Binary search trees (BST)

- used to store dictionaries.

Def. A BST is a rooted, ordered tree which is either

1) empty

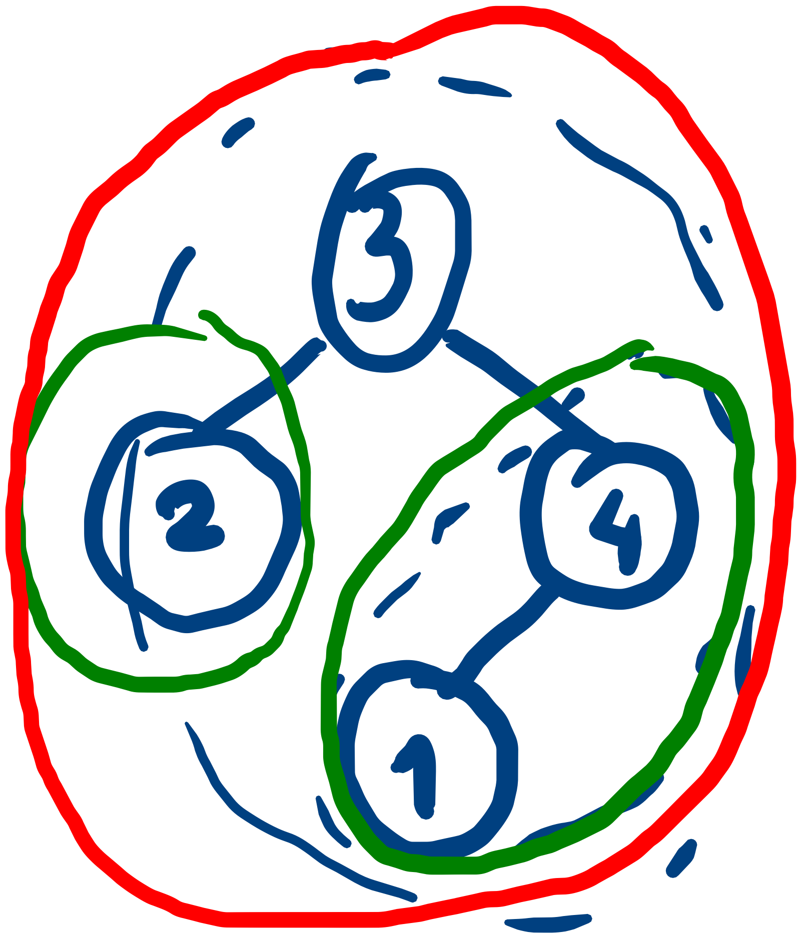
root

2) A node with a left and right subtrees that are BSTs.

