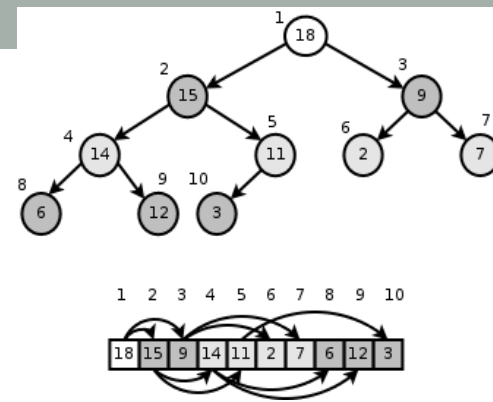
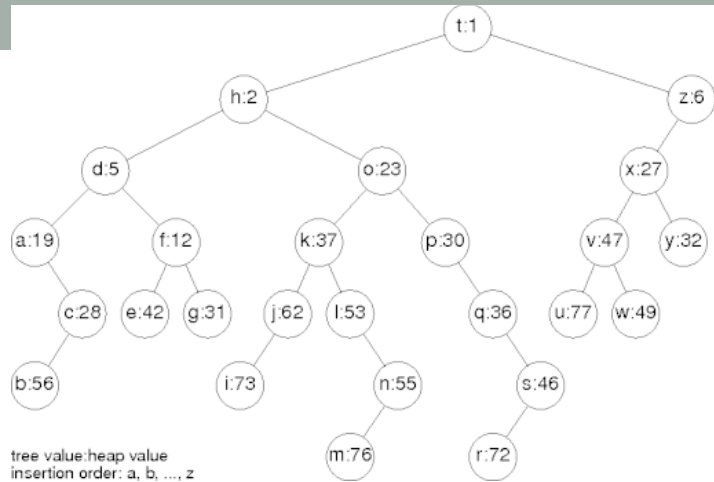
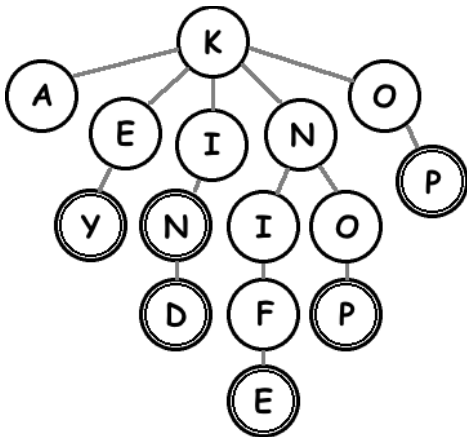




WELCOME TO CSE 100!

Advanced Data Structures in C++



C++

```
#include <iostream>
using namespace std;

int main(){
    cout<<"Hola Facebook\n";
    return 0;
}
```

Instructors

- Diba Mirza (dimirza@eng.ucsd.edu)
 - PhD (Computer Engineering, UCSD)
 - Second year @ UCSD as teaching faculty
- Office: 2124 EBU3B
- Office hours:
 - Mon: 3pm – 4pm, Wed: 3pm to 4pm
 - Or by appointment

Instructors

- Debashis Sahoo (dsahoo@ucsd.edu)
 - PhD (Electrical Engineering, Stanford University)
 - Assistant Professor (Department of Pediatrics)
- Office: BRF II – 2119 (Biomedical Research Facility II)
- Office hours:
 - Wed: 2pm – 3pm, Fri: 2pm to 3pm
 - Or by appointment

Information about TAs and tutors

TAs:

Vidya Kirupanidhi
Vineel Konduru

Office/lab hours available on the course website

<http://cseweb.ucsd.edu/classes/sp15/cse100-ab/>

Tutors:

Albert Chang
Alvin See
Avnish Patel
Daniel Lee
Elaine Hwang
Eswar Dhinakaran
Ethan Chan
Jorel Briones
Joshua Tang
Kyle Buzsaki
Kyle Huynh
Kyle Barron-Kraus
Nan Shu
Richard Lin
Jacob Tao
Patrick Chung
Sanjana Agarwal

