LECTURE 22

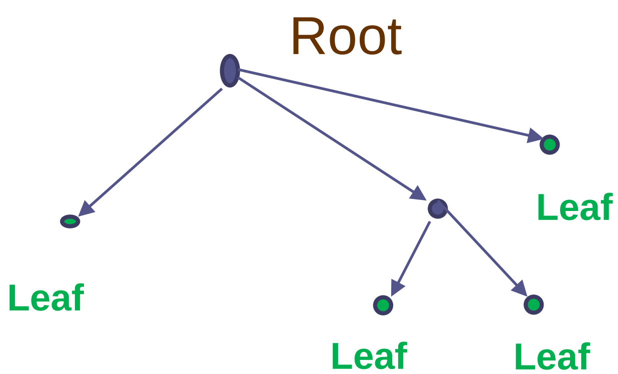
Trees

Assignment

- Read sections 6.1 and 6.2 of Chapter 6
- Do all self-check exercises (easy and … no programming involved)

A tree (definition)

- A directed *acyclic* graph in which every vertex except one has only one incoming edge

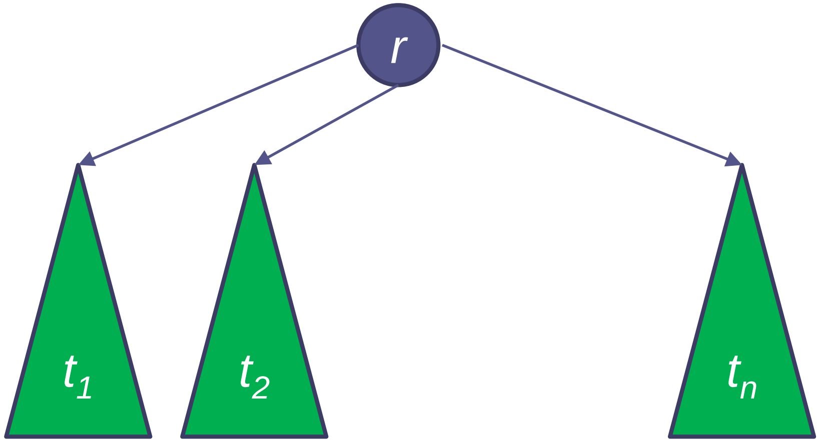


Prove that every tree has at least one leaf.

