SI Leader: Ethan Ho
Email: erh101@case.edu

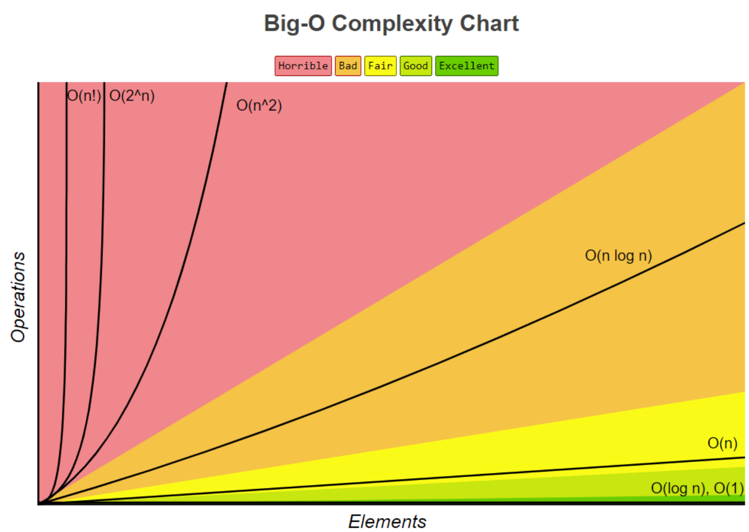
CSDS 233 - Midterm Summary Session

Objectives: This session aims to cover all of the topics from the first half of the semester.
Topics to cover:

- Runtime analysis
- Linked lists
- Stacks
- Queues
- Binary trees
- Binary search trees
- AVL trees (not deletions)

Problems:

Runtime analysis:



1. Simplify the following O-notation expressions.
   a. $O(2 + 4 + 6 + 8 + \ldots + 1000)$

      Not $O(n^2)$ $\quad O(502) \rightarrow O(1)$

   b. $O((n^2 + 2n)(4n + 1))$

      $O(n^3)$

   c. $O((\log_3 n)^2 + \log_3 n^3)$

      $O((\log n)^2)$

SI Leader: Ethan Ho
Email: erh101@case.edu

2. What foo's time complexity expressed in Big-O?

```
int foo(int arr[], int x)
{
    int l = 0, r = arr.length - 1;
    while (l <= r) {
        int m = l + (r - l) / 2;
        if (arr[m] == x)
            return m;
        if (arr[m] < x)
            l = m + 1;
        else
            r = m - 1;
    }
    return -1;
}
```

Midpoint-ish

Cutting in half every iteration

O(logn)

3. Write a method (pseudocode) that searches for the maximum element of an array in linear time.

```
if (arr.length == 0) throw err.
int max = arr[0]
for (int i = 1; i < arr.length; i++) {
    if ( arr[i] > max)
        max = arr[i];
}
return max;
```

Throw relevant exception if Array is empty

Start @ 2nd Element

update max if current element exceeds it

Linked Lists:

4. Let's say you were trying to create your own singly linked list. You'll achieve this by writing a LinkedList class and a ListNode, which represents a node in the LinkedList. What are some of the fields and methods you would include in each class to achieve this?
   a. LinkedList class fields and methods

      private ListNode head;

      LinkedList(ListNode head);

      size()

      add() remove()

      Getter/setters if required

   b. ListNode class fields and methods

      ListNode(T element, ListNode next);

      private T element; private ListNode next;

5. Explain the following procedures in a doubly linked list.
   a. Search for an element e.

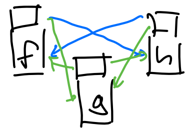
      Traverse through list to find element e and return it.

      *Same as singly linked list return False or null as requested

SI Leader: Ethan Ho
Email: erh101@case.edu

b. You are at a node containing element f and its next node contains element h. Insert a new node containing element g in between the two nodes.

f.next = g; g.prev = f
h.prev = g ; g.next = h



c. You are at node element f. Node element f has a preceding node element e and a following node element g. Delete f.

f.prev.next = f.next
f.next.prev = f.prev

Stacks and queues

6. Explain why stacks are considered last-in-first-out (LIFO) data structures and queues are considered first-in-first-out (FIFO) data structures.

