BALANCED SEARCH TREES

David Kauchak
CS 62 – Spring 2021

1

Admin

Last day for "normal" mentor hours, Friday (5/7)

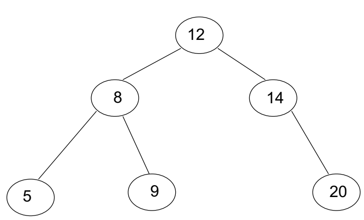
More on mentor hours next week

2

Binary Search Trees

BST – A binary tree where each each node has a key, and every node's key is:

- Larger than all keys in its left subtree. (everything left is smaller)
- Smaller than all keys in its right subtree. (everything right is larger)



3

Operations

Search – Does the key exist in the tree

Insert – Insert the key into tree

Delete – Delete the key from the tree

4

Height of the tree

Most of the operations take time
O(height)

We said trees built from random data have height
O(log n), which is asymptotically tight

Two problems:
- We can't always insure random data
- What happens when we delete nodes and insert others after building a tree?

Worst case height for binary search trees is O(n) ☹

5

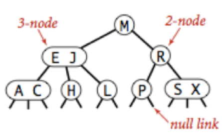
Balanced trees

Make sure that the trees remain balanced!
- Red-black trees
- AVL trees
- 2-3 trees
- 2-3-4 trees
- B-trees
- …

Height is guaranteed to be O(log n)

6

2-3 trees



Anatomy of a 2-3 search tree

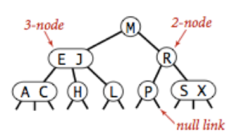
2-node: one key and two children (left and right)
- everything in left is smaller than key
- everything right is greater than (or equal to) key

3-node: two keys (k_1, k_2) and three children, left, middle and right
- $k_1 < k_2$
- everything in left is less than k_1
- everything in middle is between k_1 and k_2 (greater than or equal to k_1 and less than k_2)
- everything in right is greater than (or equal to) k_2

7

Search

How do we search for a key?



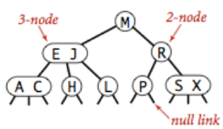
Anatomy of a 2-3 search tree

8

Search

Almost identical to BST search

Only difference: sometimes we have two keys

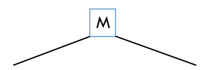


Anatomy of a 2-3 search tree

9

Search

Search(H)

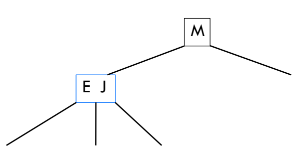


Which child?

10

Search

Search(H)

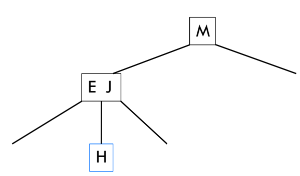


Which child?

11

Search

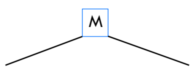
Search(H)



12

Search

Search(B)

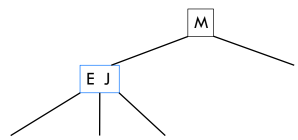


Which child?

13

Search

Search(B)

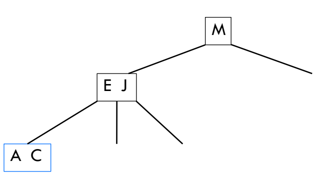


Which child?

14

Search

Search(B)

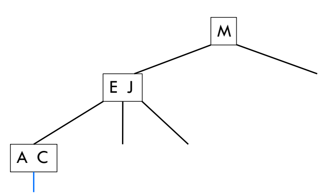


Which child?

15

Search

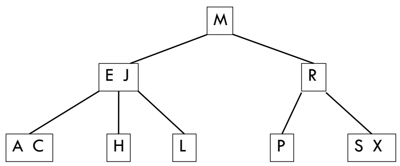
Search(B)



Not found!

16

Search



17

Insertion

Like BST, insert always happens at a leaf

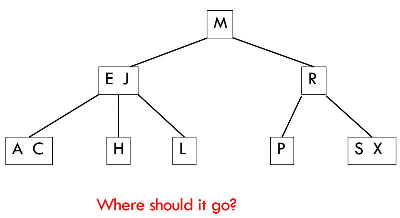
If the leaf is a 2-node, just insert it directly

18

Insertion

If the leaf is a 2-node, just insert it directly

Insert(F)

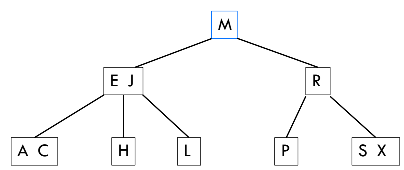


19

Insertion

If the leaf is a 2-node, just insert it directly

Insert(F)