A recursive definition

A tree may be

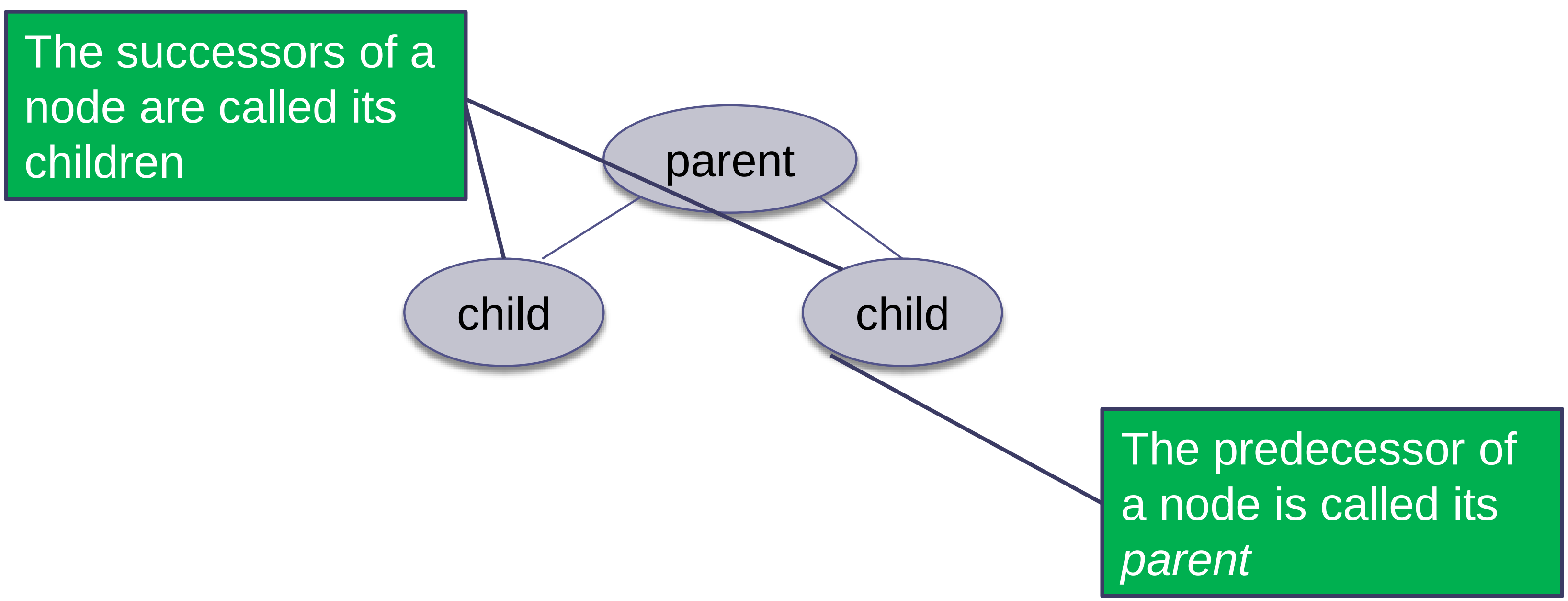
- a single node (*both* the root and the leaf) *r;* or
- $(r, \{t_1, t_2, \ldots t_n\})$, where t_i is a tree.



Binary trees

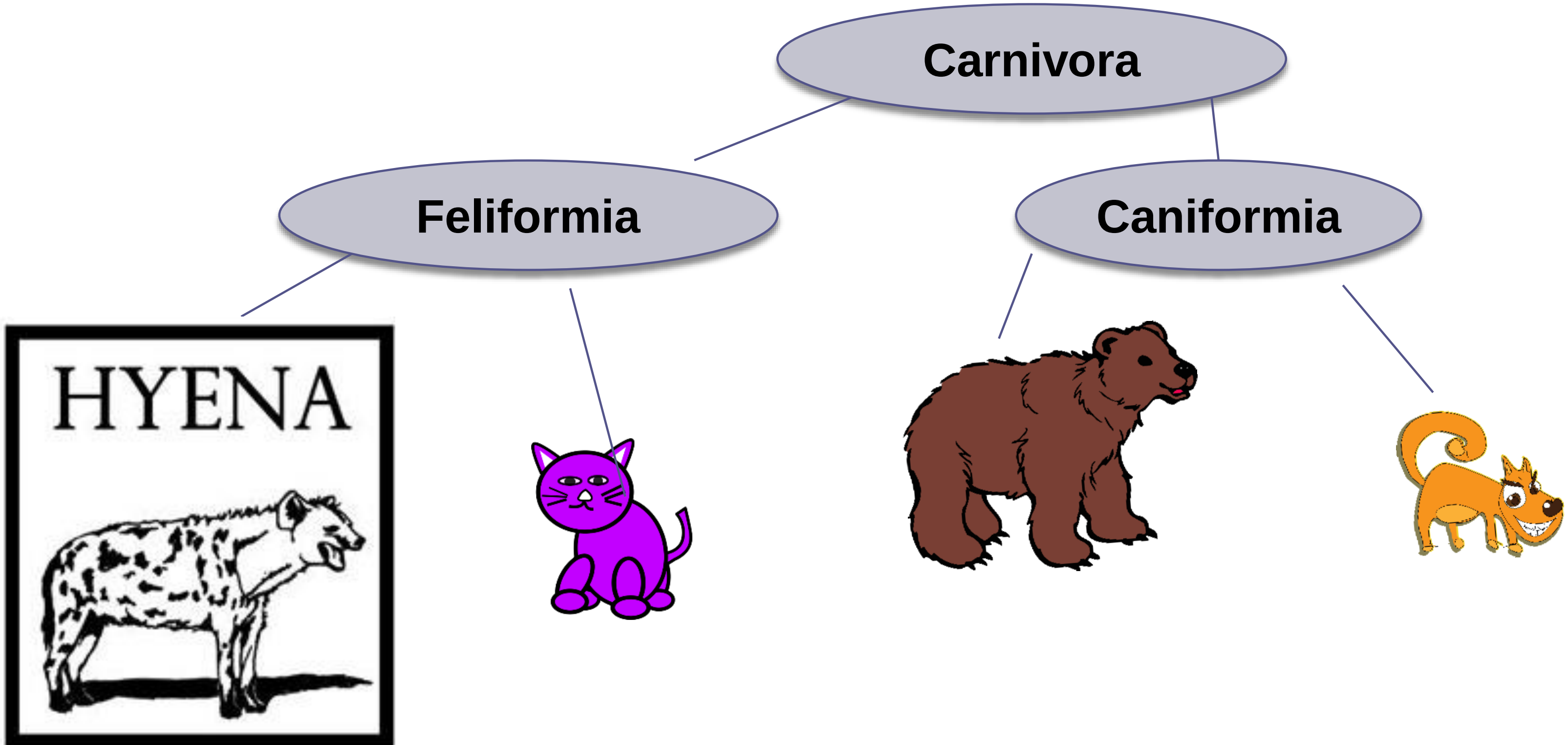
- In a binary tree each non-leaf node has two successors
- Binary trees can be implemented using both arrays and pointer data structures
- Binary trees are effective in supporting sorting and searching

