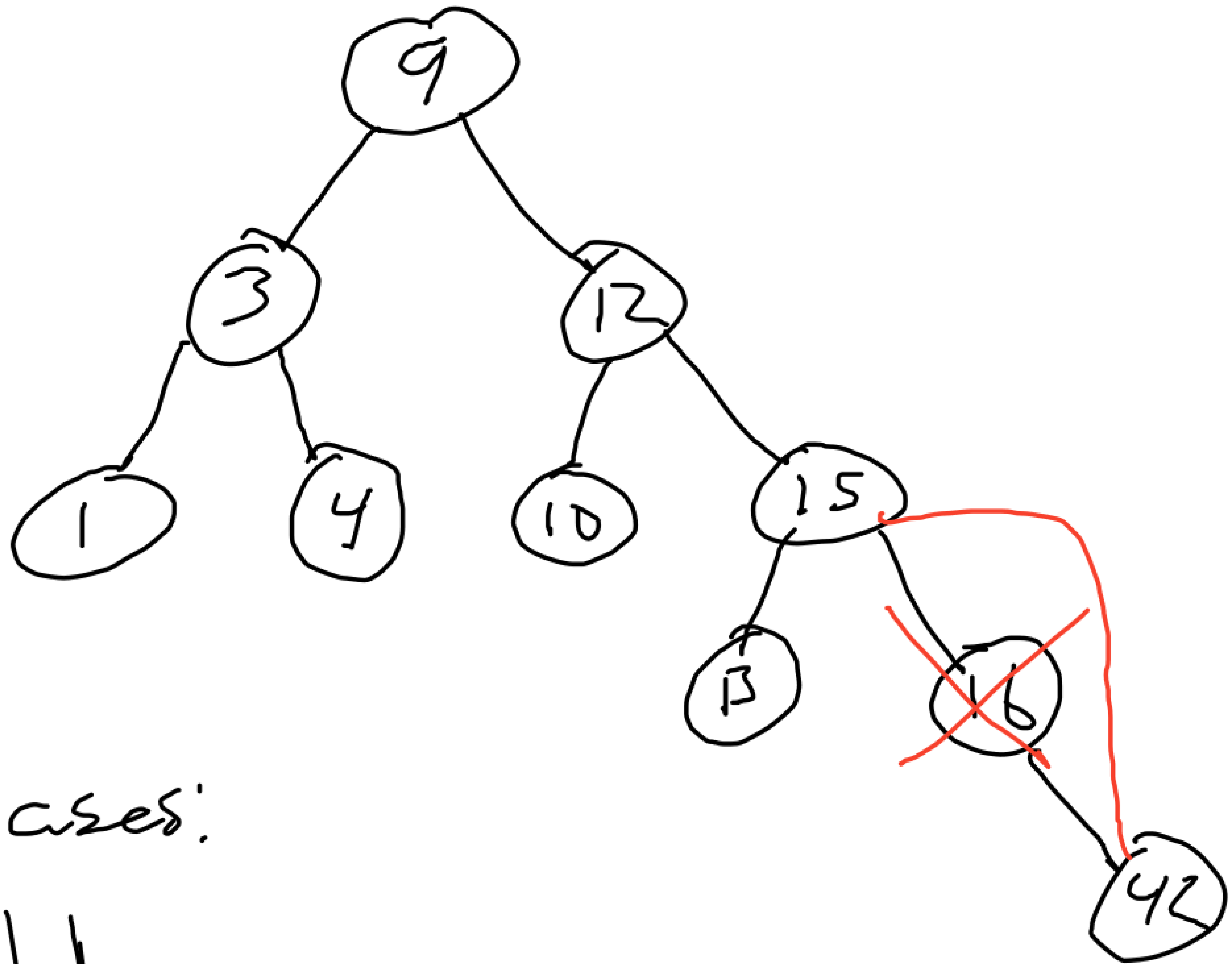


Delete Cases:

