Do Now Exercise

To prepare you for the lecture today, please do the following exercise.

Write the asymptotic worst-case running time of

bool contain(TYPE item);

method of Array and of LinkedList class.

COMP15: Data Structures

Week 7, Summer 2019

Admin

T6: **redirect (>, >>, <), pipe (|)**

(Optional) combination with: **echo, sort, uniq, wc**

Due by 6pm on Wednesday, July 10

(a quick demo)

(Renamed and Updated the due dates)

P4: **Course Registration System**

Project Due by 6pm on Sunday, July 21

Midterm Fun

(Let's aim to start it at 7:30 pm.)

Questions?

Sorting (cont.)

(Slide from Week 6)

Counting sort

(Slide from Week 6)

bounded-universe (fixed-)

(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

```
3void countingSort(int* const unsorted, int* const sorted, int n, int k){
4  int* histogram = new int[k + 1];
5  for(int i = 0; i < k + 1; i++){
6    histogram[i] = 0;
7  }
8
9  for(int i = 0; i < n; i++){
10   int number = unsorted[i];
11   histogram[number] += 1;
12 }
13
14 for(int i = 1; i < k + 1; i++){
15   histogram[i] = histogram[i] + histogram[i - 1];
16 }
17
18 for(int i = n - 1; i >= 0; i--){
19   int number = unsorted[i];
20   int index = histogram[number] - 1;
21   sorted[index] = number;
22   histogram[number] -= 1;
23 }
24
25 delete [] histogram;
26}
```

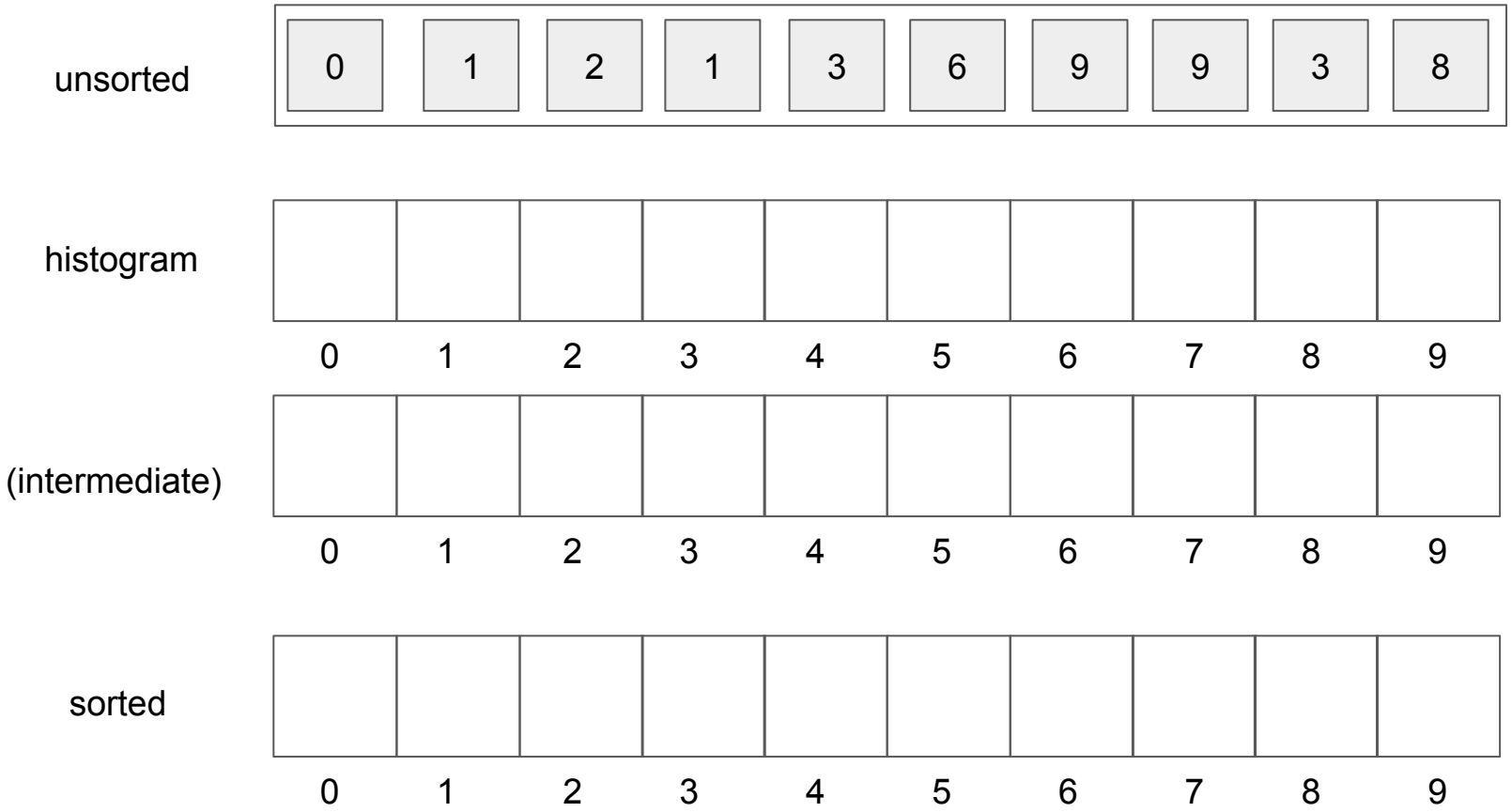
(Please let the instructor know if you find any errors in the code.)

The goal is to fill out the array pointed by "sorted".

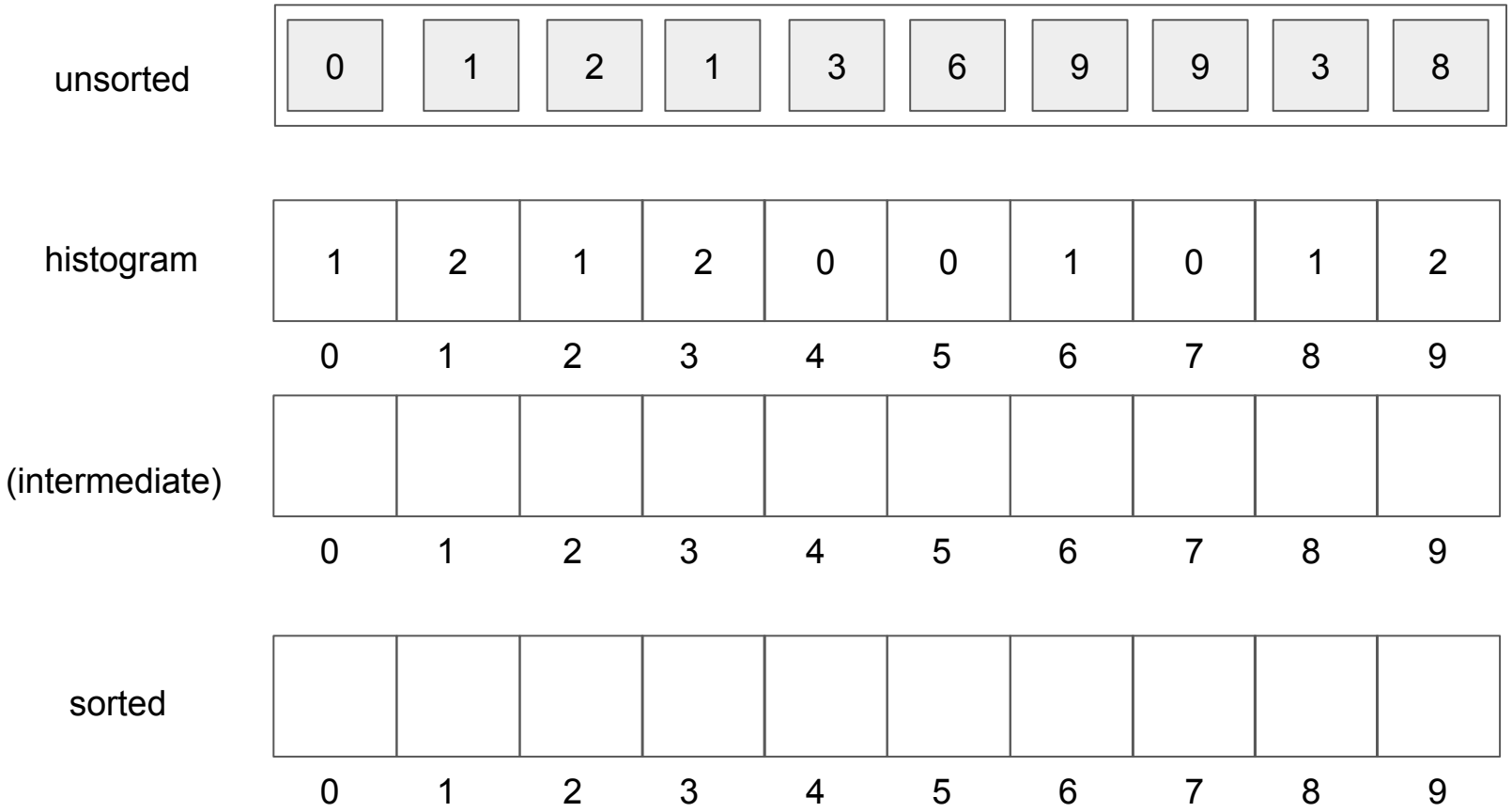
"n" is the size of the array pointed by "unsorted" and of the one pointed by "sorted".

The array pointed by "unsorted" contains integers between 0 and "k".

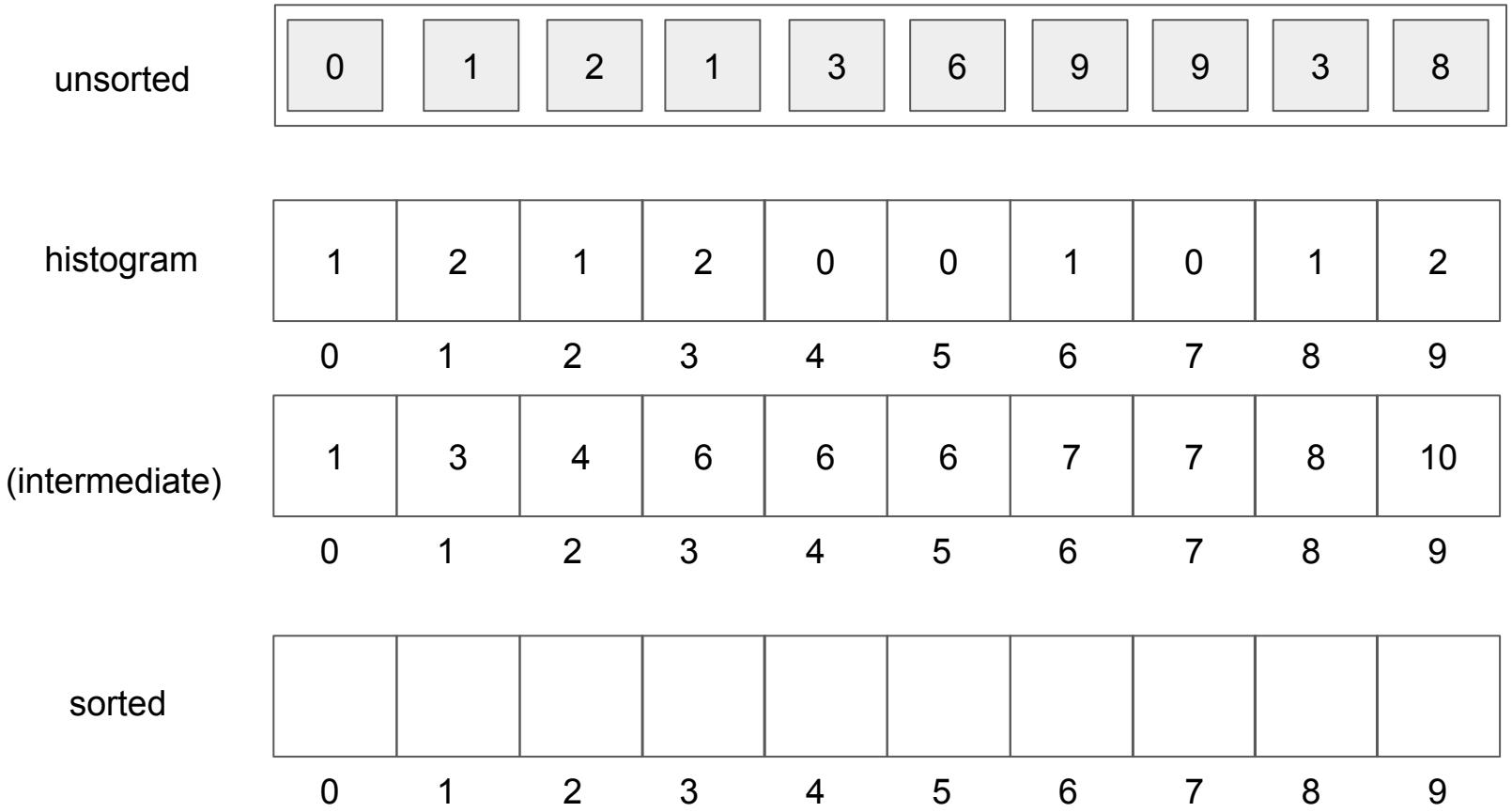
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)



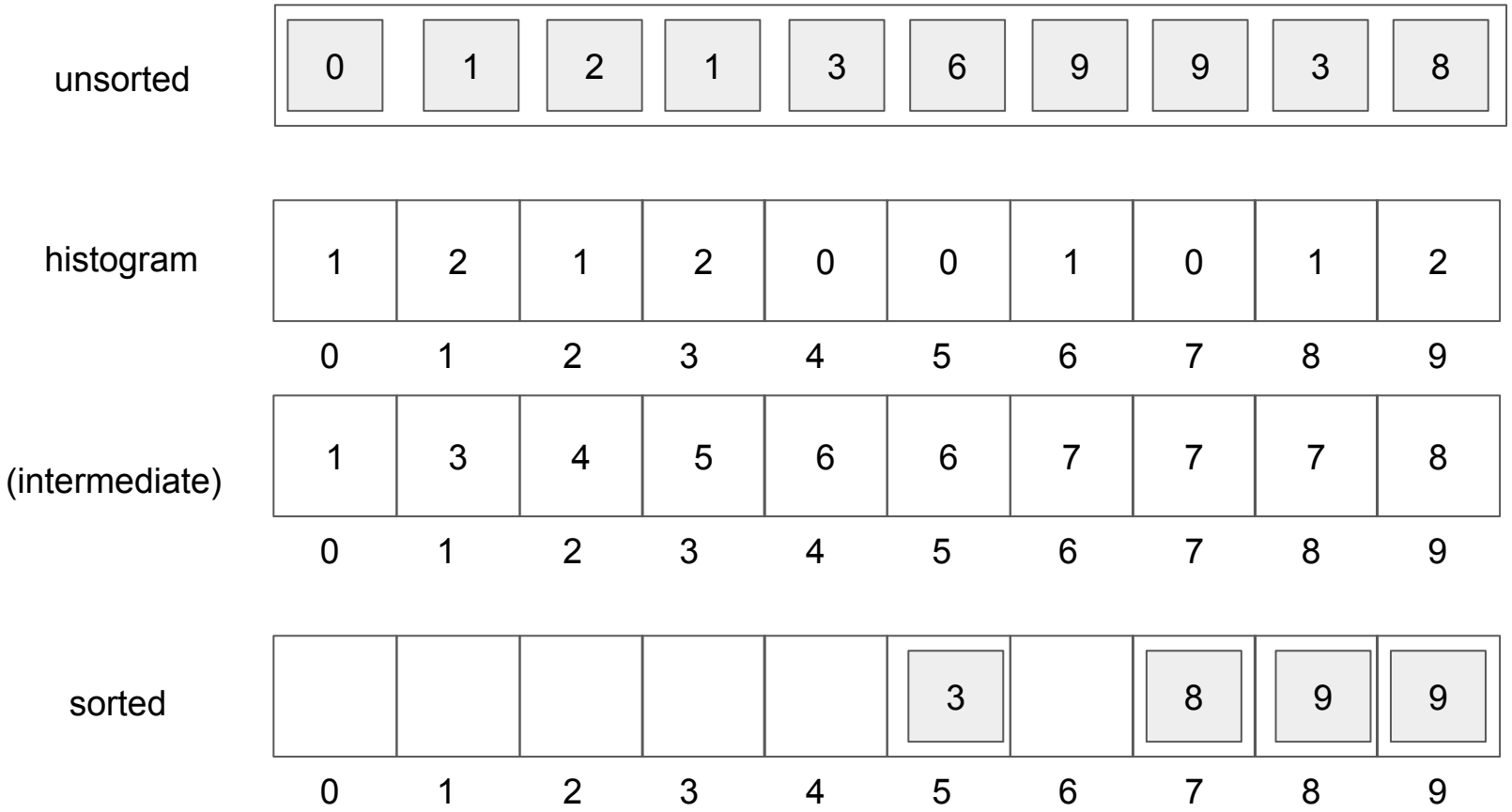
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)



