Tree / Heap

Hsuan-Tien Lin

Dept. of CSIE, NTU

March 30, 2021

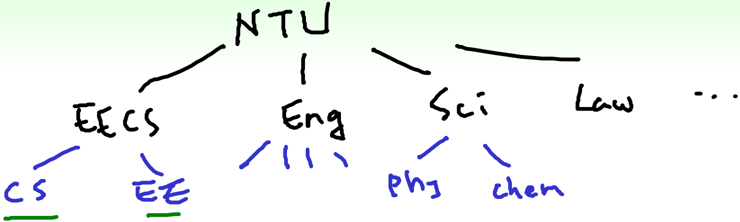
What We Have Done

- stack (LIFO): infix to postfix, postfix evaluation
- queue (FIFO): maze solving (how different data structures affect algorithm behavior), implementation by circular array
- deque: stack + queue + push_front

Nature of Data Structures

data structure	nature
array	indexed access
linked list	sequential access
stack/queue/deque	restricted (boundary) access
tree	hierarchical access

next: tree



- parent-child relationship: file system, hierarchical structure

dir

file

Formal Definition of Trees

tree sub-tree

$$T \equiv (root; T_1, T_2, \ldots, T_n)$$

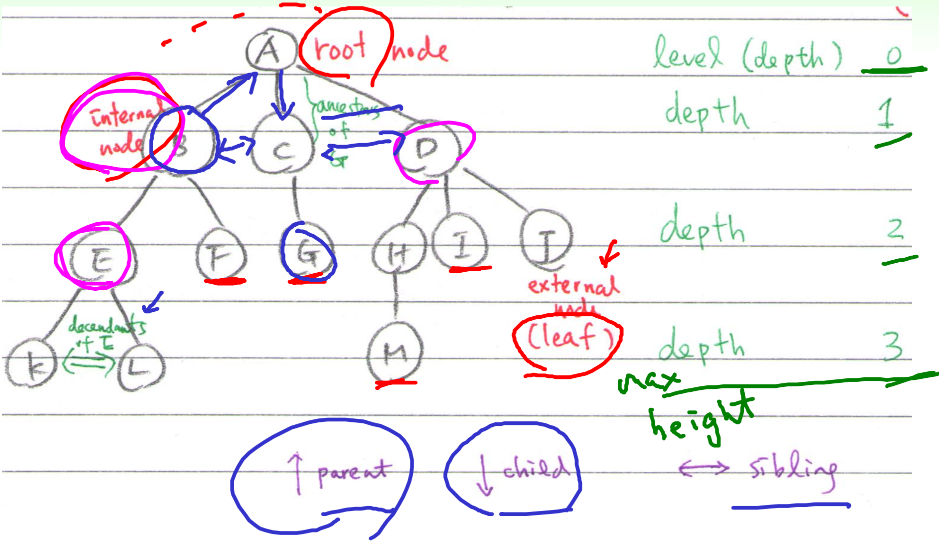
r

- recursive definition
- sub-trees $(T_1, \ldots, T_n)$ disjoint
- recursion termination?

$$T \equiv (root; nil)$$

leaf

Names in Trees



Representing Trees

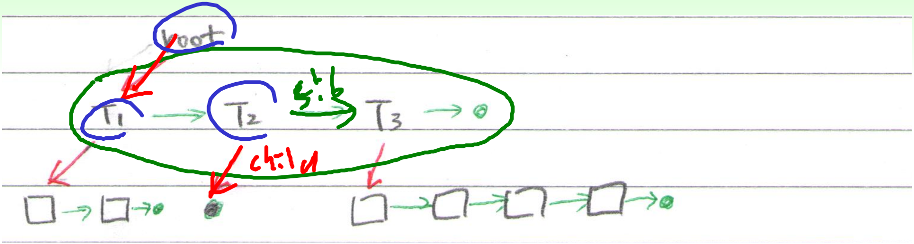
$$T \equiv (\text{root} \,;\, T_1, T_2, \cdots, T_n)$$

$$\Downarrow$$

container

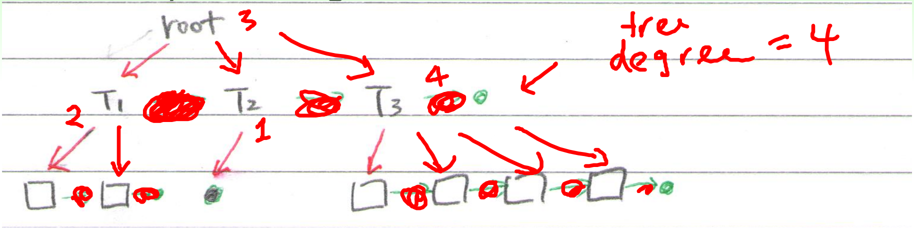
$$(\text{root} \rightarrow \boxed{T_1 \mid T_2 \mid \quad \mid \quad \mid T_n}\ )$$

Representing Trees with Linked Lists



called left-child right-sibling
link per node?