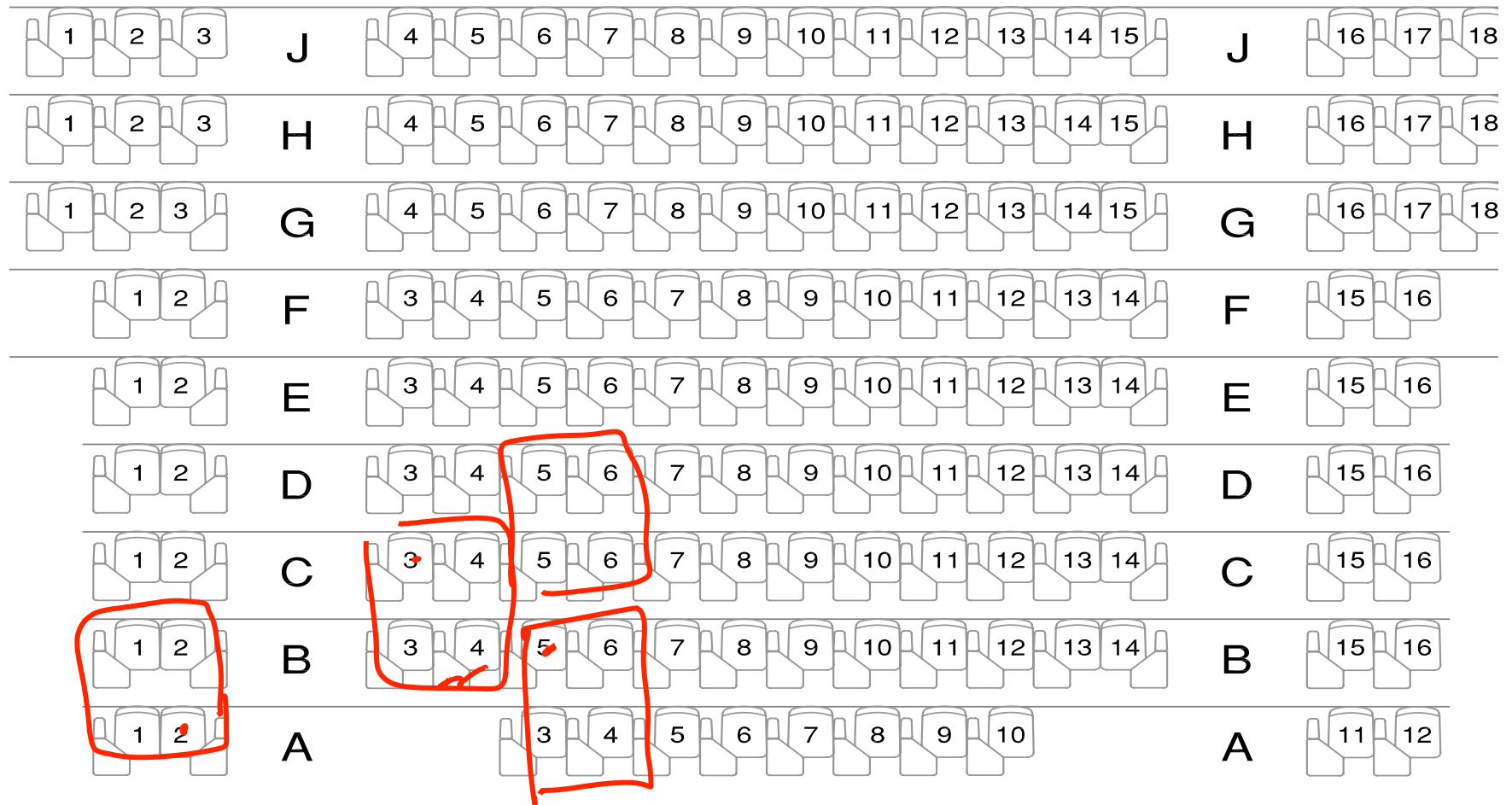
CLICKERS OUT

Set your frequency to AA

Have you been in a class that used peer instruction before?

- A. Yes
- B. No
- C. I'm not sure

Peer instruction groups



Take a minute to introduce yourself to your group members

iClickers: You must bring them

- Buy an iClicker at the Bookstore
- Register it on TED by April 4 (Saturday) .

