Lecture 23 - Wednesday, April 5

Announcements

- ProgTest 1: Jackie (Office Hour)
- Assignment 4 released
- Exam guide to be released

BST

- search property
- sorting property (in-order traversal)
- searching
  - recursive
  - RT:
    - accurate O(h)
    - best case O(log n)
    - worst case O(n)
- insertion (searching)

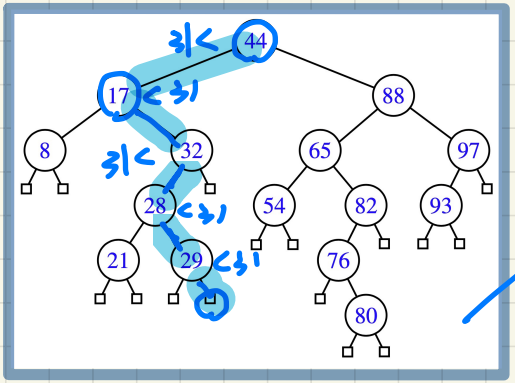
Visualizing BST Operation: Deletion

→ Exercise: Implement Case 3

Before deleting 32: 8 17 21 28 29 ~~32~~ 44

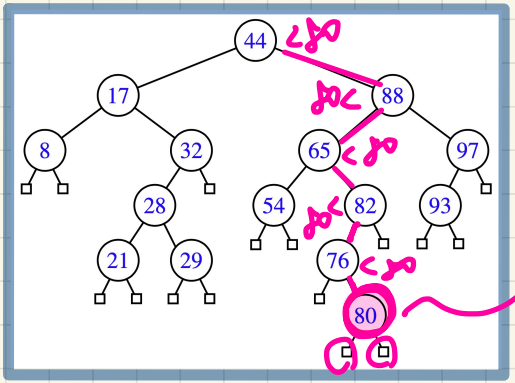
After deleting 32: 8 17 21 28 29 44

Case 1: Delete Entry with Key 31



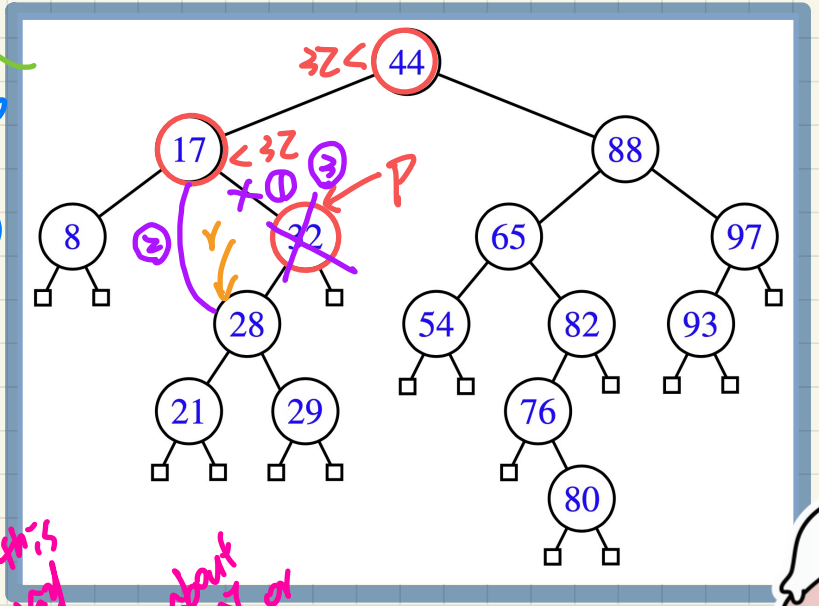
starting to delete RT: O(h)

Case 2: Delete Entry with Key 80

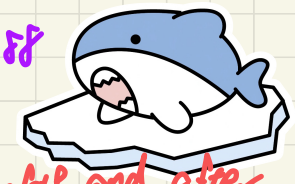


can delete this right away without worrying about any LST or RST → RT: O(h)

