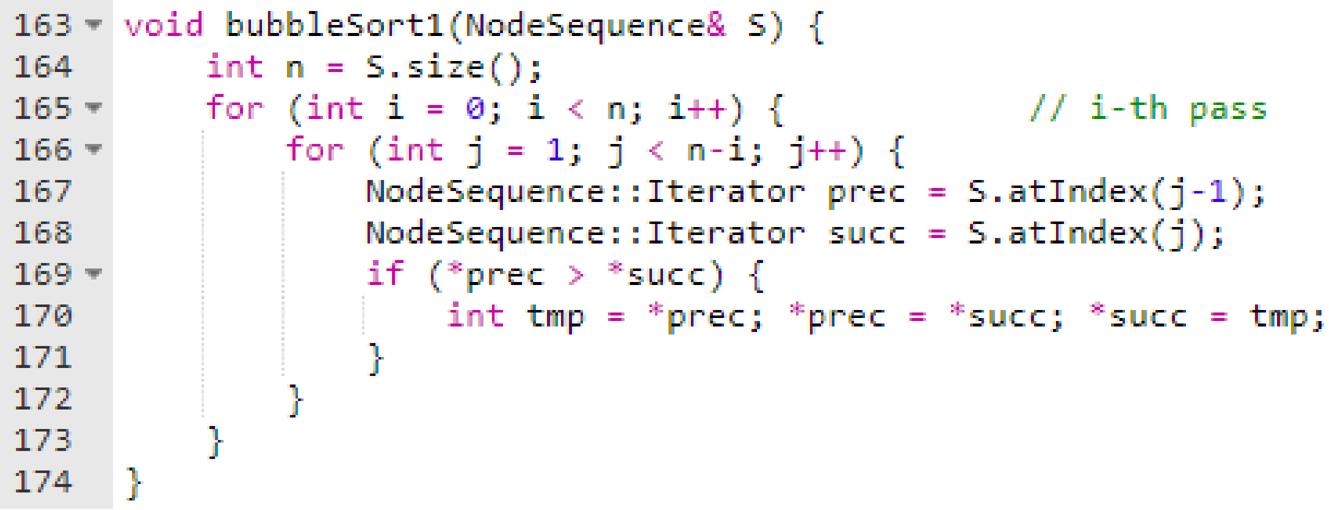
BITS F232: FOUNDATIONS OF DATA STRUCTURES & ALGORITHMS (1ST SEMESTER 2023-24) BUBBLE SORT, TREES

Chittaranjan Hota, PhD
Sr. Professor of Computer Sc.
BITS-Pilani Hyderabad Campus
hota[AT]hyderabad.bits-pilani.ac.in

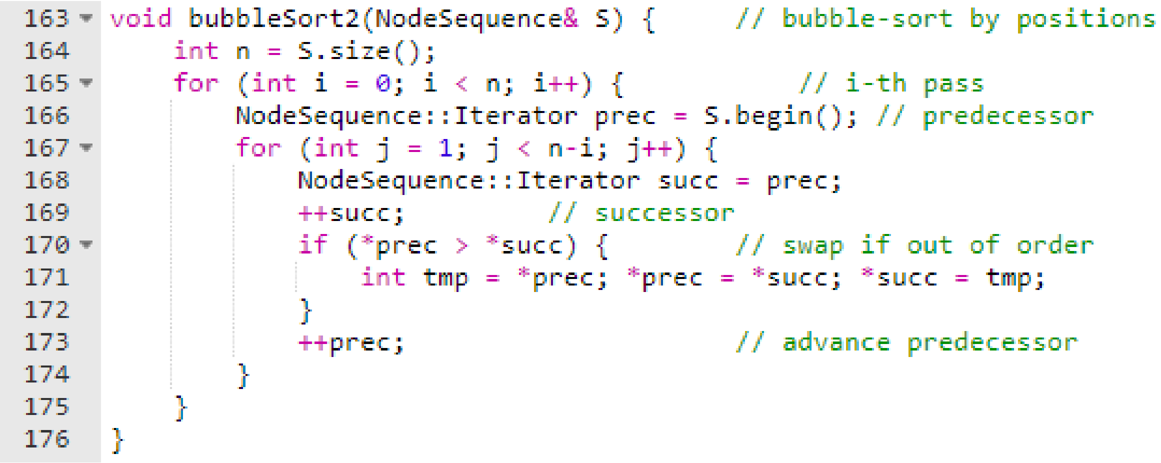
USAGE OF SEQUENCE ADT: BUBBLE SORT

- We will examine the usage of Sequence ADT and its implementation trade-offs using Bubble sort algorithm.

