Do Now Exercise

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

In regards to P3, list as many types of input arrays with different characteristics as you can come up.

COMP15: Data Structures

Week 6, Summer 2019

Admin

P3: **Sorter**

Project Due by 6pm on Sunday, June 30

Grading Rubrics for P3 (the report part)

30 points

- sorting time graphs (15 points)
- discussion (15 points)

for your report, consider the following cases:

a) list is in order
b) list is reversed
c) list is randomized
d) list has lots of copies of the same number
e) what happens as the number of random elements gets massive (~100 million elements)?
f) consider discussing your findings in terms of big-O notation - do you see what you'd expect?

Questions about P3?

Midterm Exam

on Wednesday, July 3rd

Midterm Exam

- in class; 90 mins
- closed books, closed notes, no electronic devices
- You can bring a sheet of paper (US letter size) with your handwritten notes. We will collect your paper at the end of the exam, so please put your name on it.
- Topics include everything from the lectures, in-class activities, labs, programming projects and Teach Yourself reports that we have done so far. (L6 and P3 are included.)

Midterm Exam Format

- About 10 big questions in total, each could involve sub-questions.

- Type of questions
  - (Given fragments of program code,) implement XXX functions/methods
  - Fill in blanks (terminologies, asymptotic running time, etc.)
  - Multiple choices and justification (Which XXX would you use and why?)
  - Sketch what happens in memory when XXX.
  - Short answers (Give a high-level description of a data structure that supports XXX operations. Give a high-level description of how XXX works. List operations on data structure XXX and their asymptotic running times. How would you implement XXX. etc.)
  - Given a function, write some use cases. Give suggestions on how you update it for XXX.
  - etc.

Question about the Midterm Exam?

Sorting (cont.)

- Selection sort (Week 5)

- Insertion sort (P3)
- Merge sort (P3)
- Quicksort (P3)

- Counting sort (Week 6)
- Heap sort (Week 8)

(Slide from Week 5)

A sneak peek preview (Comp 160, Algorithms)

Big-O

"asymptotically no larger than"
"asymptotic upper bound"

$f(n)$ is $O(g(n))$ if:

- There exists a positive constant c and
- There exists a positive value n_0 of n such that
- $0 \leq f(n) \leq c * g(n)$ `for all` $n \geq n_0$.

Ref: Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms.* MIT press.

(Slide from Week 5)

A sneak peek preview (Comp 160, Algorithms)

Big-Ω

"asymptotically no smaller than"
"asymptotic lower bound"

$f(n)$ is $\Omega(g(n))$ if:

- There exists a positive constant c and
- There exists a positive value n_0 of n such that
- $0 \leq c * g(n) \leq f(n)$ for all $n \geq n_0$.

Ref: Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms.* MIT press.

(Slide from Week 5)

A sneak peek preview (Comp 160, Algorithms)

Big-Θ

"asymptotically equal to"

$f(n)$ is $\Theta(g(n))$ if and only if:

- $f(n)$ is $O(g(n))$ and
- $f(n)$ is $\Omega(g(n))$.

Ref: Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms.* MIT press.

(draw graphs on the whiteboard)

Running Time Complexity of Sorting Algorithms

Selection sort

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

```
void selectionSort(int* const array, int size){
  for(int target = 0; target < size - 1; target++){

    int index = target;

     for(int i = index + 1; i < size; i++){
      if(array[i] < array[index]){
        index = i;
      }
    }

    int smallest = array[index];
    array[index] = array[target];
    array[target] = smallest;
  }
}
```

Worst-case running time

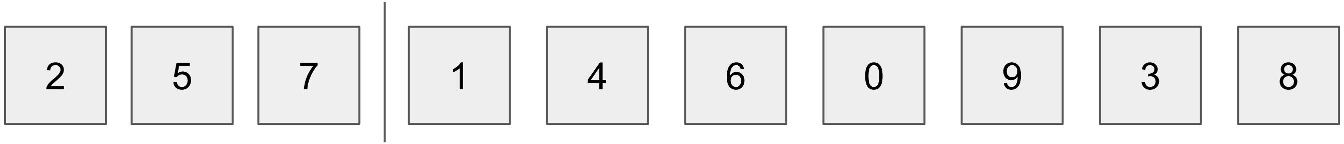
Best-case running time

Selection sort

Worst-case: $O(n^2)$
Best-case: $O(n^2)$

Insertion sort

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



Do Now Exercise

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

In regards to P3, list as many types of input arrays with different characteristics as you can come up.

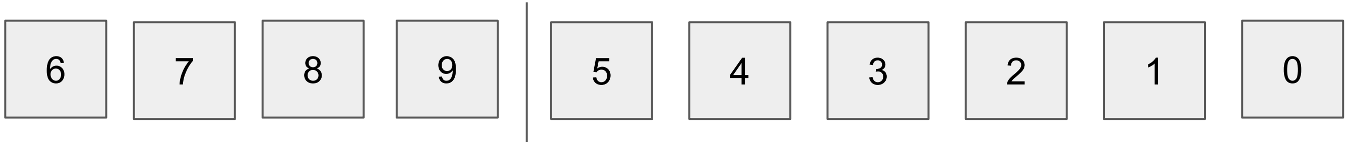
Do Now Exercise

Students' answers:

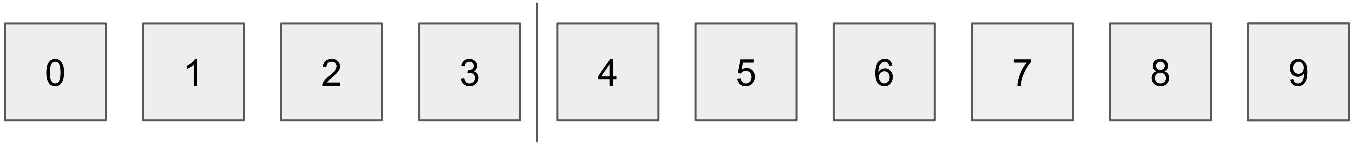
-

Insertion sort

(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.)