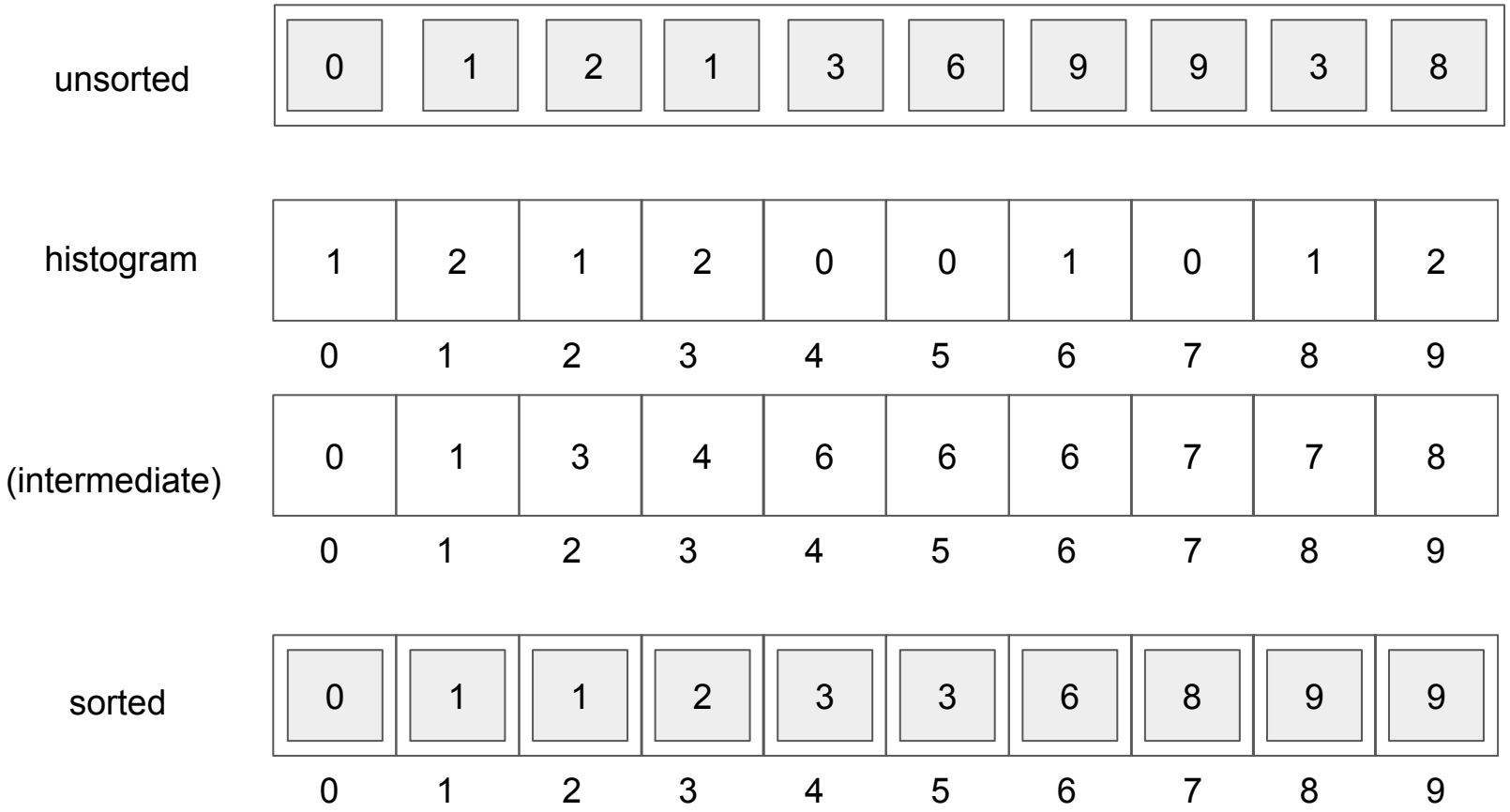
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)



(Notes from the live demo or live coding. Please do NOT assume the code is complete.)



(Notes from the live demo or live coding. Please do NOT assume the code is complete.)



Counting sort

Worst-case: $O(k + n)$

(Note: We also discussed two cases: where k < n and where k > n.)

Questions?

Trees (cont.)

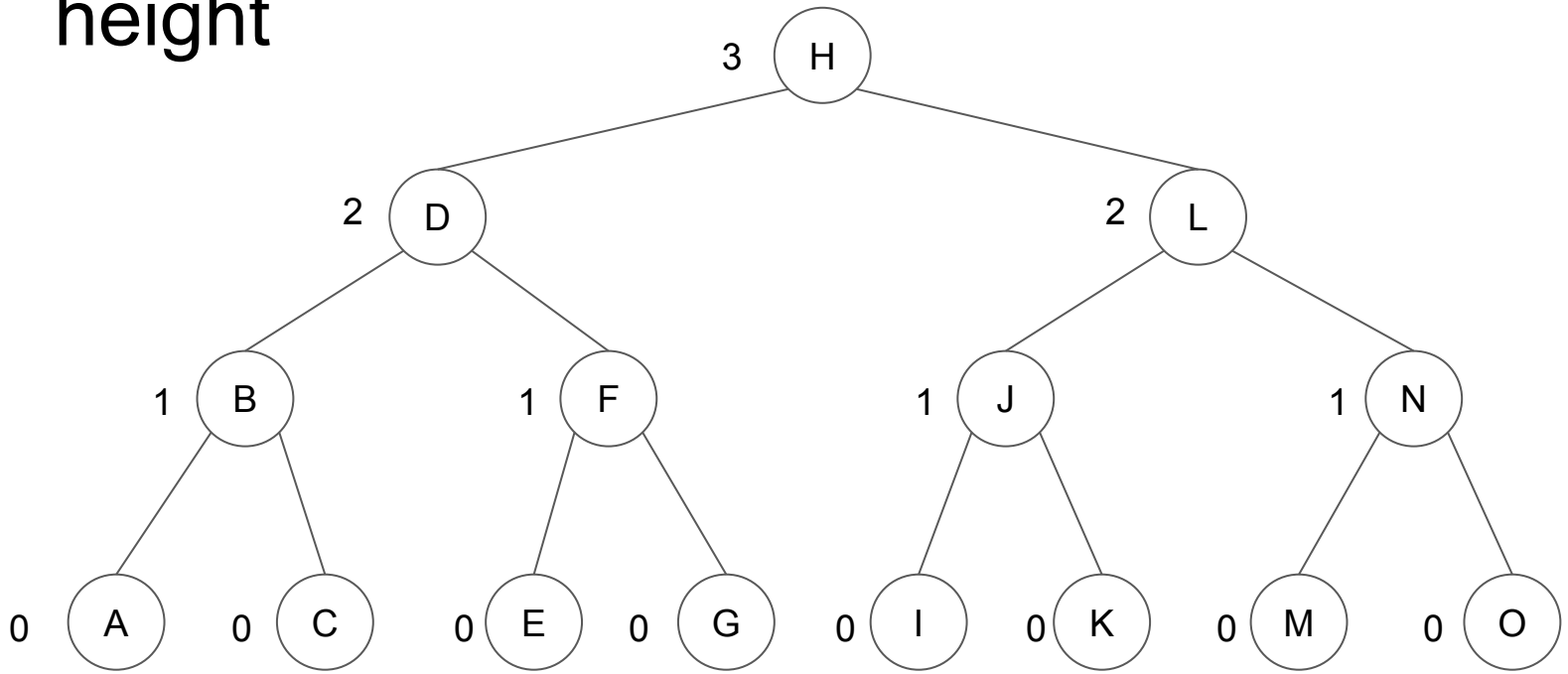
Tree:

- Terminologies (Week 6)
- Traversals (Week 6)

- Operations (Week 7)
- Rotations (Week 8)

Remaining part: Height

binary tree height



(*** For now, the height of a leaf node is 0. The height of the empty tree is not defined.)

Questions?

Binary Tree

Do Now Exercise

To prepare you for the lecture today, please do the following exercise.

Write the asymptotic worst-case running time of

bool contain(TYPE item);

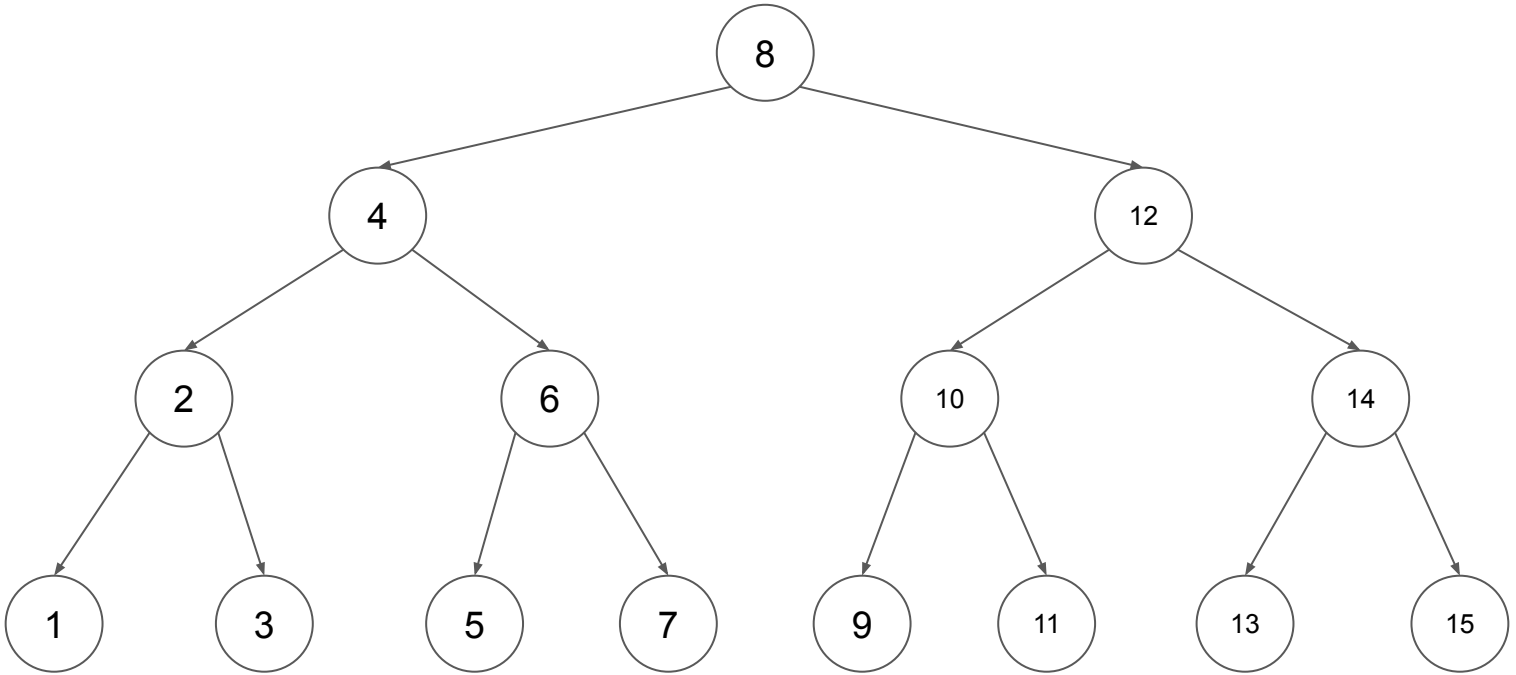
method of Array and of LinkedList class.

Do Now Exercise

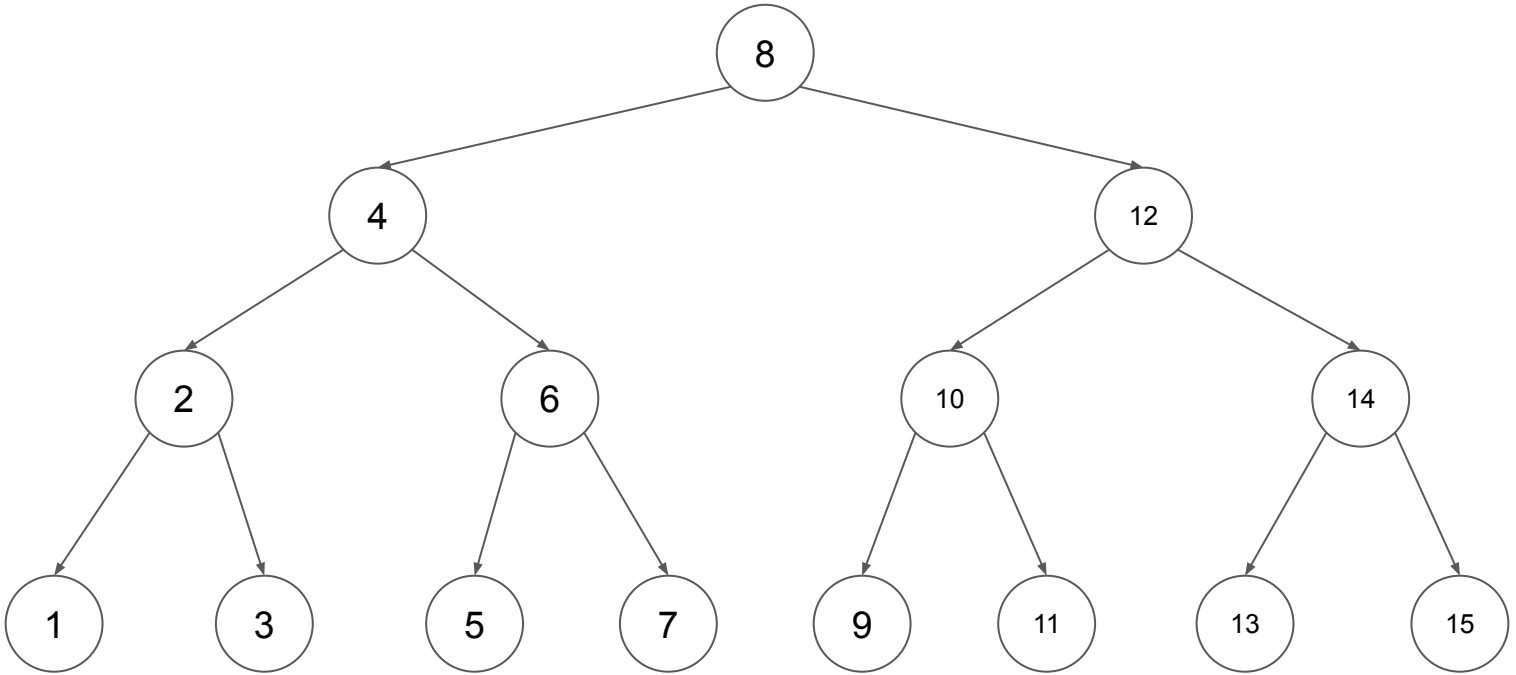
Students' answers:

Binary Search Tree (BST)

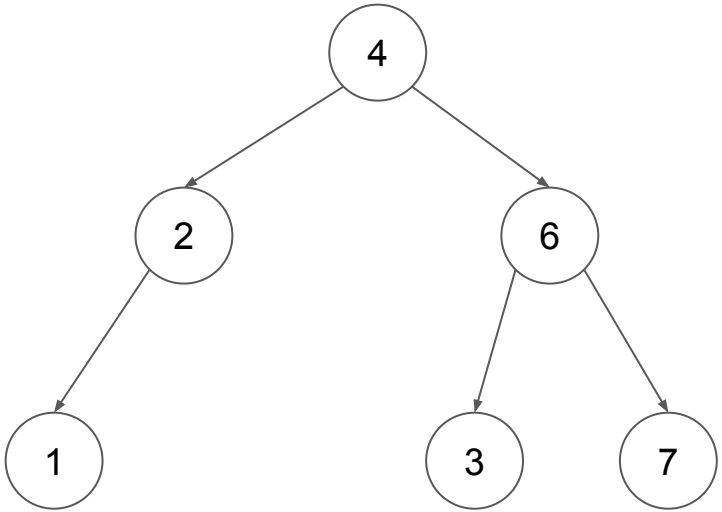
binary search tree



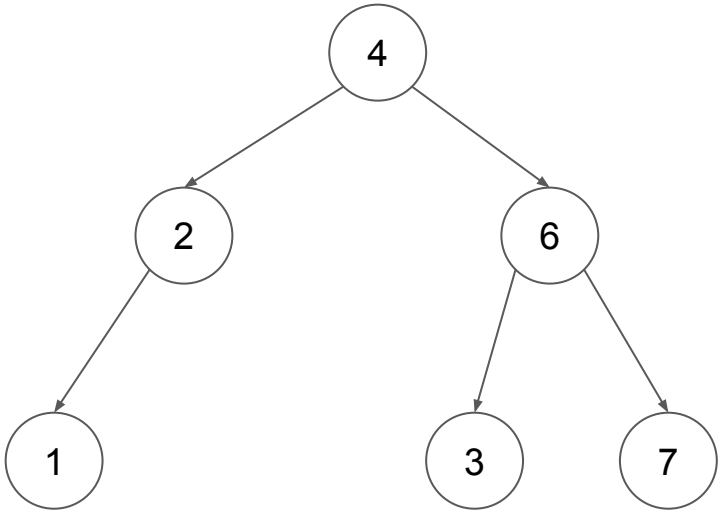
Is this a binary search tree?



Is this a binary tree?



Is this a binary search tree?