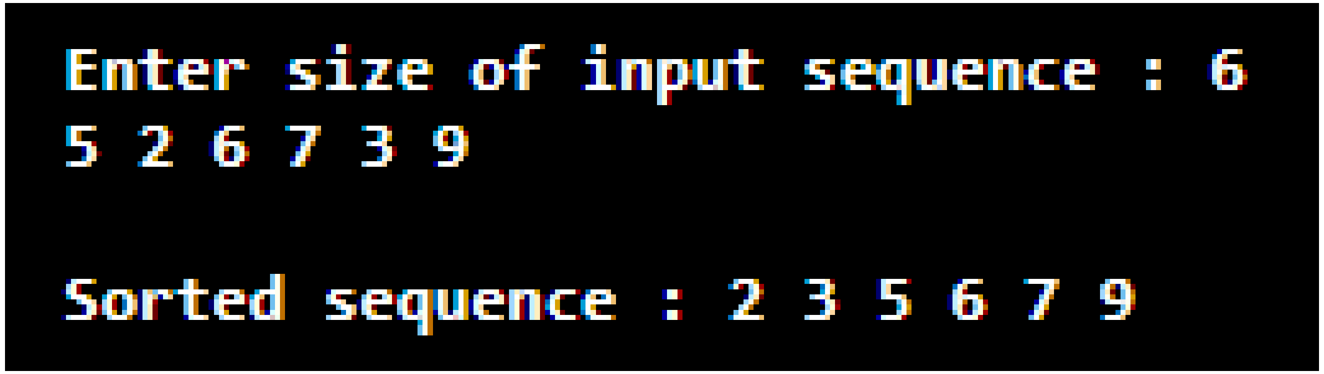
IMPLEMENTATION & ANALYSIS OF BUBBLE SORT



```cpp
void bubbleSort1(NodeSequence& S) {
    int n = S.size();
    for (int i = 0; i < n; i++) {            // i-th pass
        for (int j = 1; j < n-i; j++) {
            NodeSequence::Iterator prec = S.atIndex(j-1);
            NodeSequence::Iterator succ = S.atIndex(j);
            if (*prec > *succ) {
                int tmp = *prec; *prec = *succ; *succ = tmp;
            }
        }
    }
}
```

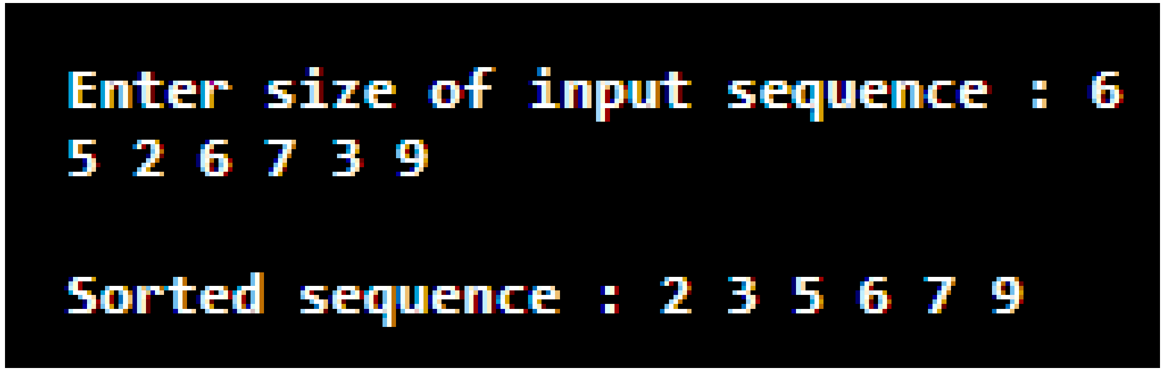


```cpp
void bubbleSort2(NodeSequence& S) {     // bubble-sort by positions
    int n = S.size();
    for (int i = 0; i < n; i++) {           // i-th pass
        NodeSequence::Iterator prec = S.begin(); // predecessor
        for (int j = 1; j < n-i; j++) {
            NodeSequence::Iterator succ = prec;
            ++succ;         // successor
            if (*prec > *succ) {        // swap if out of order
                int tmp = *prec; *prec = *succ; *succ = tmp;
            }
            ++prec;                     // advance predecessor
        }
    }
}
```



```
Enter size of input sequence : 6
5 2 6 7 3 9

Sorted sequence : 2 3 5 6 7 9
```



```
Enter size of input sequence : 6
5 2 6 7 3 9

Sorted sequence : 2 3 5 6 7 9
```

TREES: NON-LINEAR DATA STRUCTURES

- In computer science, what is a tree?