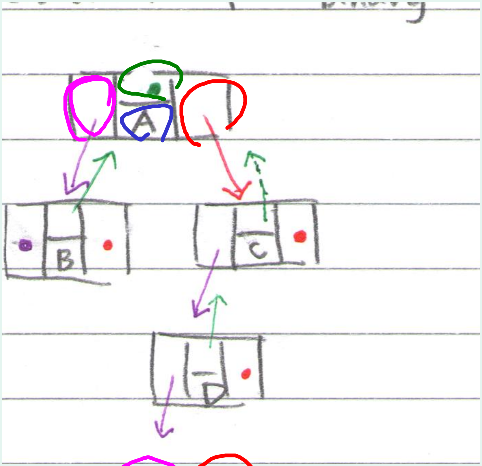
Representing Trees with Sub-Tree Links



sibling links → child links

'easiest' visual representation of trees
link per node?

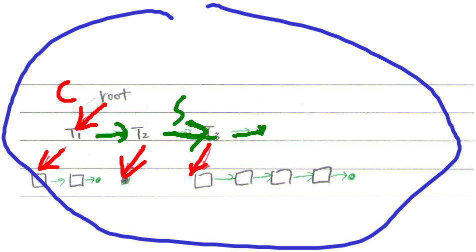
Binary Trees

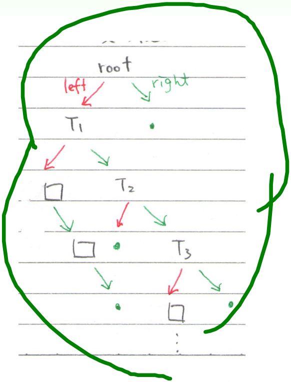


left, right, (parent)

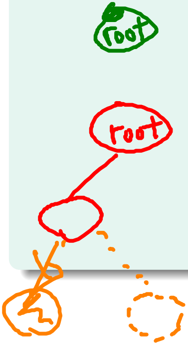
$\leq$ 2 ordered children per node

General Tree + Left-Child Right-Sibling ≡ Binary Tree

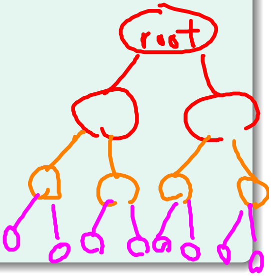




How Many Nodes in Binary Trees



h	min	max
0	1	1
1	2	3
2	3	7
3	4	15
...		