Case 3: Delete Entry with Key 32



Visualizing BST Operation: Deletion

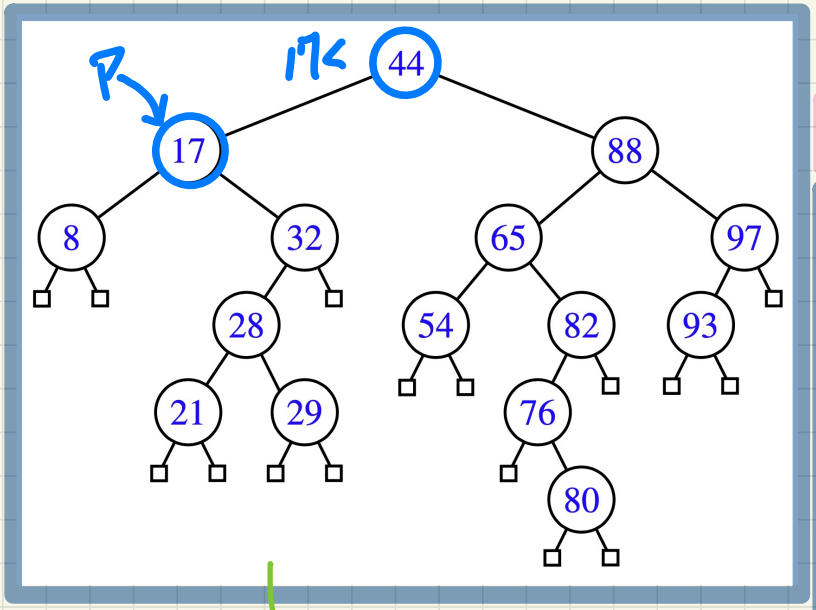


largest key that's smaller than 88

Before deleting 88: two keys before and after 88

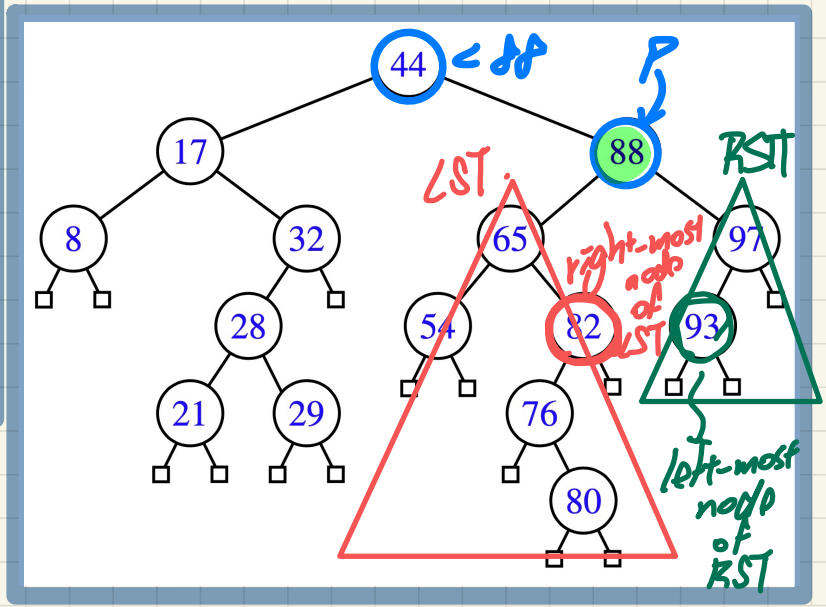
... 44 54 65 76 80 82 88 93 97

Case 4.1: Delete Entry with Key 17

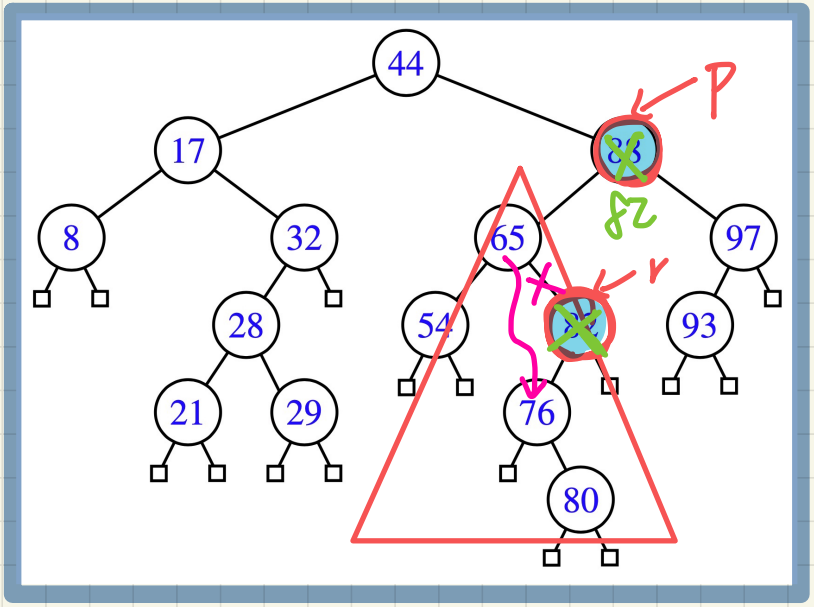


Exercise

Case 4.2: Delete Entry with Key 88



Case 4.2: Delete Entry with Key 88



P: to delete
r: largest key < 88

Choosing 82 or 93 works:
the resulting in-order traversals are identical.