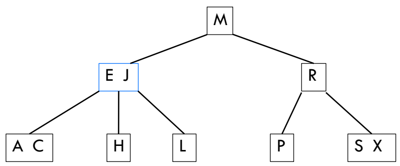


20

Insertion

If the leaf is a 2-node, just insert it directly

Insert(F)

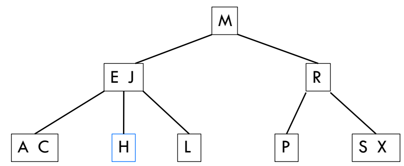


21

Insertion

If the leaf is a 2-node, just insert it directly

Insert(F)

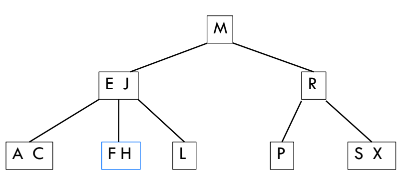


22

Insertion

If the leaf is a 2-node, just insert it directly

Insert(F)



23

Insertion

Like BST, insert always happens at a leaf

If the leaf is a 2-node, just insert it directly

If the leaf is a 3-node:
- We now have three values at this leaf
- Send the middle value up a node
- Make new 2-nodes out of the smallest and largest

24

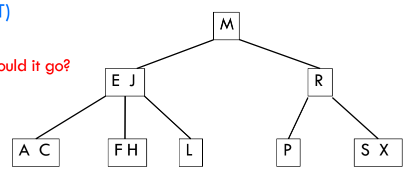
Insertion

If the leaf is a 3-node:

- We now have three values at this leaf
- Send the middle value up a node
- Make new 2-nodes out of the smallest and largest

Insert(T)

Where should it go?



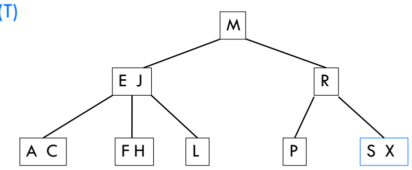
25

Insertion

If the leaf is a 3-node:

- We now have three values at this leaf
- Send the middle value up a node
- Make new 2-nodes out of the smallest and largest

Insert(T)



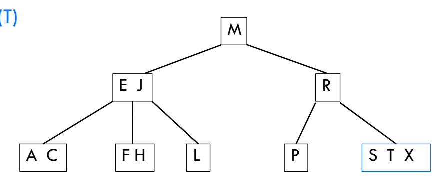
26

Insertion

If the leaf is a 3-node:

- We now have three values at this leaf
- Send the middle value up a node
- Make new 2-nodes out of the smallest and largest

Insert(T)



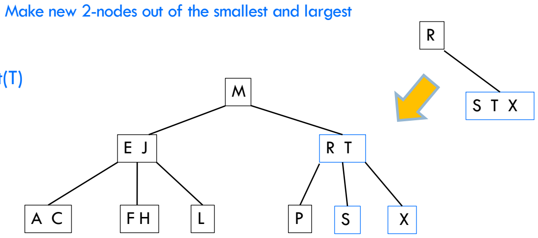
27

Insertion

If the leaf is a 3-node:

- We now have three values at this leaf
- Send the middle value up a node
- Make new 2-nodes out of the smallest and largest

Insert(T)



28

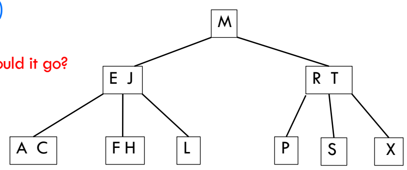
Insertion

If the leaf is a 3-node:

- We now have three values at this leaf
- Send the middle value up a node
- Make new 2-nodes out of the smallest and largest

Insert(I)

Where should it go?



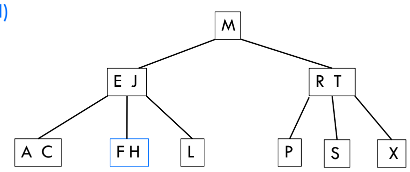
29

Insertion

If the leaf is a 3-node:

- We now have three values at this leaf
- Send the middle value up a node
- Make new 2-nodes out of the smallest and largest

Insert(I)



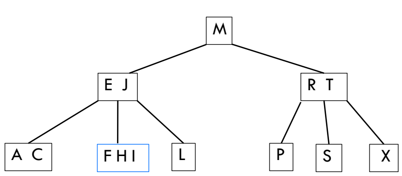
30

Insertion

If the leaf is a 3-node:

