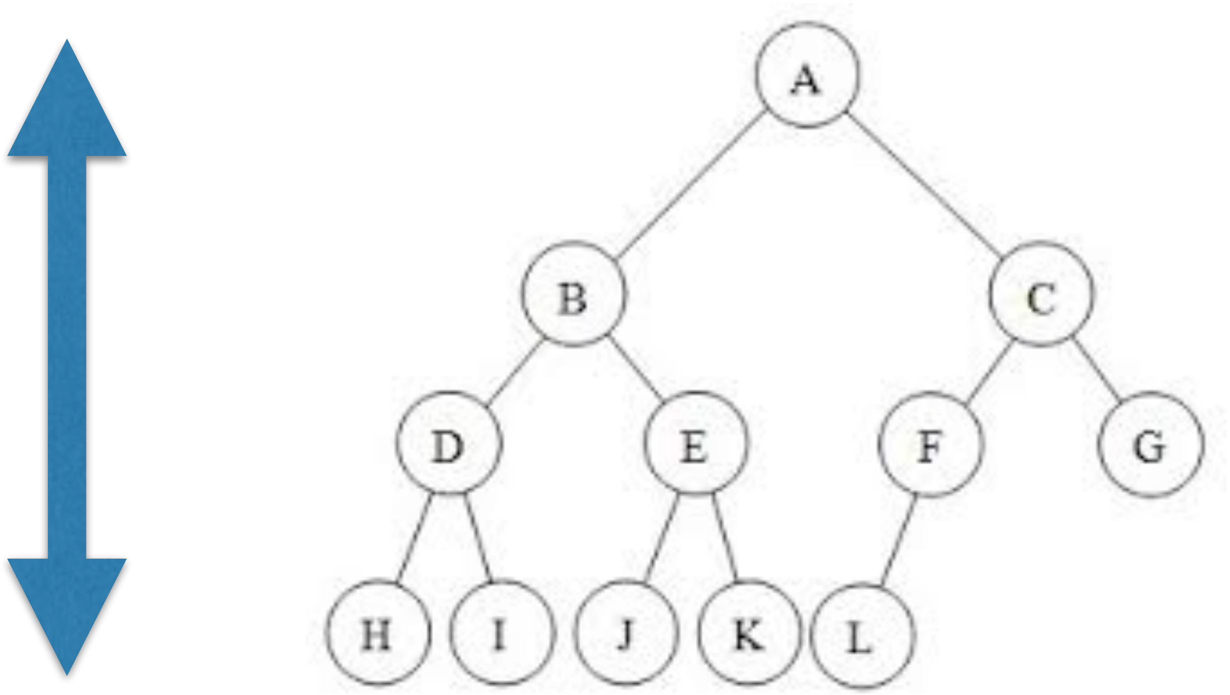
# Data structures

CS 146 - Spring 2017

# Review question



height = 3

- What is the maximum height of a binary tree with n nodes?
- What is the minimum height of a binary tree with n nodes?

# Homework tips

- Review lecture notes from the previous week
- Start early
- When getting help, come with specific questions
    - did you fully understand the question?
    - what did you try?
- Understand the reasoning that lead to the solution

# Today

- Abstract data types
- Sorted set ADT

# Why not matching?

```
List<Integer> li = new ArrayList<Integer>();
```

# Abstract vs concrete data types

- ADT is defined by its interface or **set of operations**
- Can design algorithm by choosing an appropriate ADT based on functionality and without having to worry about details

- Many concrete types can implement the same ADT
- **interchangeable** concrete types within an algorithm using that ADT
- **Different running time** for each operation of each concrete type

# Abstract data types

| ADT | API | implementations |
|---|---|---|
| **list** | get(i) | linked list, array list |
| **stack** | push, pop | list |
| **set** | contains | hash table, BST |
| **sorted set** | predecessor, successor | binary search tree |
| **map or dictionary** | get (key) -> value | hash table, BST |
| **sorted  map** | get(k), pred, succ | binary search tree |
| **priority queue** | extract min | linked list, heap |
| **graph** | neighbors (node) | map, adjacency matrix |

How are these different from their math equivalent?

# ADT in Java

| ADT | Java interface | implementations |
|---|---|---|
| **list** | List | LinkedList, Stack, Vector, ArrayList |
| **set** | Set | HashSet, TreeSet |
| **sorted set** | SortedSet, NavigableSet | TreeSet |
| **map or dictionary** | Map | HashMap, TreeMap |
| **sorted  map** | SortedMap, NavigableMap | TreeMap |
| **priority queue** | - | PriorityQueue (heap) |

# Sorted set ADT

- boolean **contains**(E e)
- **predecessor**(E e), **successor**(E e)
- **add**(E e)
- **remove**(E e)

*in Java, predecessor = lower(), successor = higher()

randomized

| sorted set ADT | linked list | binary search tree | A | B |
|---|---|---|---|---|
| contains | n | n | $2\log_2 n$ | $\log_2 n$ |
| add | n | n | $2\log_2 n$ | $\log_2 n$ |
| remove | n | n | $2\log_2 n$ | $\log_2 n$ |
| lower/higher | n | n | $2\log_2 n$ | $\log_2 n$ |

on expectation

# Go to…

- https://paper.dropbox.com/doc/CS-146-intro-to-data-structures-bBW4AVrZ85GaYQmm7TpNr

# ADTs are used as building blocks for

- algorithms
- other data structures

# Problem solving process

1. understand the problem
2. brainstorm idea
3. carry out idea
4. evaluate, repeat if necessary

For more: How to solve it