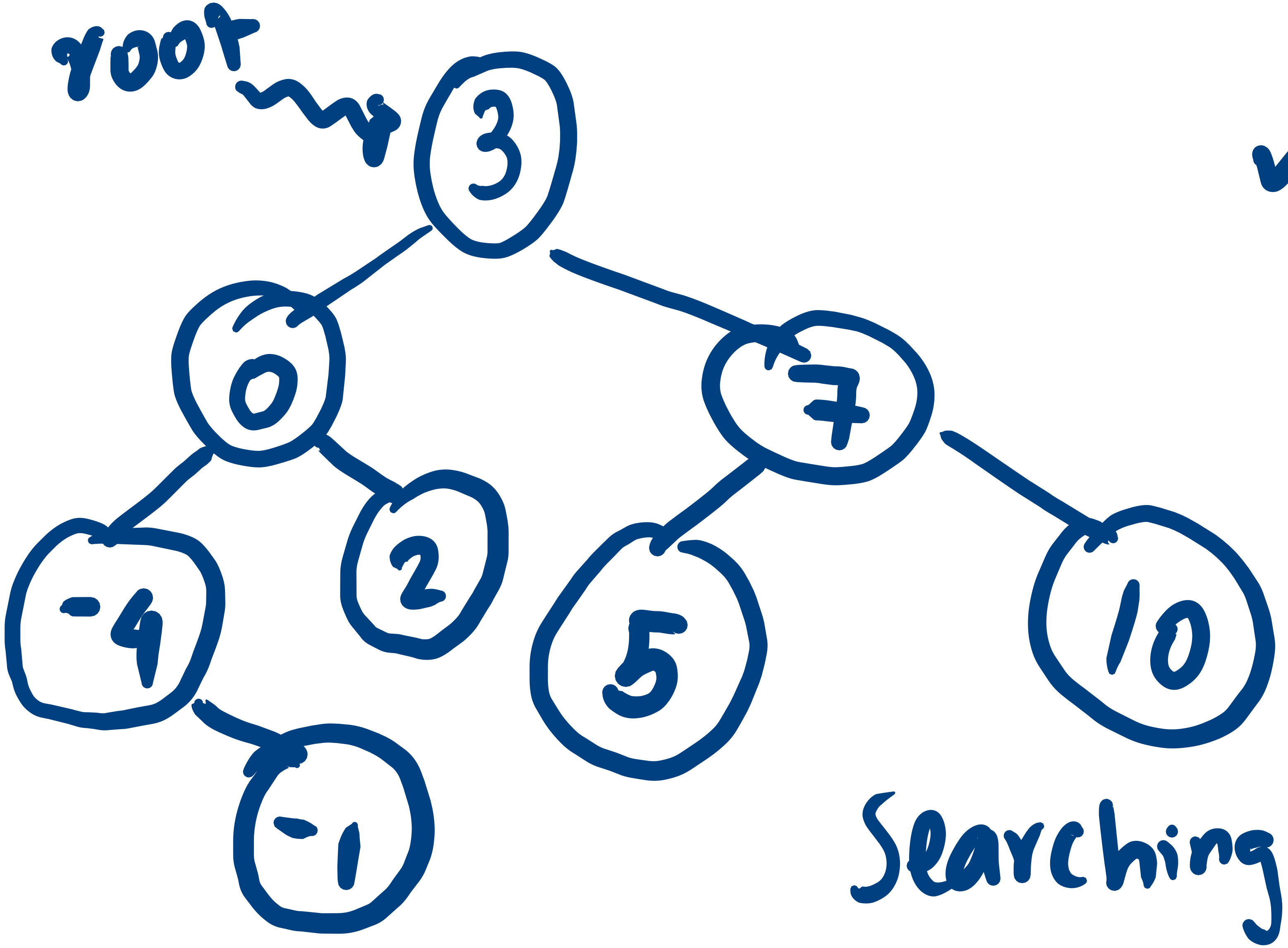
- All keys in the left subtree are less than the key in the root node
- All keys in the right subtree are greater than the key in the root node.

empty tree ✓

or



key 1 occurs in the right subtree of 3.

