# BINARY SEARCH TREES RUNNING TIME

Problem Solving with Computers-II

# How is PA01 going?

A. Done!
B. On track to finish
C. Made some progress but with difficulty
D. Haven't started

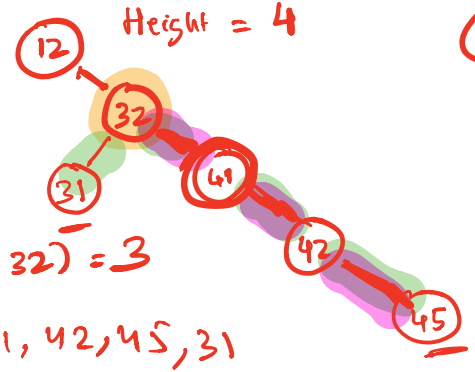
# Binary Search Trees

- WHAT are the operations supported?
- HOW do we implement them?
- WHAT are the (worst case) running times of each operation?

# Height of the tree = Height of the root

- Path – a sequence of nodes and edges connecting a node with a descendant.
- A path starts from a node and ends at another node or a leaf
- Height of node – The height of a node is the number of edges on the longest downward path between that node and a leaf.

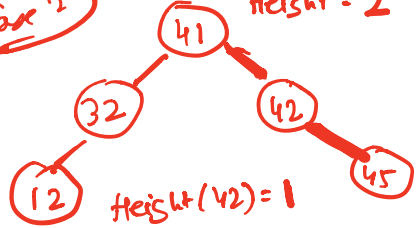
Case 1




Height (32) = 3

12, 32, 41, 42, 45, 31

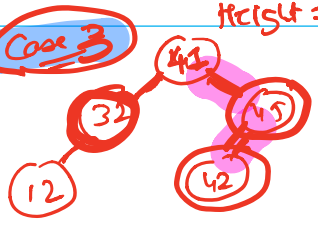
Case 2




Height (42) = 1

41, 32, 42, 45, 12

Case 3




Path: 41 → 45 → 42

45 → 42

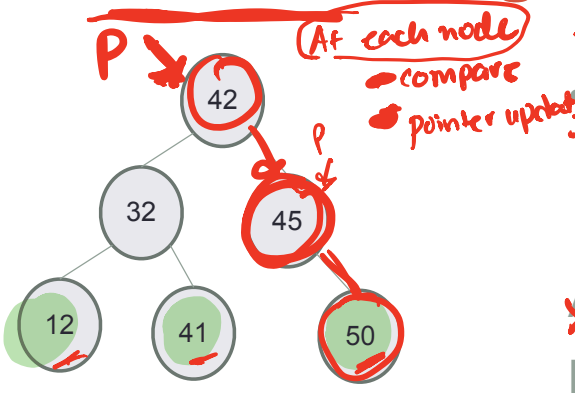
32 → 41 → 45 → 42

BSTs of different heights are possible with the same set of keys

Examples for keys: 12, 32, 41, 42, 45

41, 32, 45, 42, 12

# Worst case Big-O of search

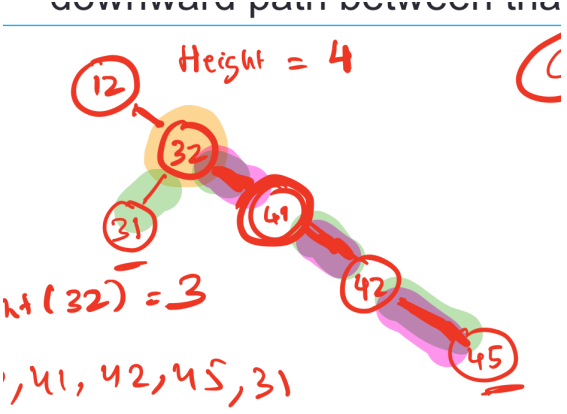



At each node
- compare
- pointer update

→ O(1)

Given a BST of height H with N nodes, what is the worst case complexity of searching for a key?