Assigned Reading from

- *Zybooks: UC San Diego CSE 100 Spring 2015: Programming in C++, Online e-book (\$25)*
- *Open Data Structures (in C++).* Pat Morin. (Free)

Text on reserve

- *Data Structures and Algorithms C++, Fourth Edition*, by Adam Drozdek
- *C++ for Java Programmers*, Mark Allen Weiss

About This Class



You must **attend** class
You must **prepare** for class
You must **participate** in class

Course Logistics

- Less than 75% iClicker response $\equiv$ missing a class
- You may miss up to 2 classes with no penalty
- Two midterms
- Grading
 - Reading quizzes: 5% *out of class via TED*
 - Participation points: 5% (lowest ~~two~~ *3* dropped)
 - Programming Assignments: 35%
PA0: 2%, PA1: 5%, PA2: 9%, PA3: 9%, PA4: 10%
 - Midterm exams: 30% (15% each)
 - Final exam: 25%

Assignment 0: Due today @ 9pm

Today < > April 2015 Day Week Month 4 Days Agenda More ▾ ⚙ ▾						
Sun	Mon	Tue	Wed	Thu	Fri	Sat
29	30 PA0 Due @ 9PM 	31 PA1 Released	Apr 1	2	3	4 Last day to register iclickers
5	6	7 PA1 Due @ 11:59PM	8 PA2 Released	9	10	11
12	13	14	15	16	17	18
19	20	21 PA2 Due @ 11:59PM	22	23	24 Midterm 1	25
26	27	28	29	30	May 1	2

Pair Programming guidelines

Basic rules

- All code written with two programmers at one machine
- You must plan *ahead of time* when you will get together
- You can change partner for each PA
- Don't be a jerk

Selecting partners: Factors to consider

- Schedule compatibility
- Roughly equal “eagerness”
- Roughly equal experience
- Partner from other section

You must choose your partner by MONDAY (tonight)

PA1 due in a week

- Discussion sections for week 1 are mandatory
 - GIT is covered only in discussion sections
 - C++ tutorial
- Get ready for serious C++ programming
 - Read zyBook section 1-8 for C++ introduction
 - Use Piazza to ask burning questions
 - Search posts before asking a new question
 - Google!
 - Refer to course website first
 - Read the PA writeups!

Reading Quizzes, due Wed and Fri before class, administered offline via TED

Today < > April 2015 Day Week Month 4 Days Agenda More ▾ ⚙						
Sun	Mon	Tue	Wed	Thu	Fri	Sat
29	30 PA0 Due @ 9PM	31 PA1 Released	Apr 1 Reading Quiz	2	3 Reading Quiz	4 Last day to register iclickers
5	6	7 PA1 Due @ 11:59PM	8 PA2 Released Reading Quiz	9	10 Reading Quiz	11
12	13	14	15 Reading Quiz	16	17 Reading Quiz	18
19	20	21 PA2 Due @ 11:59PM	22 Reading Quiz	23	24 Midterm 1	25
26	27	28	29 Reading Quiz	30	May 1 Reading Quiz	2

Discussion sections are mandatory in the first two weeks

Why study data structures?

Discuss with your group

- ① Job interviews
- ② Fast storage & retrieval of data (Big Data!)
- ③ Space efficient data representation
- ④ Good abstractions for operations on data
- ⑤ Hone your coding skills → visualize → code

What we will talk about

How do we quantify fast? "Big-Oh" → Worst case; average case

- Structures for fast storage and retrieval of different types of data (Trees, trees, and more trees! Oh, and Hashtables)
- Structures that rely on randomness for (expected) speed (skip lists)
- Structures to support compact data representation (tries)
- Structures to support topology (graphs)
- Structures that take advantage of memory and disk organization (B-Trees)

All that AND C++ and version control systems!

A lot of data (structures) but what should you take away?