More terminology



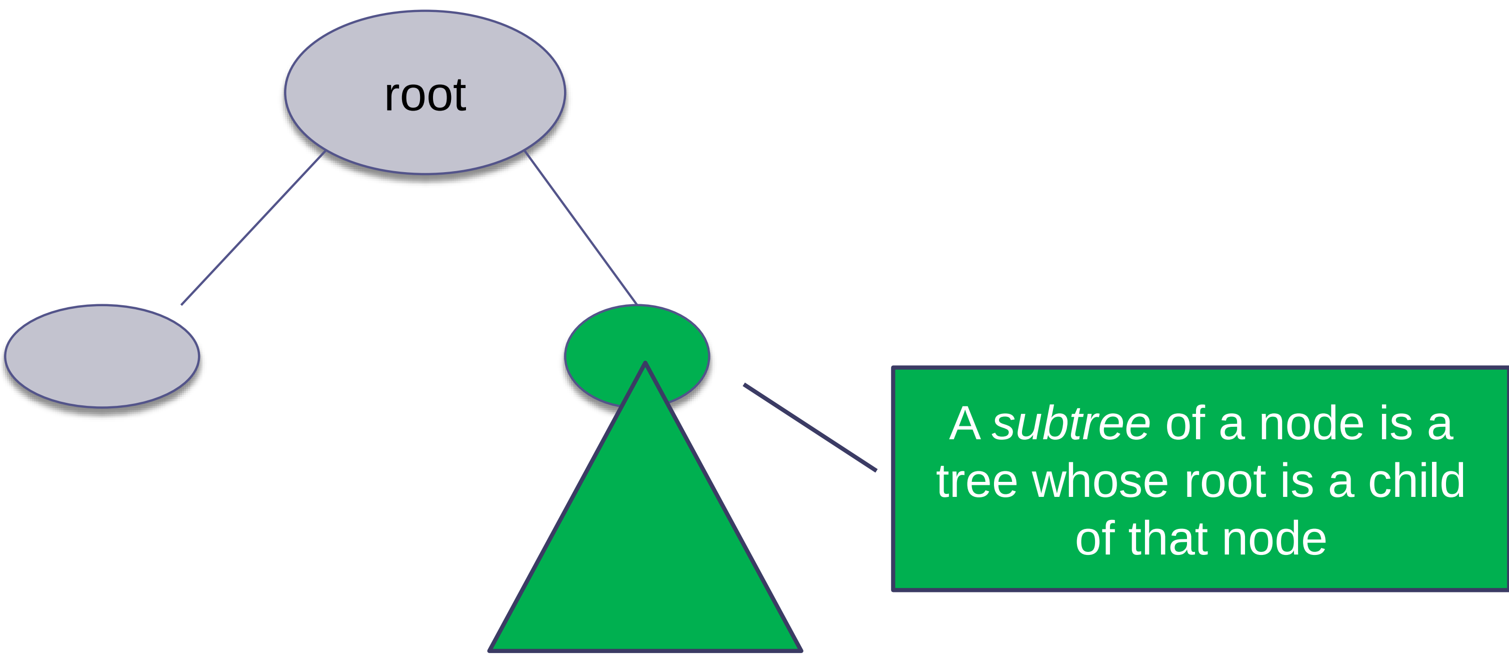
Even more terminology!