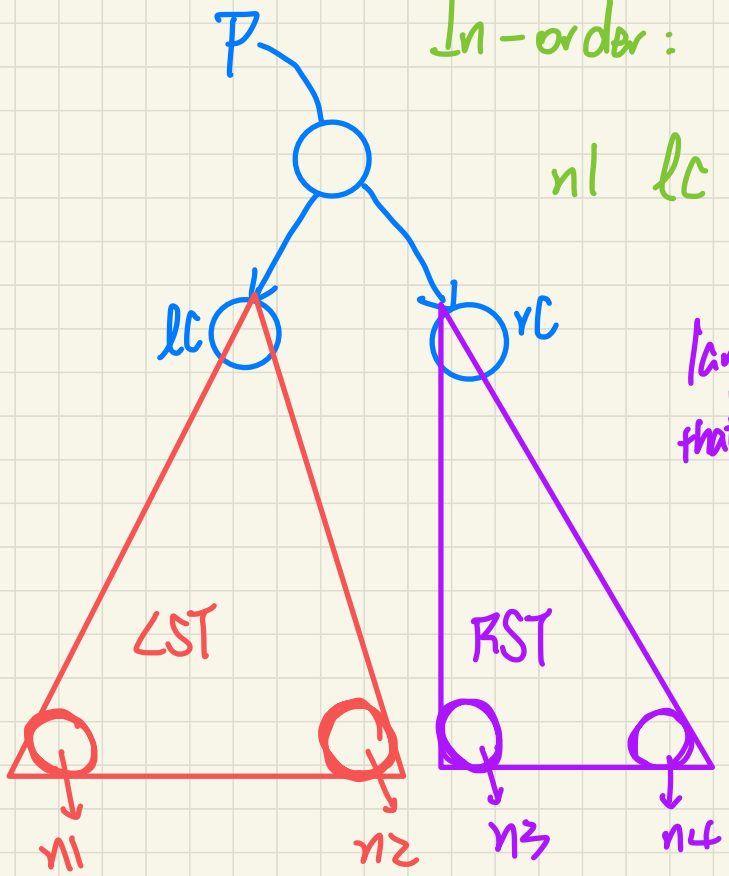
① Replace 88 by 82

② Connect 76 as the child of 65.

Exercise

Compare the in-order traversal results before & after the deletion steps.

BST



In-order:

n1 lc n2 p n3 rc n4

largest that's < p

smallest that's > p

Lecture

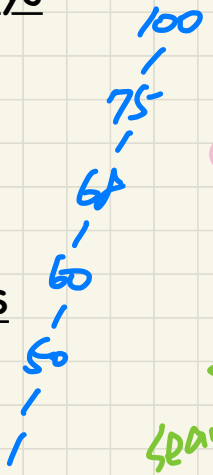
Balanced Binary Search Tree

Motivation and Property

Worst-Case RT: BST with Linear Height

Example 1: Inserted Entries with Decreasing Keys

<100, 75, 68, 60, 50, 1>



BST with height O(N)

searching/insertion/deletion can be O(N)

Example 2: Inserted Entries with Increasing Keys

<1, 50, 60, 68, 75, 100>

exercise

Example 3: Inserted Entries with In-Between Keys

<1, 100, 50, 75, 60, 68>

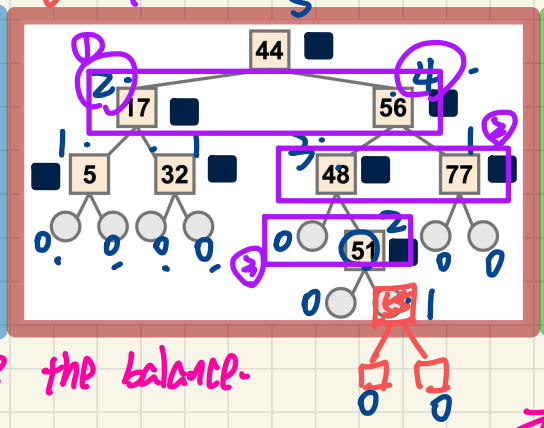
Balanced BST: Definition

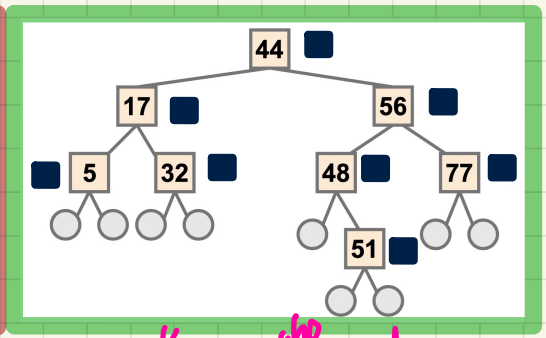
- internal node
- height
- height balance

Given a node p, the ***height*** of the subtree rooted at p is:

$$\textit{height}(p) = \begin{cases} 0 & \text{if } p \text{ is } \textit{external} \\ 1 + \textbf{MAX}\ (\{\ \textit{height}(c)\ |\ \textit{parent}\ (c) = p\ \}) & \text{if } p \text{ is } \textit{internal} \end{cases}$$

① violates height-balance prop.
$|2-4| > 1$





* should be taken to restore the balance.