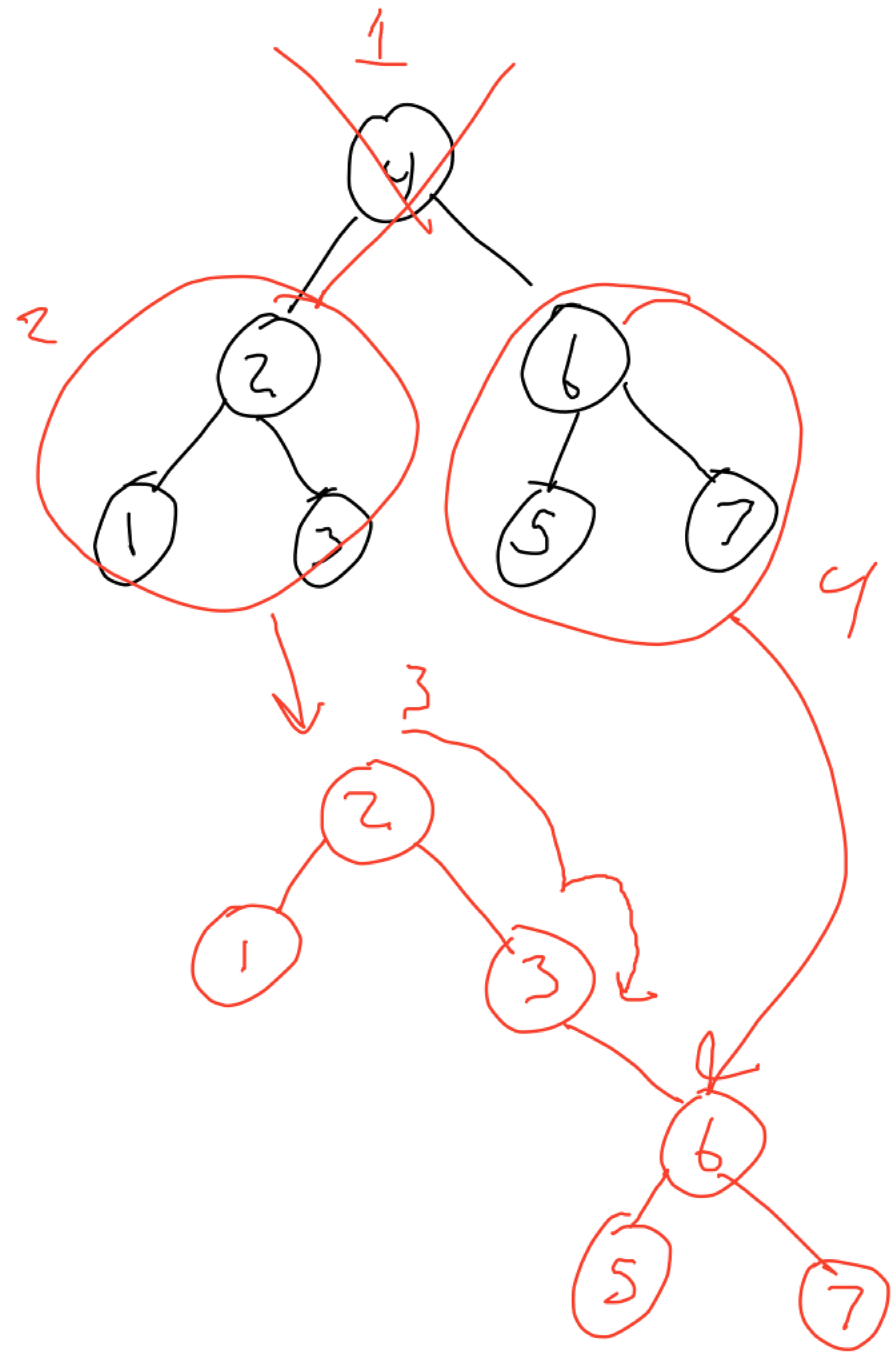
- no children
  - simply remove

- 1 child
  - promote the child (linked list delete)