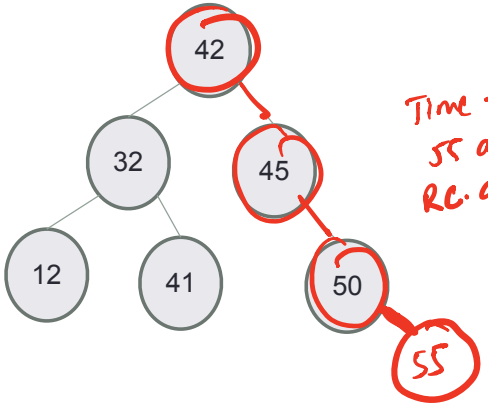
A. O(1)
B. O(log H)
C. O(H)
D. O(H*log H)
E. O(N)




Best case : Search (42)
O(1)

Worst case : Search (50)
O( )

# Worst case Big-O of insert




Time to insert 55 as the RC. of 50 O(1)

Worst case : insert (55)

O(H) + O(1)

- Given a BST of height H and N nodes, what is the worst case complexity of inserting a key?

A. O(1)
B. O(log H)
C. O(H)
D. O(H*log H)
E. O(N) ← BST looks like a linked list.

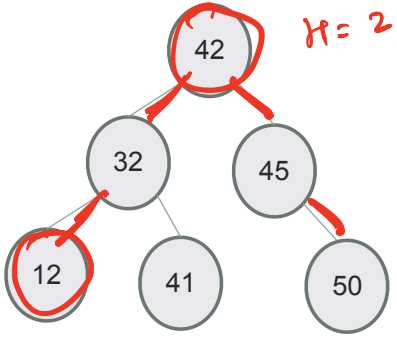
Worst case

H = (N - 1)

O(H) = O(N)

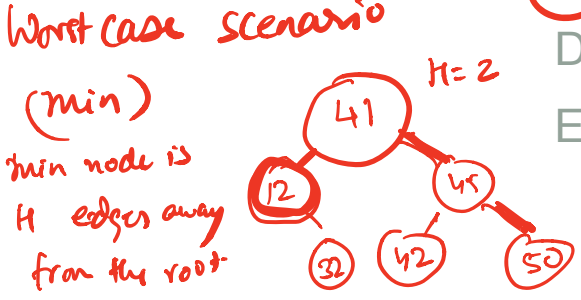
$O\left(\frac{f(N)}{N}\right)$

# Worst case Big-O of min/max



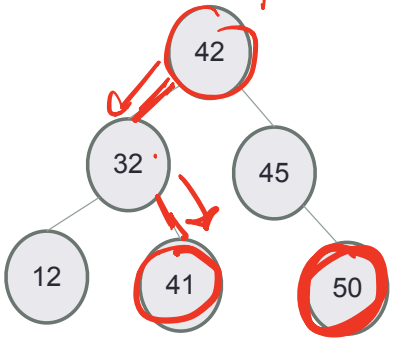

- Given a BST of height H and N nodes, what is the worst case complexity of finding the minimum or maximum key?

A. O(1)
B. O(log H)
C. O(H)
D. O(H*log H)
E. O(N)



# Worst case Big-O of predecessor/successor

pred(42) = 41

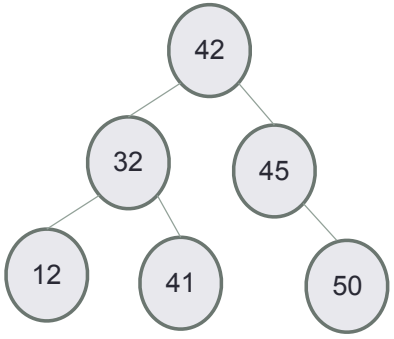



Worst: pred of root

- Given a BST of height H and N nodes, what is the worst case complexity of finding the predecessor or successor key?

