Administrative

Autocomplete

Lab tomorrow: Q&A + work session (make some progress on assignment!)

Midterm 2 next week

Pre-pre enrollment

Binary Search Trees

BST – A binary tree where each node has a value, and every node’s value is:
- Greater than all values in its left subtree. (everything left is smaller)
- Less than or equal to all values in its right subtree. (everything right is larger)

Binary Search Trees

Operations

Search – Does the key exist in the tree

Insert – Insert the key into tree

Delete – Delete the key from the tree
How do we find an element?
Finding an element

Search(9)

Finding an element

Search(13)

Finding an element

Search(13)

Finding an element

Search(13)
What is the worst case running time of search?
Finding an element

Search(9)

Worst case, have to search to the lowest leaf \( O(\text{height}) \)

Inserting

Insert(17)

How do we insert an element?
What is the worst case running time of insert?
Inserting

Insert(17)

Worst case, have to search to the lowest leaf $O(\text{height})$

Inserting duplicate

Insert(14)

Visiting all nodes

In sorted order

Visiting all nodes

In sorted order
What’s happening?
Visiting all nodes

Min/Max?

Successor and predecessor

Predecessor(12)?
Successor and predecessor

Successor in general?
- largest node of all those smaller than this node
- rightmost element of the left subtree

Successor

Successor(12)?
13

What if the node doesn't have a right subtree?
- smallest node of all those larger than this node
- leftmost element of the right subtree

Successor
What if the node doesn’t have a right subtree?

- The successor is the node that has x as a predecessor.
successor is the node that has x as a predecessor

keep going up until we’re no longer a right child

Deletion: case 1
No children
Just delete the node
Deletion: case 2

One child

Splice out the node

12

8

4

5

13

20

17

Deletion: case 3

Two children

Replace x with the smallest value of the right subtree

12

8

4

5

13

20

17

How does this maintain the search tree property?

Deletion: case 3

Two children

Replace x with the smallest value of the right subtree

- Larger than everything to the left
- Smaller than everything to the right
Deletion: case 3

Two children
Replace x with the smallest value of the right subtree

Deletion
Delete 21

Min of the right subtree

Replace the value: involves a case 2 deletion
Deletion

Replace the value: involves a case 2 deletion

Deletion: case 3

The min of the right subtree will always be either a case 1 deletion or a case 2 deletion

Why?

Deletion: case 3

The min of the right subtree will always be either a case 1 deletion or a case 2 deletion

Why?

Deletion: case 3

The min of the right subtree will always be either a case 1 deletion or a case 2 deletion

Why?
Deletion: case 3

The min of the right subtree will always be either a case 1 deletion or a case 2 deletion.

What is the worst case running time of delete?

Case 1 and Case 2: $O(1)$
Case 3: Find min and do a case 1 or case 2 delete $O(\text{height})$

Height of the tree

Most of the operations take time $O(\text{height})$

Trees built from random data have height $O(\log n)$

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 $O(n)$.

Why BSTs?

Hashables are fast at search/insert/delete, $O(1)$

Why BSTs?
Why BSTs?

Hashtables are fast at search/insert/delete, O(1)

Min/Max

Successor/predecessor

Inorder traversal

order statistics (5th largest element, etc.)

Operations

Search – Does the key exist in the tree

Insert – Insert the key into tree

Delete – Delete the key from the tree