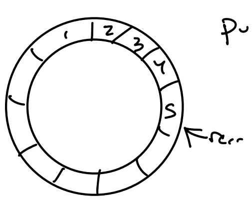
Stacks: First Element in is at bottom of stack & everything above it must be removed in order to access it

Queues: Elements added go to Queue head and must be removed to access other elements

7. Explain how a Pringles can relates to a stack. Explain the processes of pop, push, and peek on a Pringles can.

Pop: Take top pringle
Push: Put a pringle back into can
Peek: Look at top of can

8. Let's say you're making your own queue class via a circular array implementation. Write the insert method of your queue class. You have the following fields:
   a. isFull - boolean that returns if the array is full or not
   b. rear - int index of the end part of the filled section
   c. items - int array of the items




```
public boolean insert(T element) {
    if (isFull) return false;
    items += 1;
    rear = (rear + 1) % arr.length;
    arr[rear] = element;
    return true;
}
```

Binary search trees

9. Explain the process of inserting a node into a BST.

Search for the node in a tree starting with parent = null and a traversal node being the root. Look for next open spot for a node, so look until trav is null. Parent should stay one above trav. Move trav left if inserted key < trav.key, move right otherwise. Once trav is null, you can insert into the tree. Parent was one above trav so use it as a reference. If parent is null, tree is empty, insert new node as root. Otherwise check key against parent key. If key < parent.key the new node on parent's left subtree. Otherwise go on right subtree

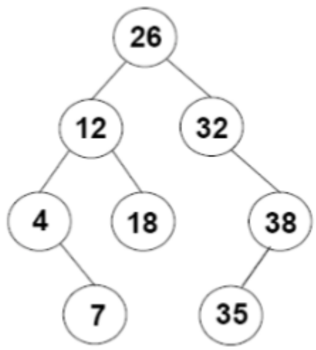
SI Leader: Ethan Ho
Email: erh101@case.edu

10. Write a recursive search method for a BST. You have the following fields:
    a. root - Node that is the root of the tree (entry point to the tree)
    b. root.key - Key of the Node

```
public T search(T key) {
    Node n = searchTree(root, key);
    return n == null ? null : n.key;
```

```
public Node searchTree(Node root, T key)
    if (root == null)
        return null;
    else if (root.key == key)
        return root;
    else if (key < root.key)
        return searchTree(root.left, key);
    else
        return searchTree(root.right, key);
```

11. Examine the following tree.



    a. If we delete node 26, what node would we use to replace it?

32 ← Smallest value in the right

    b. In general, what is the rule when we delete a node with 2 children?

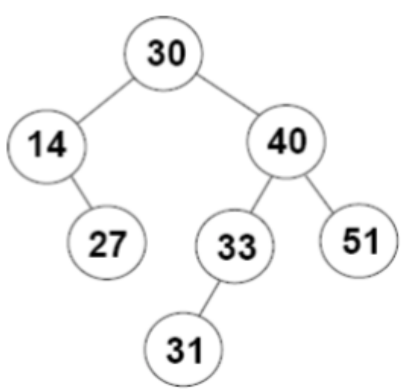
In-order successor of 26   Smallest value in node-to-delete's right subtree

    c. Write the postorder traversal of this tree.

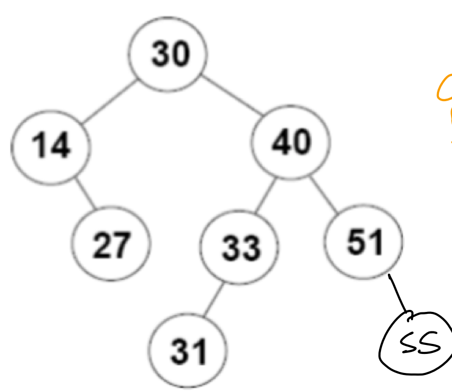
7 4 18 12 35 38 32 26

12. Examine the following AVL tree. Draw how the tree changes for each insertion. **Assume that the tree resets for each part**.

