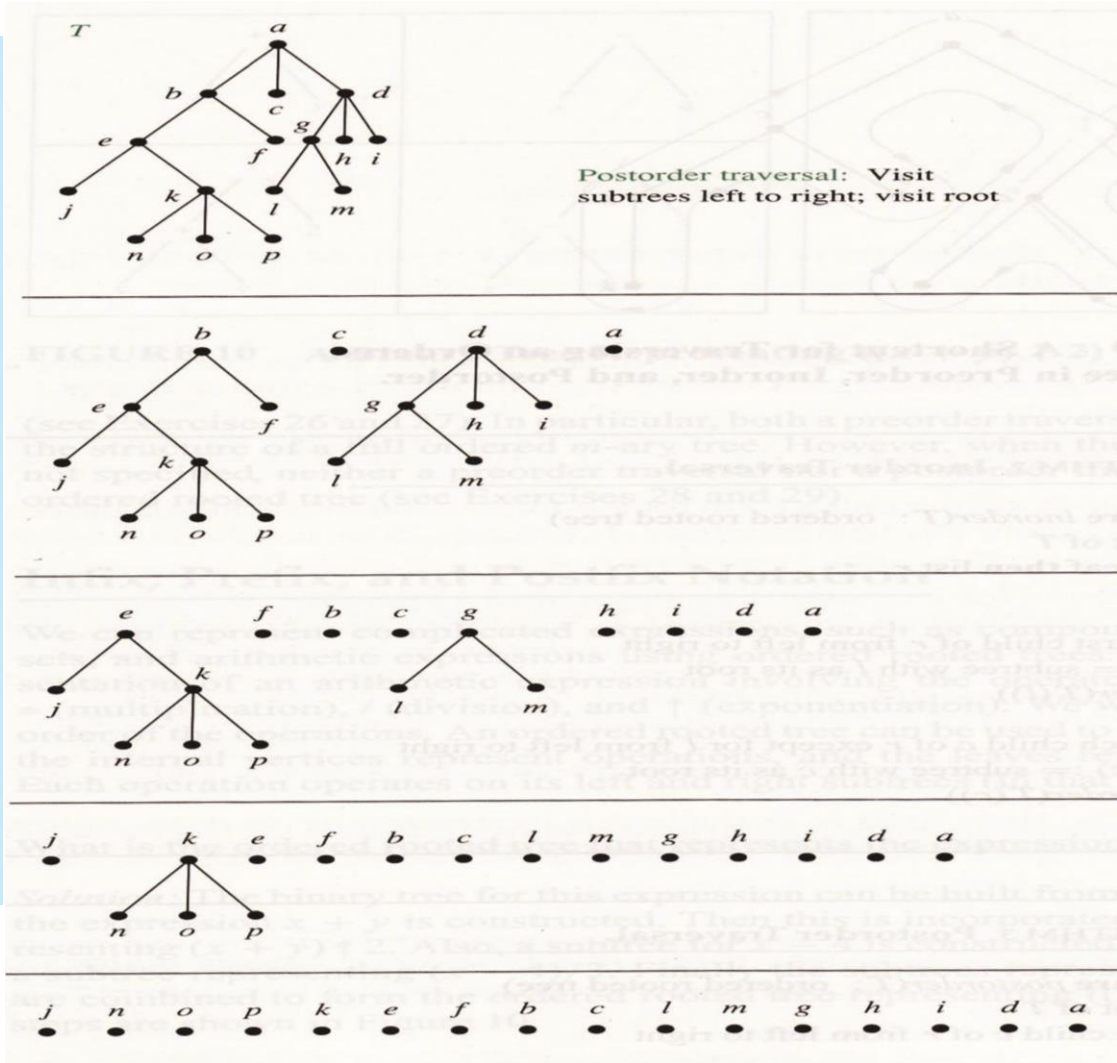
Postorder Traversal

This recursive algorithm processes the vertices of a binary tree using postorder traversal.

Input: *PT*, the root of a binary tree, or the special value *null* to indicate that no tree is input

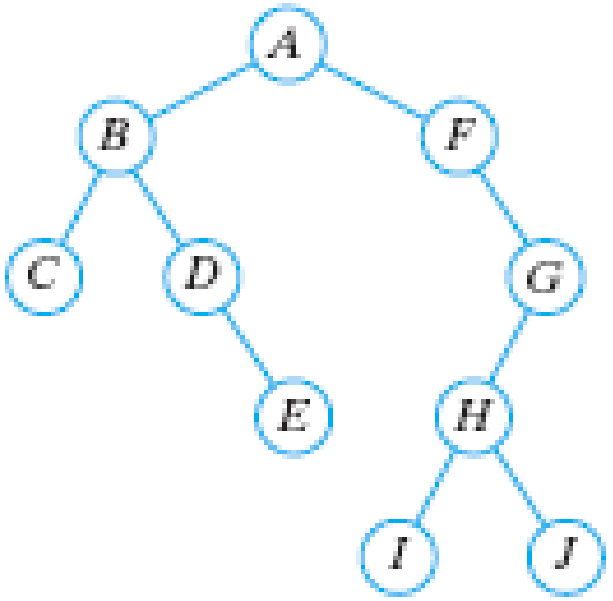
Output: Dependent on how "process" is interpreted in line 7

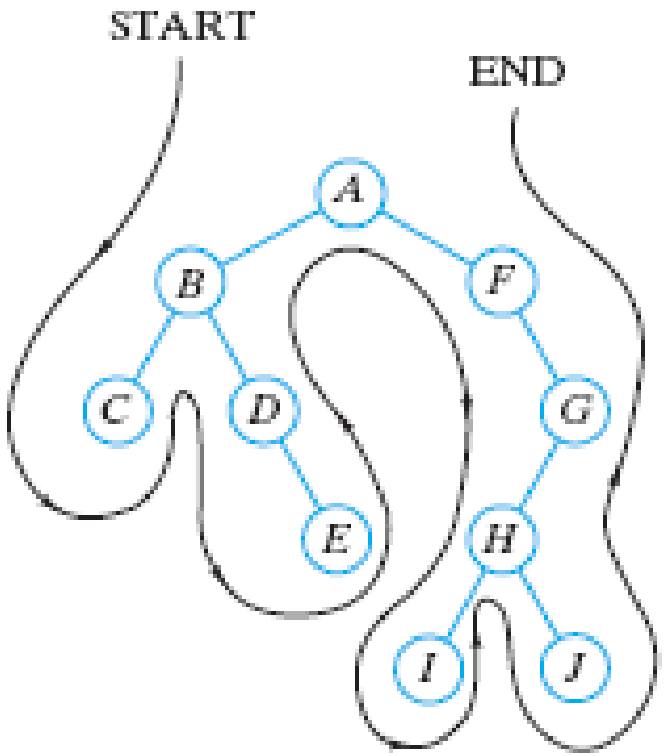
```
    postorder(PT) {
1.      if (PT == null)
2.          return
3.      l = left child of PT
4.      postorder(l)
5.      r = right child of PT
6.      postorder(r)
7.      process PT
    }
```



