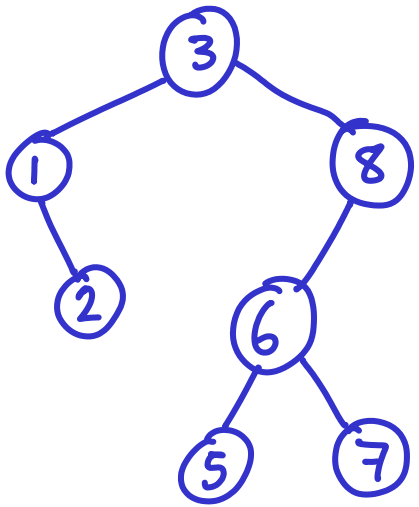
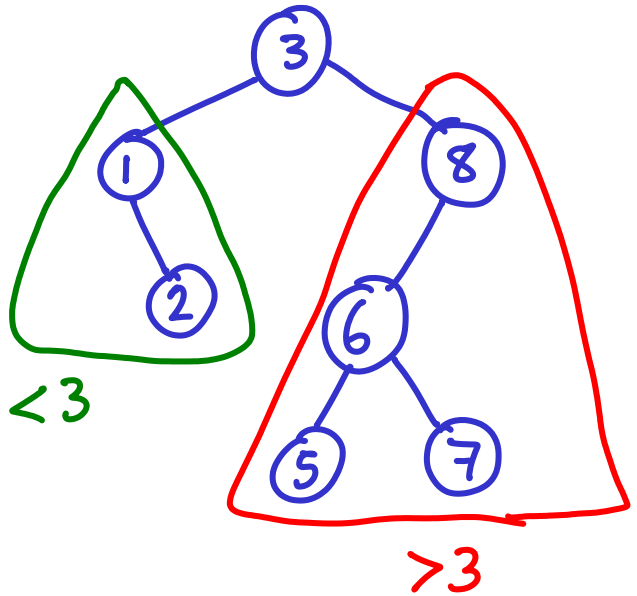
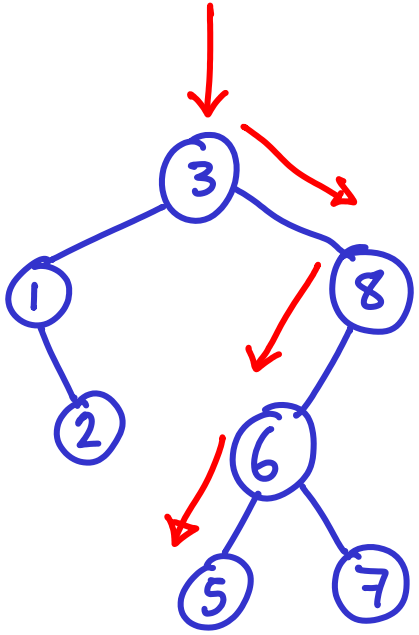
Binary search trees





Binary search trees

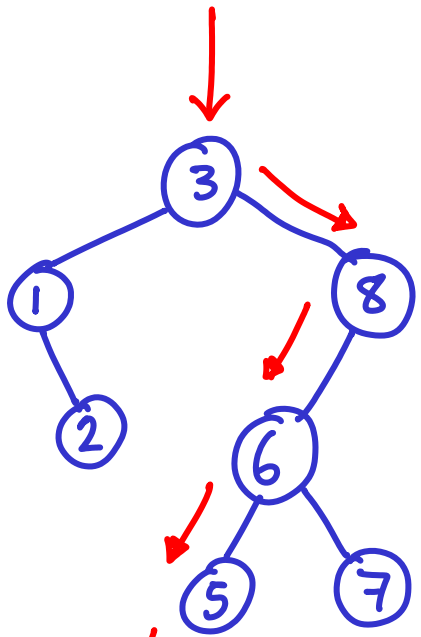
(binary) search for 5



found

time:
O(depth)