A. O(1)
B. O(log H)
C. O(H)
D. O(H*log H)
E. O(N)

# Worst case Big-O of delete



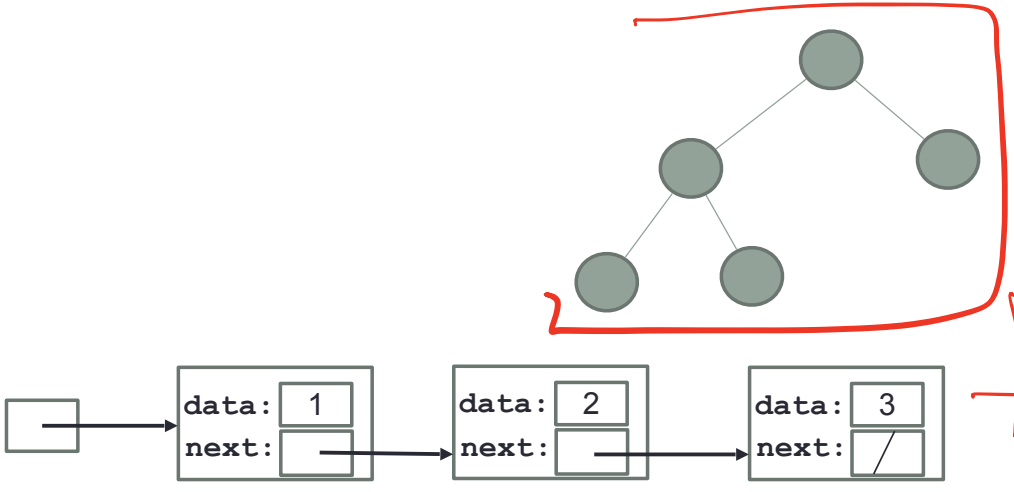

- Given a BST of height H and N nodes, what is the worst case complexity of deleting the key (assume no duplicates)?

A. O(1)
B. O(log H)
C. O(H)
D. O(H*log H)
E. O(N)

# Worst case analysis

Are binary search trees *really* faster than linked lists for finding elements?

- A. Yes
- B. No



# Balanced BST

A balanced BST, by definition is one where $H = O(\log N)$

Examples of balanced BSTs include AVL trees, Red-Black Trees.

We will show that a completely filled tree (next slide) is an example of a balanced BST i.e. $H = O(\log N)$

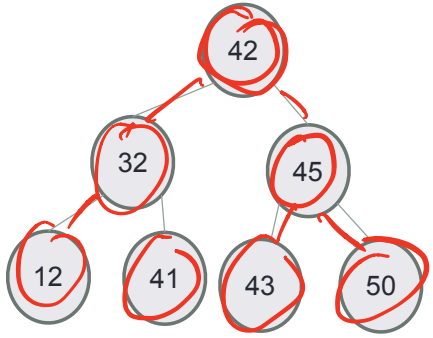
# Completely filled binary tree

Balanced BST
H = O(log N)
insert: O(log N)

Level 0

Level 1

Level 2




Nodes at each level have exactly two children, except the nodes at the last level

# Relating H (height) and N (#nodes)

find is O(H), we want to find a f(N) = H

$H = c_1 \log N + c_2$

level → # of nodes

0 → 1

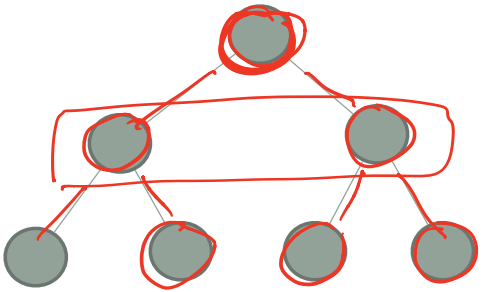
1 → 2

2 → $2^2$

3 → $2^3$

L → $2^L$

L = H




$1+2+4+8+\cdots 2^7$

$= 2^8 - 1$

$= 255$

How many nodes are on level L in a completely filled binary search tree?