Tree Traversals

- In what order are the vertices of the tree of Figure below processed if preorder, inorder, and postorder traversals are used?

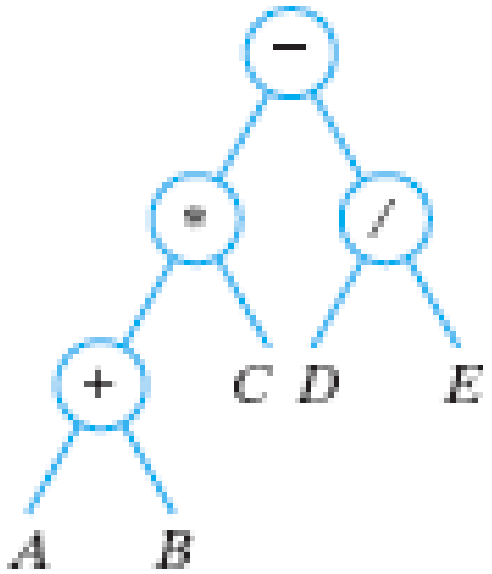




- Preorder Traversal:
  A B C D E F G H I J
- Inorder Traversal:
  C B D E A F I H J G
- Postorder Traversal:
  C E D B I J H G F A

Binary Tree Representations of Arithmetic Expressions

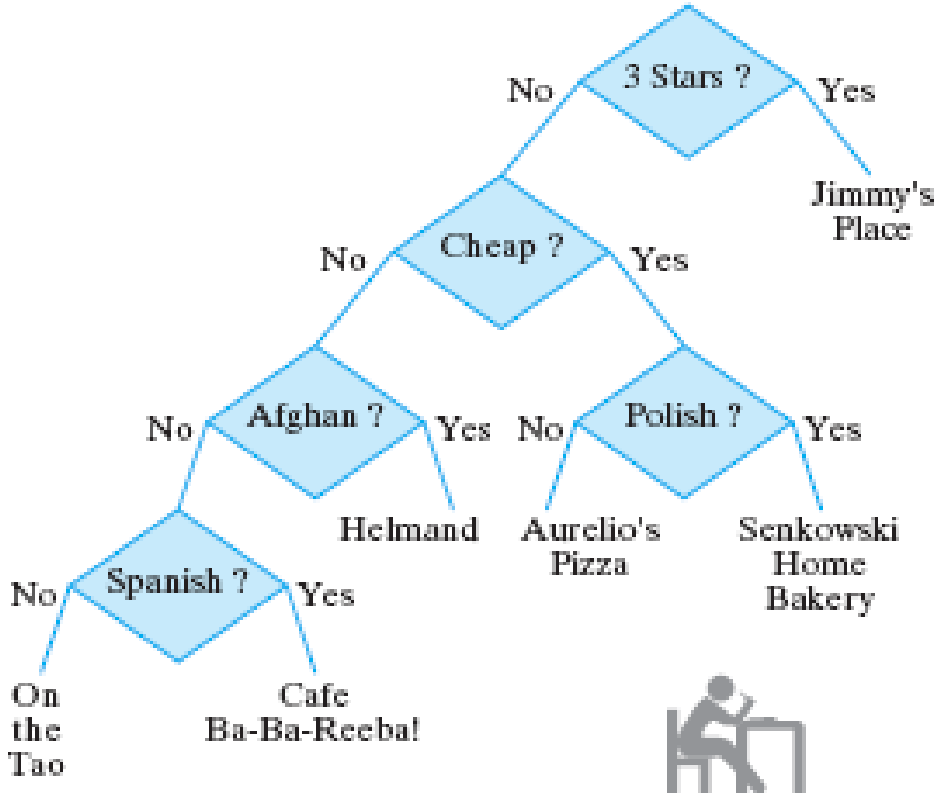
- Next, we consider binary tree representations of arithmetic expressions.
- We will restrict our operators to +, −, ∗, and /.
- An example of an expression involving these operators is (**Infix form of an expression**)
  $(A + B) * C - D/E$.



- **Fully parenthesized form of the expression:**
  $(((A + B) * C) - (D/E))$.
- **Postfix form of the expression** (or **reverse Polish notation**)
  $AB+C*DE/-$
- **Prefix form of the expression** (or **Polish notation**)
  $-*+ABC/DE$