Templates

(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

```hpp
//BSTNode.hpp
#ifndef BSTNODE_HPP
#define BSTNODE_HPP

template<typename T>
class BSTNode{
public:
  //BSTNode();
  BSTNode(T data);
  //copy constructor
  //assignment operator
  //~BSTNode();

  T getData() const;
  void setLeft(BSTNode<T>* left);
  BSTNode<T>* getLeft() const;
  void setRight(BSTNode<T>* right);
  BSTNode<T>* getRight() const;

private:
  T data;
  BSTNode<T>* left;
  BSTNode<T>* right;
};

#endif
```

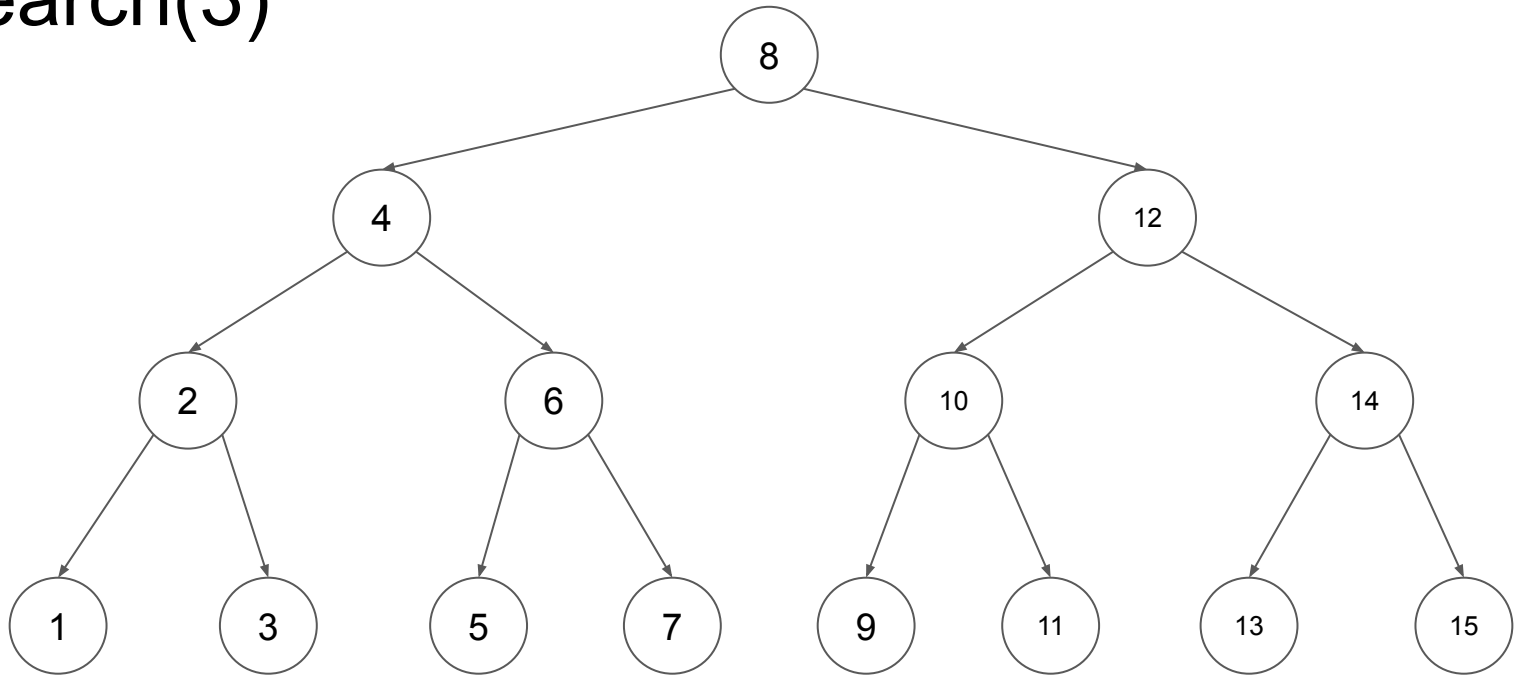
```cpp
//BSTNode.cpp
#include "BSTNode.hpp"

template<typename T>
BSTNode<T>::BSTNode(T data){
  this->data = data;
  this->left = nullptr;
  this->right = nullptr;
}

template<typename T>
T BSTNode<T>::getData() const{
  return this->data;
}

template<typename T>
void BSTNode<T>::setLeft(BSTNode<T>* left){
  this->left = left;
}

template<typename T>
BSTNode<T>* BSTNode<T>::getLeft() const{
  return this->left;
}

template<typename T>
void BSTNode<T>::setRight(BSTNode<T>* right){
  this->right = right;
}

template<typename T>
BSTNode<T>* BSTNode<T>::getRight() const{
  return this->right;
}

template class BSTNode<int>;
template class BSTNode<char*>;
```

(Notes from the live demo or live coding. Please do NOT assume the code is complete.)

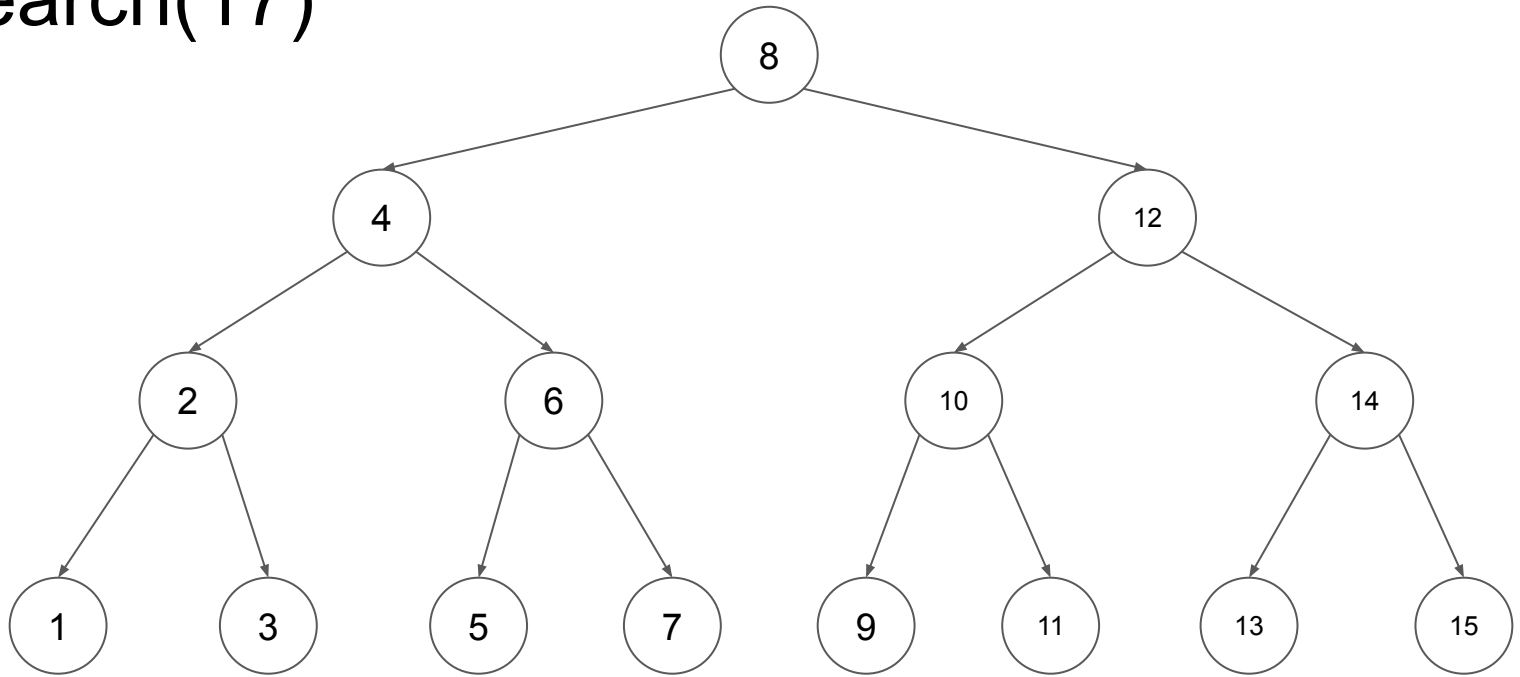
```cpp
//test.cpp
#include "BSTNode.hpp"

int main(){
  BSTNode<int>* n1 = new BSTNode<int>(1);
  BSTNode<int>* n2 = new BSTNode<int>(2);
  BSTNode<int>* n3 = new BSTNode<int>(3);

  n2->setLeft(n1);
  n2->setRight(n3);

  char* a = new char('a');
  char* b = new char('b');
  char* c = new char('c');

  BSTNode<char*>* na = new BSTNode<char*>(a);
  BSTNode<char*>* nb = new BSTNode<char*>(b);
  BSTNode<char*>* nc = new BSTNode<char*>(c);

  nb->setLeft(na);
  nb->setRight(nc);

  delete n1;
  delete n2;
  delete n3;
  delete na;
  delete nb;
  delete nc;
  delete a;
  delete b;
  delete c;

  return 0;
}
```

search(item)

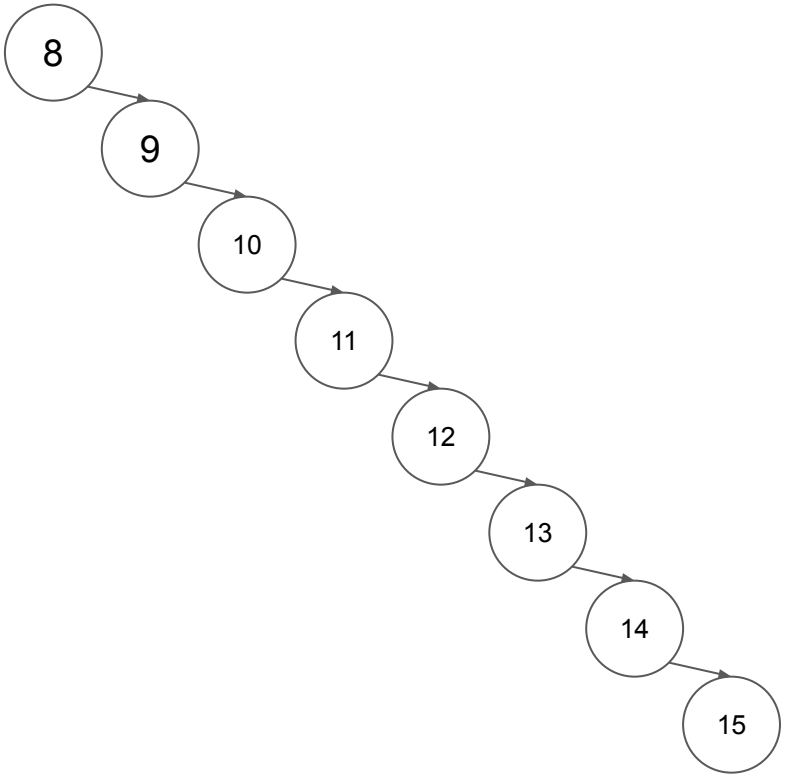
binary search tree search(3)



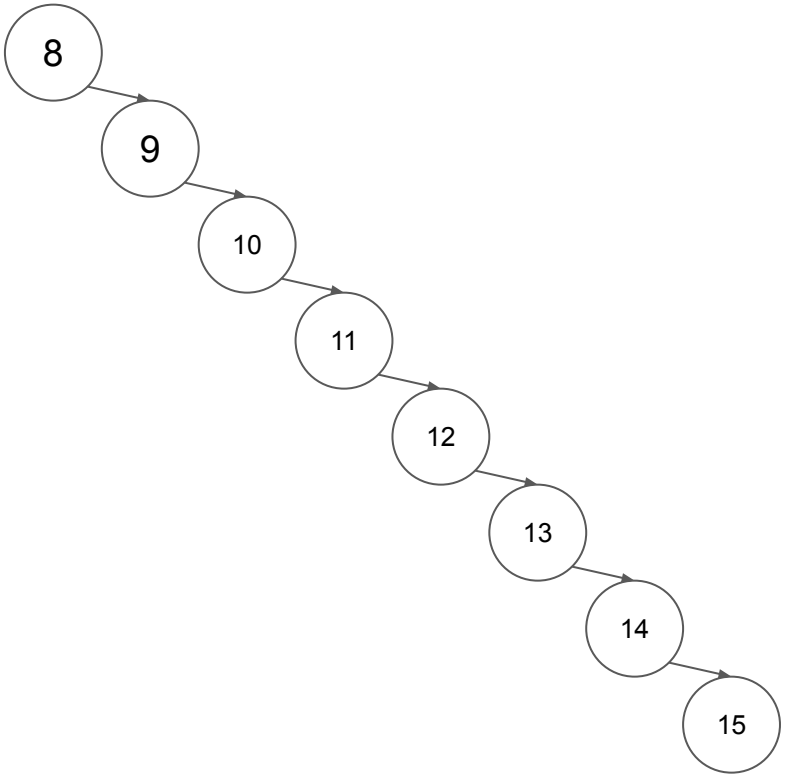
binary search tree
search(17)



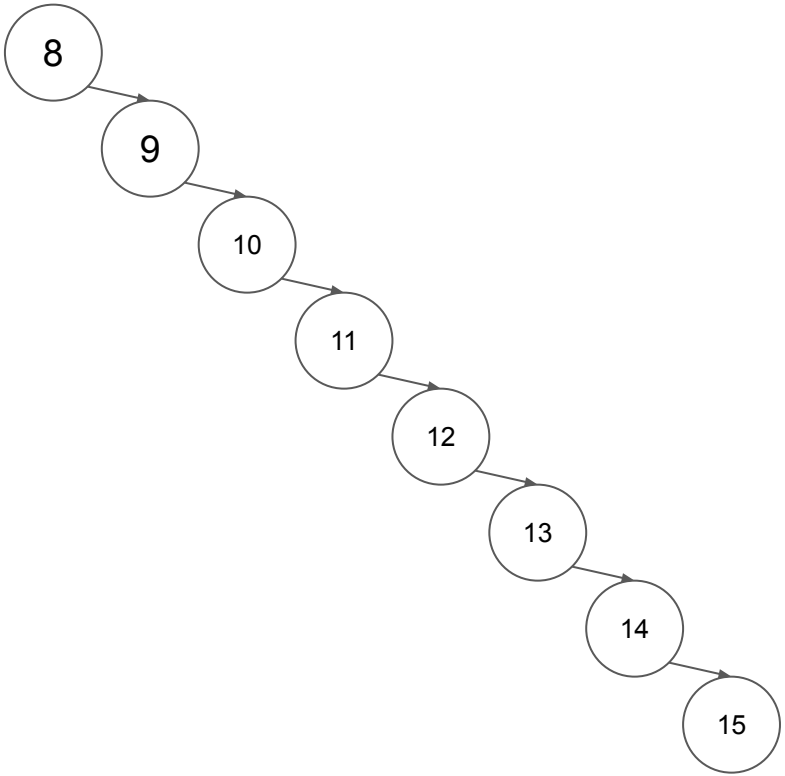
Is this a binary tree?



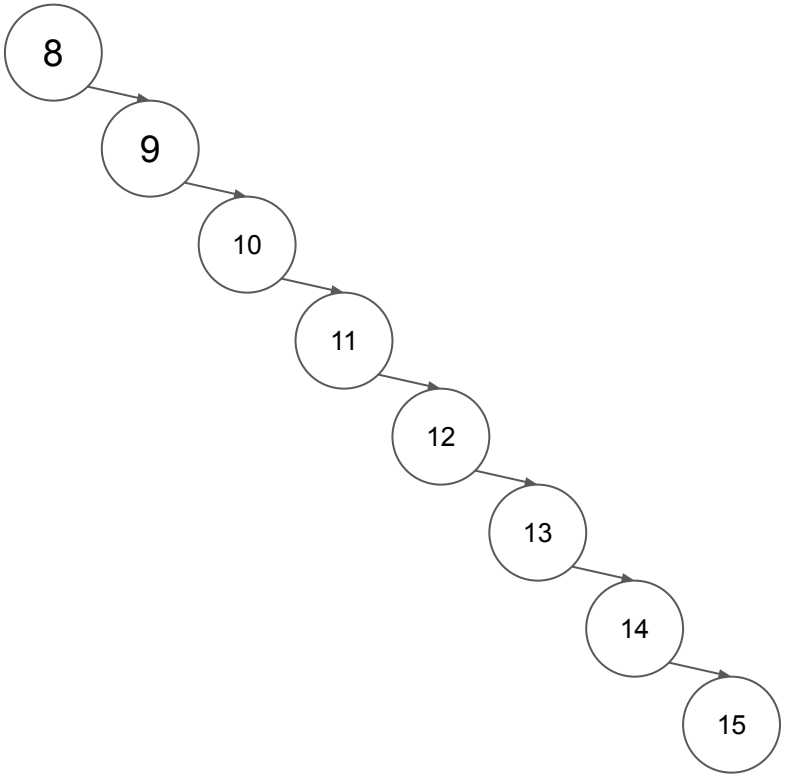
Is this a binary search tree?



Is this binary search tree balanced?



binary search tree
search(15)

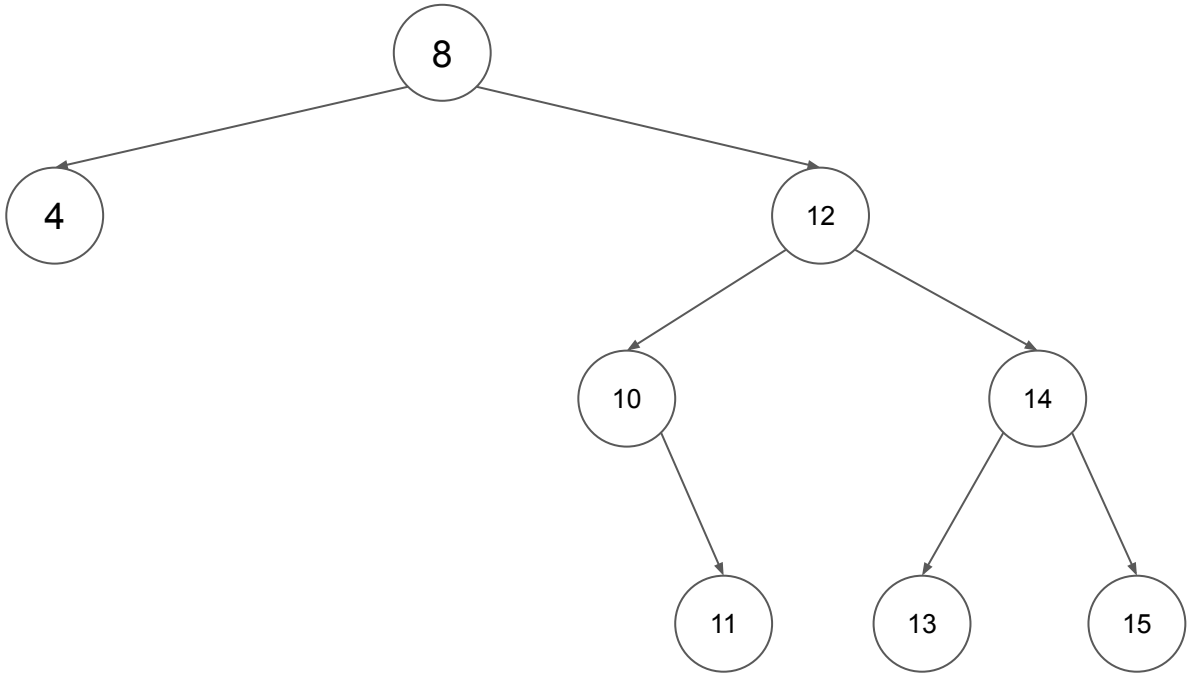


Asymptotic running time of search()?