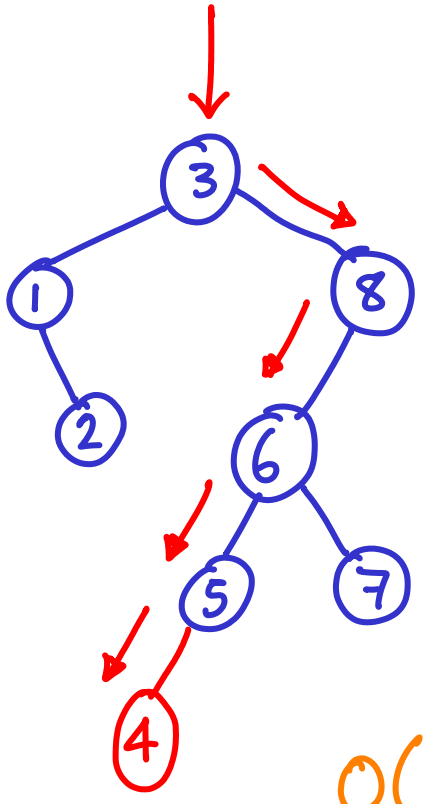
(binary) search for 4



x not found

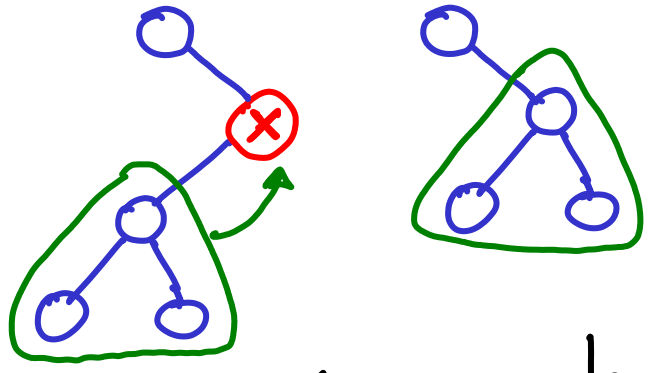
Binary search trees

insert (4) ~ search



O(depth)

instant delete (x)



x: any node w/ < 2 children

O(1)

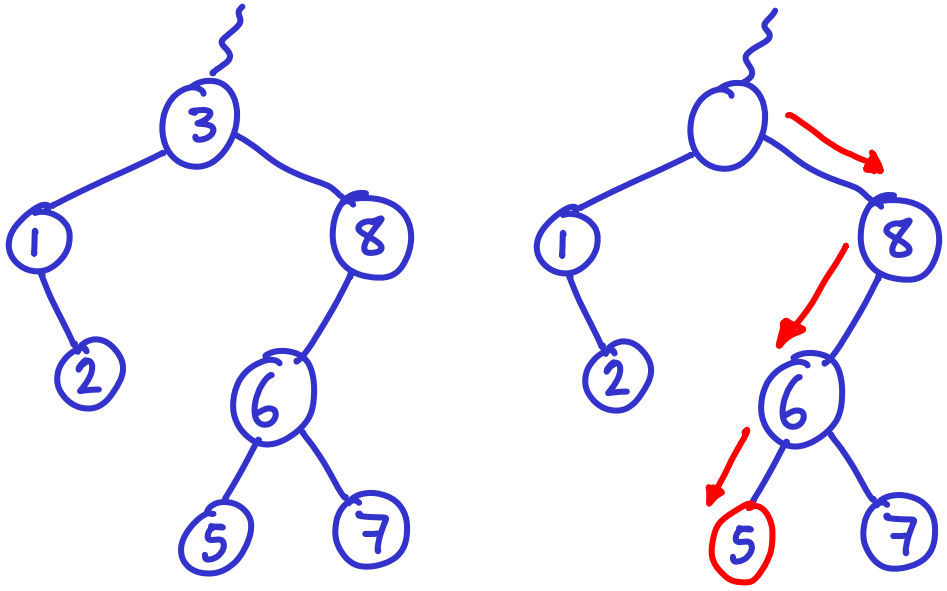
If (one) subtree exists, promote it.

BINARY SEARCH TREES

non-instant

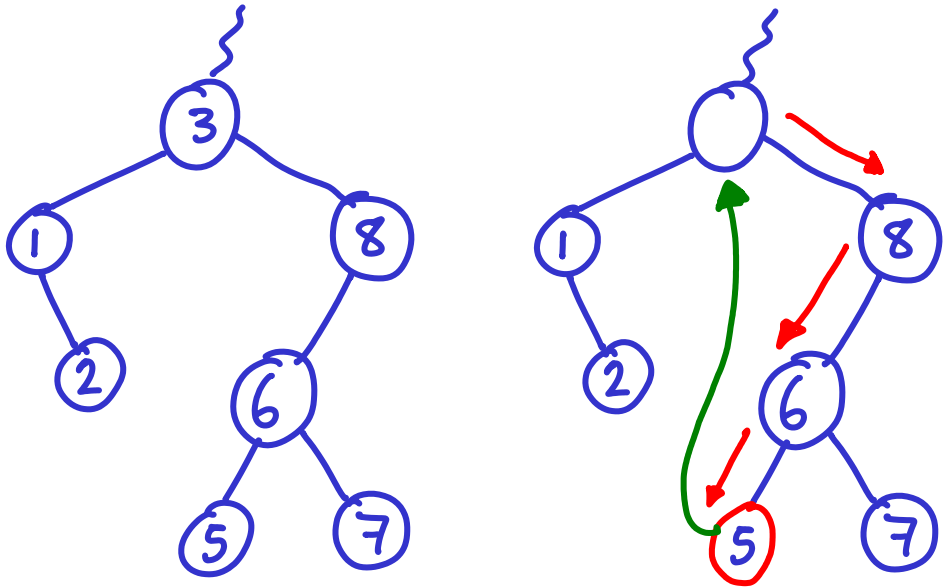
delete(3) → find successor : smallest element greater than 3

(which exists because : 2 children)