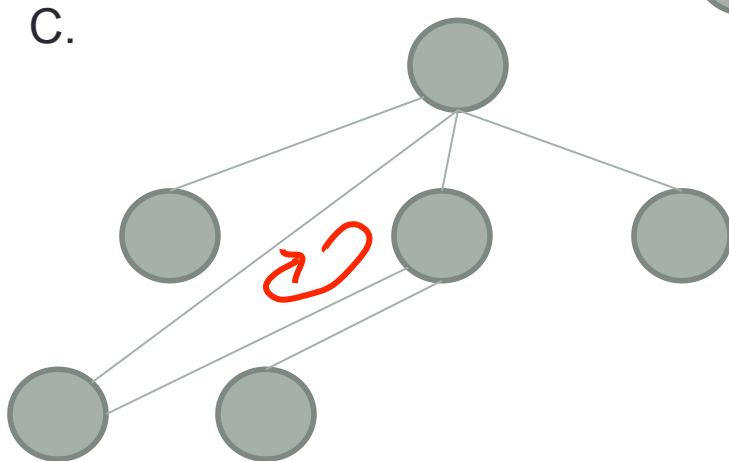
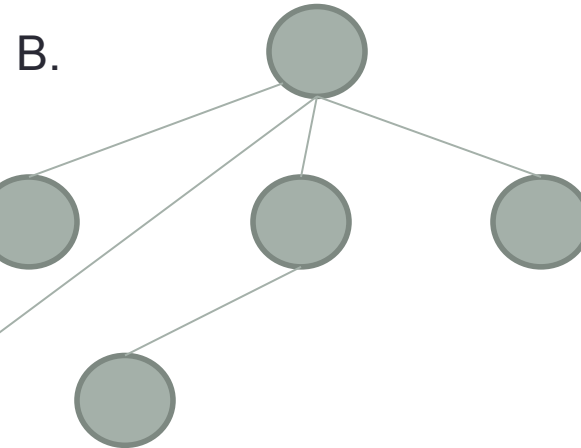
- Ability to choose the correct data structure -> requires understanding data structures from a client perspective:
 - Operations supported
 - Time and space complexity (should be able to analyze)
- Ability to implement some number of non-trivial data structures, correctly, efficiently and elegantly.
 - You may never have to do it (the STL and all...) but if you ever need to, you'll be glad you have the skill.
 - The ability to write and understand good, clean, well-designed C++ code.

If you only remember one thing today

`http://cseweb.ucsd.edu/classes/sp15/cse100-ab/`

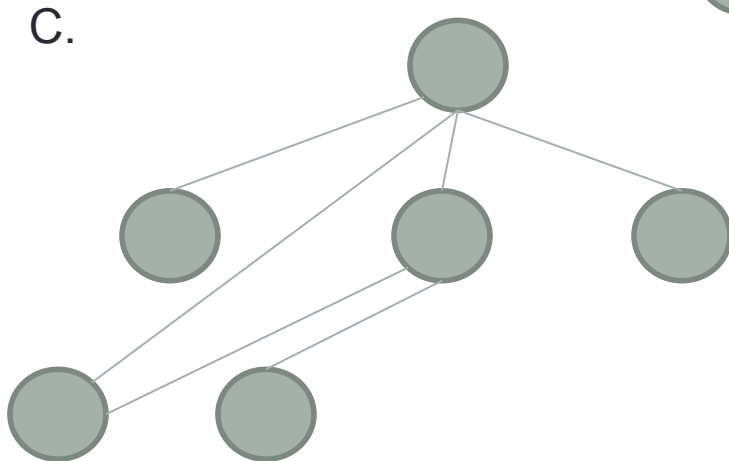
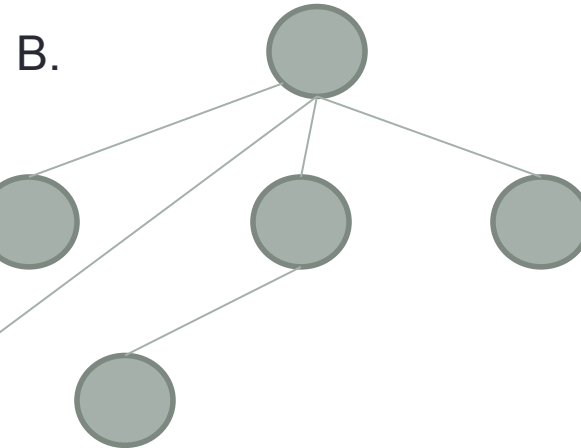
Read the syllabus. Know what's required. Know how to get help.

Which of the following is/are a tree?



