Algorithmic Analysis

Joe

Introduction

Session Objective

Please clone:
https://github.com/drJAGartner/big_o_demo

1. Write an implementation of Insertion sort

2. Use big-O notation to describe the performance of:
   * Insertion sort
   * Binary Search

Scenario

Algorithmic Time Complexity

Sorting

- Sorting a list of numbers is a common question in computer science, and will often make it's way into DS interviews.

  - The reason is that it is a good test of several things:
    - Estimating the growth in time for an algorithm as the input size grows
    - Evaluate your knowledge of data structures
    - They test your ability to think about what you are doing

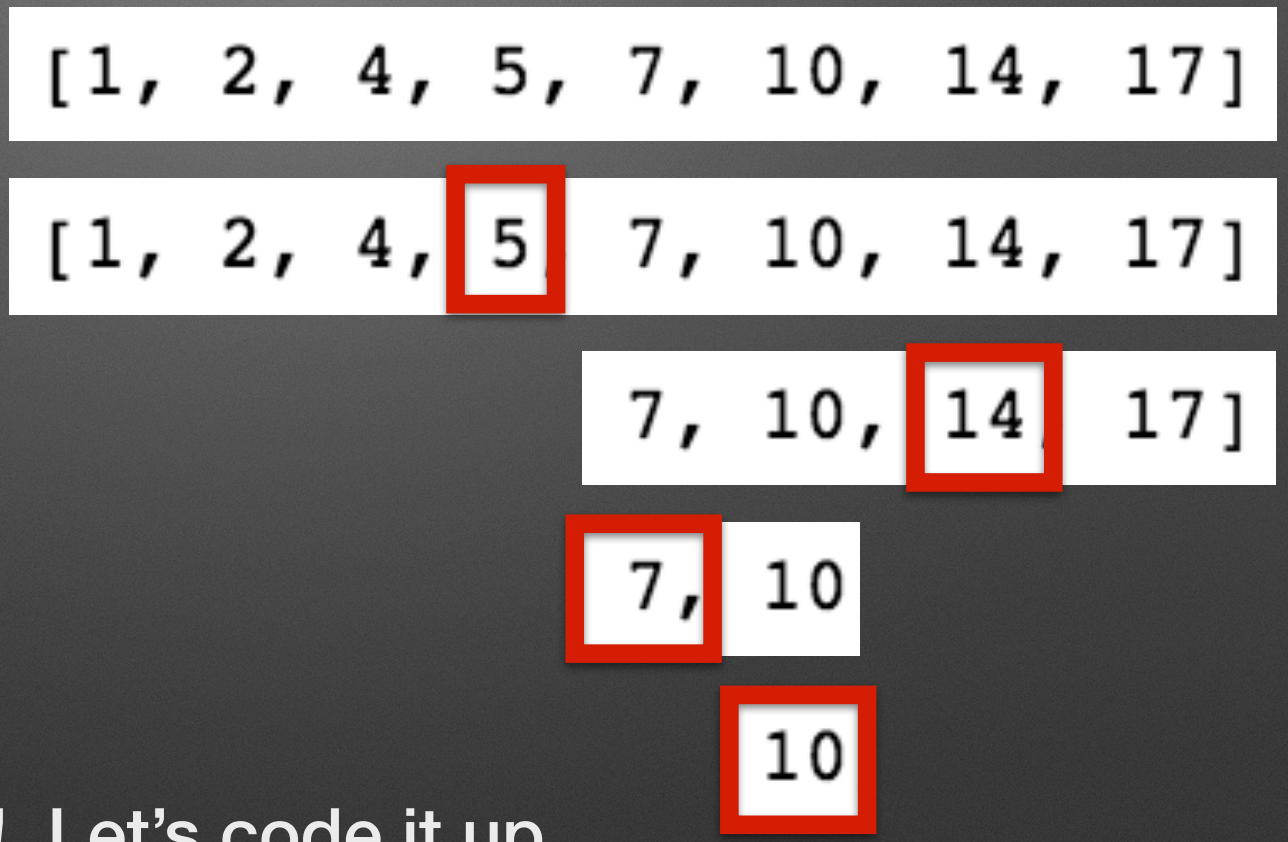
Big O Notation

- Big O Notation - Used to describe how the runtime (and to a lesser extent, memory) of function increases as the size of the input array increases.
- This is an order of magnitude approximation, meaning we only worry about the leading term
  - Example: Whiteboard bubble sort

Common Data Structures/ Algorithms to Know

Binary Search

Is 8 in my list?