A generalization of the parent-child relationship is the *ancestor-descendant relationship*

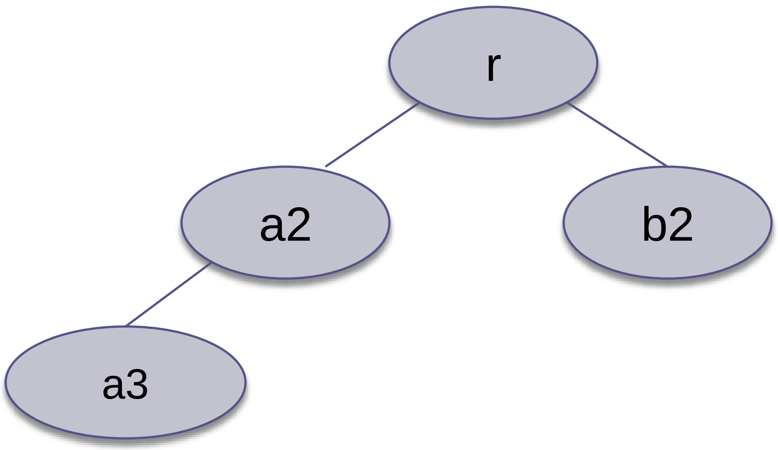


And more

A tree consists of a collection of elements or nodes, with each node linked to its successors

