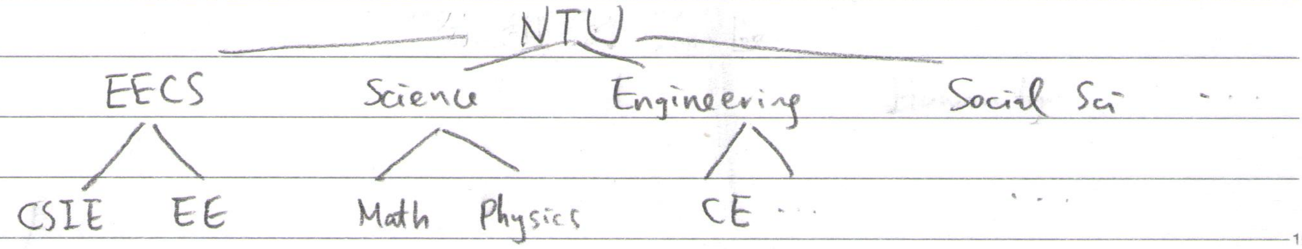
* Trees

vector ~~array~~ (indexed access)   stack/queue (restricted access)

list (sequential access)

tree : hierarchical access



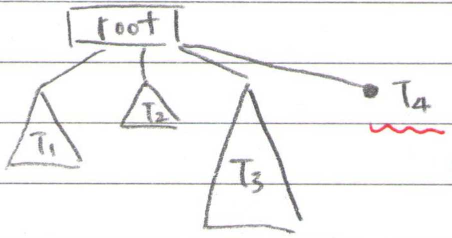
* similarly, directory/files in your filesystem

* formal definition

$T \equiv (\text{root} ; \underbrace{T_1, T_2, \cdots, T_n}_{\text{disjoint subtrees}})$ recursive definition

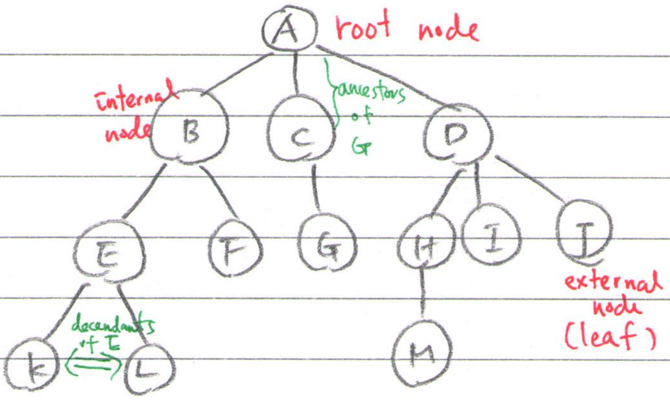
disjoint subtrees : no cross links

(ordered or unordered)



termination of recursion (no subtrees)

*



level (depth) 0

depth 1

depth 2

depth 3 = height of tree

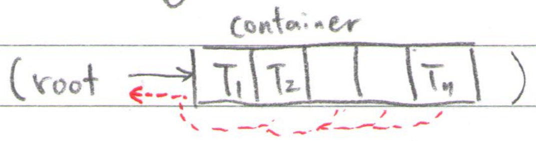
↑ parent   ↓ child   ⟷ sibling

degree of node : # child   degree of tree: max degree of node

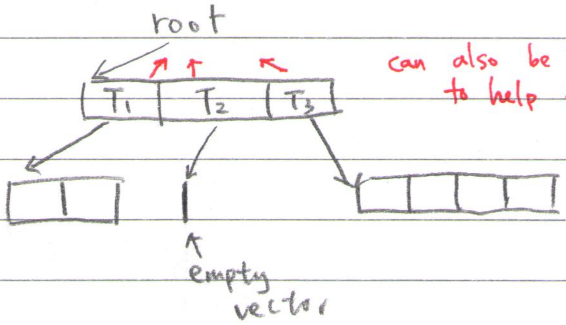
* representing trees

$$T \equiv (\text{root} \; ; \; T_1, T_2, \cdots, T_n)$$

⇓



* use vector as container (space : O(n) in general)

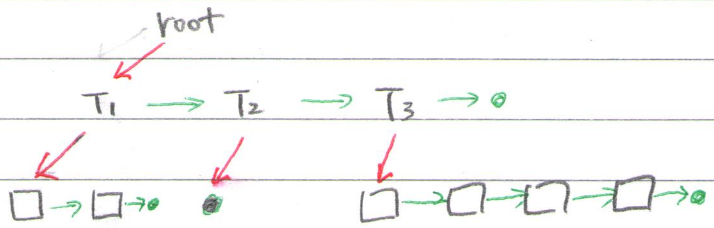


can also be "doubly" to help find parent

isRoot(p) : O(1)

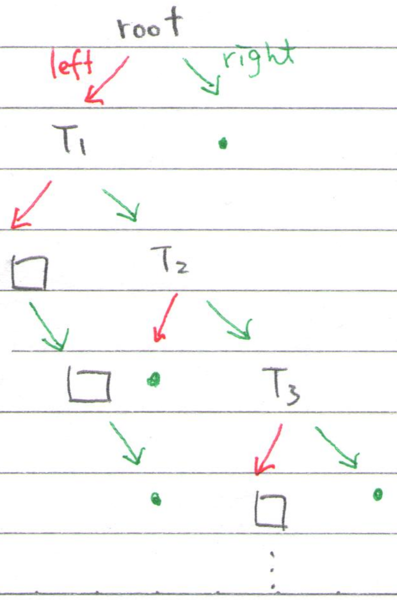
isLeaf(p) : O(1)

* use l-list as container



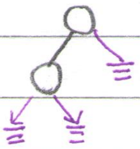
called left-child right-sibling

* rotate a little



an equivalent representation of general tree by $\boxed{\text{binary tree}}$

deg ≤ 2, ordered, allowing empty subtrees



usually don't draw ⏚