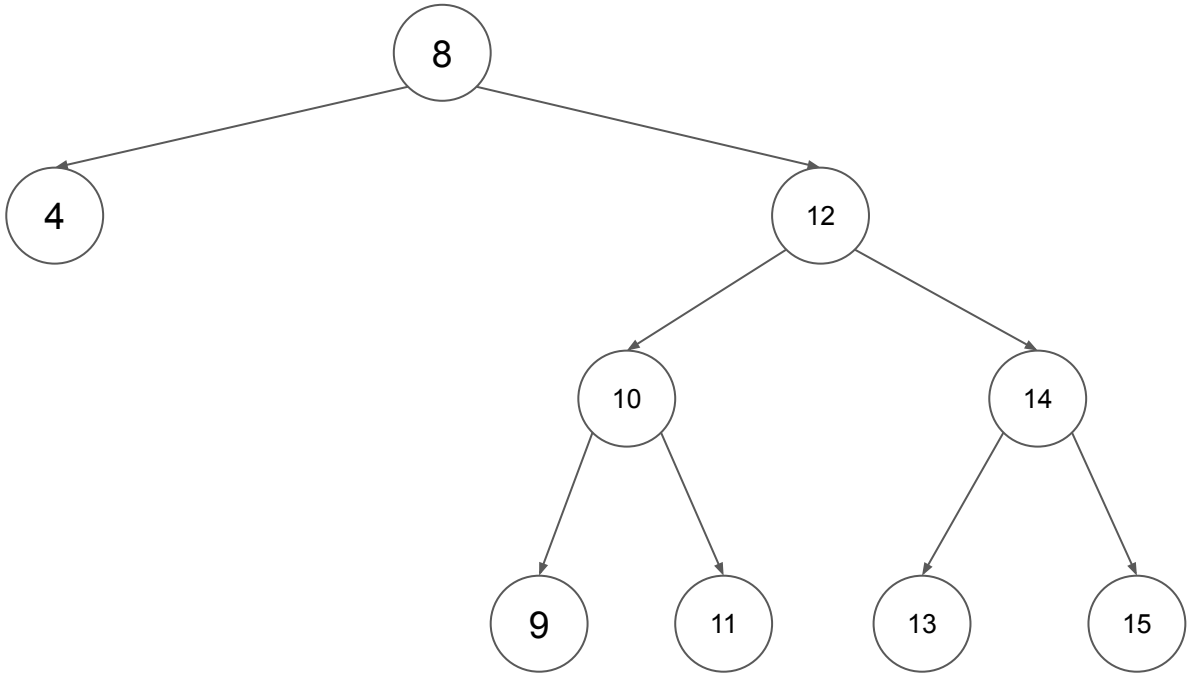
$O(h)$ where h is the height to the tree

insert(item)

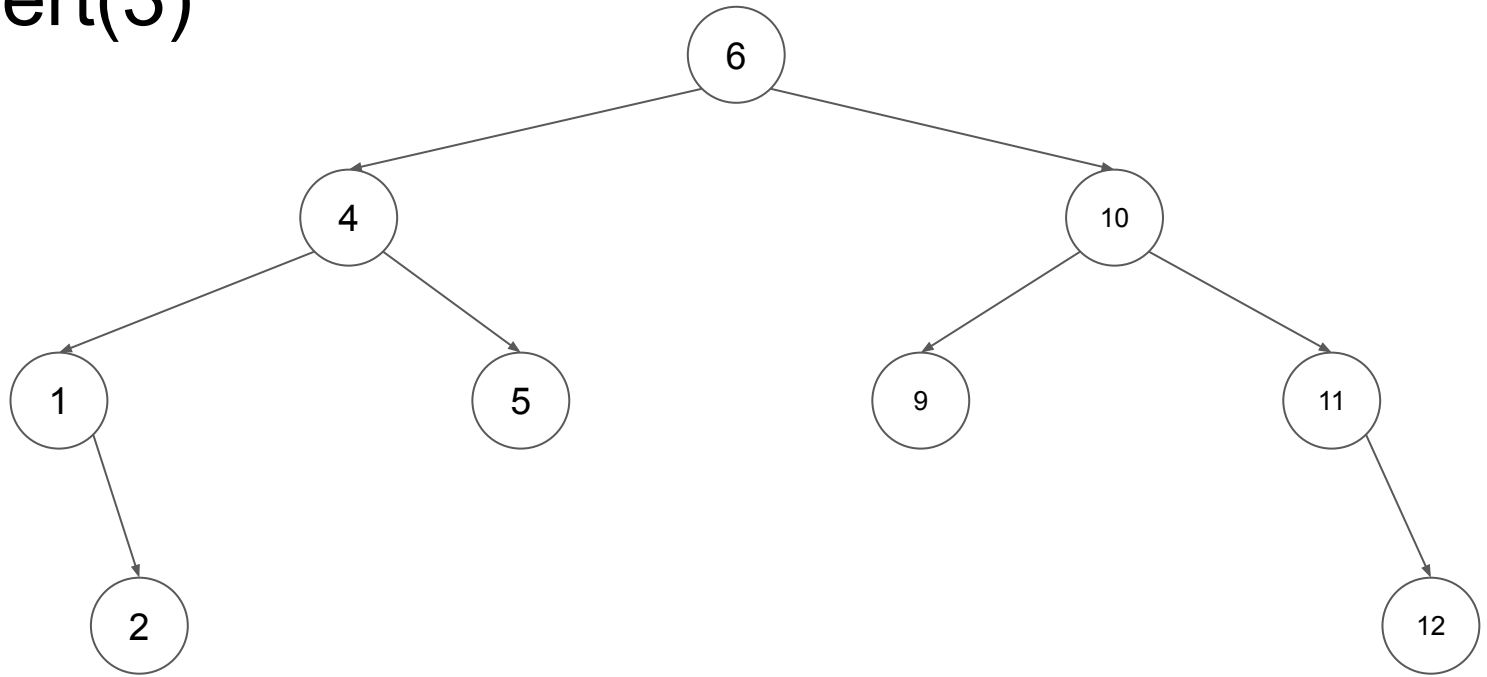
binary search tree
insert(9)



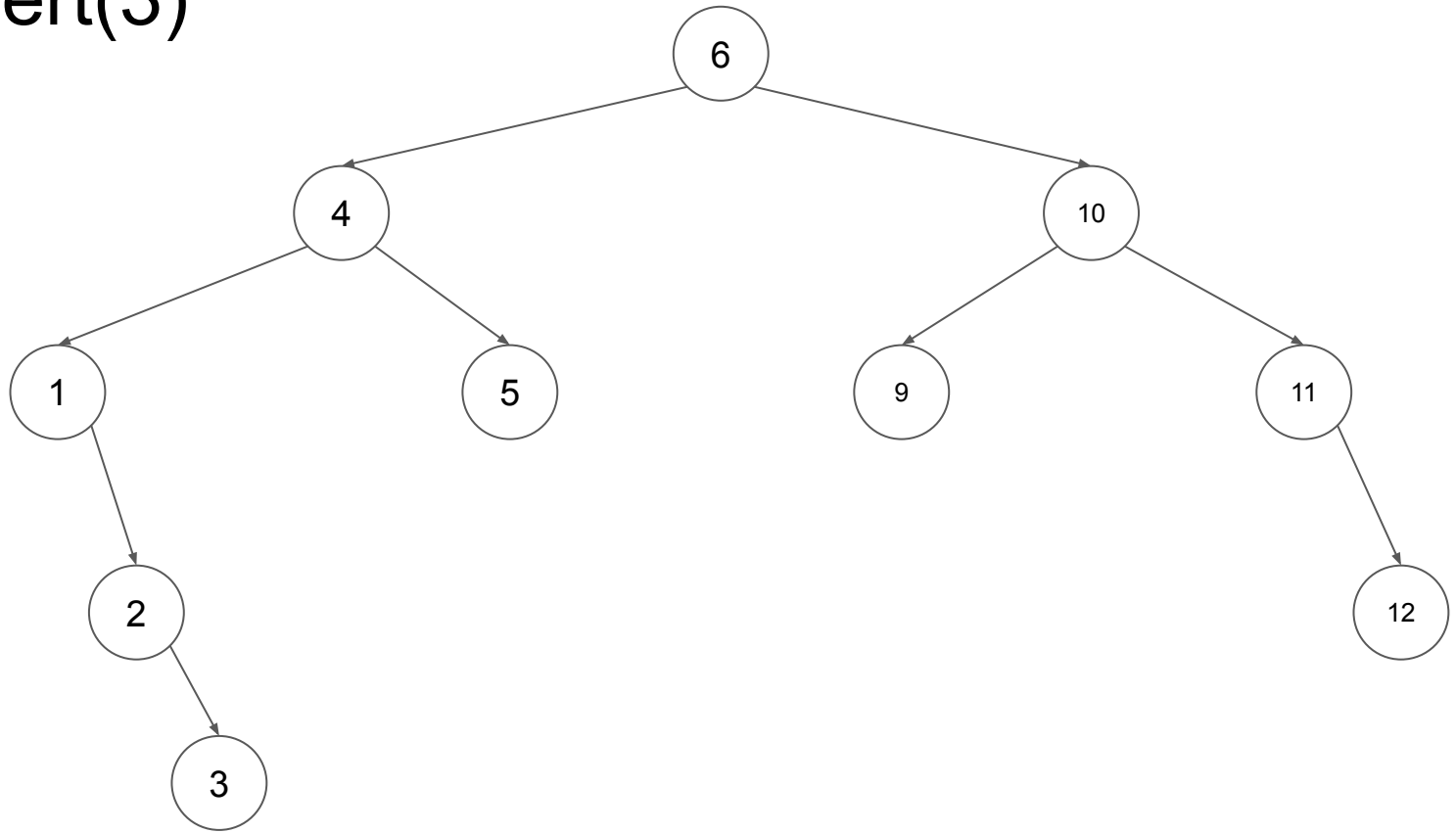
binary search tree
insert(9)



binary search tree
insert(3)



binary search tree
insert(3)

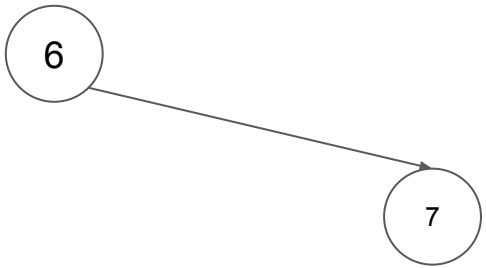


binary search tree
insert(6)

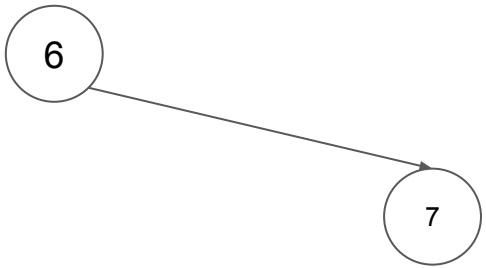
binary search tree
insert(6)

binary search tree
insert(7)

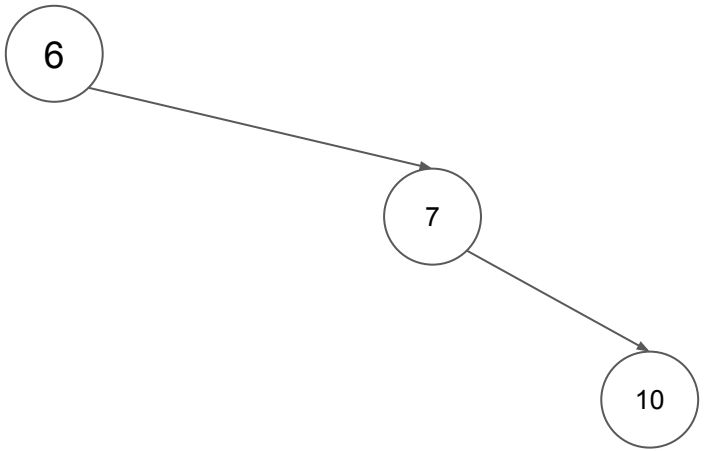
binary search tree
insert(7)



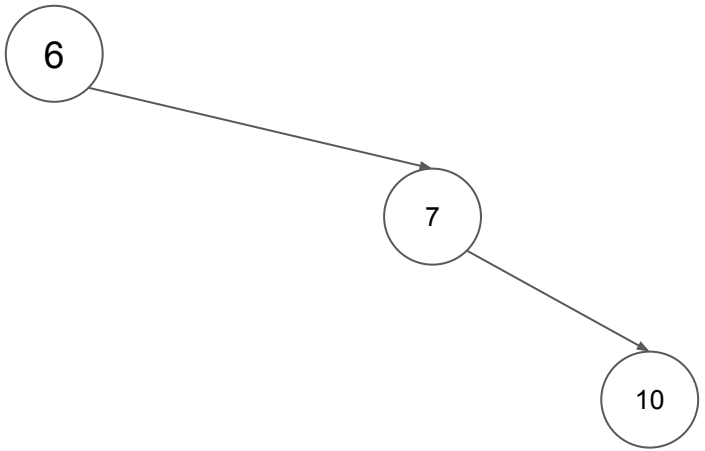
binary search tree
insert(10)



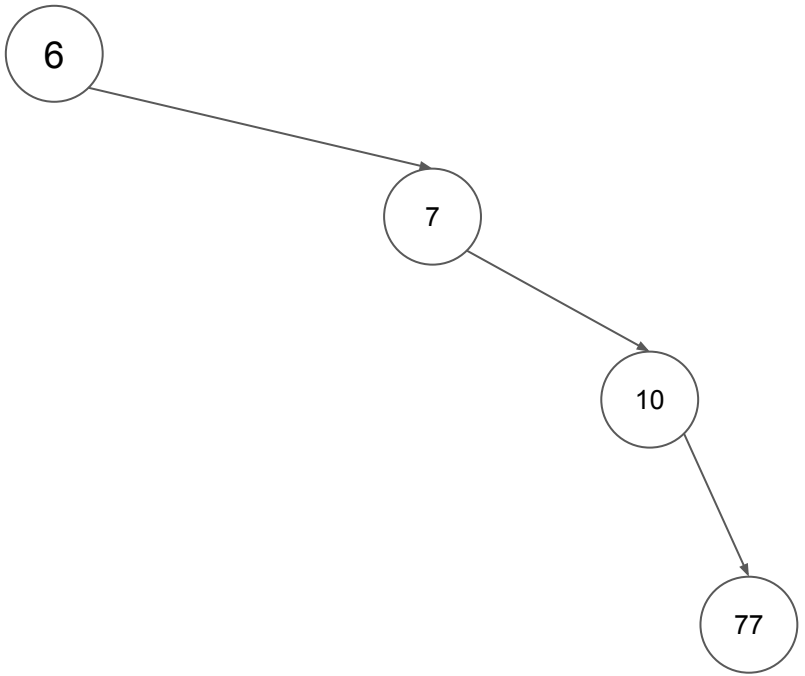
binary search tree
insert(10)



binary search tree
insert(77)



binary search tree
insert(77)

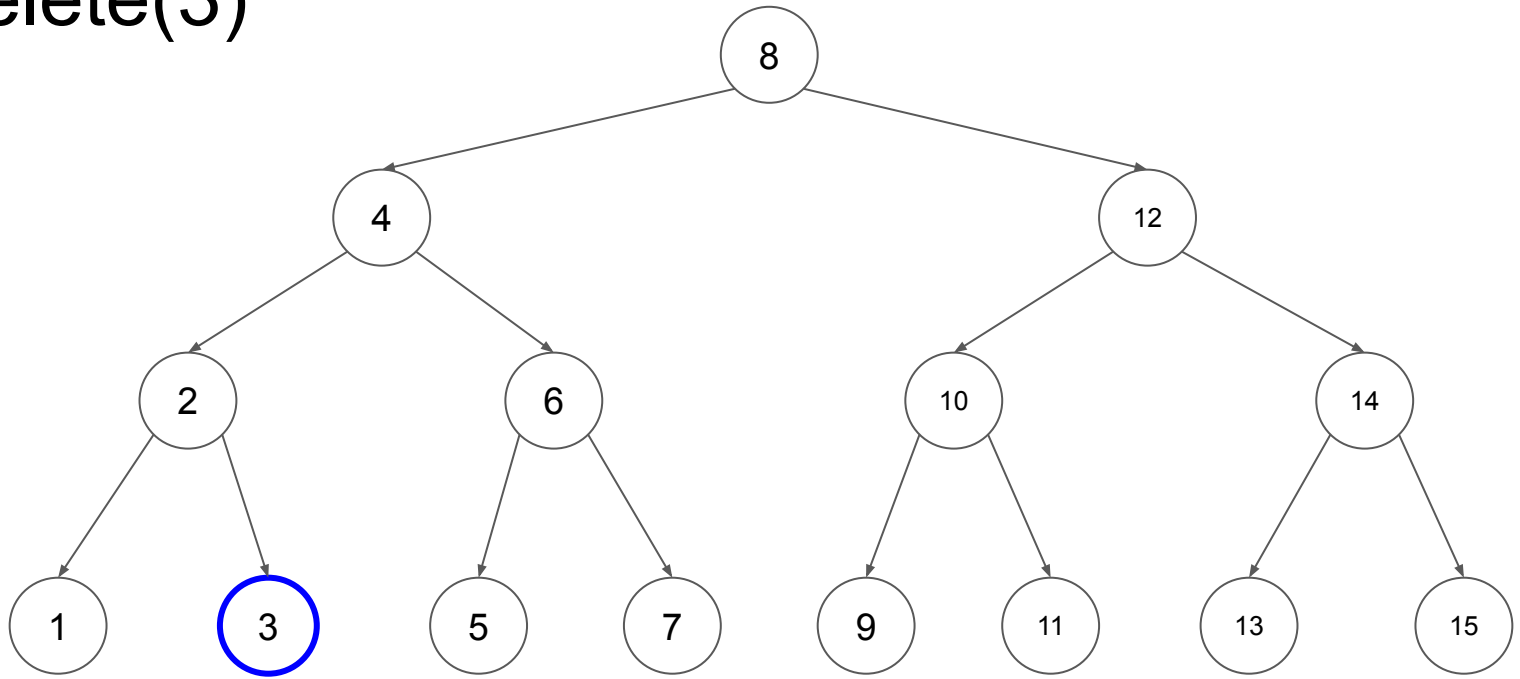


Asymptotic running time of insert()?

