Lec15-binary-search-tree

Sunday, October 1, 2023 3:38 PM

We often don't want to just find a number, but an object associated with that number

Dictionary abstract data type:

- insert (key, value)
- delete (key)
- value = find (key)

Many different ways to implement.

Ex. Linked list with attached values

Ex. Skip lists

Ex. Hash tables

Ex. Binary search tree

Binary search tree: (rooted) — attached items (w/ values) to keys

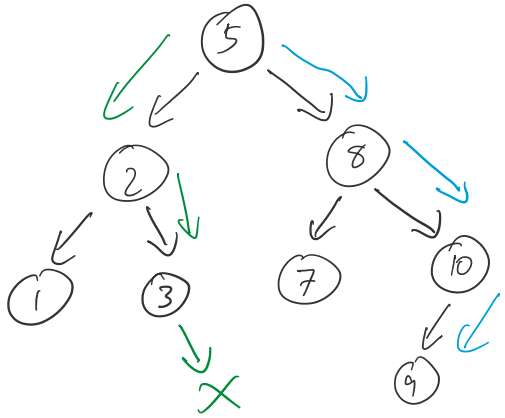
↑ every node has ≤2 children

BST property:

Given a node (k, v), all nodes in left subtree have keys $< k$, in right subtree have keys $> k$.

(disallow duplicate keys for simplicity)

Ex.



(sorted [1, 2, 3, 5, 7, 8, 9, 10])

Find (9)
- 9 > 5, right
- 9 > 8, right
- 9 < 10, left

Find:
- if bigger, go right
- if smaller, go left

Find (4)
- 4 < 5, left
- 4 > 2, right
- 4 > 3, right ✗

⇒ not present

Find Min ()
- always walk left

Insert: same as find, but add a child when you reach an appropriate empty spot. (otherwise, emit error on finding duplicate)

Pseudo code:

```
insert (T, K):
    q = NULL
    p = root (T)
    while p ≠ nil and p.key ≠ K:
```

```
p = root(T)
while p ≠ nil and p.key ≠ K:
    q = p
    if p.key < K, p = p.right
    else p = p.left
if p ≠ nil, error Duplicate
N = new Node(K)
if q.key > K, q.left = N
else, q.right = N
```

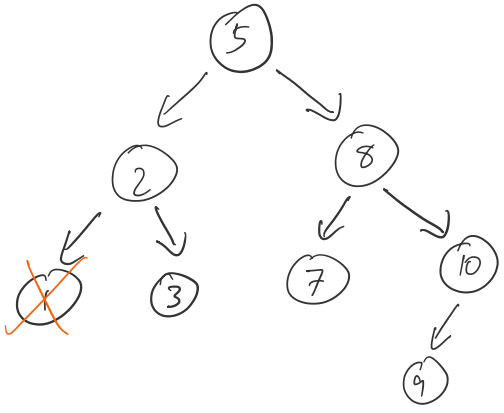
} find

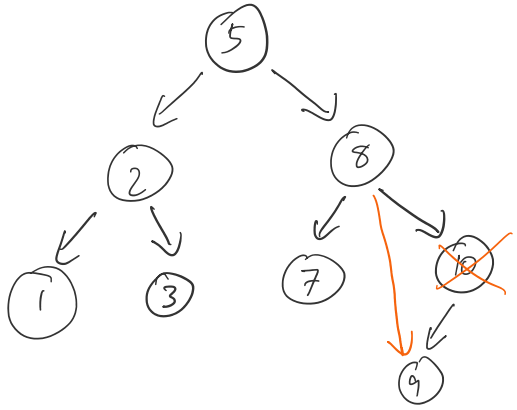
Delete: Find node u with parent p.

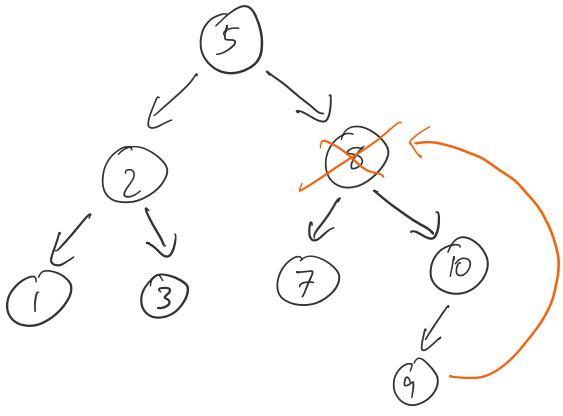
If u is a leaf, just delete

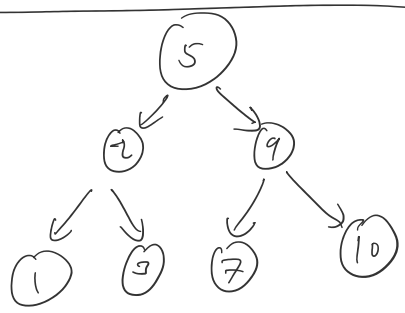
If u has 1 child c, delete u, and make c a child of p.

If u has 2 children, find smallest node in right subtree of u, delete it, and replace u with it.









9 still bigger than left subtree and smaller than right subtree

What can go wrong? Linearity

What would be optimal? Balance.

How can we achieve balance?

- many options, such as AVL trees, red-black trees, splay trees, etc.