- 2 children
  - merge
  - -OR-
  - replace

Delete by merging

- delete 4

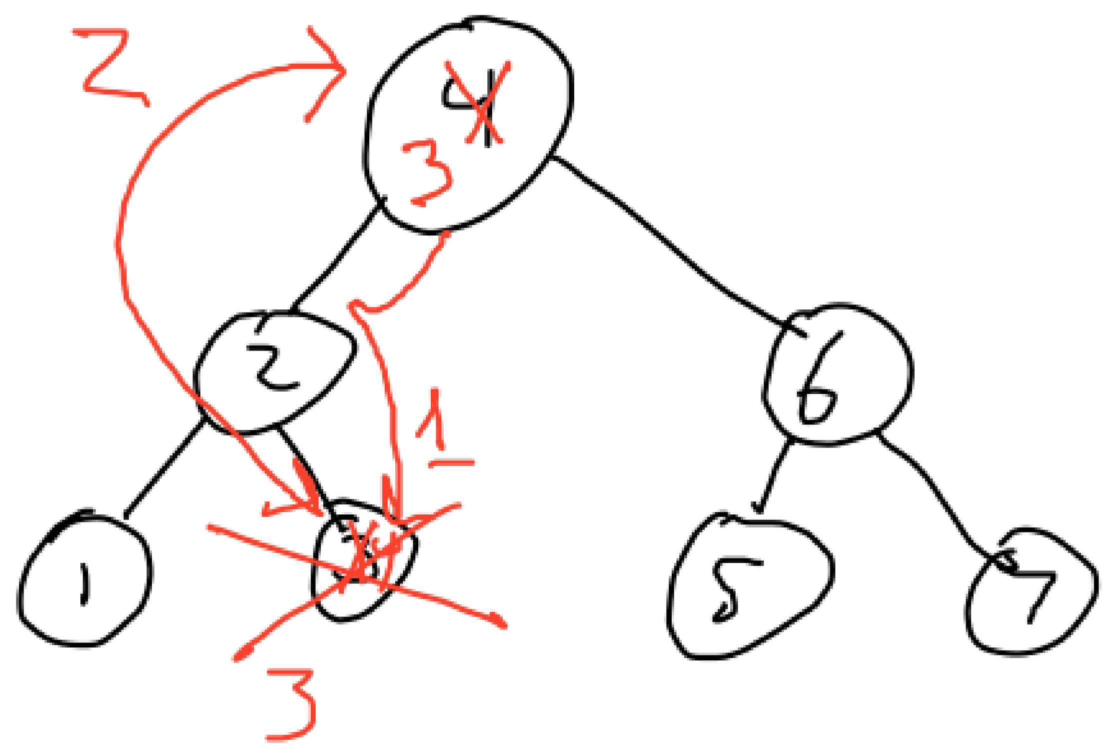


1. delete node — $\sim O(1)$
2. pick one side as root (arbitrary) — $O(1)$
3. find first right child space (left child if we picked the other tree) — $O(h) \approx O(\lg n)$
4. add other subtree as that child — $O(1)$

$O(\lg n)$

Delete by replacement

- delete 4



1. find greatest predecessor (left child) or least successor (right child) - $O(h) \approx O(\lg n)$

2. swap value of deleted node and the found node - $O(1)$

3. delete the found node - $O(1)$
   - will be 0 or 1 child case

$$O(\lg n)$$

To implement a set or map:

- find
  - need to check at each level

    $O(h)$, generally $O(\lg)$
- insert (after find)
  - create node, link to parent

    $O(1)$
- delete (after find)
  - 0 children $O(1)$
  - 1 child $O(1)$
  - 2 children $O(\lg n)$