Formally, we define tree T to be a set of nodes storing elements in a parent-child relationship with the following properties:

- If T is nonempty, it has a special node, called the root of T, that has no parent.

- Each node v of T different from the root has a unique parent node w; every node with parent w is a child of w.

Applications?

TREE TERMINOLOGIES AND PROSPERITIES

Root: ???

Internal node: ???

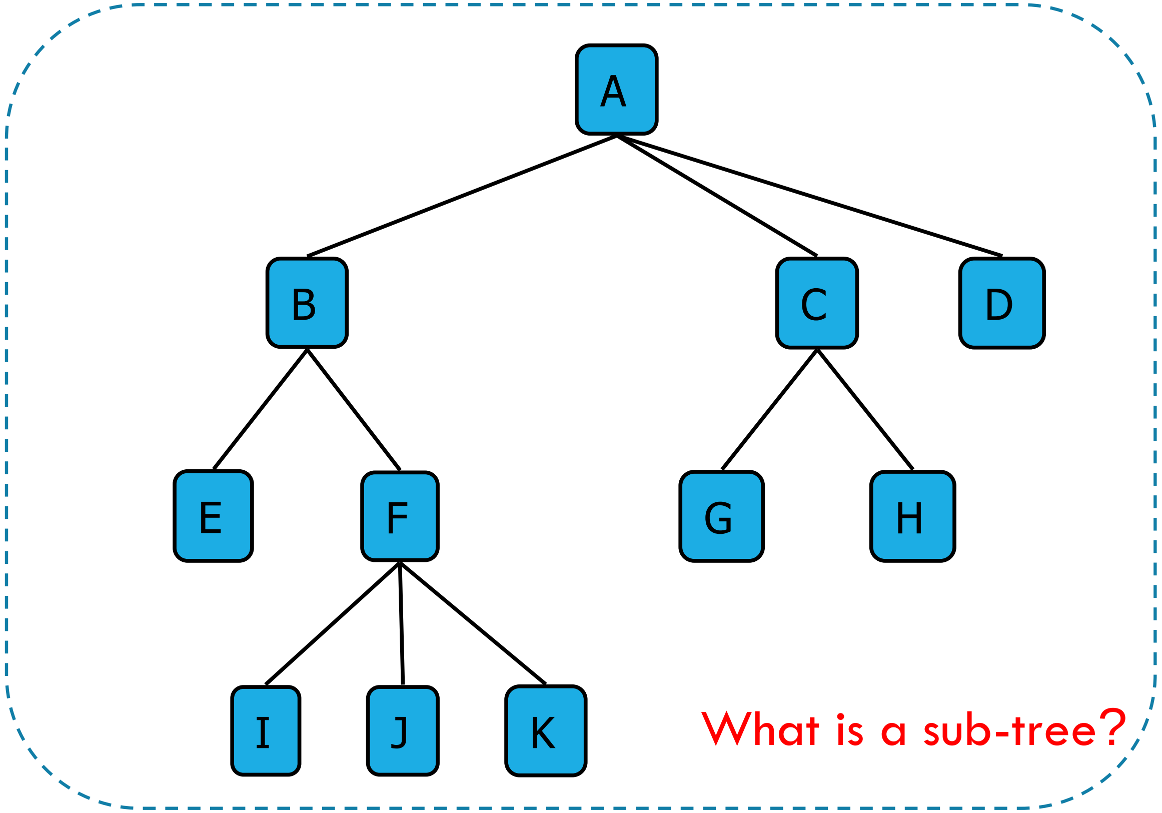
External node: ???

Ancestors of a node: parent, grandparent, grand-grandparent, etc.

Level of a node and Depth: ???

Height of a tree: maximum depth of any node (???)

Descendant of a node: child, grandchild, grand-grandchild, etc.



