Programming Abstractions

CS106B

Cynthia Lee
Topics:

- **Map** implemented as a Binary Search Tree (BST)
  - Starting with a dream: binary search in a linked list?
  - How our dream provided the inspiration for the BST
  - BST insert
  - Big-O analysis of BST

- Next time:
  - BST balance issues
  - Traversals
    - Pre-order
    - In-order
    - Post-order
    - Breadth-first
  - Applications of Traversals
Question about our put() algorithm:

Pretty simple!
- If key > node’s key
  › Go right!
- If key < node’s key
  › Go left!

FAQ. What do we do if the key is equal to the node’s key?

Stanford Map example:

```cpp
Map<int, string> mymap;
mymap.put(5, "five");
mymap.put(5, "cinco"); // what should happen?
cout << mymap.get(5) << endl; // what should print?
```
BST put() algorithm:

- If key > node’s key
  - Go right!
  - (if doesn’t exist—place here)
- If key < node’s key
  - Go left!
  - (if doesn’t exist—place here)
- If key is equal
  - Update value here!
BST put()

Insert: 22, 9, 34, 18, 3

How many of these result in the same tree structure as above?

22, 34, 9, 18, 3
22, 18, 9, 3, 34
22, 9, 3, 18, 34

A. None of these
B. 1 of these
C. 2 of these
D. All of these
Performance analysis:
What is the WORST CASE cost for doing `containsKey()` in BST?

A. \(O(1)\)
B. \(O(\log n)\)
C. \(O(n)\)
D. \(O(n \log n)\)
E. \(O(n^2)\)
What is the worst case cost for doing containsKey() in BST *if the BST is balanced*?

O(logN)—awesome!

BSTs are **great when balanced**

BSTs are **bad when unbalanced**

…and **Balance depends on order of insert** of elements…

…but user controls this, not “us” (author of the Map class)

…no way for “us” (author of Map class) to ensure our Map doesn’t perform terribly! 😞 😞
Ok, so, long-chain BSTs are bad, should we worry about it? [math puzzle time]

One way to create a bad BST is to insert the elements in *decreasing* order: 34, 22, 9, 3
That’s not the only way…

How many **distinctly structured** BSTs are there that exhibit the worst case height
(height equals number of nodes) for a tree with the 4 nodes listed above?

A. 1-3
B. 4-5
C. 6-7
D. 8-9
E. More than 9

*Bonus question: general formula for any BST of size n?*
*Extra bonus question (CS109): what is this as a fraction of all trees (i.e., probability of worst-case tree).*
BST and Heap quick recap/cheat sheet
BST and Heap Facts (cheat sheet)

**Heap** *(Priority Queue)*
- **Structure:** must be “complete”
- **Order:** parent priority must be $\leq$ both children
  - This is for min-heap, opposite is true for max-heap
  - *No rule* about whether left child is $>$ or $<$ the right child
- **Big-O:** guaranteed log(n) enqueue and dequeue
- **Operations:** always add to end of array and then “bubble up”; for dequeue do “trickle down”

**BST** *(Map)*
- **Structure:** any valid binary tree
- **Order:** leftchild.key $<$ self.key $<$ rightchild.key
  - *No duplicate keys*
  - Because it’s a Map, values go along for the ride w/keys
- **Big-O:** log(n) if balanced, but might not be balanced, then $O(n)$
- **Operations:** recursively repeat: start at root and go left if key $<$ root, go right if key $>$ root