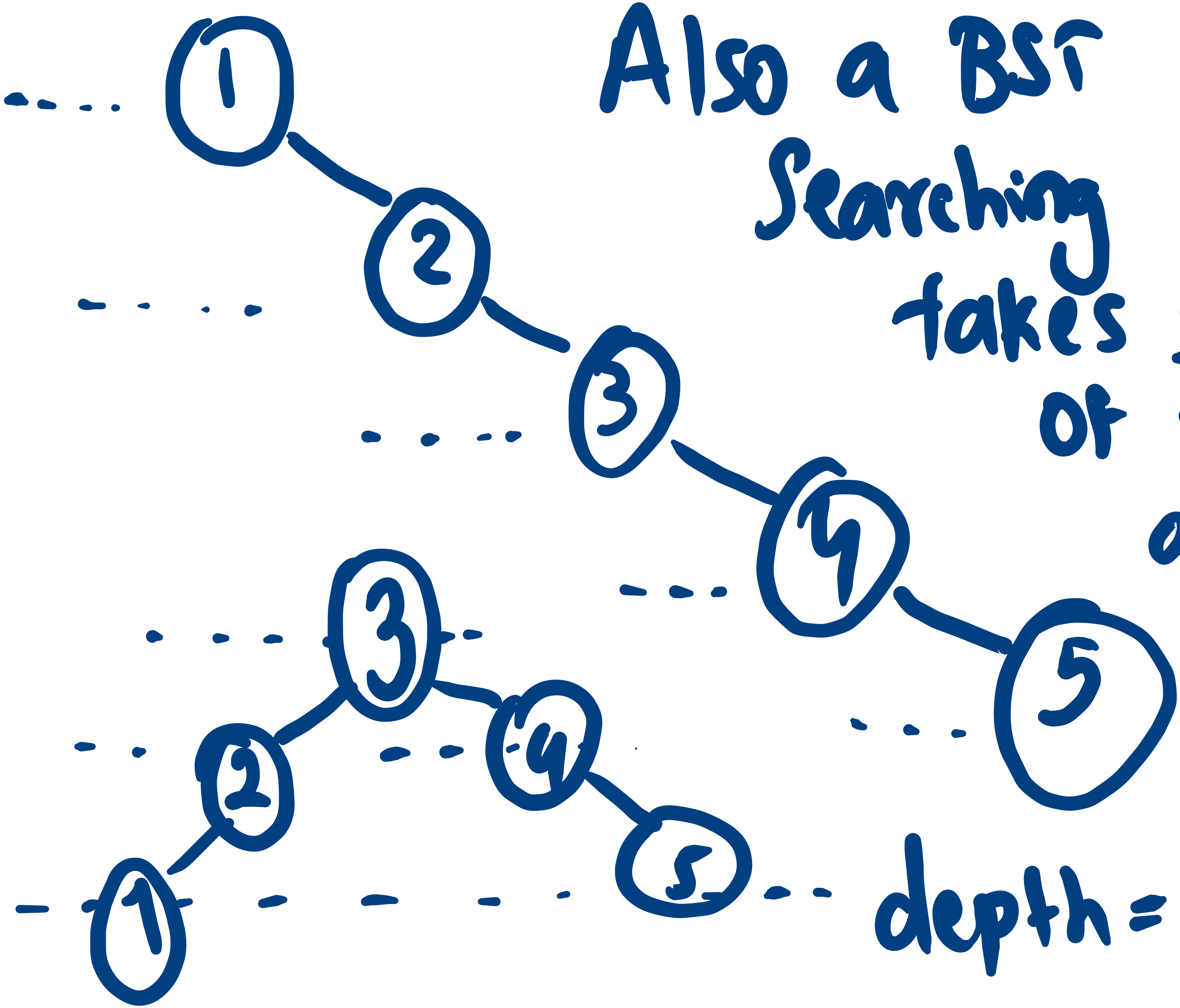


✓ BST

Recursively satisfies all properties

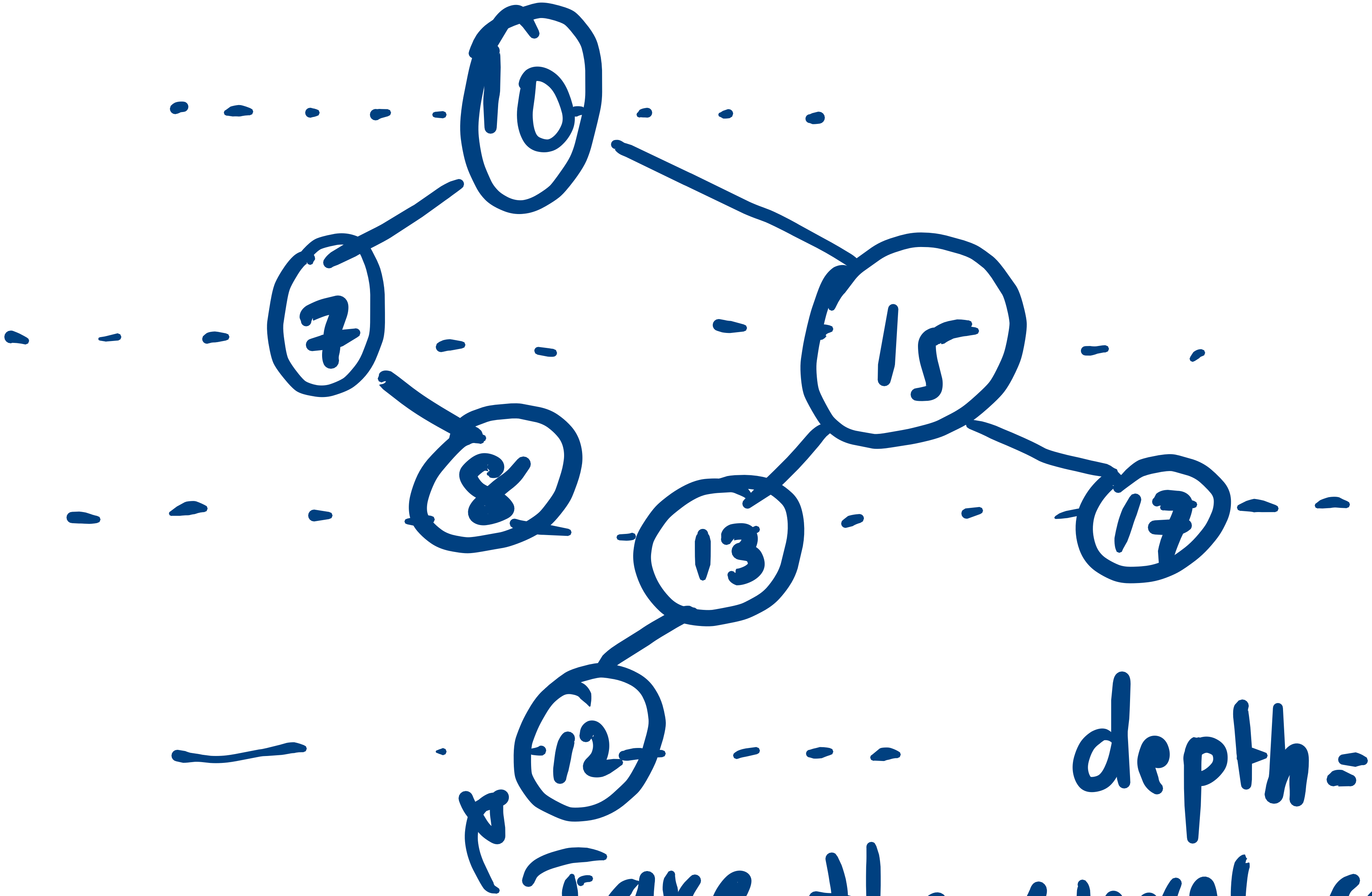
Searching for 5 takes ~ 3 units of time.

Also a BST
Searching for 5
takes 5 units
of time.



depth=5

depth=3



depth = 4

Take the worst case

Key obs.

Any search in a BST takes
$\sim$ depth units of time.

Why?. The search reduces the depth of the BST it is searching in by 1 at each step.

Running-time for BST search

$$\Theta(d)$$

lower bound. Searching for a key in the lower most level takes $\geq d$ units of time.

A - algorithm $\quad$ x - input

$t^*_A(x) :=$ time taken by A on x.

$$t_A(n) := \max_{x : |x| = n} t^*_A(x)$$

$f(n)$ is a lower bound if $t_A(n) \geq f(n)$

$$\max_{x : |x| = n} t^*_A(x) \geq f(n)$$

Prove this by exhibiting an x with $|x|=n$ and

$$t_A^*(x) \geq f(n)$$

$f(n)$ is an upper bound $t_A(n) \leq f(n)$

$$\max_{x:|x|=n} \underbrace{t_A^*(x)}_{\text{Prove an upper bound for all } x.} \leq f(n)$$

$$\max\{\cdot,\cdot,\cdot,\cdot,101,\cdot,\cdot,\cdot\} \geq 101 \checkmark$$

$$\max\{\cdot,\overset{102}{\cdot},\cdot,\cdot,101,\cdot,\cdot,\cdot\} \leq 101$$

$\times$