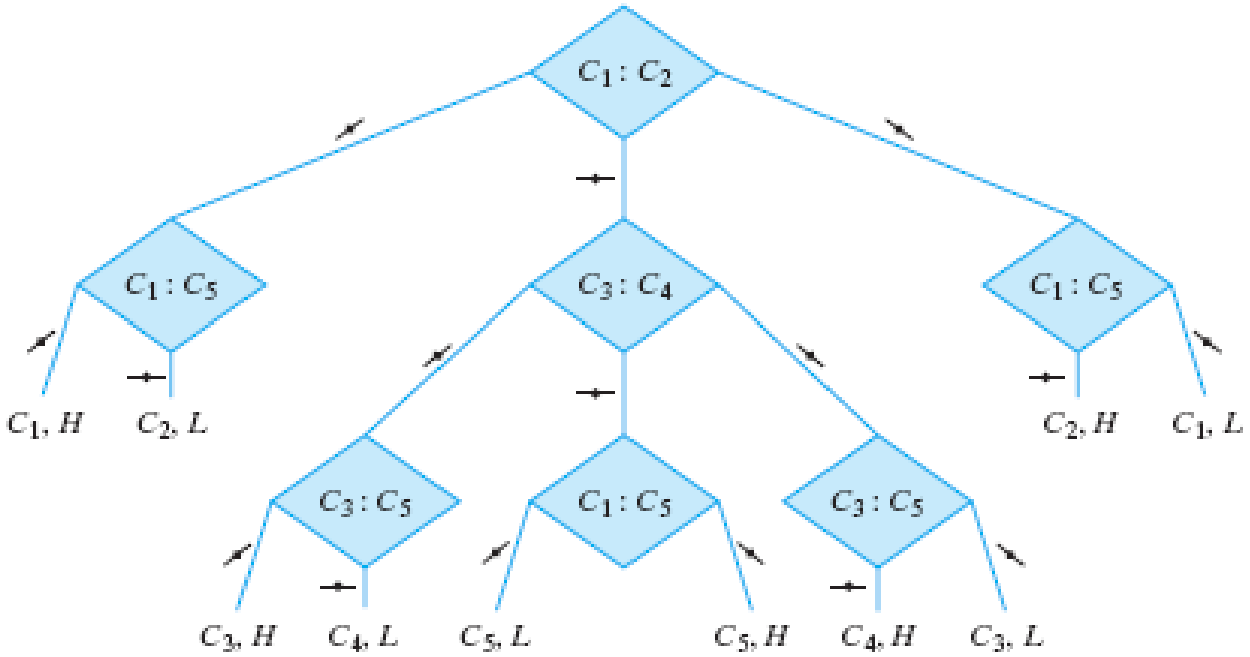
Decision Tree

- The binary tree of Figure below gives an algorithm for choosing a restaurant. Each internal vertex asks a question. If we begin at the root, answer each question, and follow the appropriate edge, we will eventually arrive at a terminal vertex that chooses a restaurant.
- Such a tree is called a **decision tree.**



Decision Tree

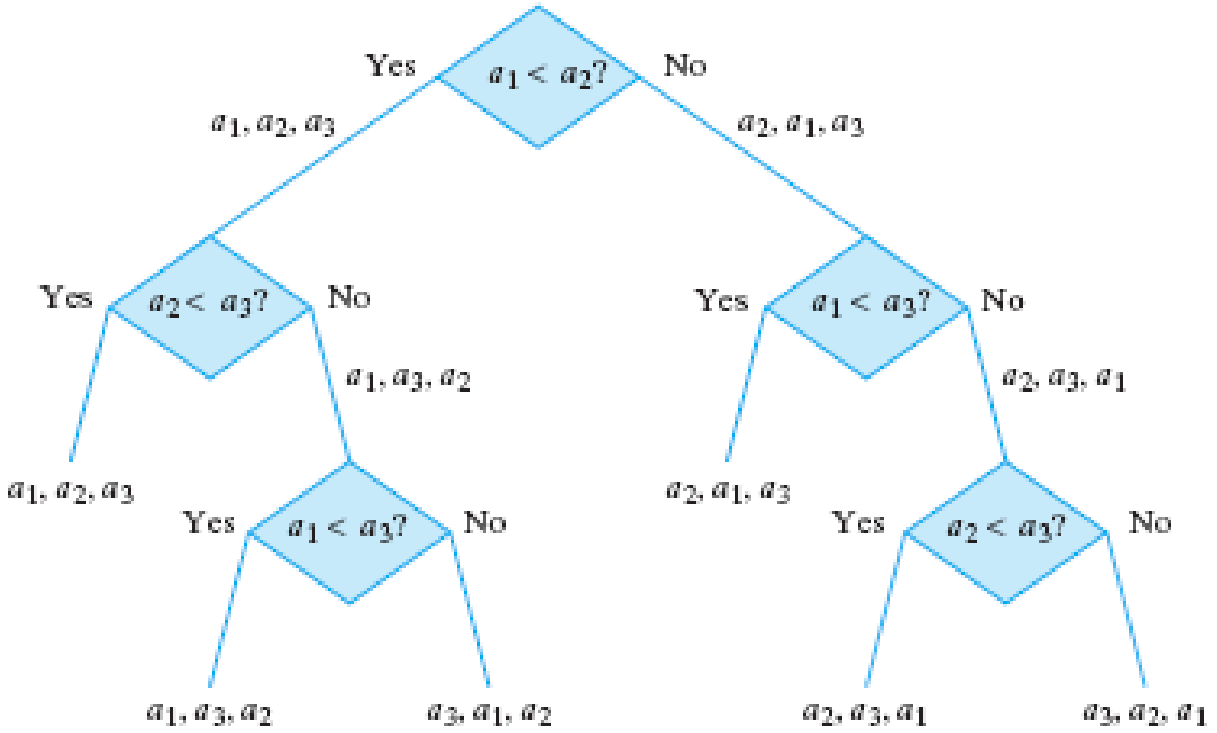
- Five coins are identical in appearance, but one coin is either heavier or lighter than the others, which all weigh the same. The problem is to identify the bad coin and determine whether it is heavier or lighter than the others using only a pan balance, which compares the weights of two sets of coins.



- If we define the worst-case time to solve a coin-weighing problem to be the number of weightings required in the worst case, it is easy to determine the worst-case time from the decision tree; the worst case time is equal to the height of the tree.

Decision Tree

- An algorithm to sort a_1, a_2, a_3 is given by the decision tree of Figure below. Each edge is labeled with the arrangement of the list based on the answer to the question at an internal vertex. The terminal vertices give the sorted order.