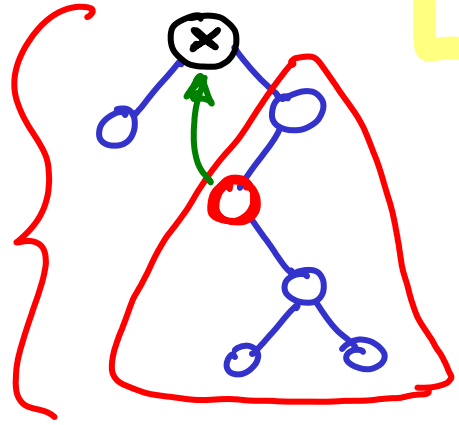
BINARY SEARCH TREES

delete(3) → find successor & replace



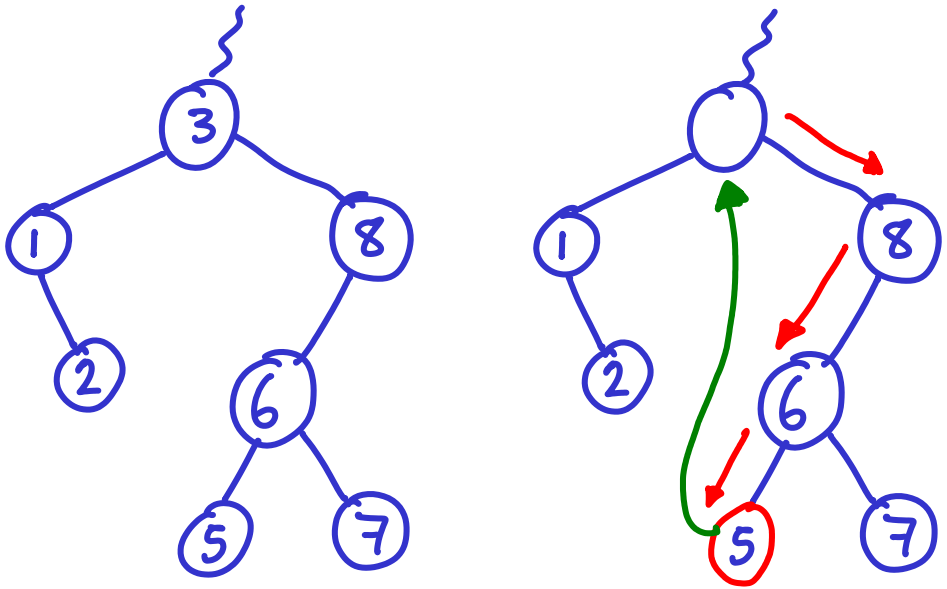
total O(depth) ← might need to recurse

By definition,
successor is
the last node visited on a
path from R-child(3)

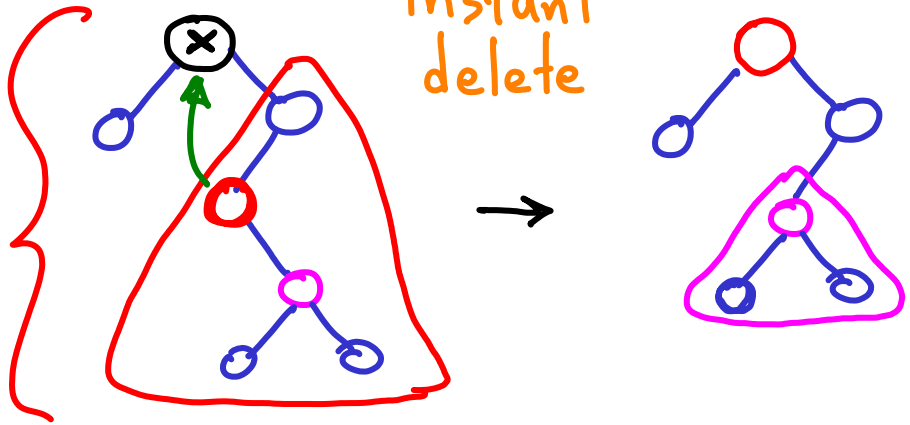


BINARY SEARCH TREES

delete(3) → find successor & replace

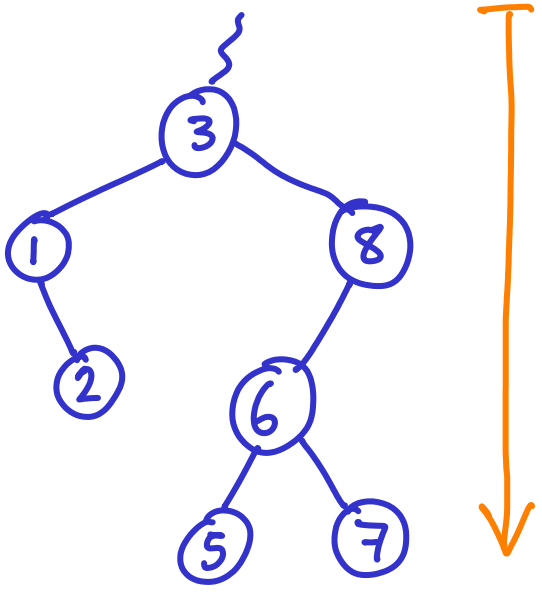


By definition,
successor is
the last node visited on a
path from R-child(3)



still O(depth) ← no need to recurse

Binary search tree summary



INSERT → SEARCH → O(depth)

DELETE → SEARCH → O(depth)

We should keep the tree balanced as much as possible