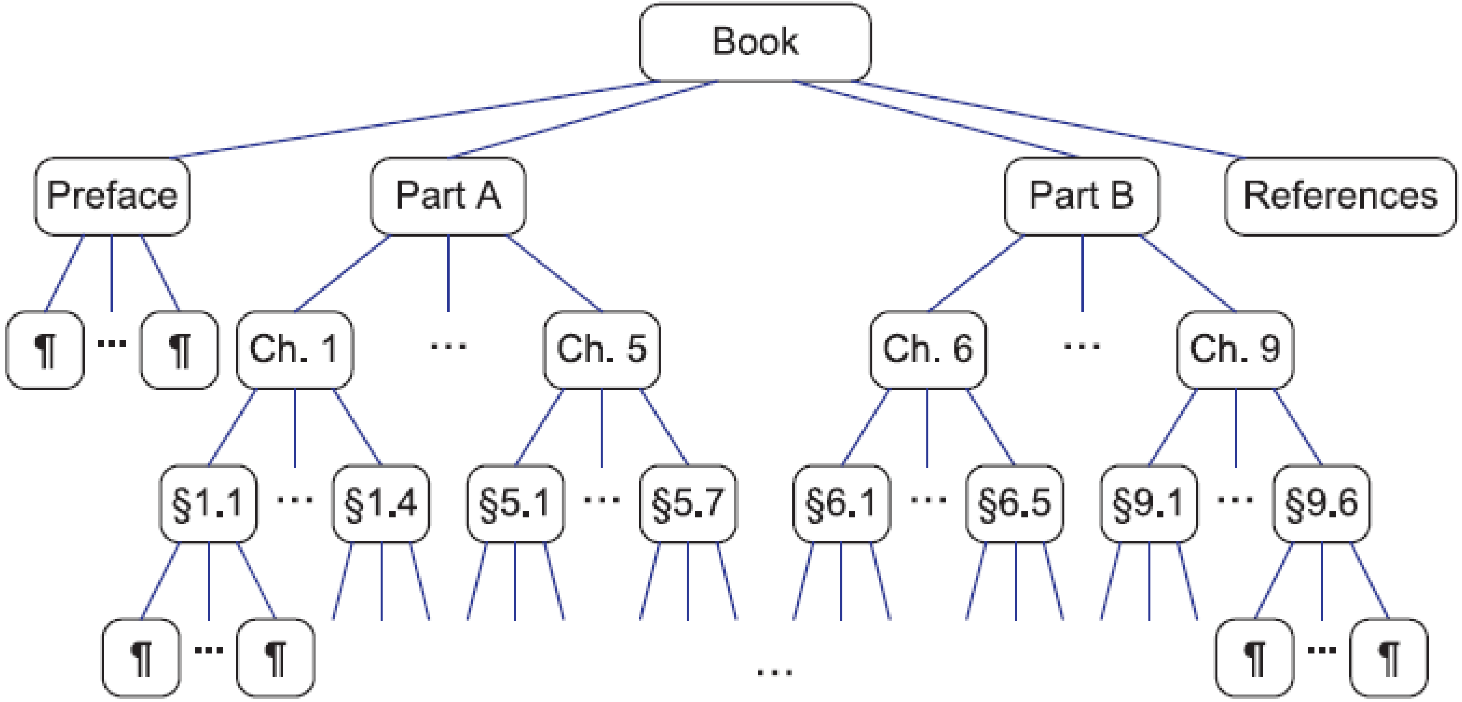
What is a sub-tree?