How Many Nodes in Binary Trees

$n = 10 \quad \log_2(n+1) - 1 \le h \le n - 1$

h	min	max
0	1	1
1	2	3
2		
3		
$\cdots$		
h	$h+1$	$2^{h+1}-1$
	(skewed)	full

$$h + 1 \le n \le 2^{h+1} - 1$$

$\Leftrightarrow$

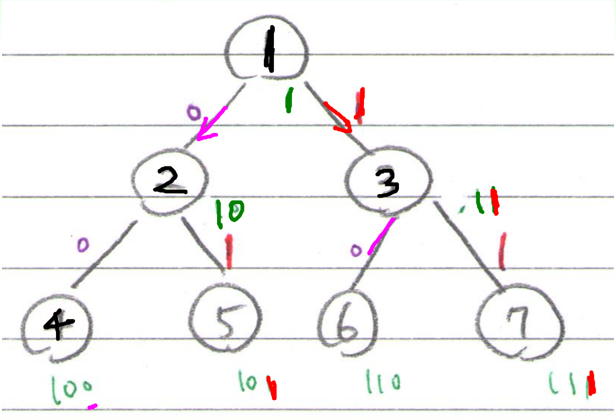
#node

How Many Nodes in Binary Trees

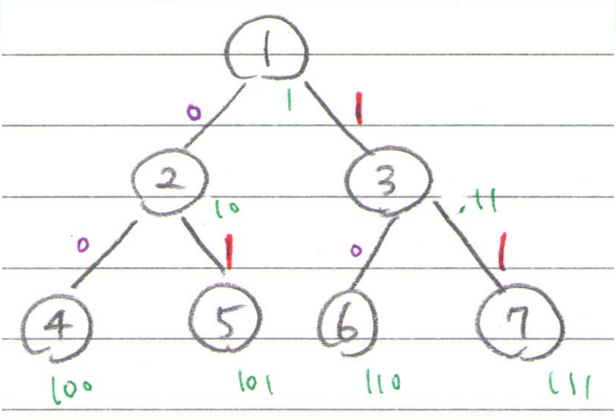
h	min	max
0	1	1
1	2	3
2		
3		
$\cdots$		
h	$h+1$	$2^{h+1}-1$
	(skewed)	full

$$h+1 \le n \le 2^{h+1}-1$$
$$\Leftrightarrow \lg(n+1)-1 \le h \le \tfrac{n-1}{2} \cdot Z$$

Node Index in Full Binary Tree

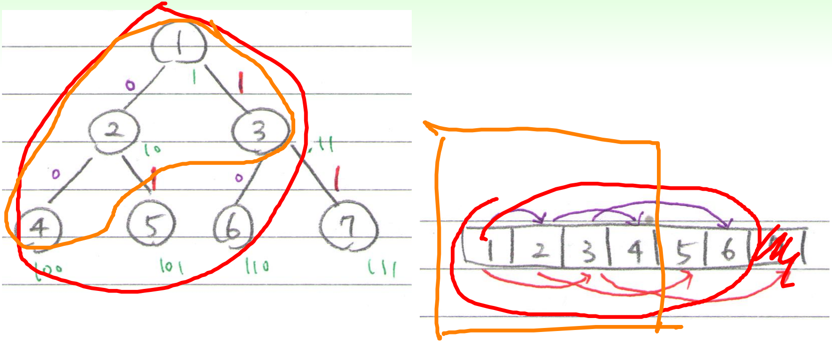


Node Index in Full Binary Tree



node index $= (1\textit{path code})_2$

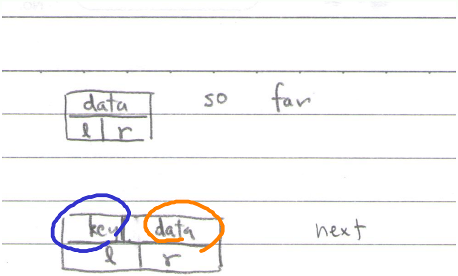
Representing/Packing Full Binary Tree in Array



- implicit links: no need for explicit pointers
- can similarly pack any binary tree if NIL can represent NO-DATA (with space wasting in data field)

> complete binary tree: full binary tree with first n nodes
> (no waste with array representation)

Need: Priority Queue (with Binary Tree)



- key: priority
- data: item in todo list

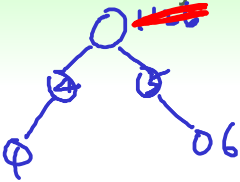
goal: get the node with largest key

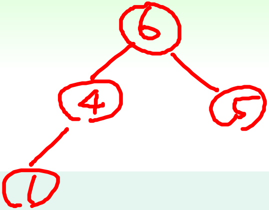
Design Thoughts of Priority Queue with Tree (1/4)

getLargest

- entry point of tree? root
- to allow fastest access, put largest close to entry point

Design Thoughts of Priority Queue with Tree (1/4)





- entry point of tree?
- to allow fastest access, put largest close to entry point

but what if not just `getLargest` but need `removeLargest`?

Design Thoughts of Priority Queue with Tree (2/4)

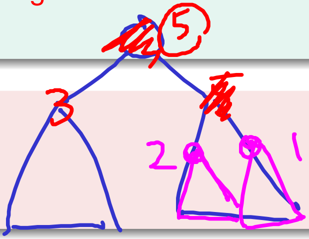
`removeLargest` needs 2nd largest to replace

- to allow fastest removal, put 2nd largest close to next entry points
- next entry points of tree? children of root

Design Thoughts of Priority Queue with Tree (2/4)

`removeLargest` needs 2nd largest to replace

- to allow fastest removal, put 2nd largest close to next entry points
- next entry points of tree?



max tree:

- root key $\geq$ other node's key
- every sub-tree also max tree

`removeLargest` (version 0): recursive duel of sub-tree roots

$O(h)$