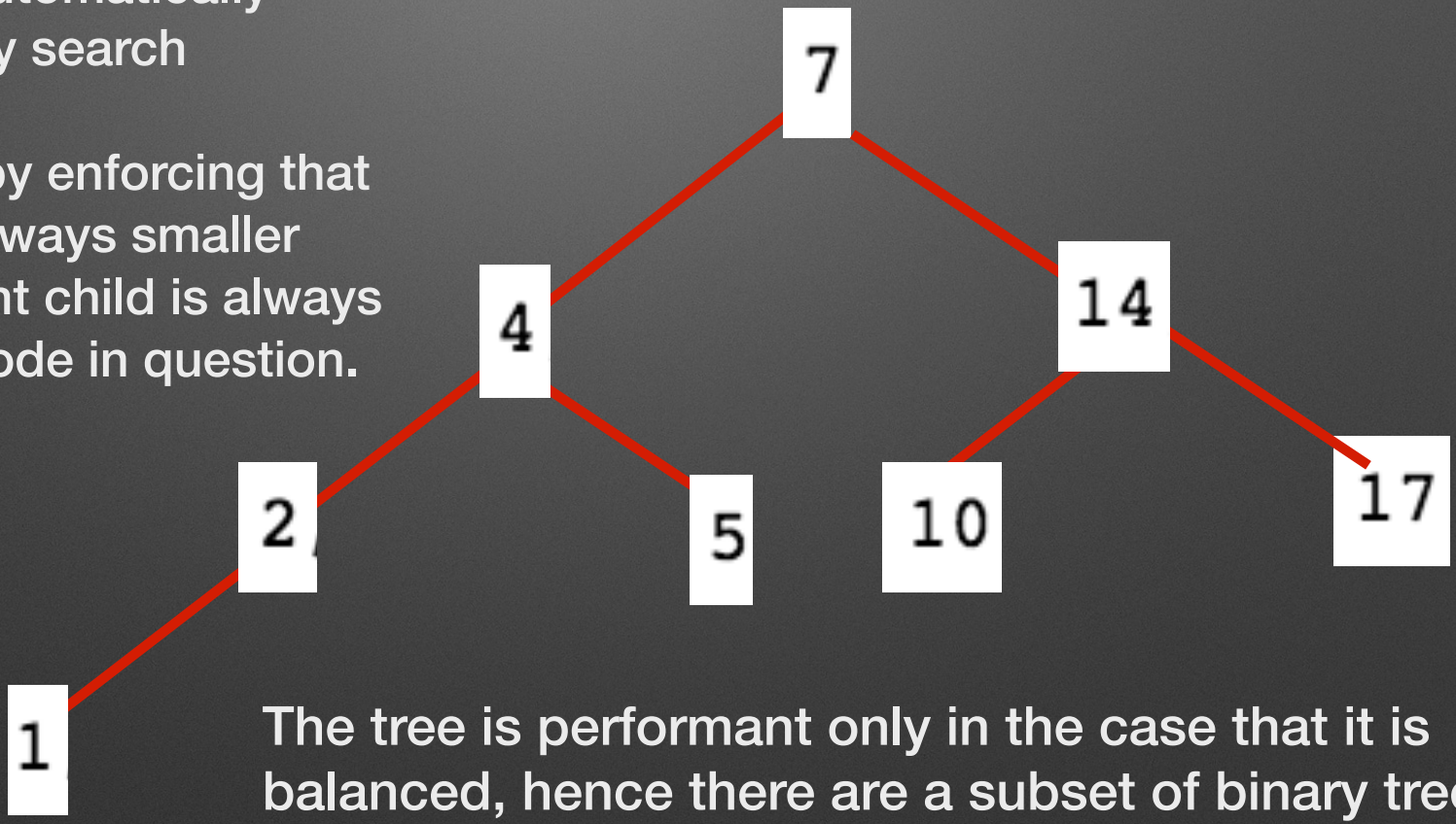


Nope! Let's code it up.

Binary Tree

Binary Search Trees are built with the purpose of automatically performing binary search

The tree is built by enforcing that the left child is always smaller than, and the right child is always larger than the node in question.



The tree is performant only in the case that it is balanced, hence there are a subset of binary trees called ‘self balancing trees’, the most famous of which are called ‘red black trees’

A Few More Concepts

- Better search algorithms:
  - Heap Sort
  - Quick Sort
- Graph Search - depth first vs. breadth first
- A* - shortest path

Parting Thoughts

- Read - Introduction to Algorithms
- Practice - Cracking the Coding Interview
- Avoid nested loops, $O(n^2)$ is bad, anything worse is bad(^{er})