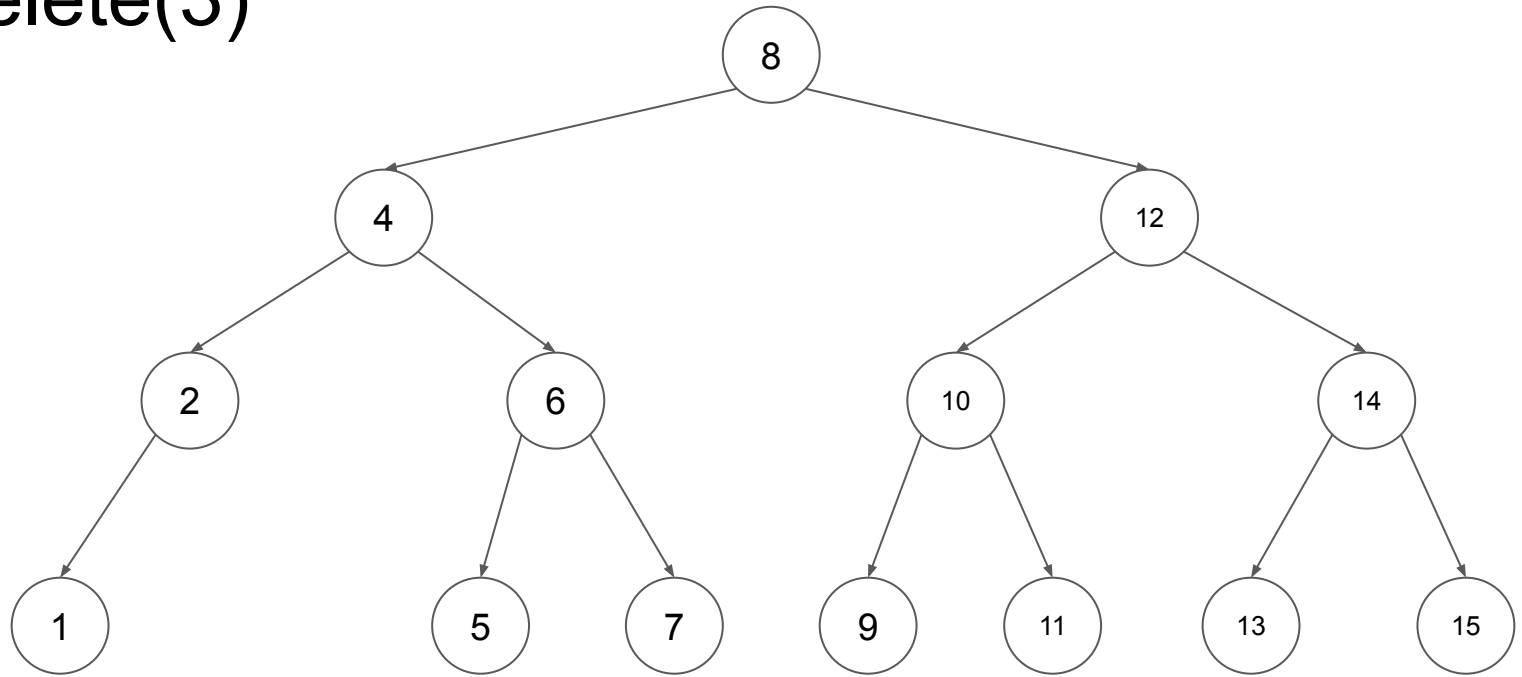
$O(h)$ where h is the height to the tree

delete(item)

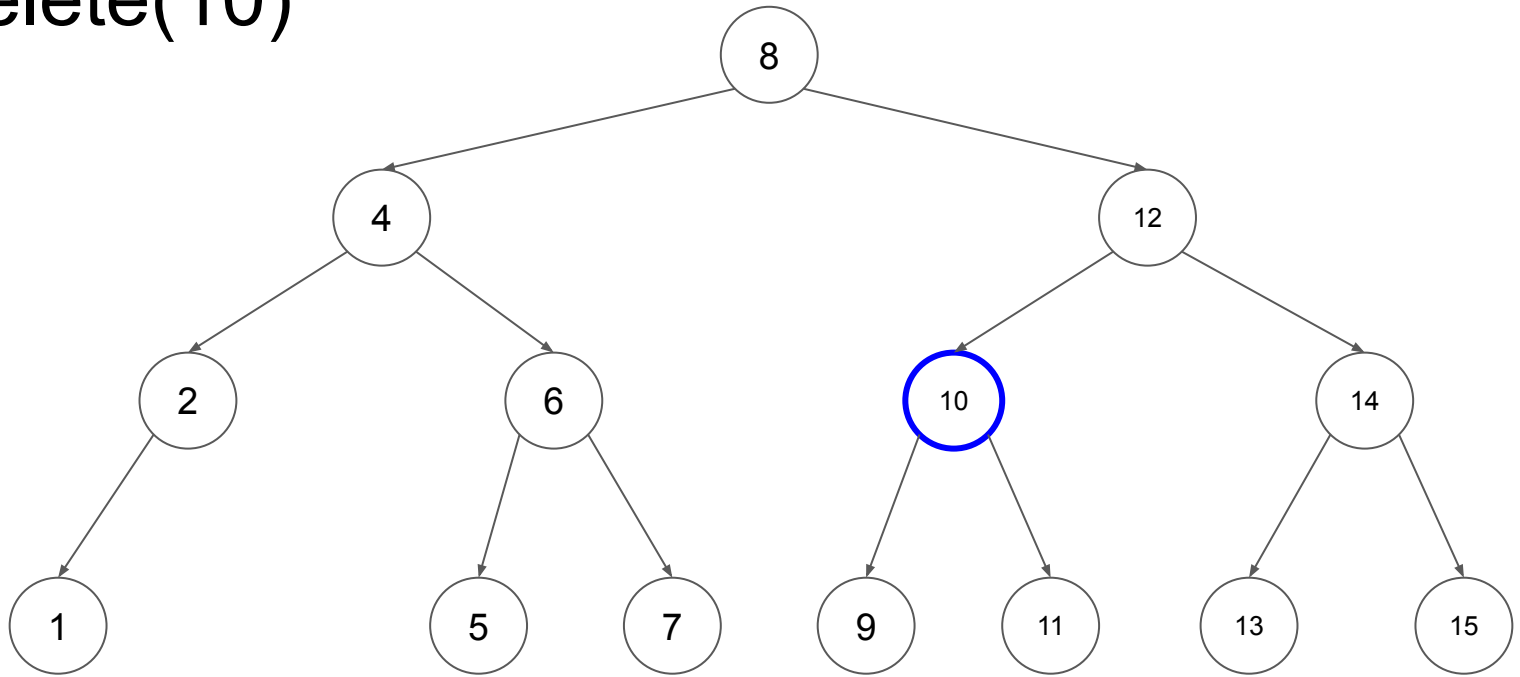
binary search tree
delete(3)



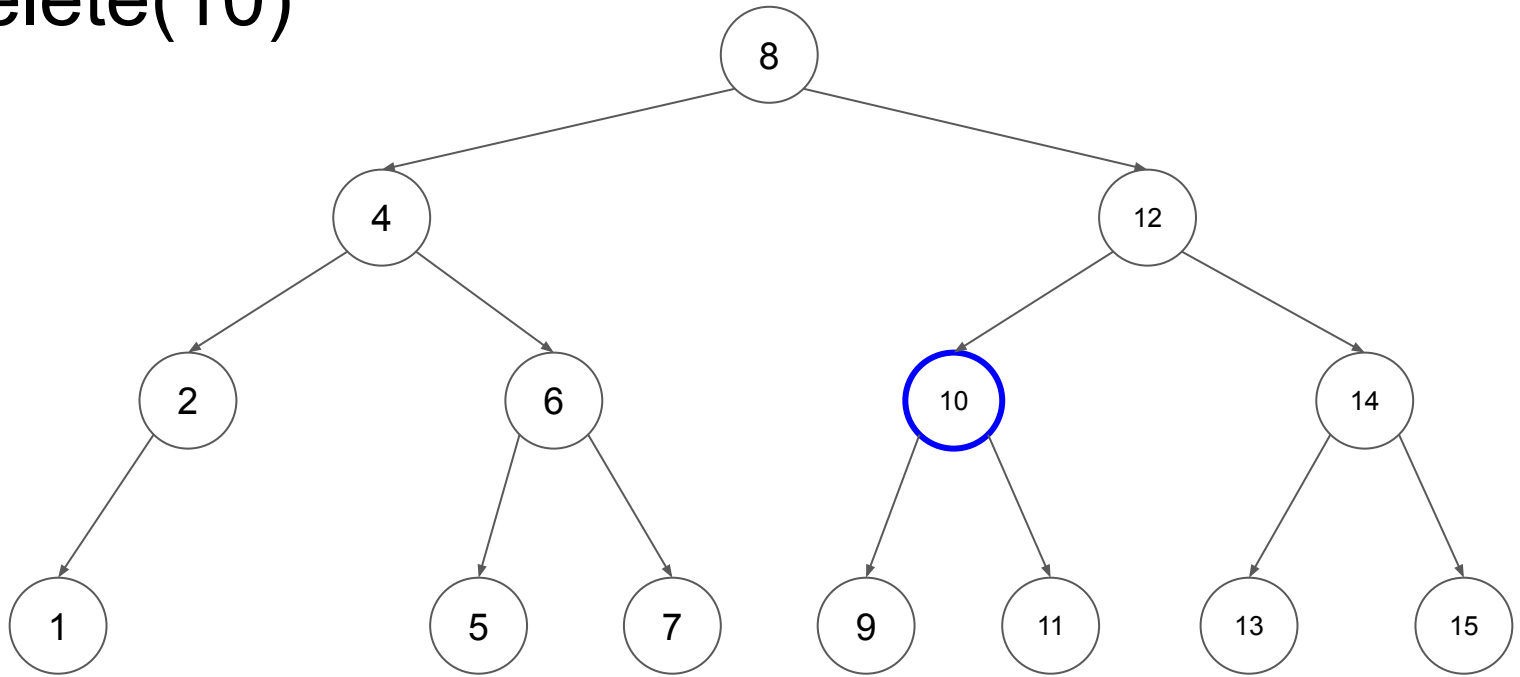
binary search tree
delete(3)



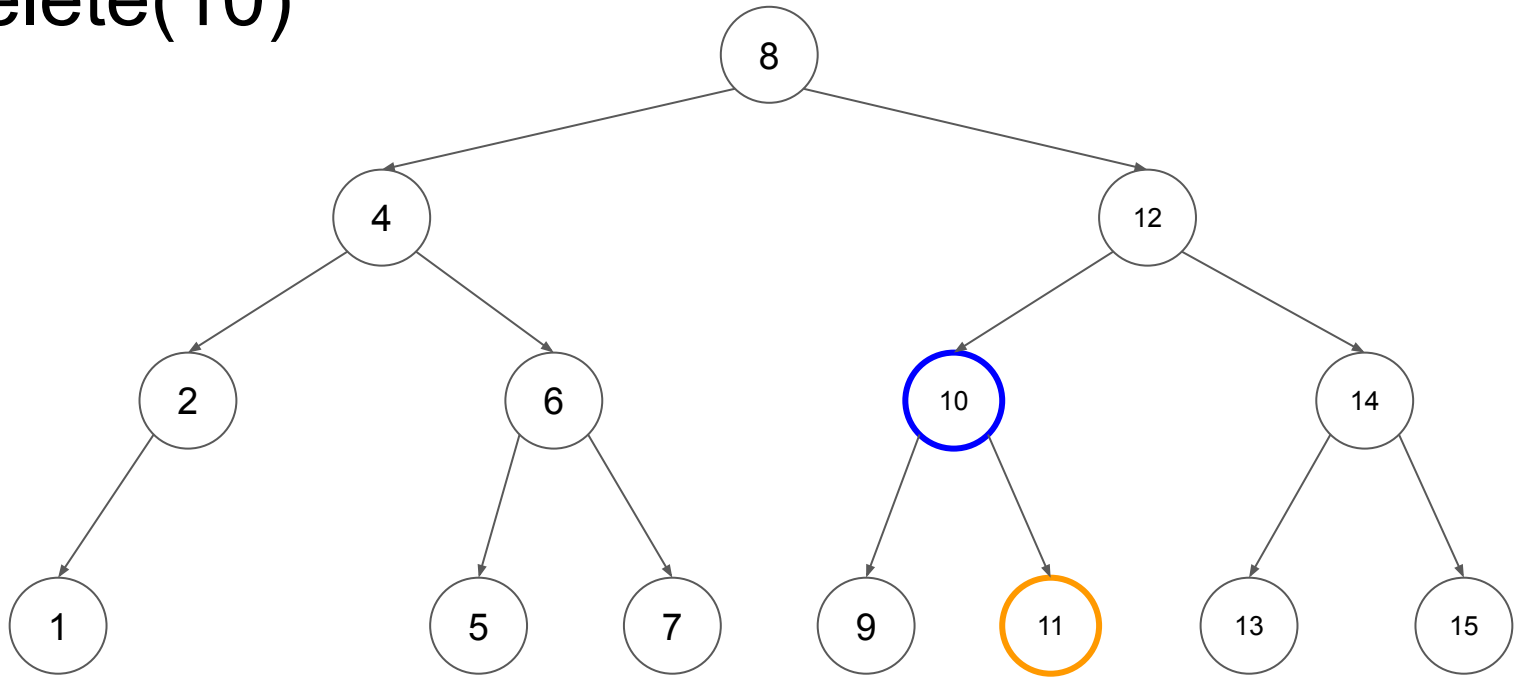
binary search tree
delete(10)



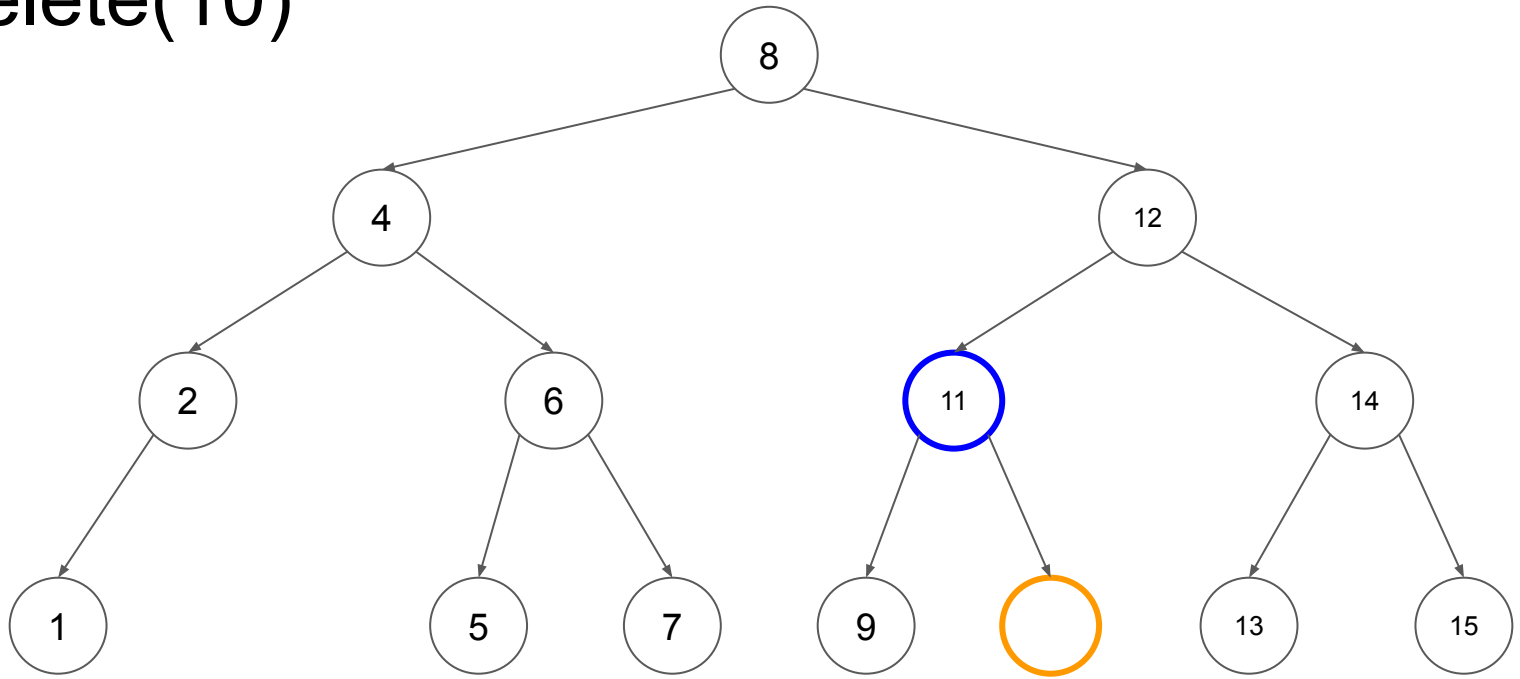
binary search tree
delete(10)