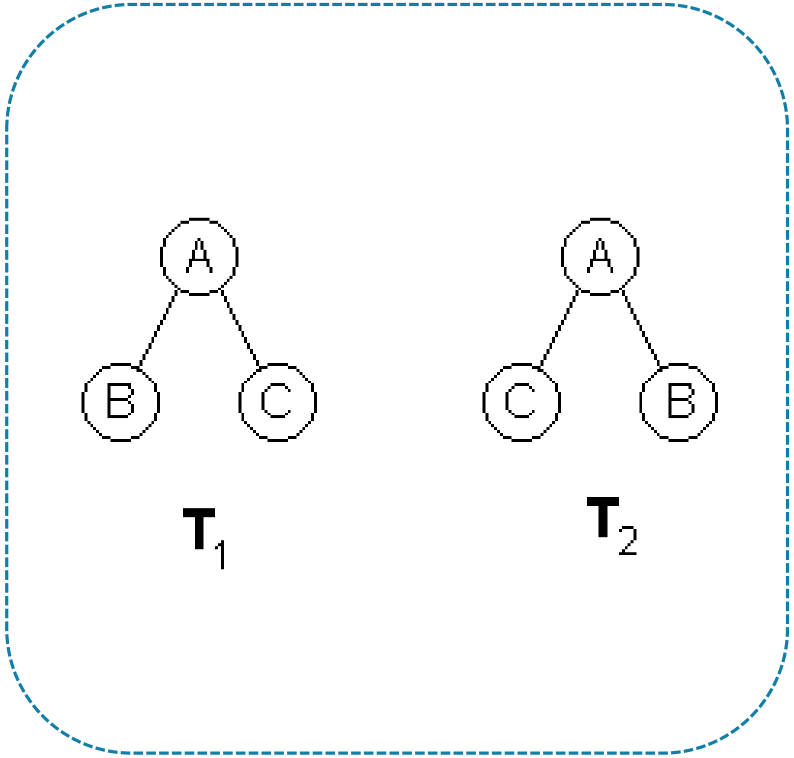
No of edges???

Degree of a node???
↓
count of subtrees

ORDERED TREES

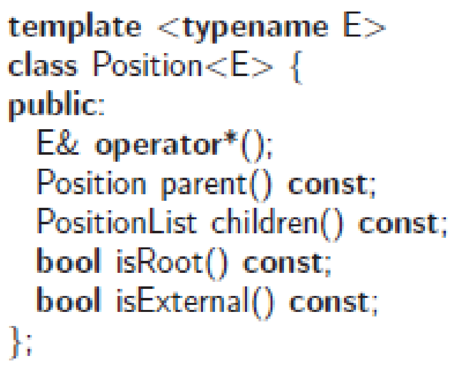
What are Ordered Trees?





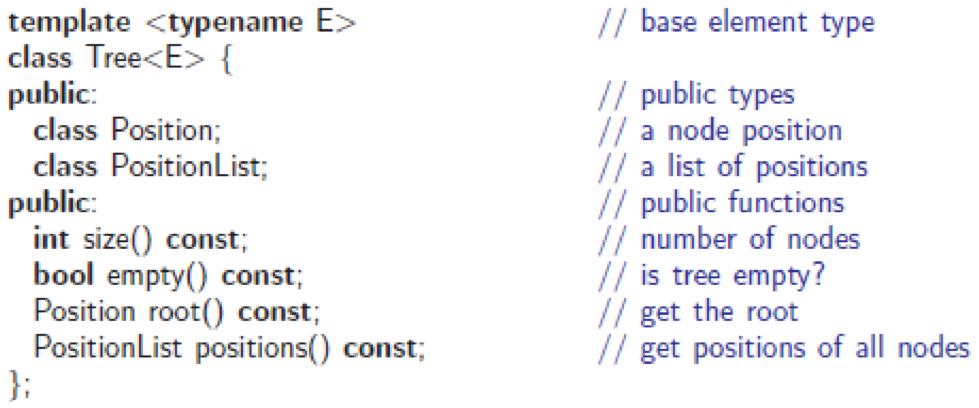
TREE ADT

- Generic methods:
  - integer size()
  - boolean empty()
- Accessor methods:
  - position root()
  - list<position> positions()
- Position-based methods:
  - position p.parent()
  - list<position> p.children()
- Query methods:
  - boolean p.isRoot()
  - boolean p.isExternal()



```
template <typename E>
class Position<E> {
public:
  E& operator*();
  Position parent() const;
  PositionList children() const;
  bool isRoot() const;
  bool isExternal() const;
};
```

(An informal interface for a position in a tree)

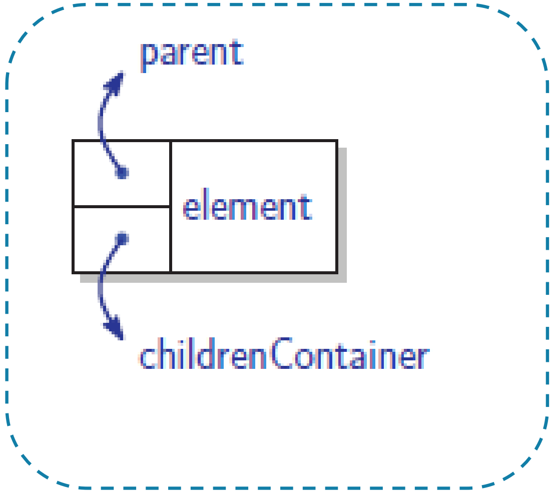


```
template <typename E>                          // base element type
class Tree<E> {
public:                                        // public types
  class Position;                              // a node position
  class PositionList;                          // a list of positions
public:                                        // public functions
  int size() const;                            // number of nodes
  bool empty() const;                          // is tree empty?
  Position root() const;                       // get the root
  PositionList positions() const;              // get positions of all nodes
};
```