Should be big on cheat sheet!!



    a. Insert 55



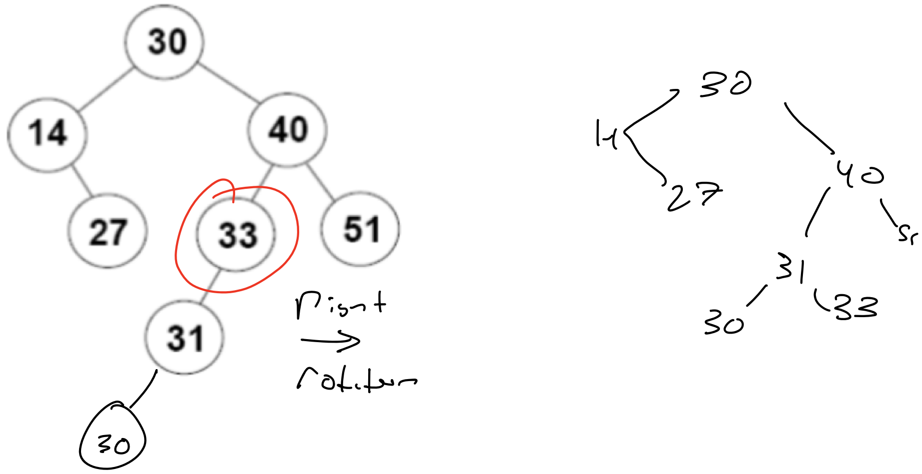
Check each ancestor and check balance

Check if Tree is Balanced after insertion

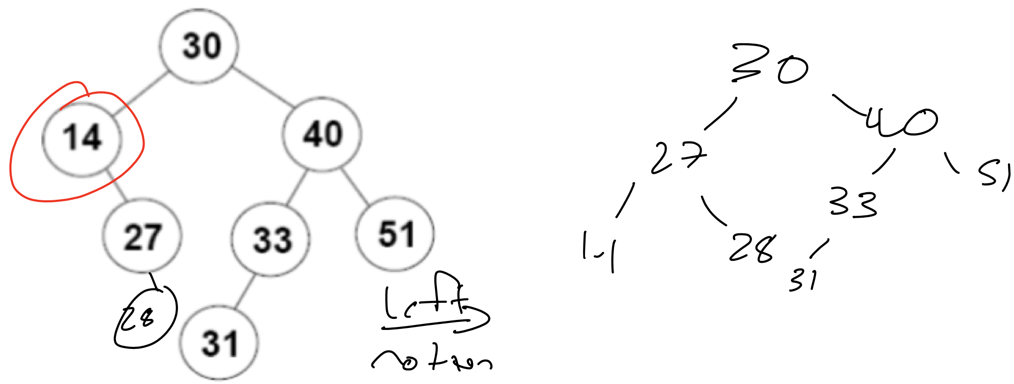
All node ±1 balance

SI Leader: Ethan Ho
Email: erh101@case.edu

b. Insert 3(

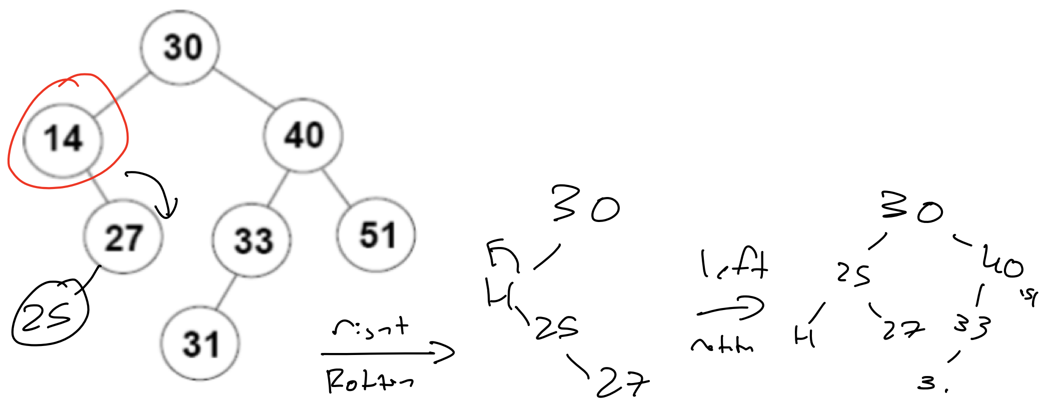


c. Insert 28



d. Insert 25

Double Rotation!



Conclusion:
Please review the lecture slides and rewatch lectures on Echo 360.

Midterm is on 10/15.

Websites

Big O Cheatsheet

University of San Francisco Data structures visualizer link in email