Binary search trees

Diagram of binary search trees:
- Example 1
- Example 2
- Example 3
**Binary Search Trees**

(binary) search for 5

![Binary search tree diagram for searching 5]

Time: $\mathcal{O}(\text{depth})$

(found)

(binary) search for 4

![Binary search tree diagram for searching 4]

$\times$ not found
**Binary Search Trees**

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

<table>
<thead>
<tr>
<th>Insert (4)</th>
<th>O(depth)</th>
</tr>
</thead>
</table>

- **Instant Delete (x)**

  - **x**: any node with < 2 children
  - If (one) subtree exists, promote it.

  ![Diagram](image.png)
**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

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

total $O(\text{depth})$ → 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)

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

- Insert
- Search $O(\text{depth})$
- Delete

We should keep the tree balanced as much as possible.