- ☒ D. A & B
E. All of A-C

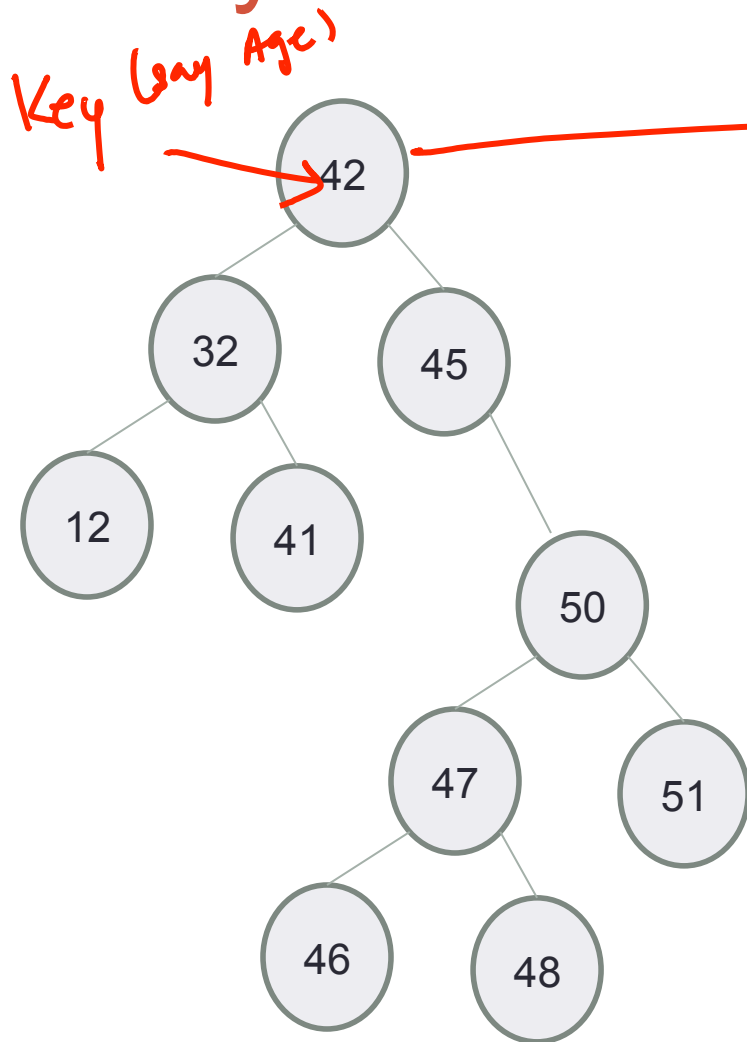
Which of the following is/are a tree?



D. A & B

E. All of A-C

Binary Search Tree – What is it?



This node may in general hold more data, only the key is shown. Say if node was an employee record, the data may be multiple fields

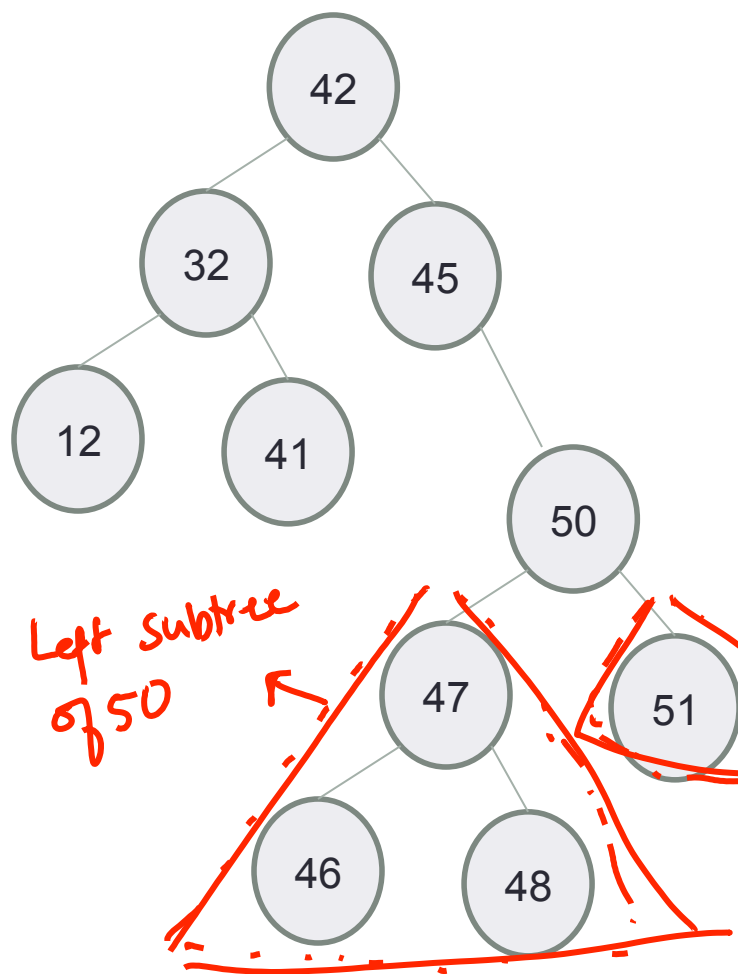
- Employer name
 - Age
 - SSN
 - Salary
- Pick one to be the key (depends on application)