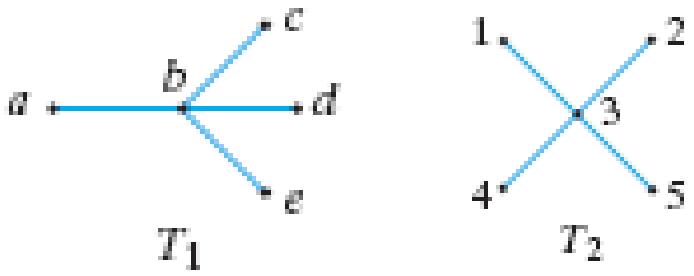
Isomorphisms of Trees

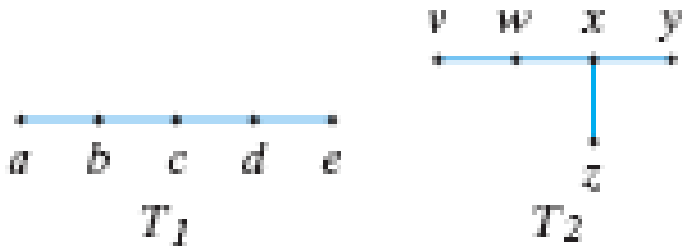
- In the two previous weeks, we defined what it means for two graphs to be isomorphic.
- In this section we discuss **isomorphic trees**, **isomorphic rooted trees**, and **isomorphic binary trees**.

❑ The function f from the vertex set of the tree T_1 to the vertex set of the tree T_2 shown in Figure below (left) defined by

$$f(a) = 1, f(b) = 3, f(c) = 2, f(d) = 4, f(e) = 5$$

is a one-to-one, onto function that preserves the adjacency relation. Thus the trees T_1 and T_2 are isomorphic.





- The trees T_1 and T_2 of Figure above (right) are not isomorphic because T_2 has a vertex (x) of degree 3, but T_1 does not have a vertex of degree 3.

Isomorphisms of Trees

- **Definition 13.1**: Let T_1 be a rooted tree with root r_1 and let T_2 be a rooted tree with root r_2. The rooted trees T_1 and T_2 are *isomorphic* if there is a one-to-one, onto function f from the vertex set of T_1 to the vertex set of T_2 satisfying the following:

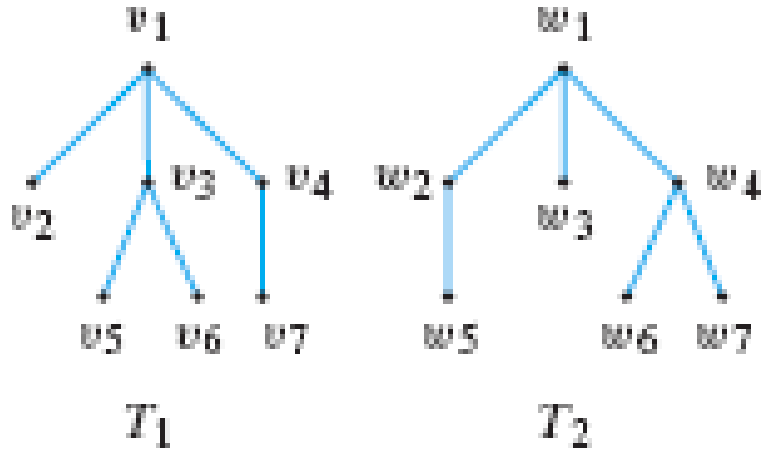
a) Vertices v_i and v_j are adjacent in T_1 if and only if the vertices $f(v_i)$ and $f(v_j)$ are adjacent in T_2.

b) $f(r_1) = r_2$.

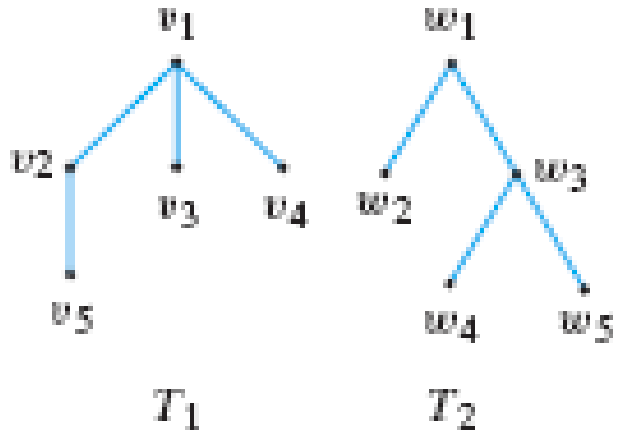
- We call the function f an **isomorphism**.

❑ The rooted trees T_1 and T_2 in Figure on the right are isomorphic. An isomorphism is

$f(v_1) = w_1, f(v_2) = w_3, f(v_3) = w_4, f(v_4) = w_2, f(v_5) = w_7, f(v_6) = w_6, f(v_7) = w_5$



Isomorphisms of Trees



- The rooted trees T_1 and T_2 of Figure on the right are not isomorphic since the root of T_1 has degree 3 but the root of T_2 has degree 2. These trees are isomorphic as *free* trees.

- **Definition 13.2**: Let T_1 be a binary tree with root r_1 and let T_2 be a binary tree with root r_2. The binary trees T_1 and T_2 are **isomorphic** if there is a one-to-one, onto function f from the vertex set of T_1 to the vertex set of T_2 satisfying the following:

a) Vertices v_i and v_j are adjacent in T_1 if and only if the vertices $f(v_i)$ and $f(v_j)$ are adjacent in T_2.

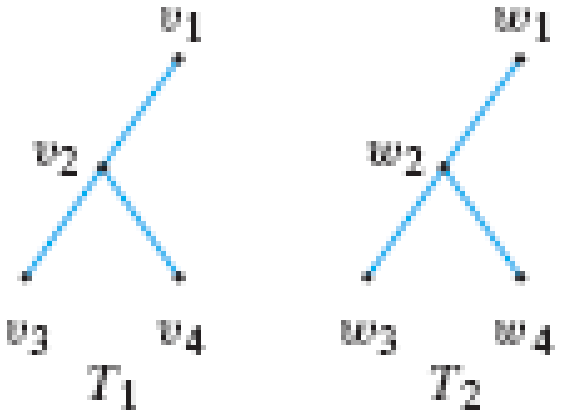
b) $f(r_1) = r_2$.

