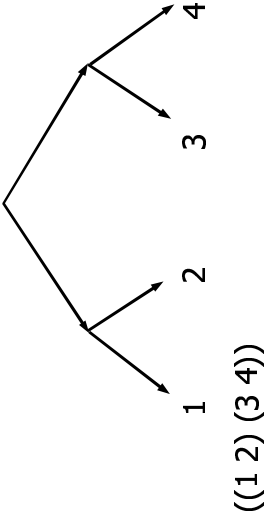


## Hierarchical structures (trees)

- Pairs are like nodes of binary trees
- Binary trees can be used to emulate arbitrary trees



## Tree representation

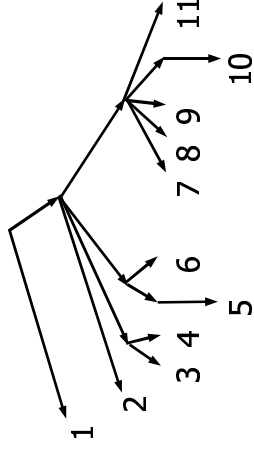
- Data at the leaves represent themselves
- Each nonleaf node is represented by a list of the representations of the children of that node

Trinary tree:

((1 2 3) (4 5 6) (7 8 9))

## An arbitrary tree

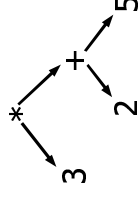
(1 (2 (3 4) ((5) 6) (7 8 9 (10) 11))))



## Labeled trees

- If each node of a tree has a label (or value), we can just make that label (or value) be the first element of the list representing the node

(\* 3 (+ 2 5))



## Trees (inductive definition)

- A primitive data element is a tree, specifically, it is a leaf
  - A nonempty list of trees is a tree
- Operations on trees are usually defined recursively, patterned after this definition

## Counting leaves

```
(define (count-leaves x)
  (cond ((null? x) 0)
        ((not (pair? x)) 1)
        (else
         (+ (count-leaves (car x))
            (count-leaves (cdr x))))))
```

## Multiplying all leaves by the same number

```
(define (scale-tree tree num)
  (cond ((null? tree) ())
        ((not (pair? tree)) (* num tree))
        (else (cons (scale-tree (car tree)
                                 num)
                      (scale-tree (cdr tree)
                                 num)))))

if x --> ((2 1) (4 3)), then
(scale-tree x 5) --> ((10 5) (20 15))
```

## A definition using map

```
(define (scale-tree tree num)
  (if (pair? tree)
      (map (lambda (x)
             (scale-tree x num))
           tree)
      (* num tree))) ; node case
                    ; leaf case

(better than book's definition; works on one-leaf tree)
```

## Solution to exercise 2.32

```
(define (subsets s)
  (if (null? s)
      (list ())
      (let ((rest (subsets (cdr s))))
        (append rest
                  (map (lambda (x)
                        (cons (car s) x))
                       rest)))))

(subsets (list 1 2 3) -->
((() (3) (2) (2 3) (1) (1 3) (1 2) (1 2 3)))
```