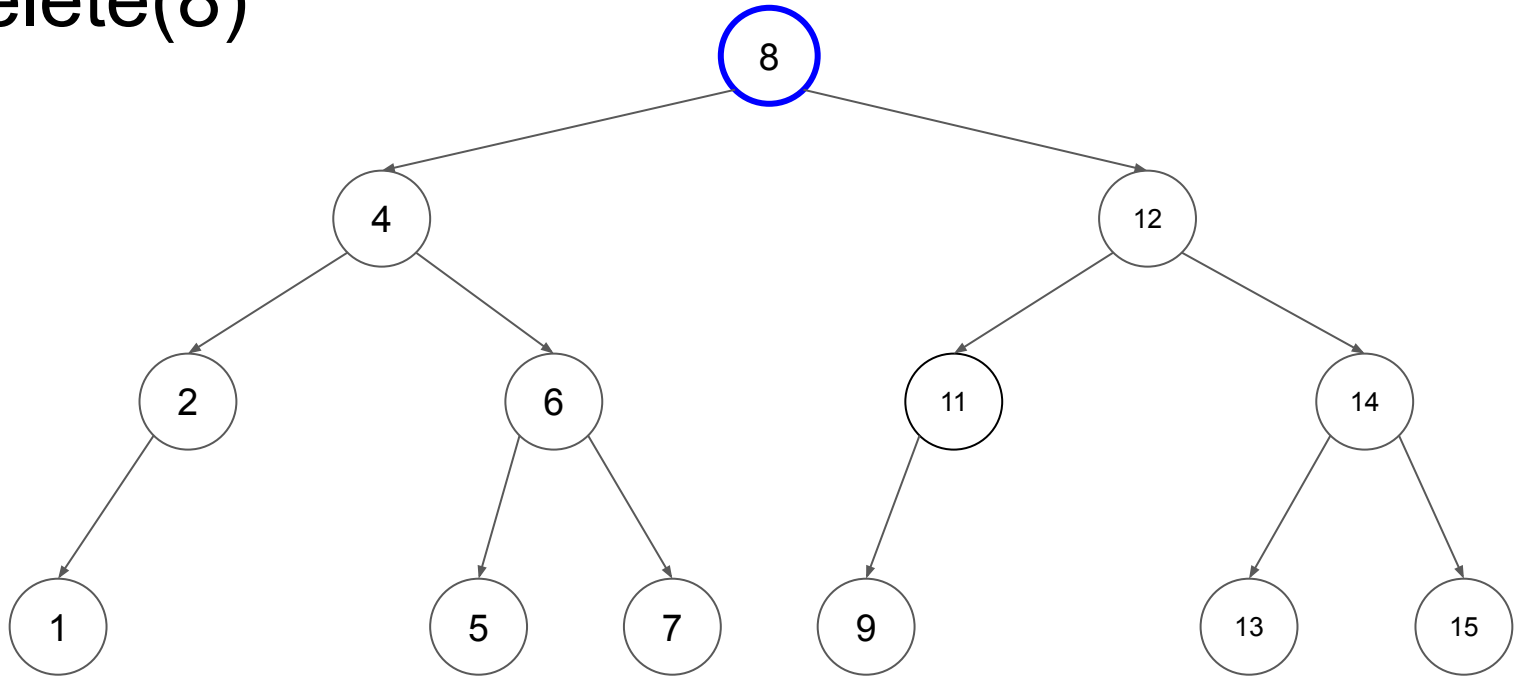
binary search tree
delete(10)



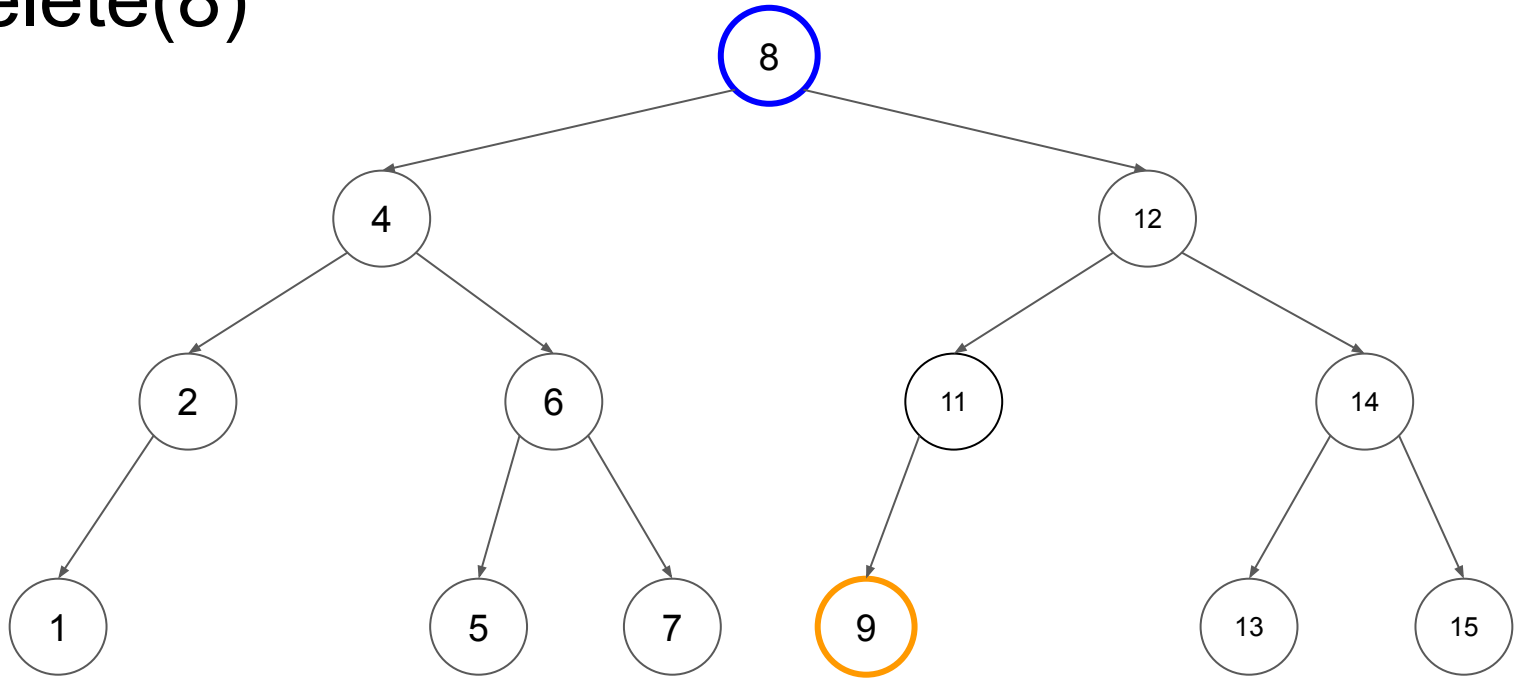
binary search tree
delete(10)



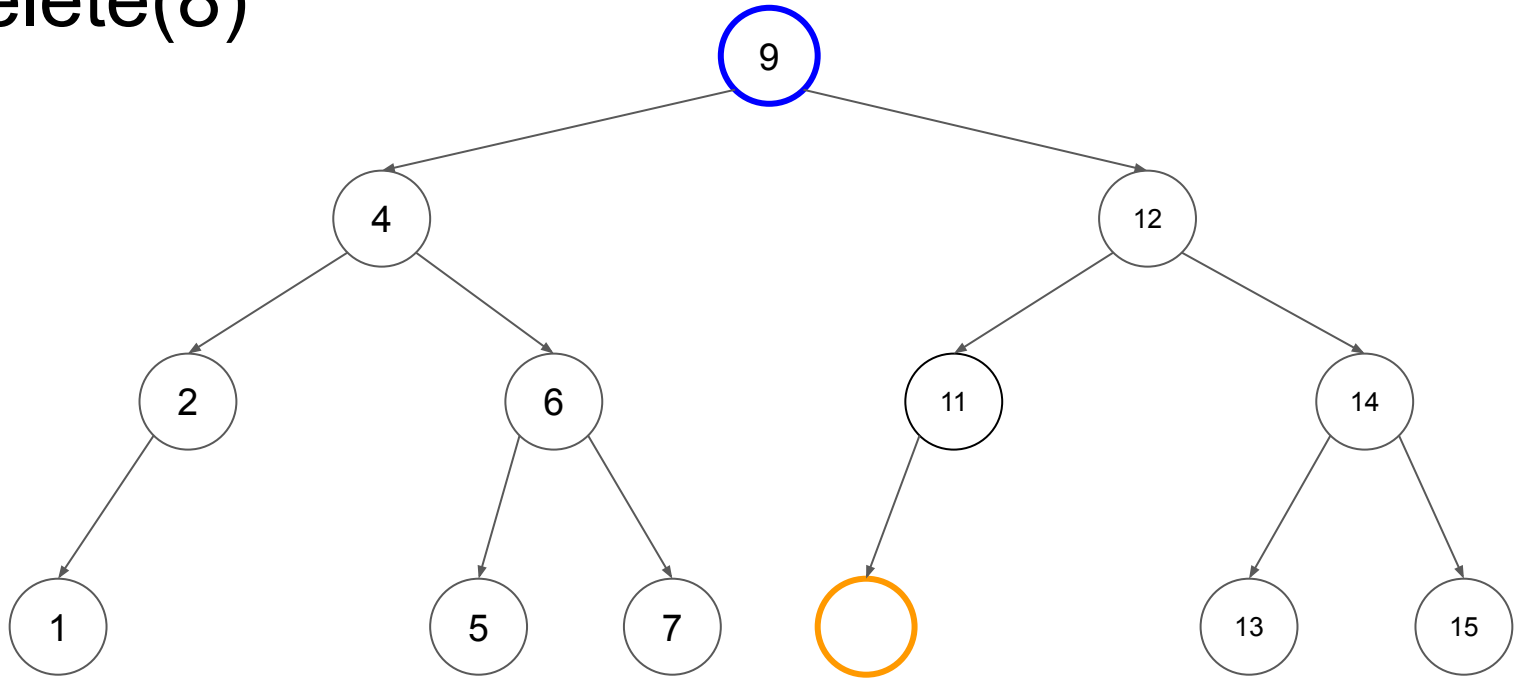
binary search tree
delete(8)



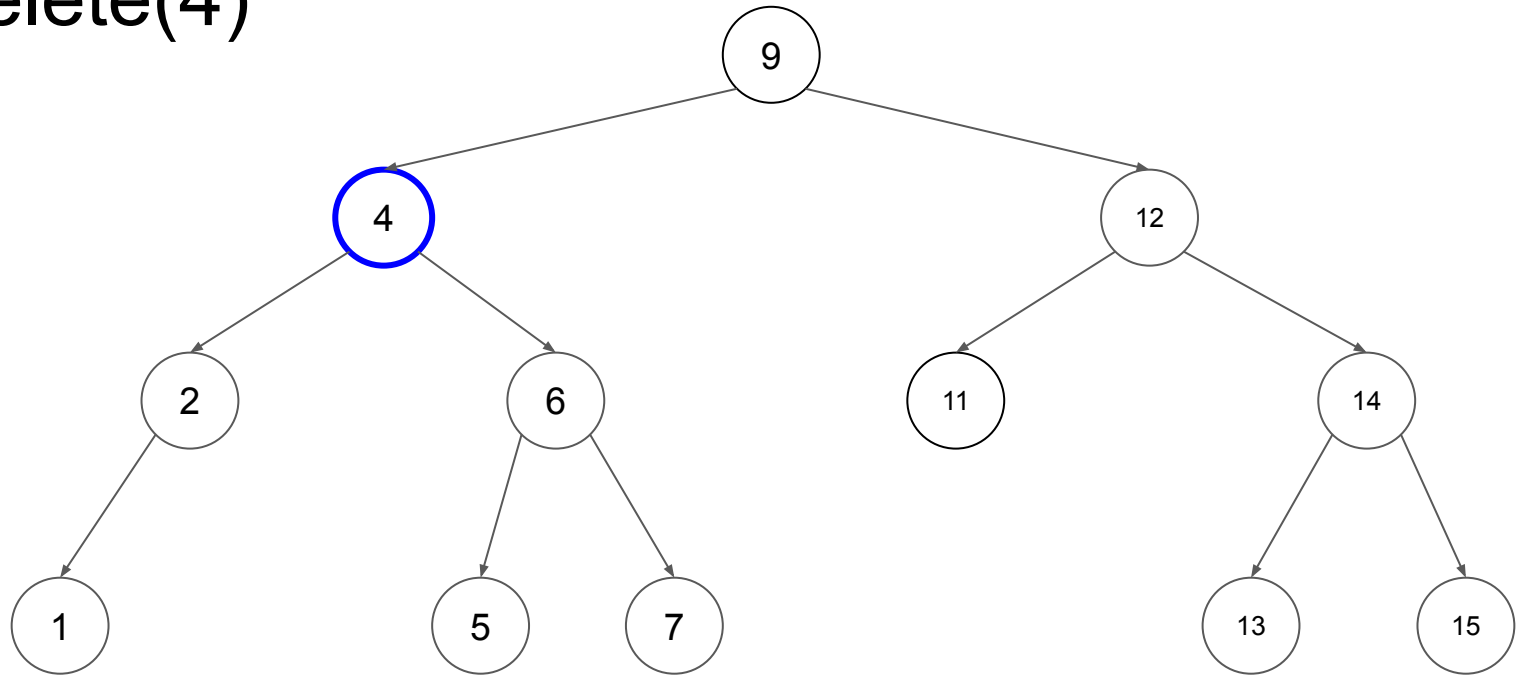
binary search tree
delete(8)



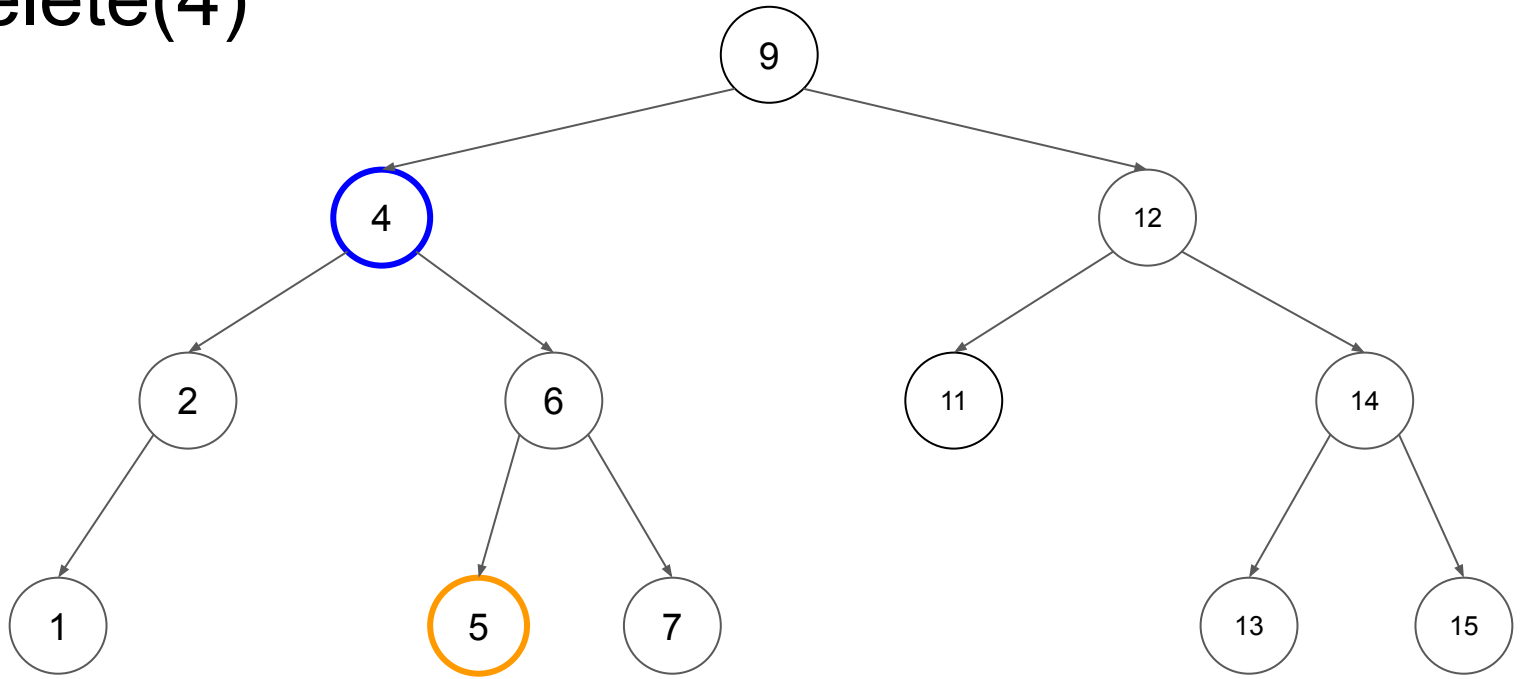
binary search tree
delete(8)