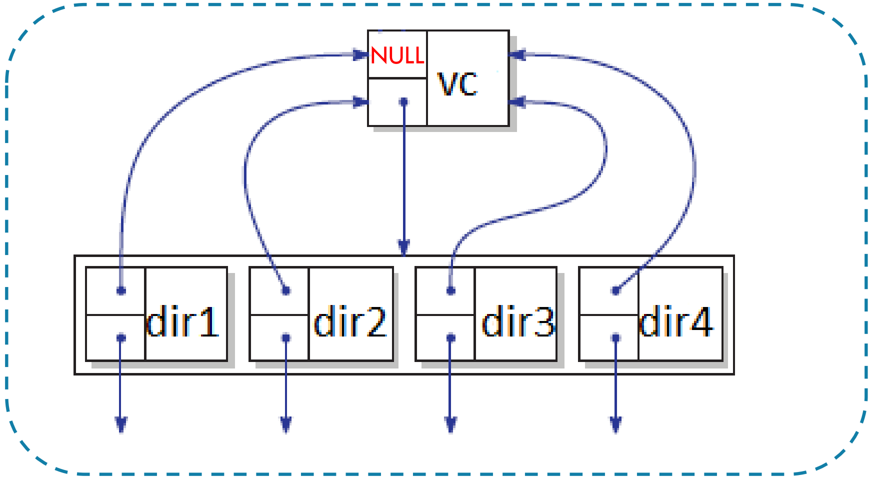
(An informal interface for the tree ADT)

- Additional update methods may be defined by data structures implementing the Tree ADT.

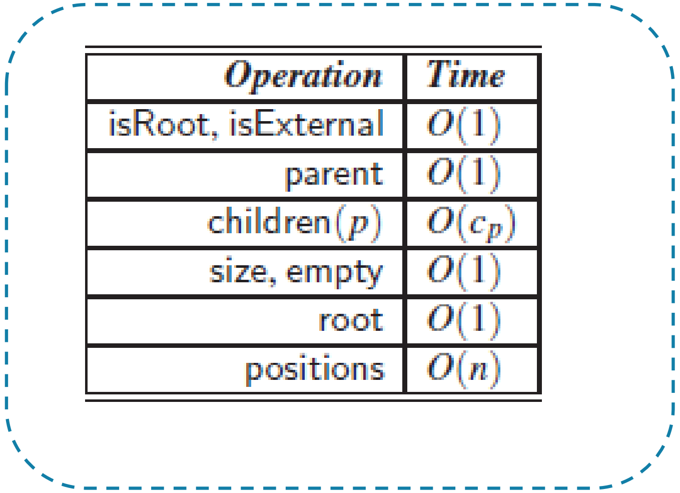
A LINKED STRUCTURE FOR GENERAL TRESS



(The node structure)



(The portion of the data structure associated with root node and its children)



Operation	*Time*
isRoot, isExternal	$O(1)$
parent	$O(1)$
children(p)	$O(c_p)$
size, empty	$O(1)$
root	$O(1)$
positions	$O(n)$

(Running times of the functions of an *n*-node linked tree structure)

DEPTH AND HEIGHT OF A TREE

```
Algorithm depth(T, p):
  Let us write it out using Recursion…
```

Complexity?

The height of a node p in a tree T is also defined recursively:

-If p is external, then the height of p is ???.

-Otherwise, the height of p is one plus the ??? height of a child of p

```
Algorithm height(T):
  h = 0
  for each p ∈ T.positions() do
    if p.isExternal() then
      h = max(h,depth(T, p))
  return h
```

(The height of a tree is equal to the maximum depth of its external nodes)

```
Algorithm height(T, p):
  if p.isExternal() then
    return 0
  else
    h = 0
    for each q ∈ p.children()
    do h=max(h,height(T,q))
    return 1+h
```