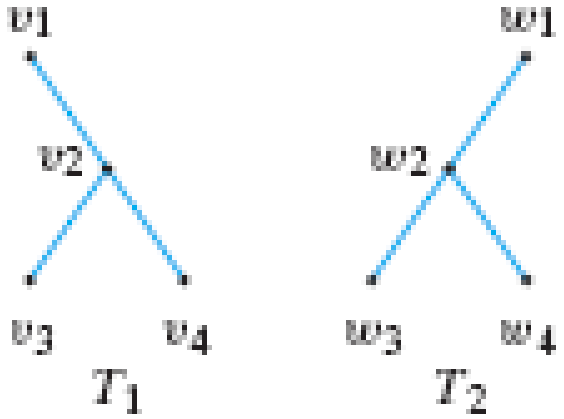
c) v is a left child of w in T_1 if and only if $f(v)$ is a left child of $f(w)$ in T_2.

d) v is a right child of w in T_1 if and only if $f(v)$ is a right child of $f(w)$ in T_2.

Isomorphisms of Trees

- The binary trees T_1 and T_2 in Figure on the left are isomorphic. The isomorphism is $f(v_i) = w_i$ for $i = 1, \ldots, 4$.





- The binary trees T_1 and T_2 in Figure on the right are not isomorphic. The root v_1 in T_1 has a right child, but the root w_1 in T_2 has no right child.

Game Trees

- Trees are useful in the analysis of games such as tic-tac-toe, chess, and checkers, in which players alternate moves.
- All possible move sequences can be listed in a **game tree.**
- The first player is represented by a box and the second player is represented by a circle.
- The process by which circle seeks the minimum of its children and box seeks the maximum of its children is called the **minimax procedure.**
- Evaluation of a game tree, or even a part of a game tree, can be a time-consuming task, so any technique that reduces the effort is welcomed.
- The most general technique is called **alpha-beta pruning.**
- In general, alpha-beta pruning allows us to bypass many vertices in a game tree yet still find the value of a vertex.
- The value obtained is the same as if we had evaluated all the vertices.

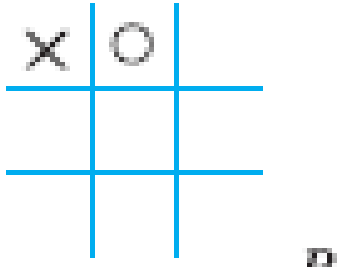
Game Trees

- Apply the minimax procedure to find the value of the root in tic-tac-toe using a two-level, depth-first minimax search. Use the evaluation function *E*, which assigns a position the value

$$NX - NO$$

where *NX* (respectively, *NO*) is the number of rows, columns, or diagonals containing an X (respectively, O) that X (respectively, O) might complete. For example, position *P* of Figure below has *NX* = 2, since X might complete the column or the diagonal, and *NO* = 1, since O can complete only a column. Therefore,