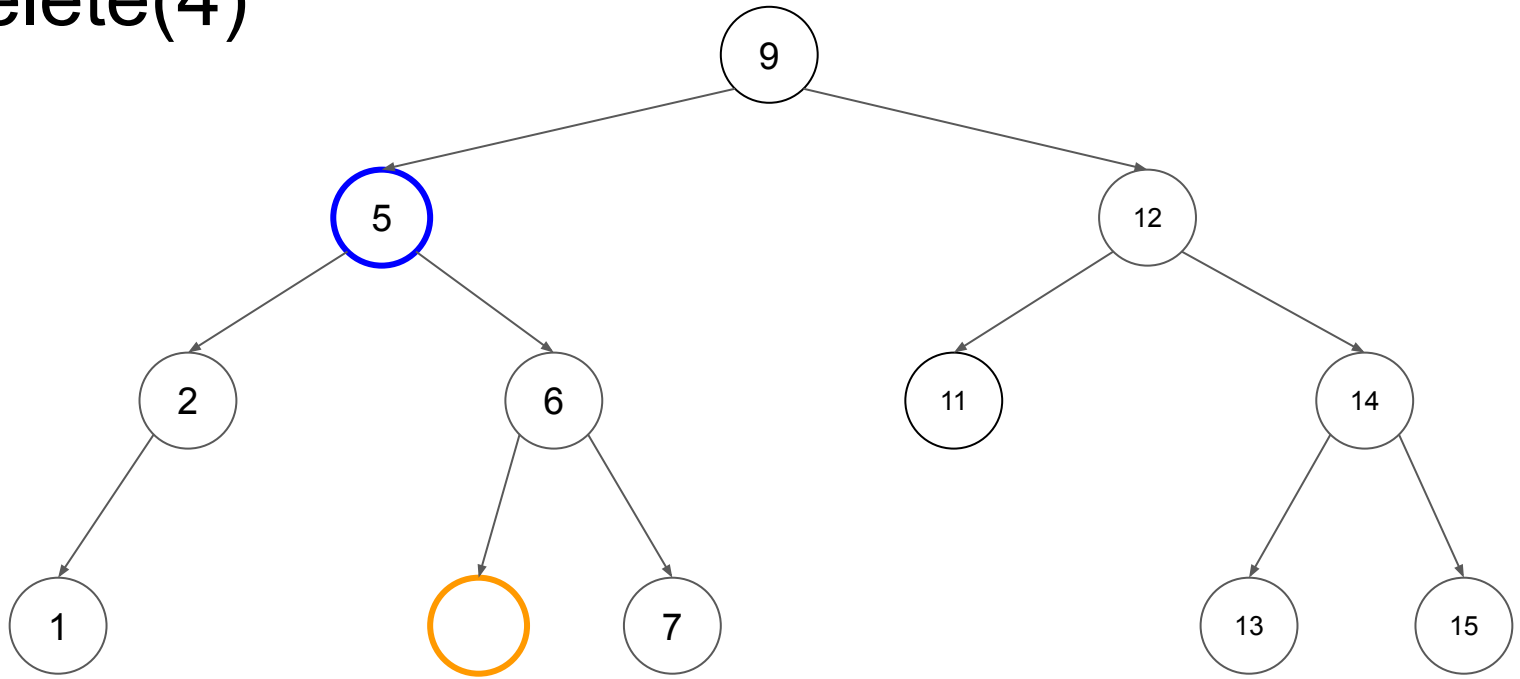
binary search tree
delete(4)



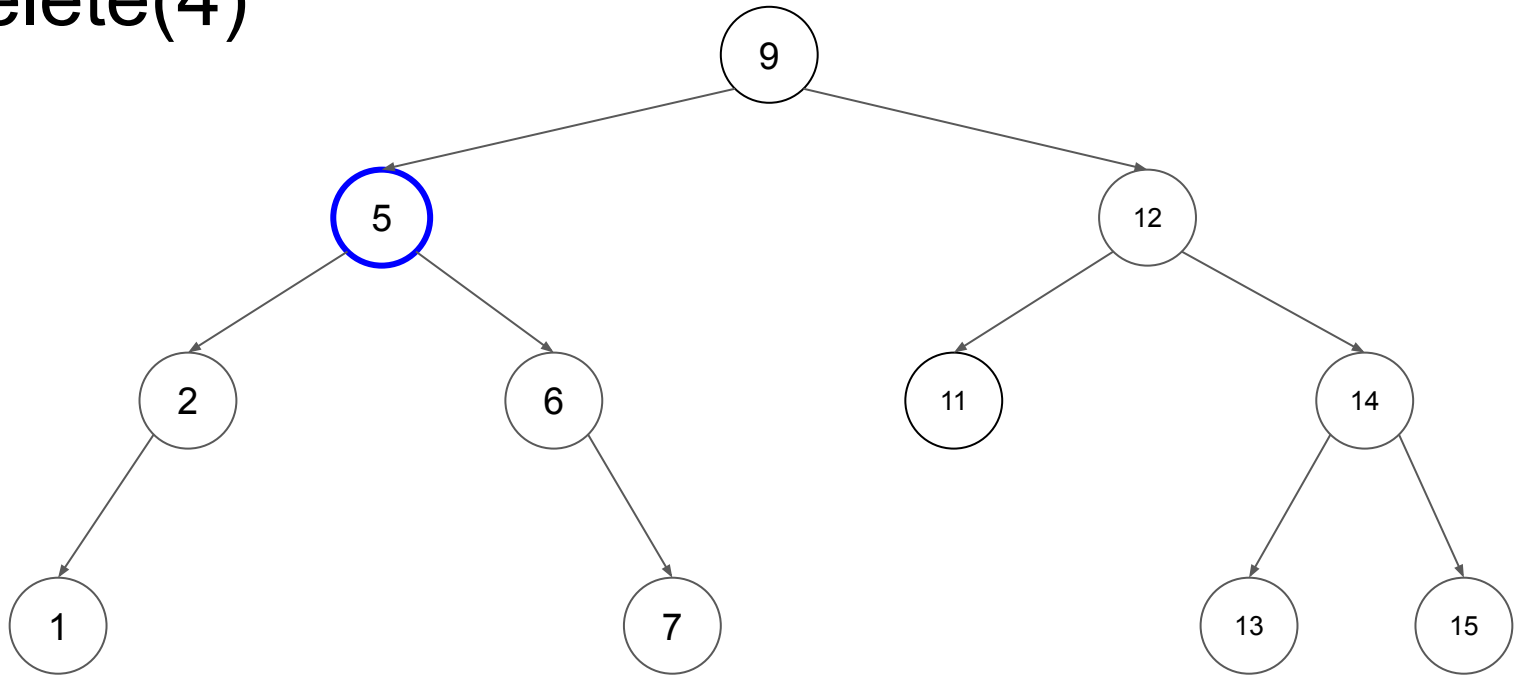
binary search tree
delete(4)



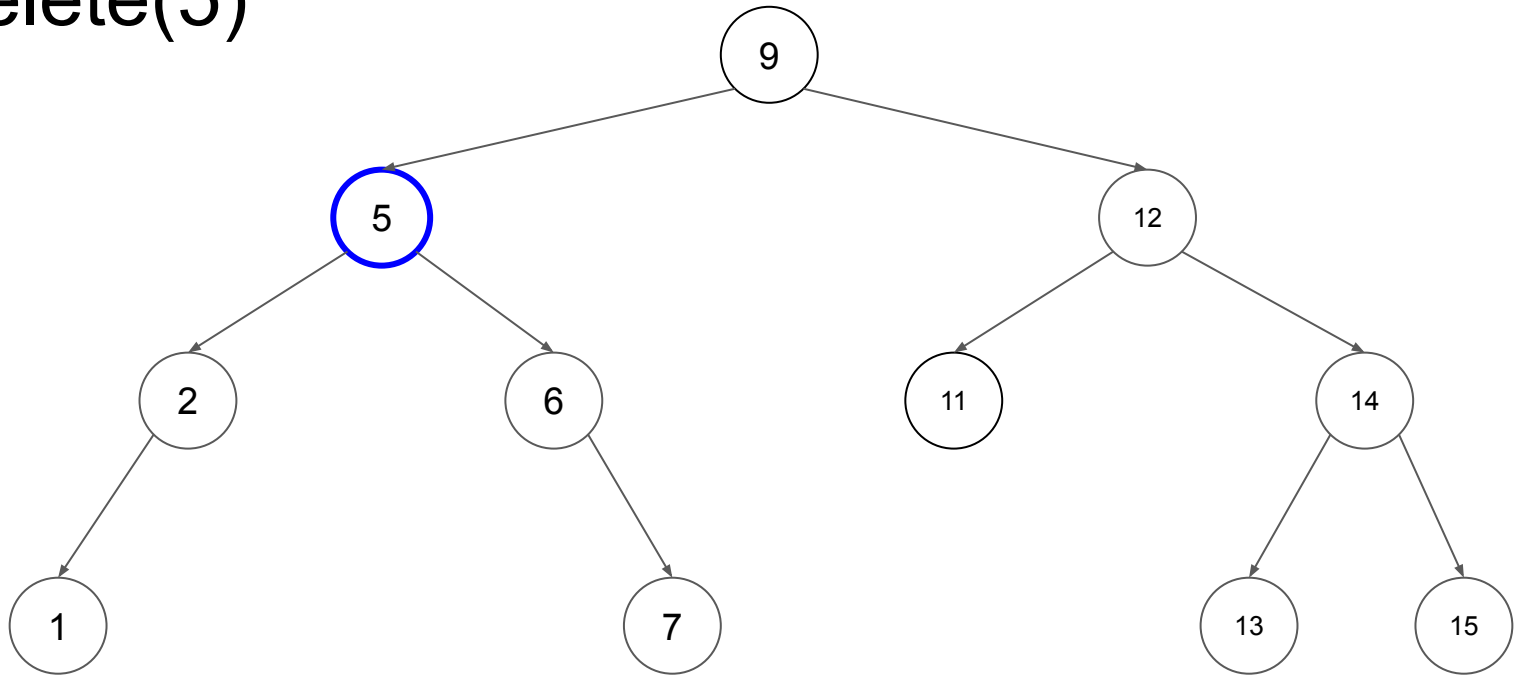
binary search tree
delete(4)



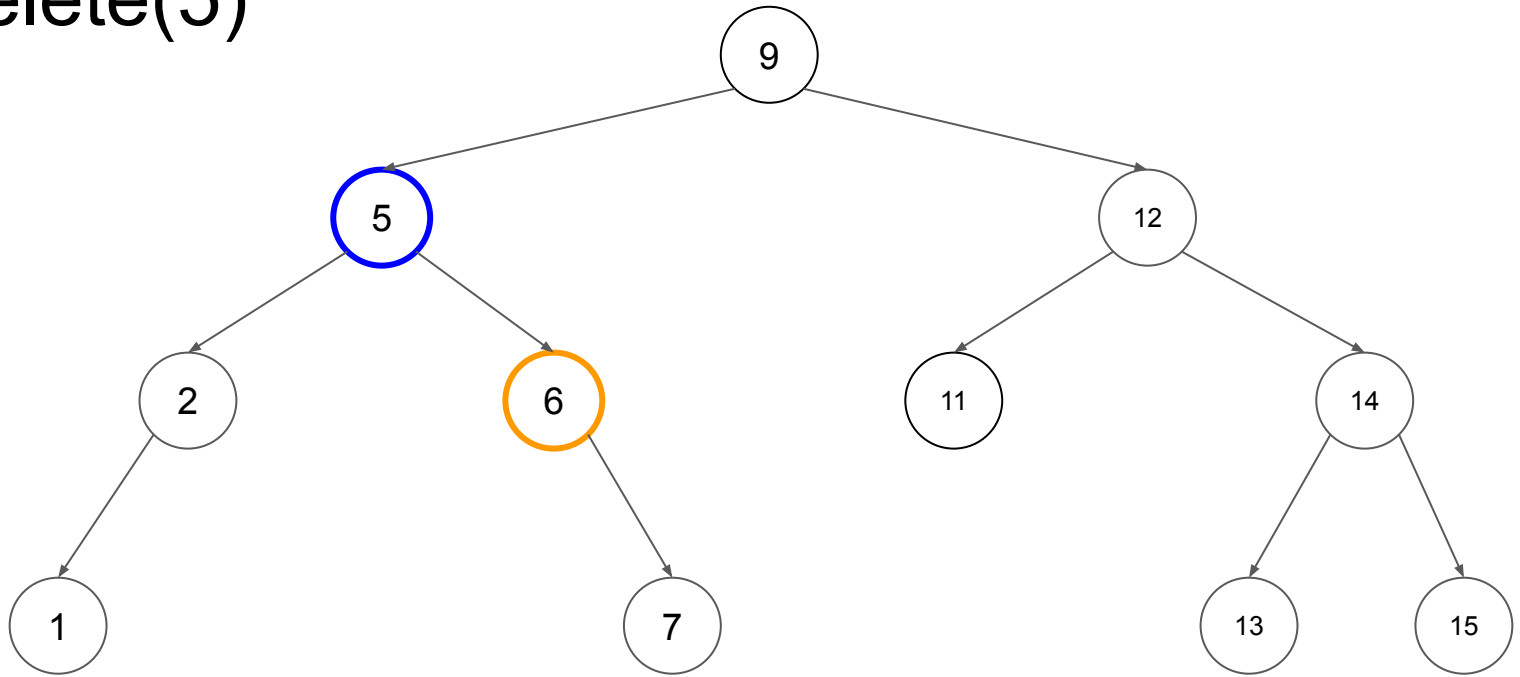
binary search tree
delete(4)



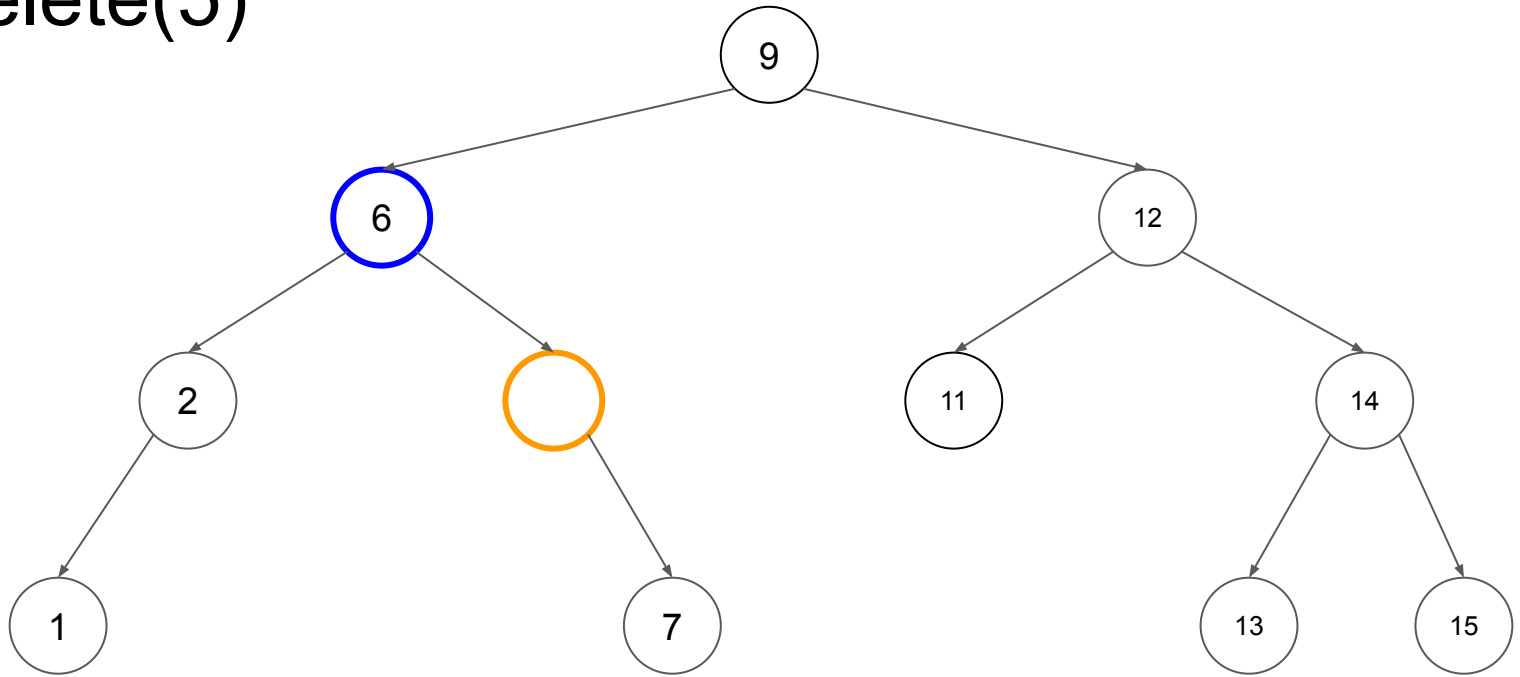
binary search tree
delete(5)