The BST is organized based on the keys



What are the numbers in the nodes?

Binary Search Tree – What is it?

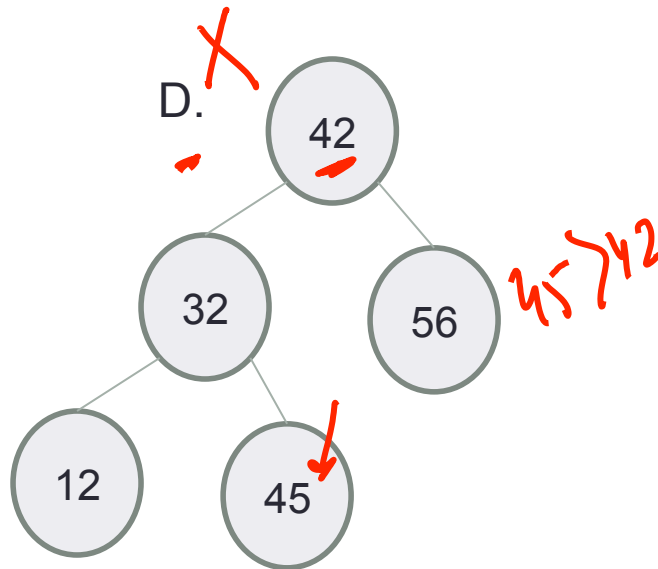
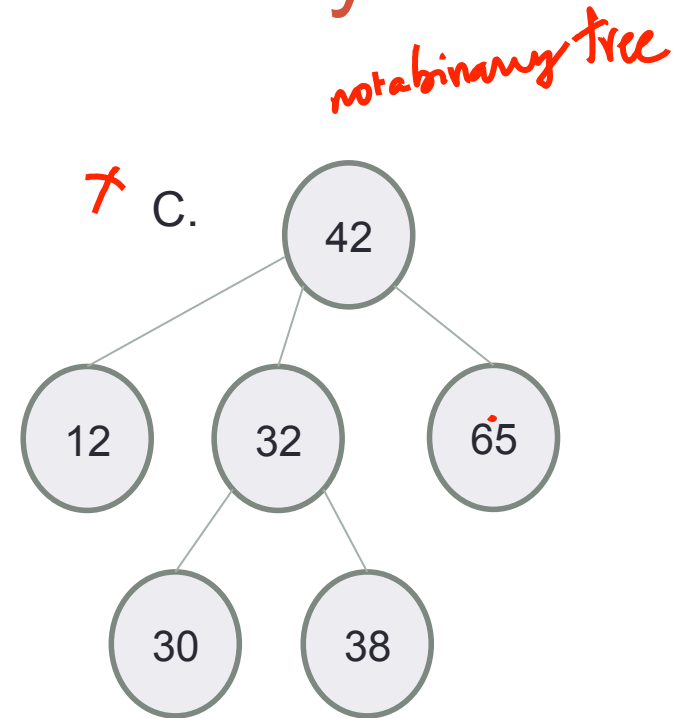
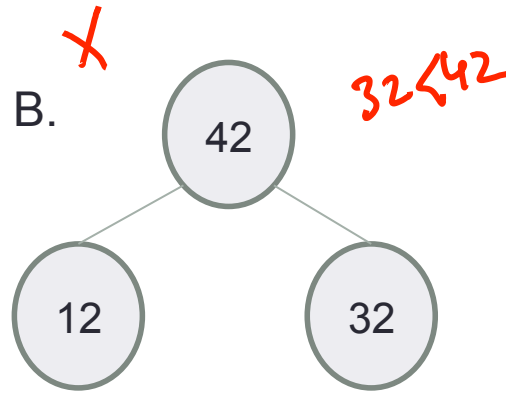
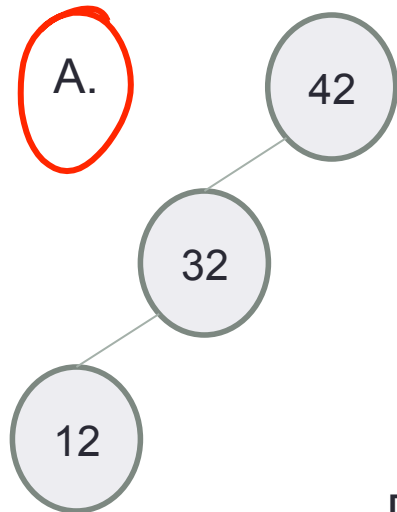