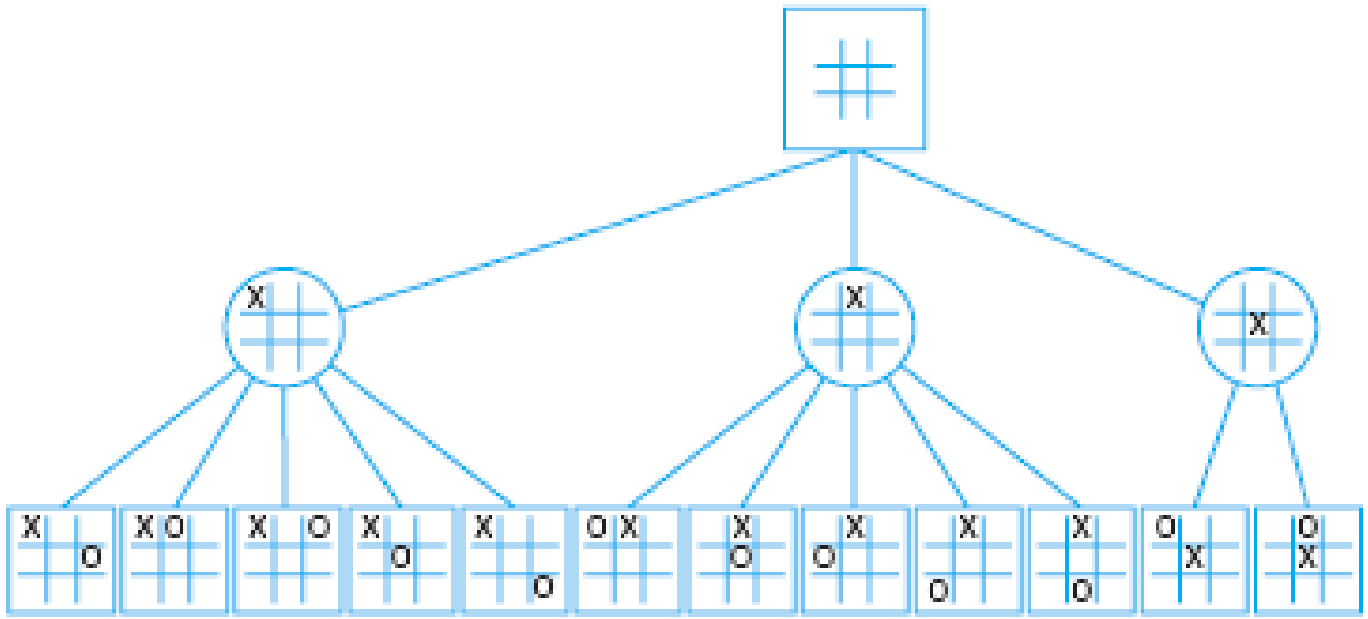
$$E(P) = 2 - 1 = 1.$$

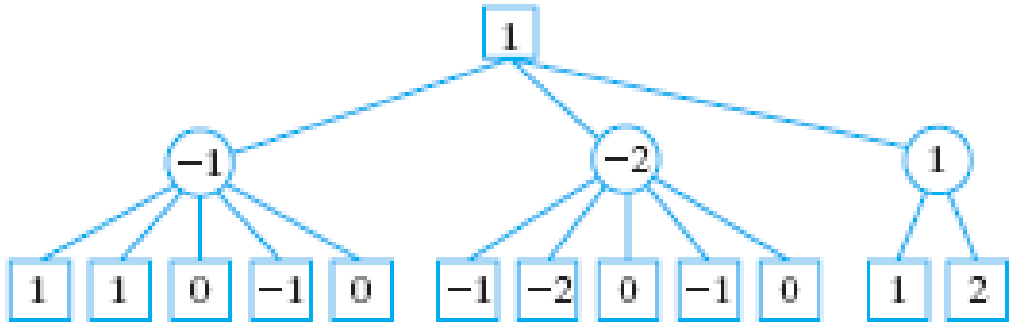


P

Game Trees

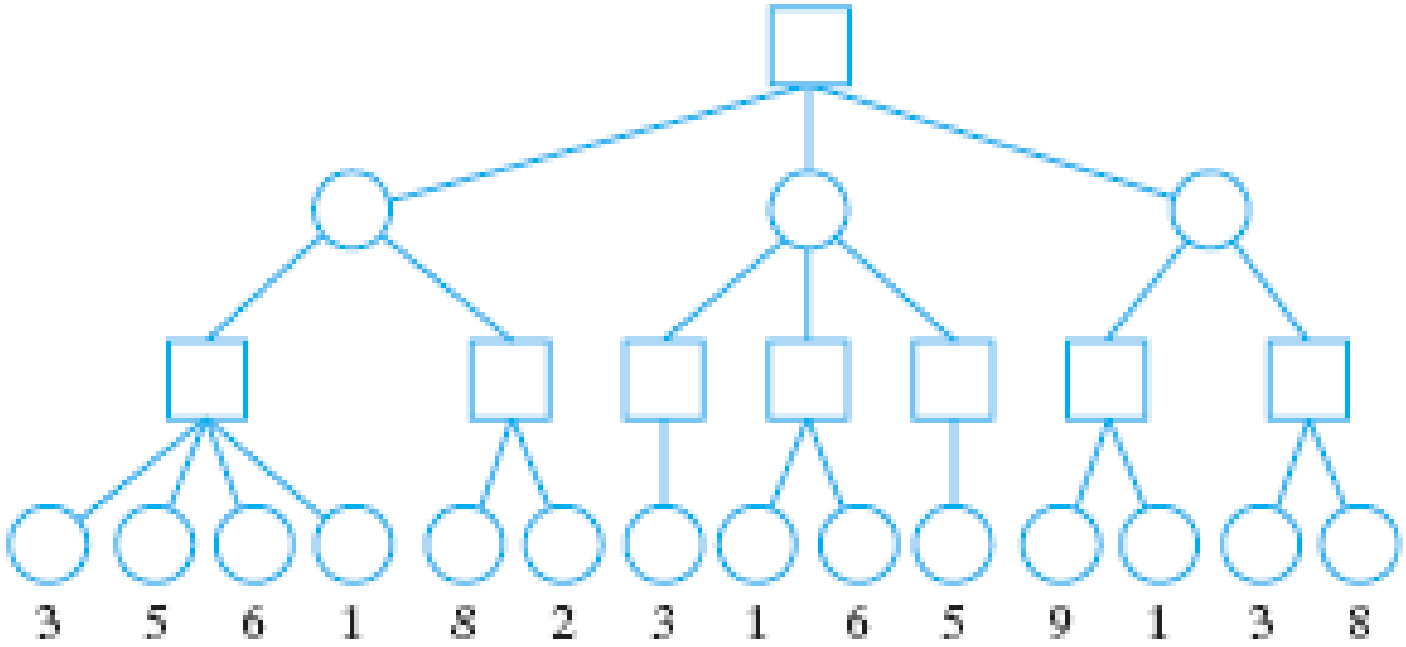
In Figure on the left, we have drawn the game tree for tic-tac-toe to level 2. We have omitted symmetric positions. We first assign to the vertices at level 2 the values given by *E* (see Figure on the right). Next, we compute circle's values by minimizing over the children. Finally, we compute the value of the root by maximizing over the children. Using this analysis, the first move by the first player would be to the center square.