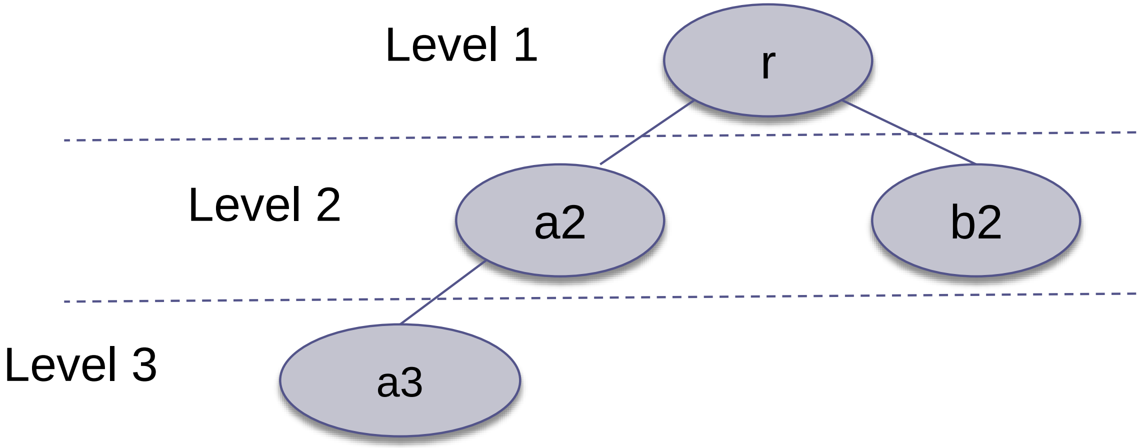


Node level



The *level of a node* is a measure of its distance from the root

A recursive property of level

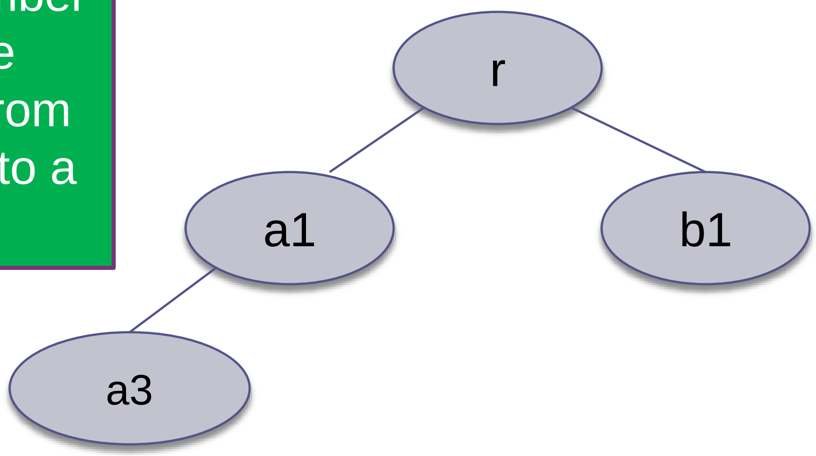


The *level of a node* is defined recursively

- If node *n* is the root of tree T, its level is 1
- If node *n* is not the root of tree T, its level is 1 + the level of its parent

Tree height

The *height of a tree* is the number of nodes in the longest path from the root node to a leaf node

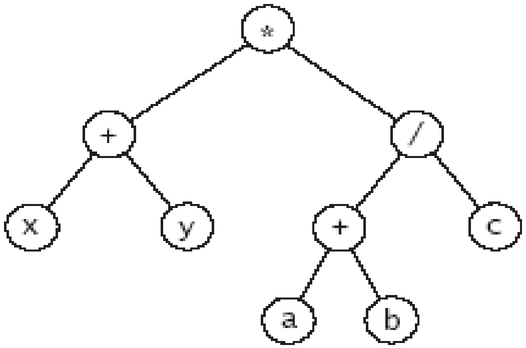


Binary Trees

- In a binary tree, each node has two subtrees
- A set of nodes T is a binary tree if either of the following is true
  - T has only one node, which is both the root and the leaf; or
  - Its root node has two subtrees, T_L and T_R, such that T_L and T_R are binary trees

  (T_L = left subtree; T_R = right subtree)