For any node,
Keys in node's left subtree $\leq$ Node key
Node key $\leq$ Keys in node's right subtree



Do the keys have to be integers?

Which of the following is/are a binary search tree?



E. More than one of these

Binary Search Trees

- What are the operations supported? *If available as part of a standard library (say C++ STL) know how to use it.*
- What are the running times of these operations?
Later Lectures
- How do you implement the BST i.e. operations supported by it? *PA1*

Binary Search Trees

- What is it good for?
 - If it satisfies a special property i.e. Balanced, you can think of it as a dynamic version of the sorted array

Supports all operations of a sorted array + Fast Inserts & Deletes

PA1: Implementing BST operations in C++

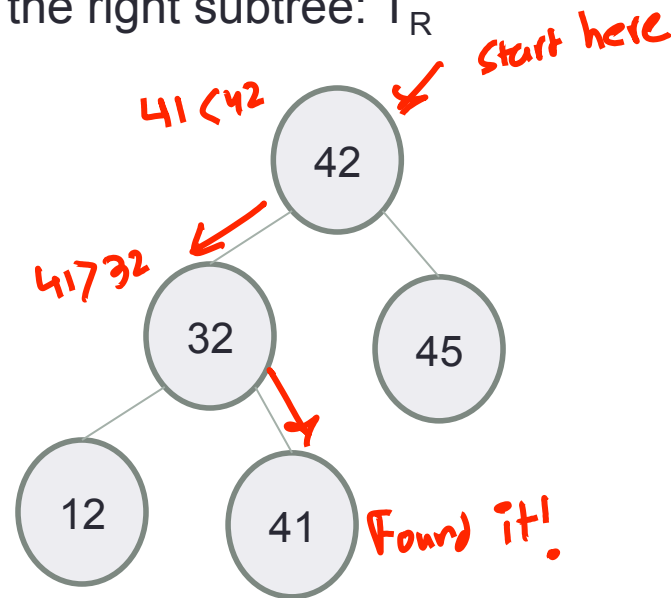
Don't cheat!

- You need to implement
 - find () – find an element
 - size() – returns total number of elements
 - clear() – deletes all the elements ← C++ memory management important here!
 - empty() – checks if the BST is empty
 - successor() – returns the next element in an in-order traversal
- And the iterator pattern (we will talk about it later this week)