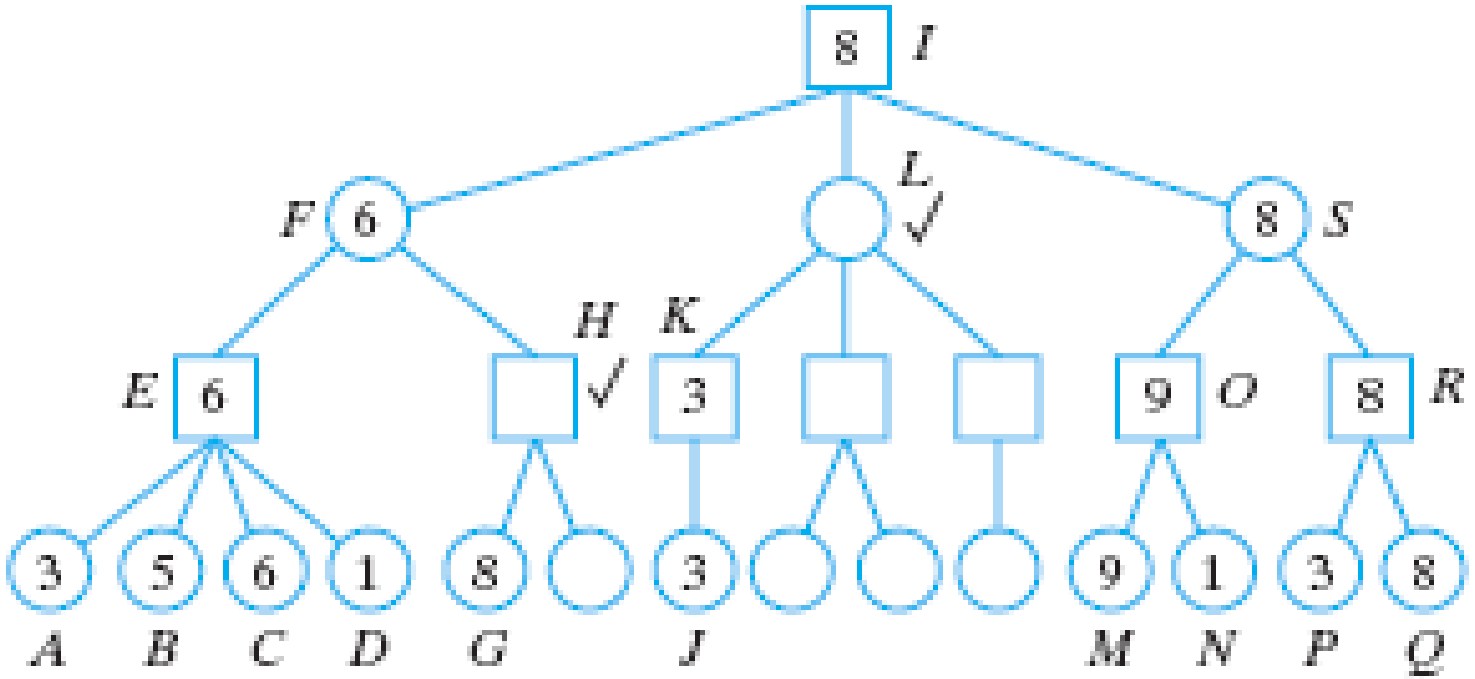


Game Trees

- Evaluate the root of the tree of Figure below using depth-first search with alpha-beta pruning. Assume that children are evaluated left to right. For each vertex whose value is computed, write the value in the vertex. Place a check by the root of each subtree that is pruned. The value of each terminal vertex is written under the vertex.



We begin by evaluating vertices *A*, *B*, *C*, and *D* (see Figure below). Next, we find that the value of *E* is 6. This results in a beta value of 6 for *F*. Next, we evaluate vertex *G*. Since its value is 8 and 8 exceeds the beta value of *F*, we obtain a beta cutoff and prune the subtree with root *H*. The value of *F* is 6. This results in an alpha value of 6 for *I*. Next, we evaluate vertices *J* and *K*. Since the value 3 of *K* is less than the alpha value 6 of *I*, an alpha cutoff occurs and the subtree with root *L* may be pruned. Next, we evaluate *M*, *N*, *O*, *P*, *Q*, *R*, and *S*. No further pruning is possible. Finally, we determine that the root *I* has value 8.

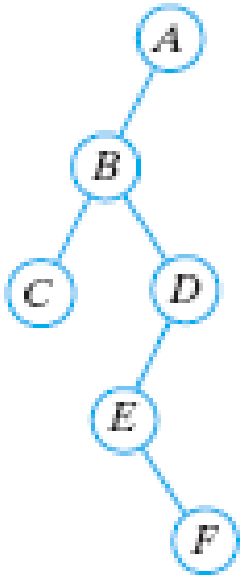


PRACTICE

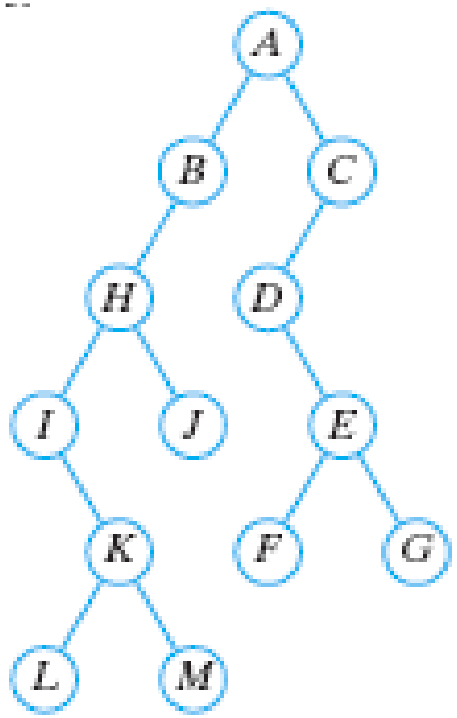
PRACTICE I

- *In Exercises below, list the order in which the vertices are processed using preorder, inorder, and postorder traversal.*

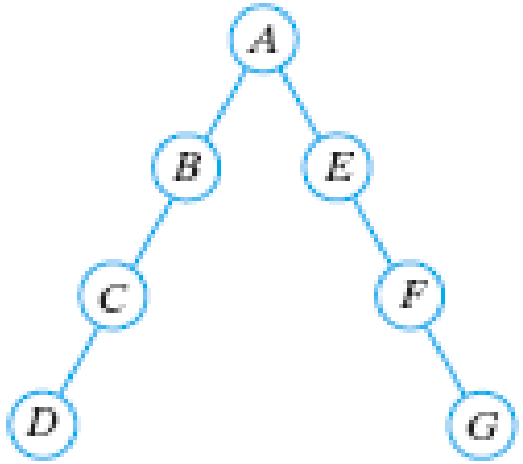
1.



2.



3.



PRACTICE 2

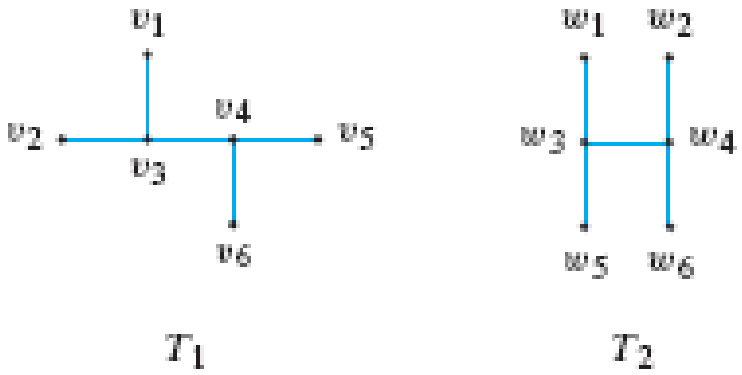
- *In Exercises below, represent the expression as a binary tree and write the prefix and postfix forms of the expression.*

1. $(A + B) * (C - D)$
2. $((A - C) * D) / (A + (B + D))$
3. $(A * B + C * D) - (A/B - (D + E))$

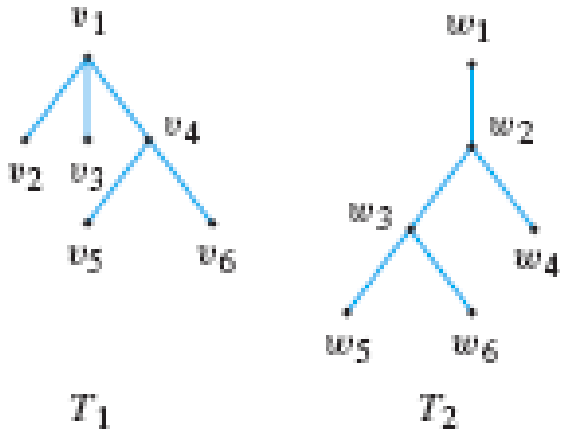
PRACTICE 3

- *In Exercises below, determine whether each pair of free trees is isomorphic. If the pair is isomorphic, specify an isomorphism. If the pair is not isomorphic, give an invariant that one tree satisfies but the other does not.*