- We now have three values at this leaf
- Send the middle value up a node
- Make new 2-nodes out of the smallest and largest

Insert(I)



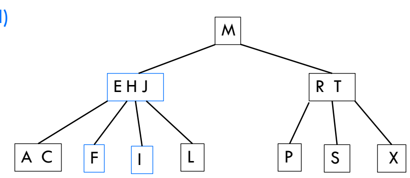
31

Insertion

If the leaf is a 3-node:

- We now have three values at this leaf
- Send the middle value up a node
- Make new 2-nodes out of the smallest and largest

Insert(I)



32

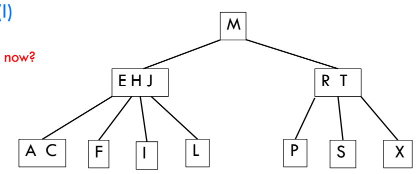
Insertion

If the leaf is a 3-node:

- We now have three values at this leaf
- Send the middle value up a node
- Make new 2-nodes out of the smallest and largest

Insert(I)

What now?



33

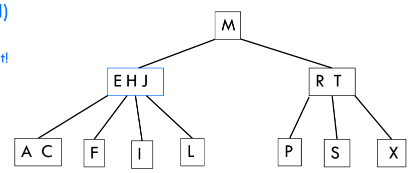
Insertion

If the leaf is a 3-node:

- We now have three values at this leaf
- Send the middle value up a node
- Make new 2-nodes out of the smallest and largest

Insert(I)

Repeat!



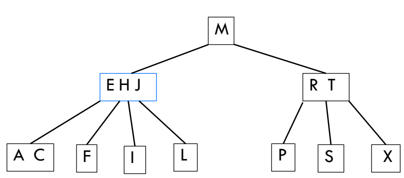
34

Insertion

If the leaf is a 3-node:

- We now have three values at this leaf
- Send the middle value up a node
- Make new 2-nodes out of the smallest and largest

Insert(I)



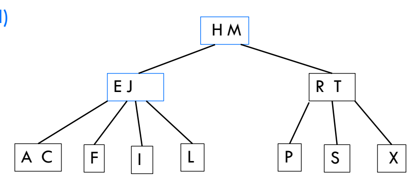
35

Insertion

If the leaf is a 3-node:

- We now have three values at this leaf
- Send the middle value up a node
- Make new 2-nodes out of the smallest and largest

Insert(I)



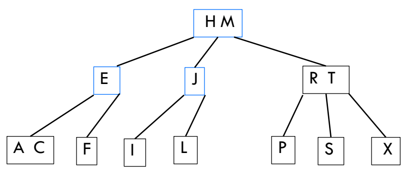
36

Insertion

If the leaf is a 3-node:

- We now have three values at this leaf
- Send the middle value up a node
- Make new 2-nodes out of the smallest and largest

Insert(I)



37

Insertion

If the leaf is a 2-node, just insert it directly

If the leaf is a 3-node:

- We now have three values at this leaf
- Send the middle value up a node
- Make new 2-nodes out of the smallest and largest

When will the height of the tree change?

38

Insertion

If the leaf is a 2-node, just insert it directly

If the leaf is a 3-node:

- We now have three values at this leaf
- Send the middle value up a node
- Make new 2-nodes out of the smallest and largest

Only when the root is a 3-node and we insert into a path that is all 3-nodes!

Effect: The tree can hold quite a few values before having to increase the height

39

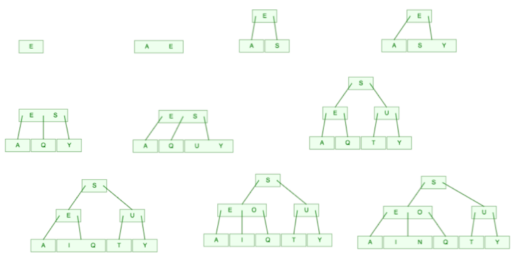
Practice

Draw the 2-3 tree that results when you insert the keys:
E A S Y Q U T I O N in that order in an initially empty tree.

40

Practice

Draw the 2-3 tree that results when you insert the keys:
E A S Y Q U T I O N in that order in an initially empty tree.



41

Running time

Worst case height: O(log n)

What does that mean?

42

Running time

Worst case height: O(log n)

Insert, search and delete are all O(log n)

43

2-3 search trees in practice

A pain to implement

Overhead can often make slower than standard BST

Other balanced trees exist that provide the same worst case guarantee, but are faster (e.g, red-black trees)

44

Red-black tree high-level

https://www.cs.usfca.edu/~galles/visualization/RedBlack.html

45