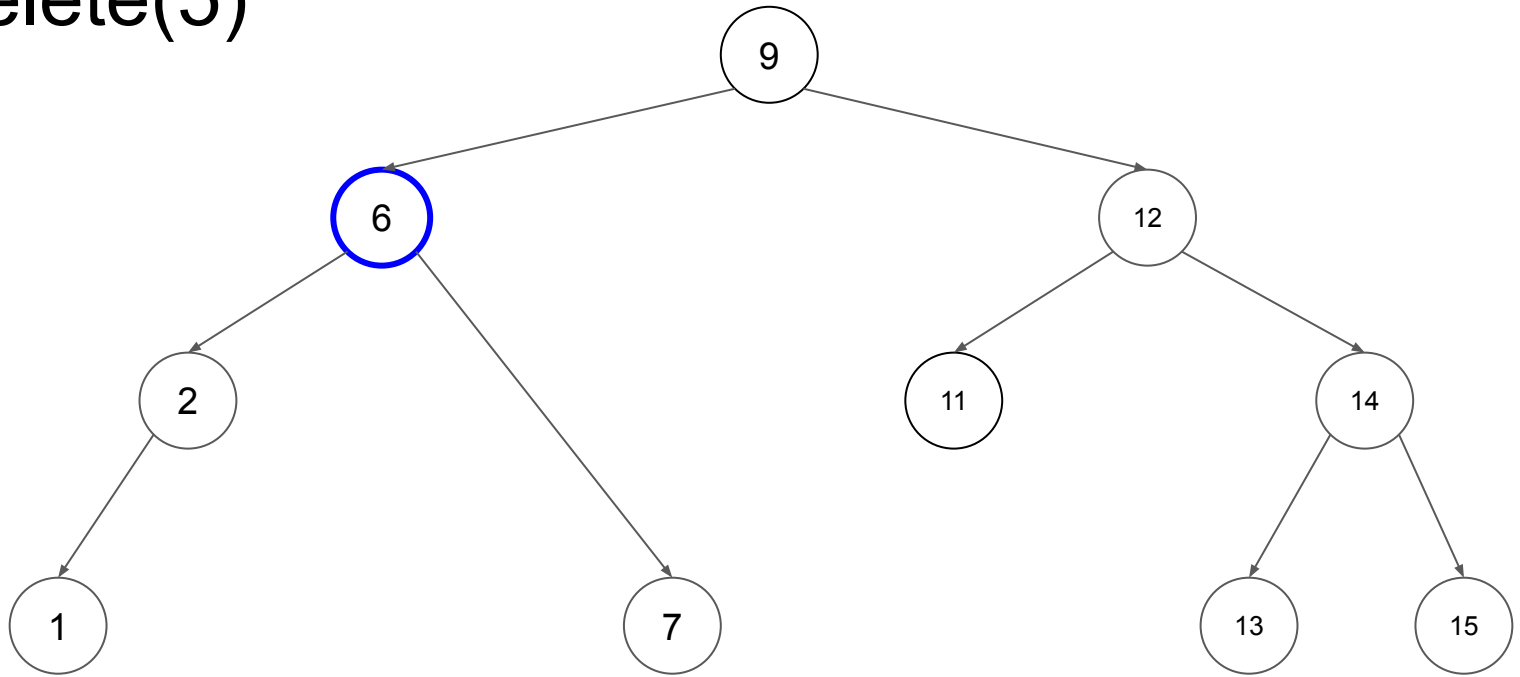
binary search tree
delete(5)



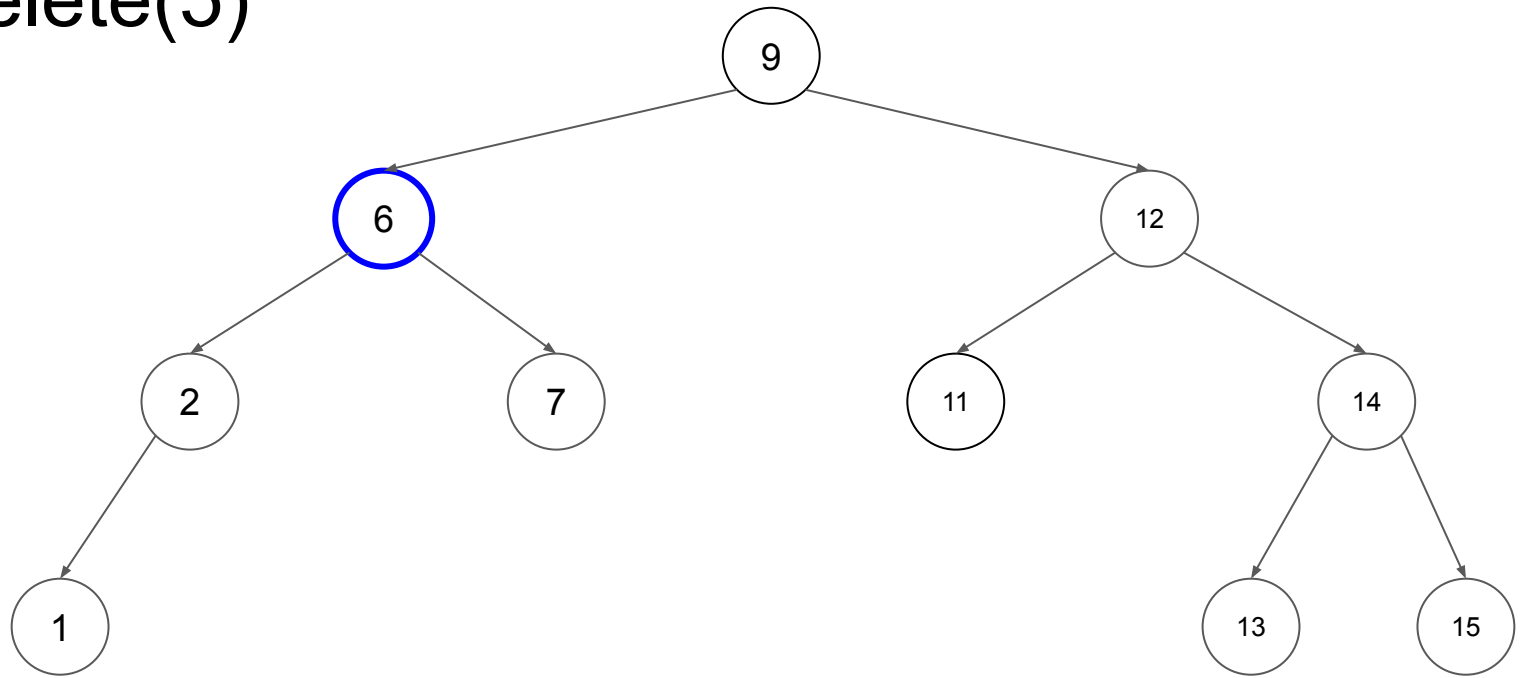
binary search tree
delete(5)



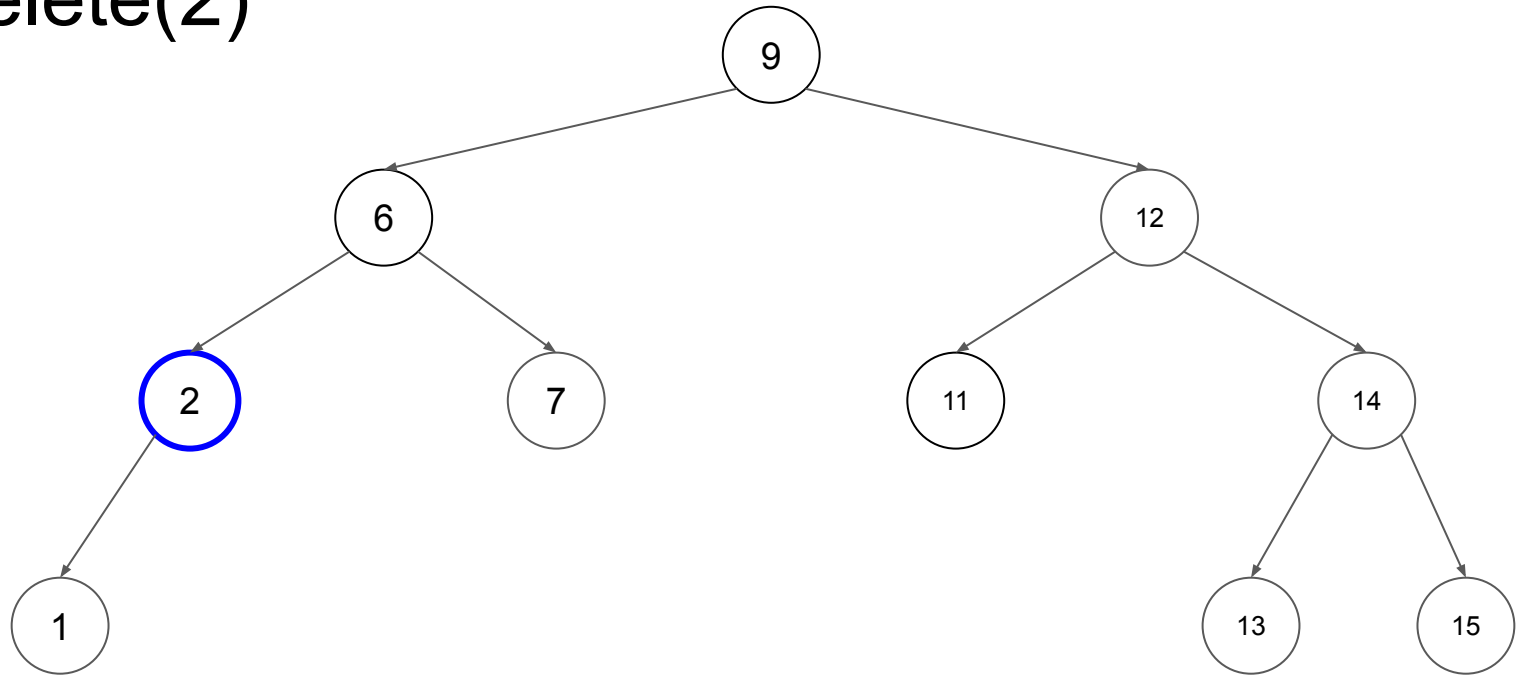
binary search tree
delete(5)



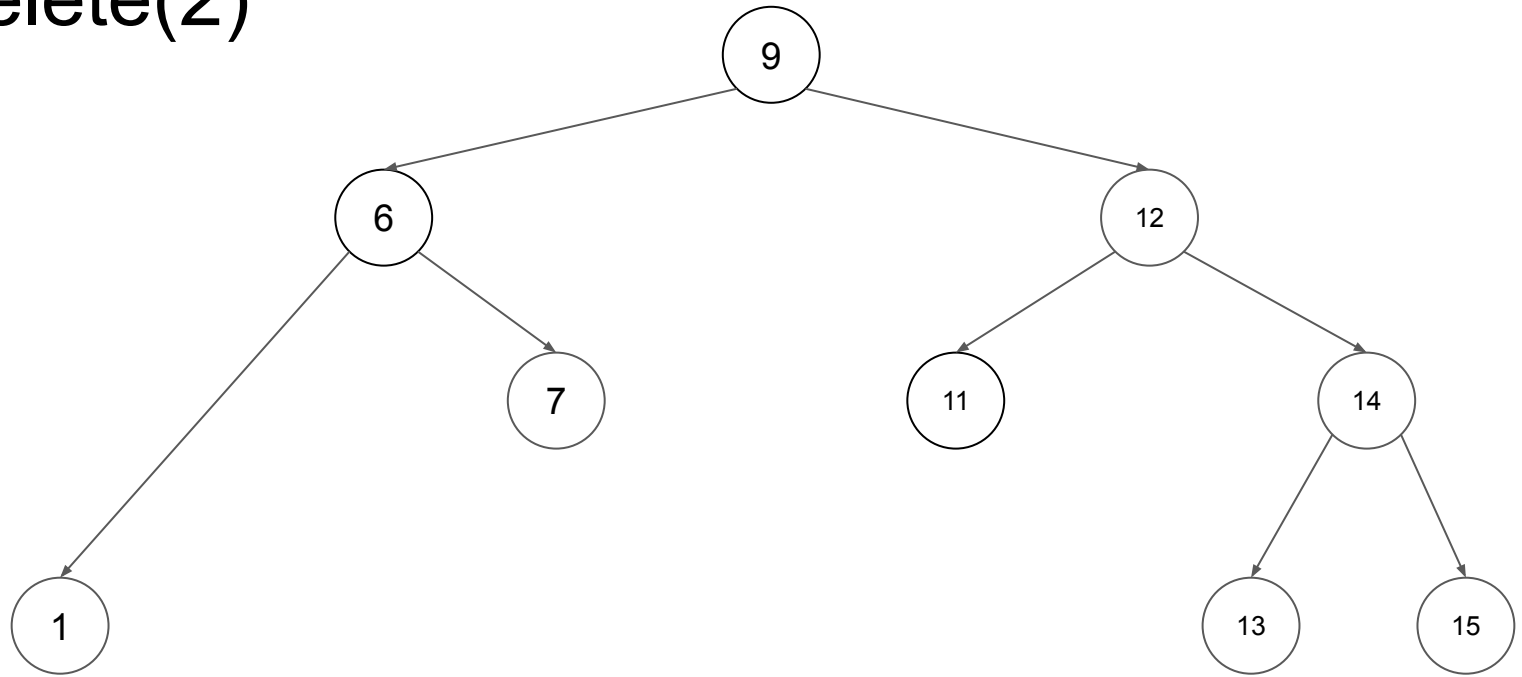
binary search tree
delete(5)



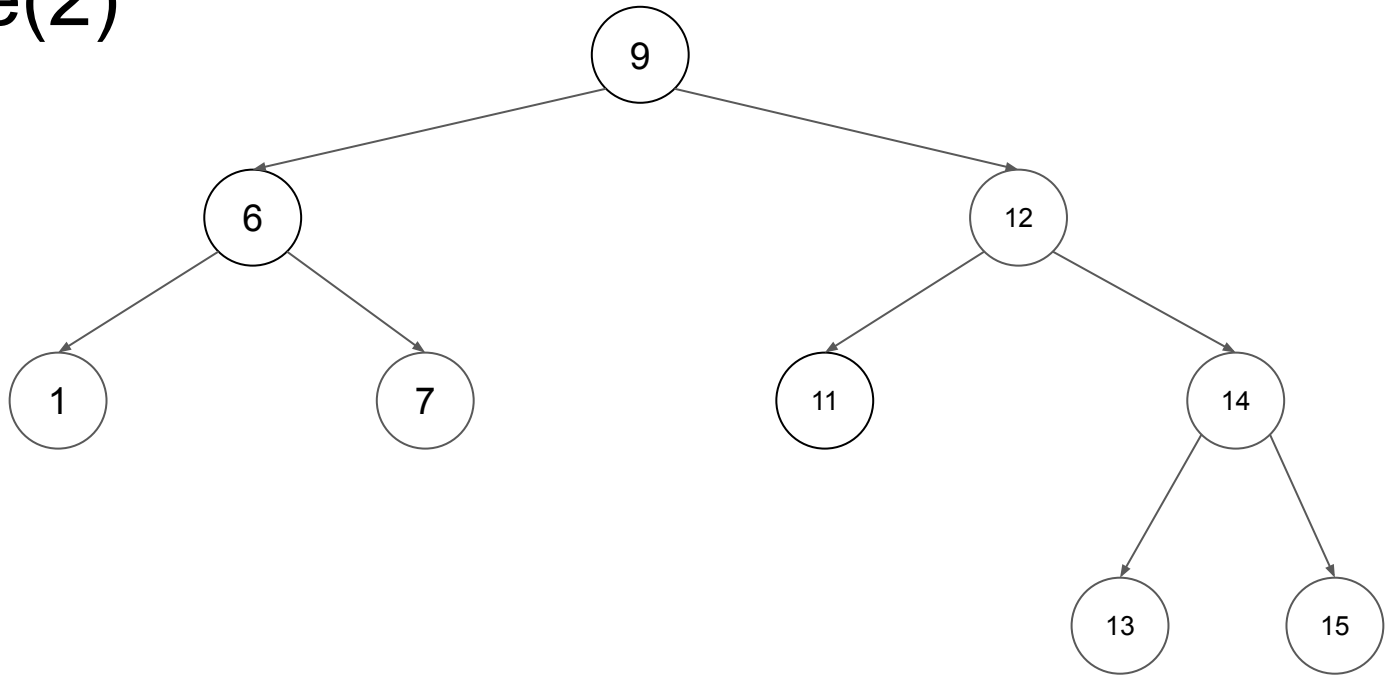
binary search tree
delete(2)



binary search tree
delete(2)



binary search tree
delete(2)



Asymptotic running time of delete()?

$O(h)$ where h is the height to the tree

In Your Pocket

arrays
linked lists
stacks
queues
(trees)

man ssh exit pwd cd ls
valgrind touch mkdir cp
rm rmdir mv cat head
tail less

Sorting Algorithms
- Selection sort
- Insertion sort
- Merge sort
- Quicksort
- Counting sort

Some keywords from today's lecture:

- redirect, pipe
- counting sort
- height of tree
- binary search tree (BST)
- operations performed on binary search trees, search, insert, delete
- (C++) templates

Midterm Fun

starts 7:30 pm; and,
ends 9:00 pm