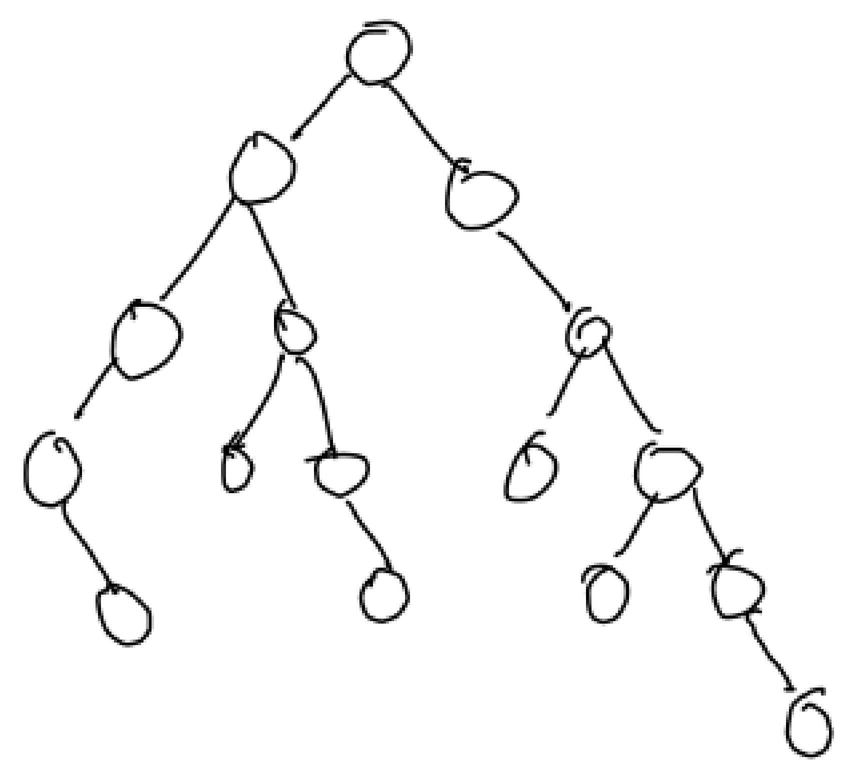
---

get = find = $O(\lg n)$

put = find + insert = $O(\lg n)$

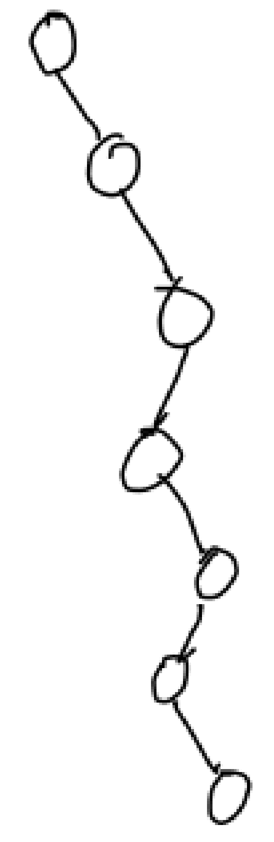
delete = find + delete = $O(\lg n)$

height



n nodes

worst case



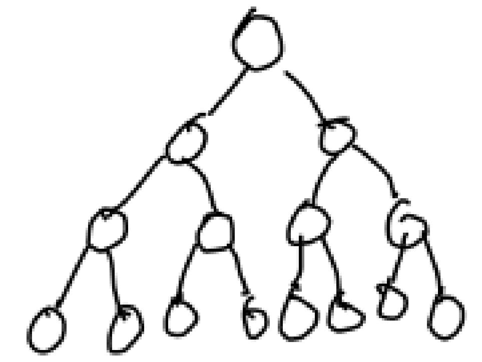
$h = n$

$O(h) = O(n)$

average case

$O(h) = O(\lg n)$

best case



h	n
1	1
2	2
2	3
3	3
⋮	⋮
3	7
4	8
⋮	⋮
4	15
5	16
⋮	⋮
5	

biggest n for each h

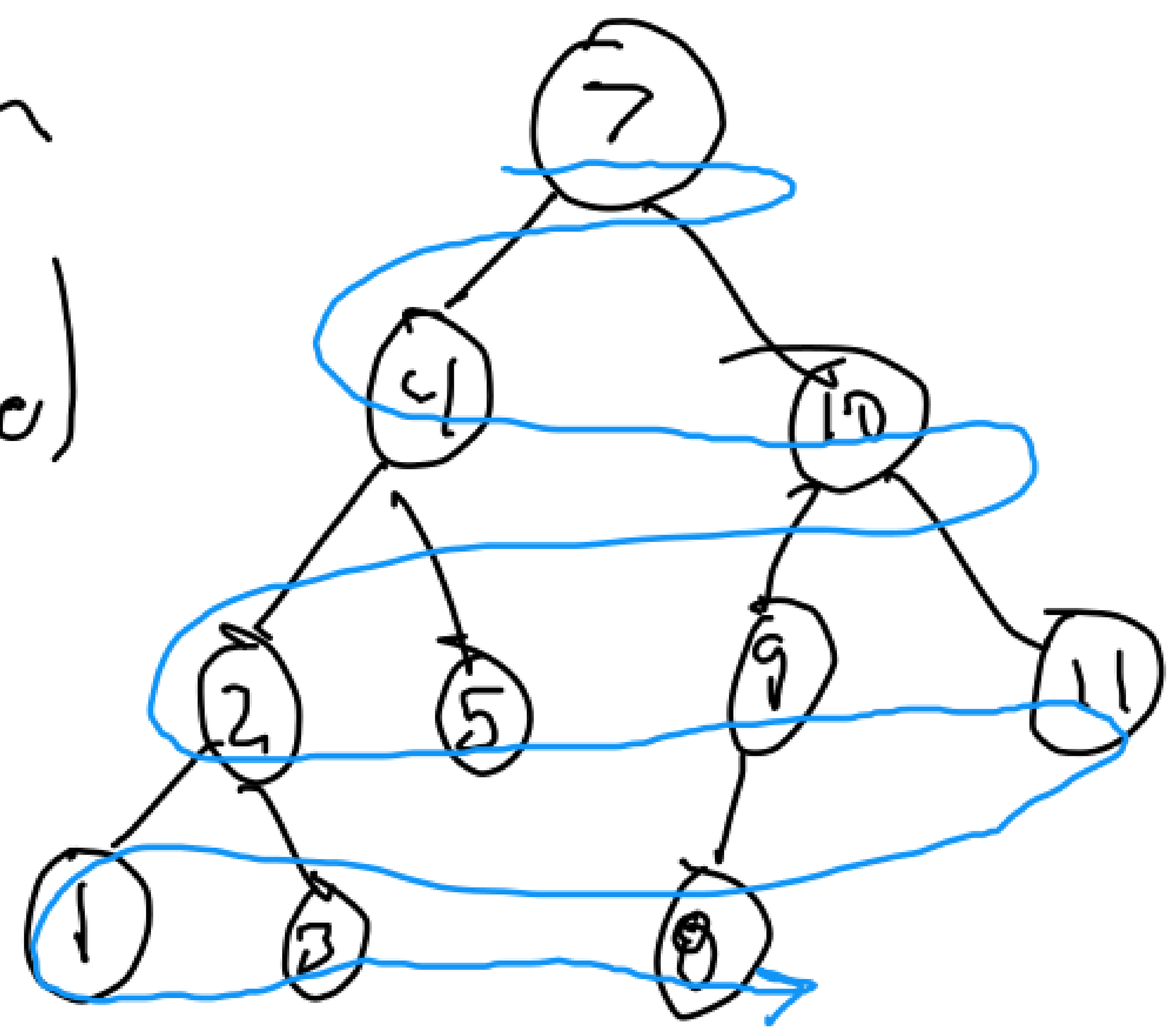
$n = 2^h - 1$

$h = \lfloor \log_2(n) \rfloor + 1$

$O(h) = O(\lg n)$

Breadth-first traversal

visit each node in a level before moving to the next level



We use a queue of children to visit it

- add root
- while not empty
    - remove a node
    - visit it
    - add its children

queue ~~8~~ ~~3~~ ~~1~~ ~~11~~ ~~9~~ ~~5~~ ~~2~~ ~~10~~ ~~4~~ ~~7~~

current ~~7~~ ~~4~~ ~~10~~ ~~2~~ ~~5~~ ~~9~~ ~~11~~ ~~1~~ ~~3~~ 8

visit order: 7, 4, 10, 2, 5, 9, 11, 1, 3, 8