Expression Tree



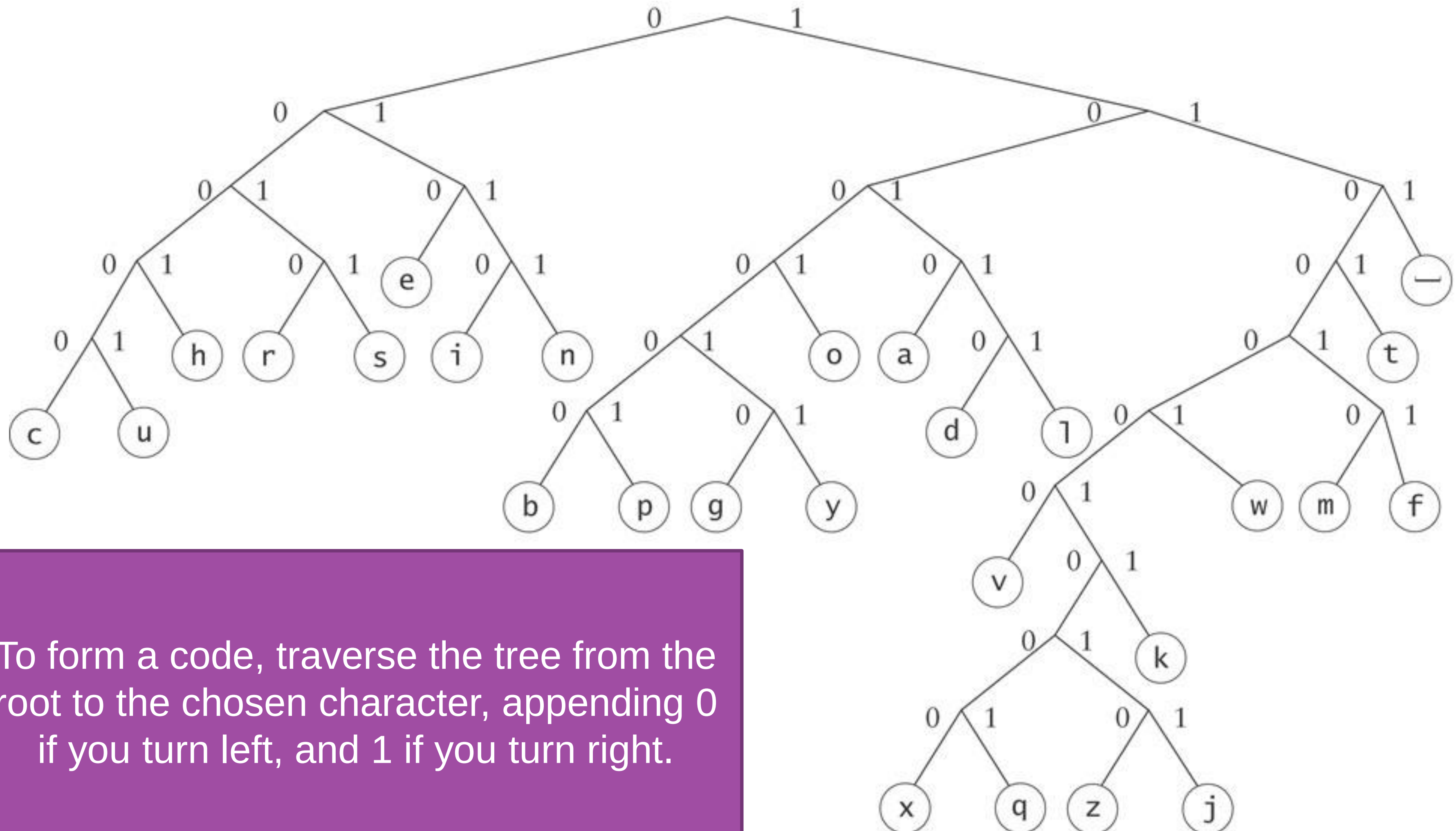
(x + y) * ((a + b) / c)

- Each node contains an operator or an operand
- Operands are stored in leaf nodes
- Parentheses are not stored in the tree because the tree structure dictates the order of operand evaluation
- Operators in nodes at higher levels are evaluated after operators in nodes at lower levels

