Trees

- Trees are the second most important data structure in computing, next to lists
  - Trees are extremely useful for efficiently organizing information and performing many kinds of calculations
- Trees illustrate well goal-oriented programming
  - Many tree data structures are based on a global property, that must be maintained during the calculation
- In this lesson we will define trees and use them to store and look up information
  - We will define ordered binary trees and algorithms to add information, look up information, and remove information

Trees

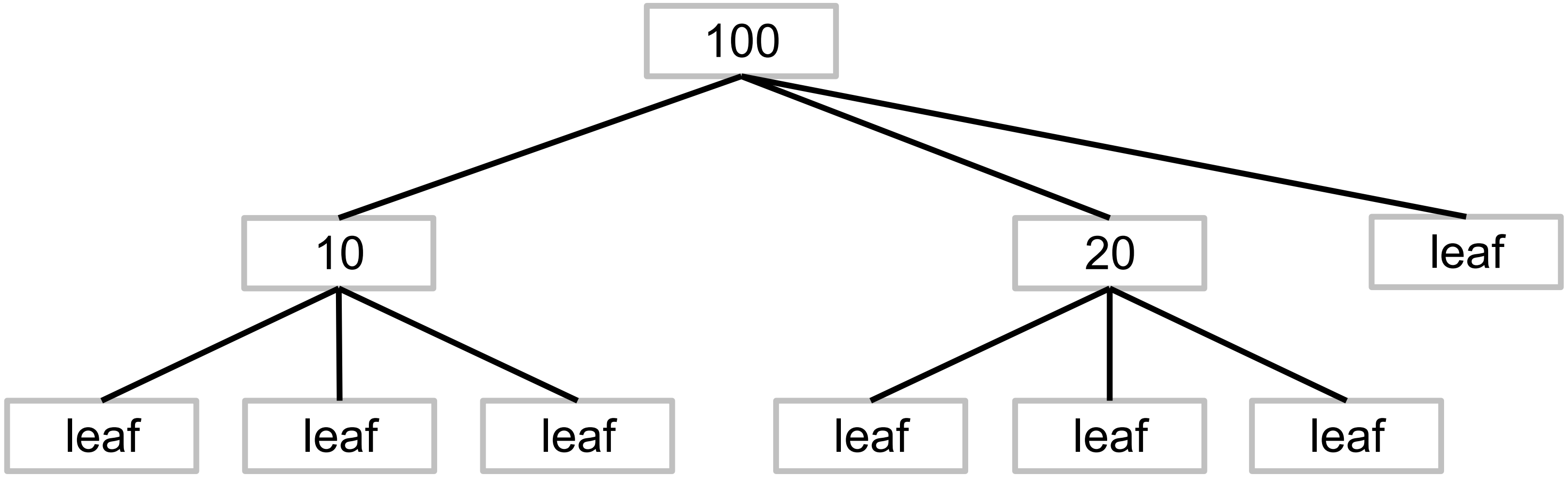
- A tree is a recursive structure: it is either an empty tree (called a leaf) or an element and a set of trees

$$\langle \text{tree } T\rangle ::= \text{leaf} \mid \overbrace{\text{t}(T\ \langle \text{tree } T\rangle\ \dots\ \langle \text{tree } T\rangle)}^{\text{root node}}$$

subtrees

Example tree

- **declare**
  T=t(100 t(10 leaf leaf leaf) t(20 leaf leaf leaf) leaf)



Trees compared to lists

- A tree is a recursive structure: it is either an empty tree (called a leaf) or an element and a set of trees

  <tree T> ::= leaf | t(T <tree T> ... <tree T>)

  <list T> ::= nil | '|'(T <list T>)

Notice the similarity with lists!