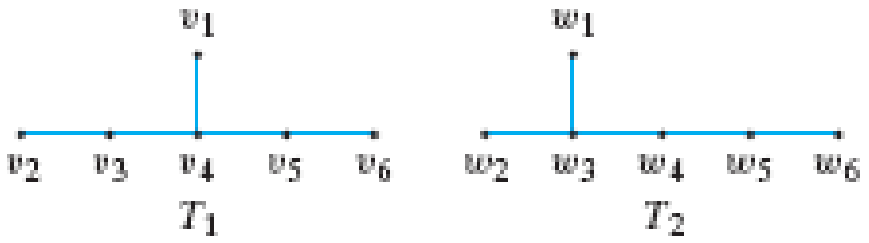
1.



2.



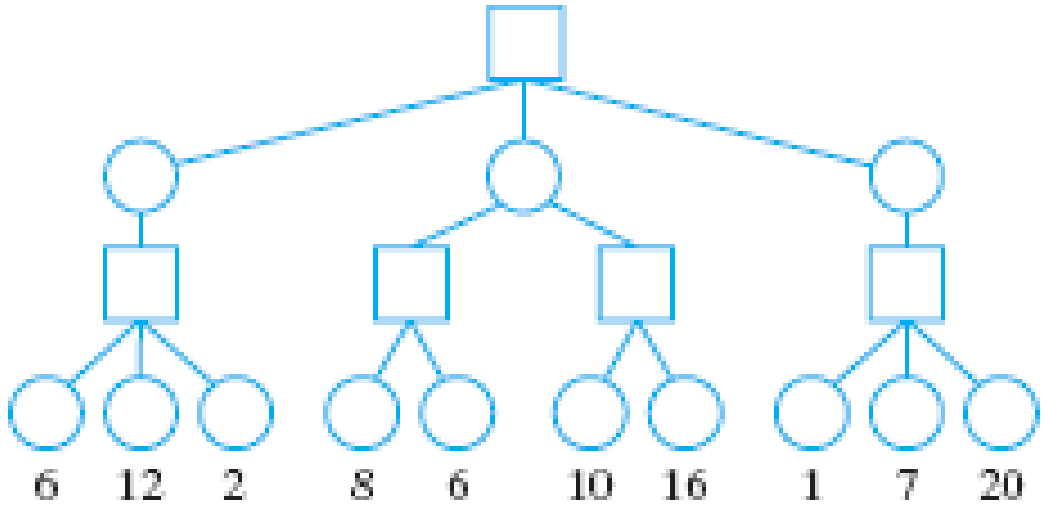
3.



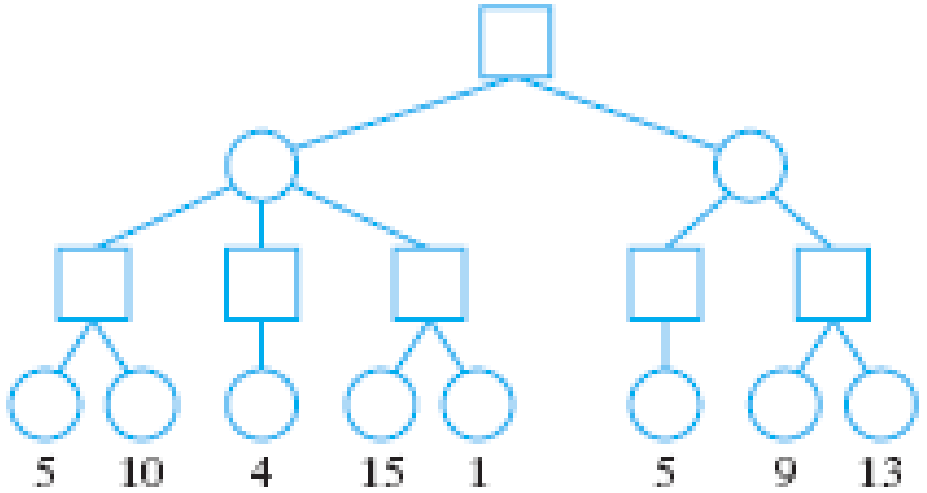
PRACTICE 4

- *Evaluate each vertex in each game tree.The values of the terminal vertices are given.*

1.



2.



NEXT WEEK'S OUTLINE

- Combinatorial Circuits
- Boolean Algebra

REFERENCES

- Johnsonbaugh, R., 2005, *Discrete Mathematics*, New Jersey: Pearson Education, Inc.
- Rosen, Kenneth H., 2005, *Discrete Mathematics and Its Applications*, 6th edition, McGraw-Hill.
- Hansun, S., 2021, *Matematika Diskret Teknik*, Deepublish.
- Lipschutz, Seymour, Lipson, Marc Lars, *Schaum's Outline of Theory and Problems of Discrete Mathematics*, McGraw-Hill.
- Liu, C.L., 1995, *Dasar-Dasar Matematika Diskret*, Jakarta: Gramedia Pustaka Utama.
- Other offline and online resources.

Visi

Menjadi Program Studi Strata Satu Informatika **unggulan** yang menghasilkan lulusan **berwawasan internasional** yang **kompeten** di bidang Ilmu Komputer (*Computer Science*), **berjiwa wirausaha** dan **berbudi pekerti luhur**.

INFORMATIKA
UMN
UNIVERSITAS MULTIMEDIA NUSANTARA

Misi

1. Menyelenggarakan pembelajaran dengan teknologi dan kurikulum terbaik serta didukung tenaga pengajar profesional.
2. Melaksanakan kegiatan penelitian di bidang Informatika untuk memajukan ilmu dan teknologi Informatika.
3. Melaksanakan kegiatan pengabdian kepada masyarakat berbasis ilmu dan teknologi Informatika dalam rangka mengamalkan ilmu dan teknologi Informatika.