Under the hood: Searching an element in the BST

To search for element with key k

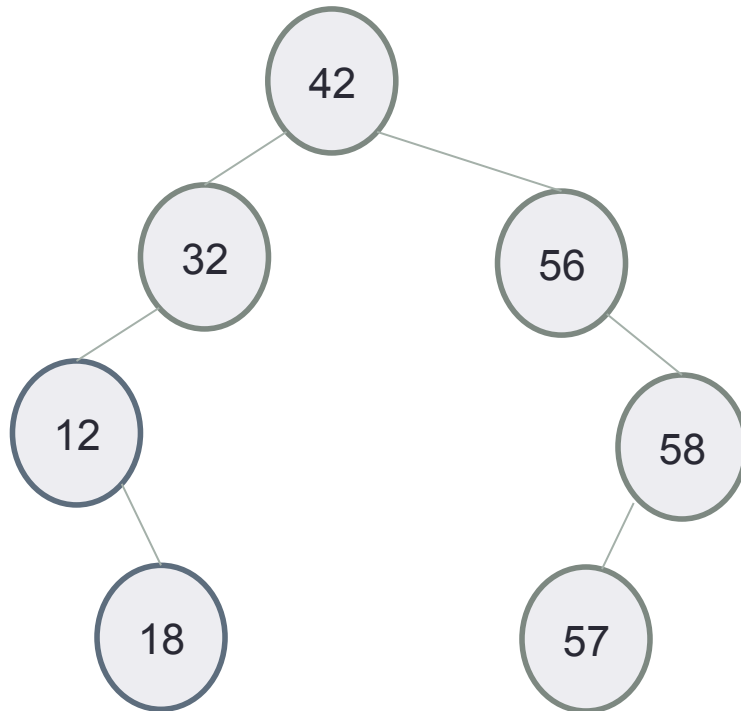
1. Start at the root
2. If $k = \text{key}(\text{root})$, found key, stop.
3. Else If $k < \text{key}(\text{root})$, recursively search the left subtree: T_L
Else recursively search the right subtree: T_R



Search for 41.
Now search for 43.



Under the hood: Finding the successor of an element in the BST



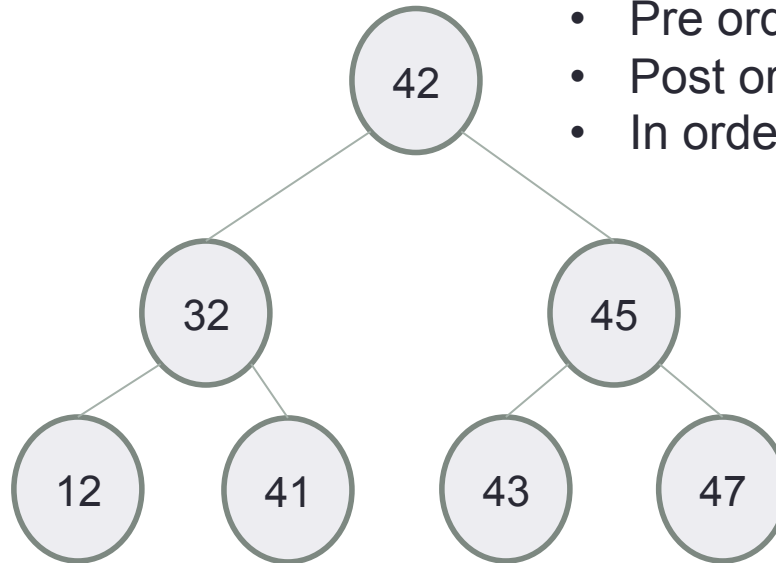
Which node is the successor of 56?
How would you find it?



Traversing the BST in sorted order

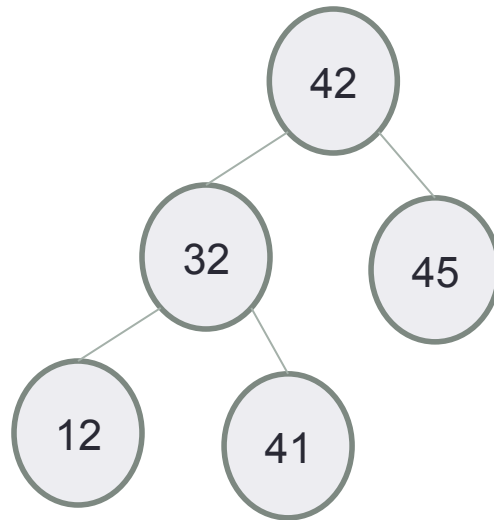
Different methods of tree traversal:

- Pre order traversal
- Post order traversal
- In order traversal



`successor()` – returns the next element in an in-order traversal

In-order traversal of BST



Which of the following results from an in-order traversal of a BST?

- A. Nodes are visited in the order in which they were inserted into the BST
- B. Nodes are visited in order of the number of children that they have
- C. Nodes are visited in sorted order of their keys
- D. None of the above

PA0

- Don't forget to choose a partner and complete PA0 by tonight (Monday, March 30 @9 pm).
- We should have you setup on gitLab by the next day (Tuesday, April 1st @noon). Wait for the announcement.
- You can start working on PA1 soon after.
- PA1 is due in 1 week on Tues (April 7@11:59PM)