Binary search trees
Binary search trees

(binary) search for 5

```
  3
 /|
/  |
1   8
```

```
  6
 /|
/  |   |
5   7
```
Binary Search Trees

(binary) search for 5
Binary search trees

(binary) search for 5

found
Binary Search Trees

(binary) search for 5

found

time: $O(d)$
Binary search trees

(binary) search for 5

found

\text{time: } \mathcal{O}(\text{depth})

(binary) search for 4

x not found
Binary search trees

Insert (4)  \lor search
Binary Search Trees

- Insert (4)
- Search
- Instant delete (x)

Diagram:

- Tree with nodes 1, 2, 3, 4, 5, 6, 7, 8
- Node 3 is the root
- Node 4 is a leaf

$x$: leaf
**Binary Search Trees**

- **Insert (4)**
- **Search**

![Tree Diagram]

- **Instant Delete (x)**
  - \( x: \text{leaf} \)
  - \( x: \text{root with 1 child} \)
    \[ \Rightarrow x = \text{min or max} \]
Binary Search Trees

Insert (4) vs Search

Instant Delete (x)

x: Leaf

x: Root w/ 1 child
\[ \Rightarrow x = \text{MIN or MAX} \]

x: Any node w/ 1 child
Binary search trees

- Insert: \( O(1) \) (4) vs search \( O(\text{depth}) \)
- Instant delete: \( O(1) \) for any node with \( \leq 2 \) children, promote subtree.
Binary search trees

delete(3)
**Binary Search Trees**

non-instant delete(3) \[\rightarrow\] find successor: smallest element greater than 3 (which exists because: 2 children)
**Binary Search Trees**

delete(3) → find successor & replace
**Binary Search Trees**

- **delete(3)** → **find successor & replace**

By definition, the 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)

Might need to recurse
Binary search trees

**delete(3)** → find successor & replace

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

- **total O(depth)** → might need to recurse
**Binary Search Trees**

Delete (3) → Find successor & replace

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

```
   3
  / 
 1   8
 /   /
2    6
 |    |
5    7
```

```
   3
  / 
 1   8
 /   /
2    6
 |    |
5    7
```

Total $O(\text{depth})$ might need to recurse

OR?
Binary Search Trees

delete(3) → find successor & replace

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

still $O(\text{depth})$ → no need to recurse
Binary Search Tree Summary

We should keep the tree balanced as much as possible.