Q. Is the above tree a balanced BST?

Q. Still a balanced BST after inserting 55?

Q. Still a balanced BST after inserting 63?

after it's detected that the insertion caused a violation of the height-bal prop. some steps*

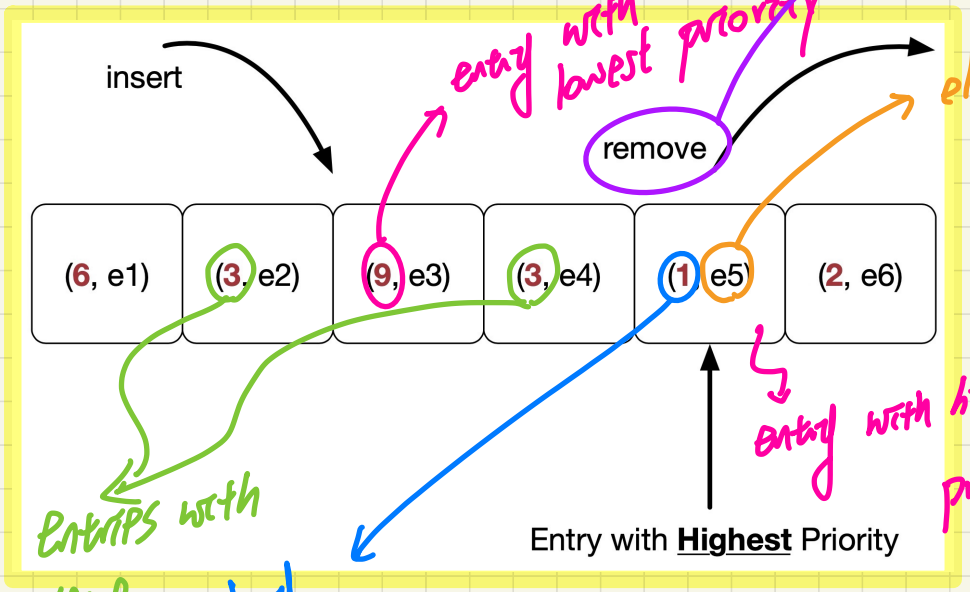
exercise

Lecture

Priority Queue

Intro & List-Based Implementations

What is a Priority Queue (PQ)



entry with lowest priority

remove entry with the highest priority element (i.e., lowest key value)

entry with highest priority

Entries with the same priority (when needed, doesn't matter which one is chosen)

key, denoting the priority

Compare PQ with FIFO queue

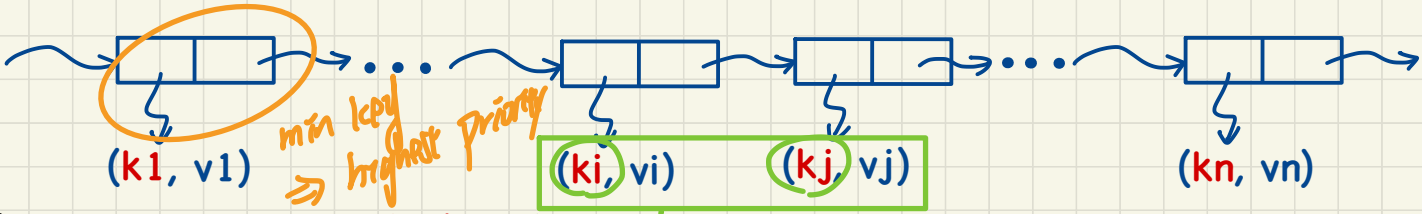
- entries in a FIFO queue is returned in chronological order.
- entries in a PQ is returned according to key val.

List-Based Implementations of Priority Queue (PQ)

PQ Method	List Method			
	SORTED LIST		UNSORTED LIST	
size	list.size *O(1)*			
isEmpty	list.isEmpty *O(1)*			
min	list.first	*O(1)*	search min	*O(n)*
insert	insert to "right" spot	*O(n)*	insert to front	*O(1)*
removeMin	list.removeFirst	*O(1)*	search min and remove	*O(n)*

highest priority

Approach 1: Sorted List



Approach 2: Unsorted List