A. 2

B. L

C. 2*L

D. $2^L$

$$N = \sum_{l=0}^{H} 2^l = 2^{H+1} - 1$$

A nice analogy is thinking about the largest number that can be represented in b bits

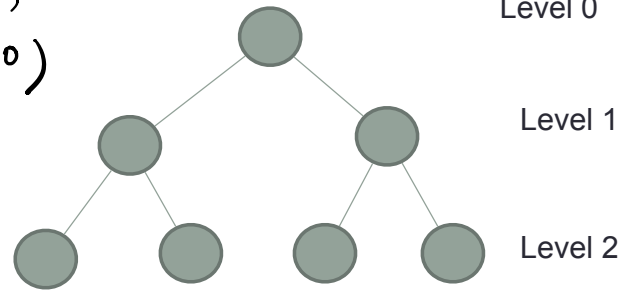
# Relating H (height) and N (#nodes)
# find is O(H), we want to find a f(N) = H

8 bits. Binary: (1 1 1 1 1 1 1 1)

$(2^7 + \dots + 2^2 + 2^1 + 2^0)$

$= 2^8 - 1$

$N = 2^{H+1} - 1$




Finally, what is the height (exactly) of the tree in terms of N?

$2^{H+1} = N + 1$

$H = \log_2(N+1) - 1$

$O(H) = O(\log_2 N)$

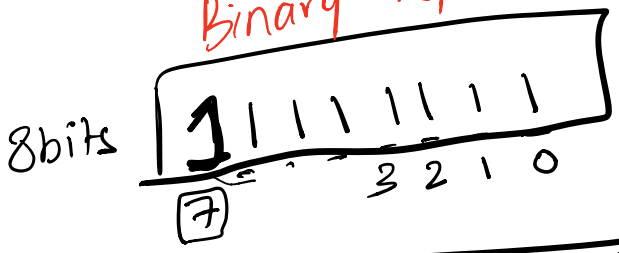
$$\sum_{i=0}^{n-1} 2^i = 2^n - 1$$

$$\sum_{i=0}^{H} 2^i = 2^{H+1} - 1$$

Binary representation

Value in decimal

8bits




$2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0$

$= 255 = 2^8 - 1$

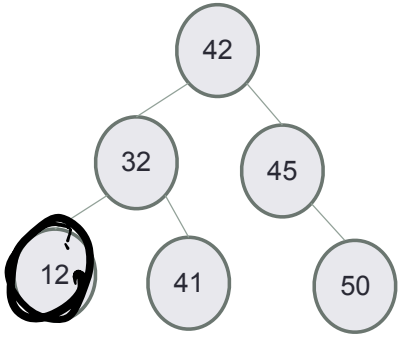
$2^7 =$ 1 0 0 0 0 0 0 0 → 128

$2^8 =$ 1 0 0 0 0 0 0 0 0 → 256 $= 2^8$

$2^0 + 2^1 + 2^2 + \cdots 2^7 = 2^8 - 1$

$2^0 + 2^1 + 2^2 + \cdots 2^H = 2^{H+1} - 1$

# Big O of traversals




In Order: O(N)

Pre Order: O(N)

Post Order: O(N)

# Balanced trees

- Balanced trees by definition have a height of O(log N)
- A completely filled tree is one example of a balanced tree
- Other Balanced BSTs include AVL trees, red black trees and so on
- Visualize operations on an AVL tree: https://visualgo.net/bn/bst

# Summary of operations

| Operation | Sorted Array | BST | Balanced BST | Linked List |
|---|---|---|---|---|
| Min | | | | |
| Max | | | | |
| Median | | | | |
| Successor | | | | |
| Predecessor | | | | |
| Search | | | | |
| Insert | | | | |
| Delete | | | | |