Huffman Tree

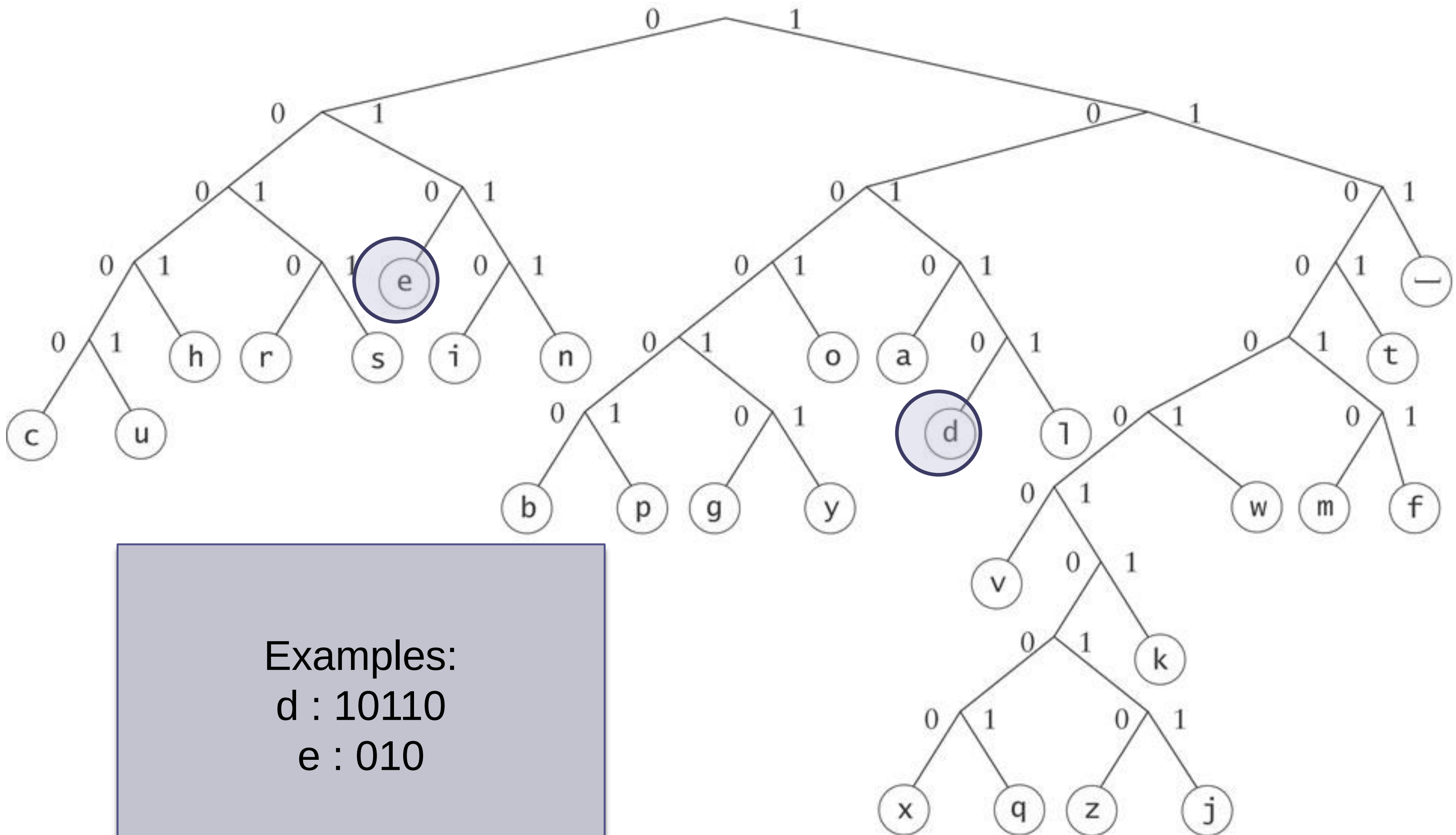
- A *Huffman tree* represents *Huffman codes* for characters that might appear in a text file
- As opposed to ASCII or Unicode, Huffman code uses different numbers of bits to encode letters; more common characters use fewer bits
- Use: Encoding, cryptography, cryptanalysis.

Huffman Tree example



To form a code, traverse the tree from the root to the chosen character, appending 0 if you turn left, and 1 if you turn right.

Huffman Tree example



Examples:
d : 10110
e : 010