Insertion sort

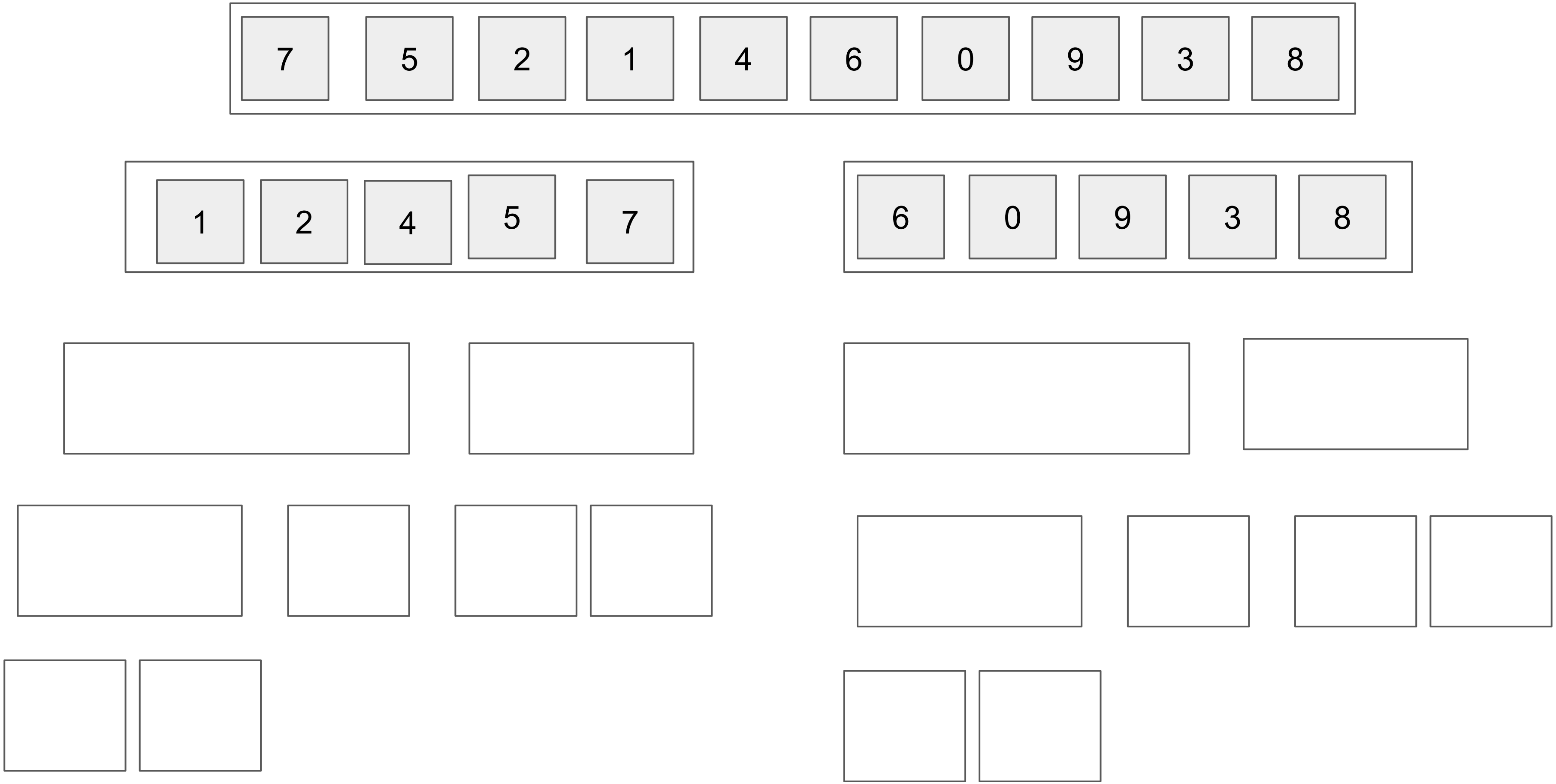
Worst-case: $O(n^2)$
Best-case: $O(n)$

In-place

Merge sort

Divide and conquer

(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.)

(merge sort, very rough procedures, not precise)

```
mergeSort(A, /* (your will figure out) */)
  if /* (check stop condition) */
    find the middle of A
    mergeSort(A, /* (represent first half) */)
    mergeSort(A, /* (represent second half) */)
    merge(A, /* (first half and second half) */)
```

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

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 2 * T(\frac{n}{2}) + \Theta(n) & \text{if } n > 1 \end{cases}$$

(We used the whiteboard to derive)

$$\Theta(n * log(n))$$

A sneak peek preview (Comp 160, Algorithms)

Master theorem

(Introduced only the word...)

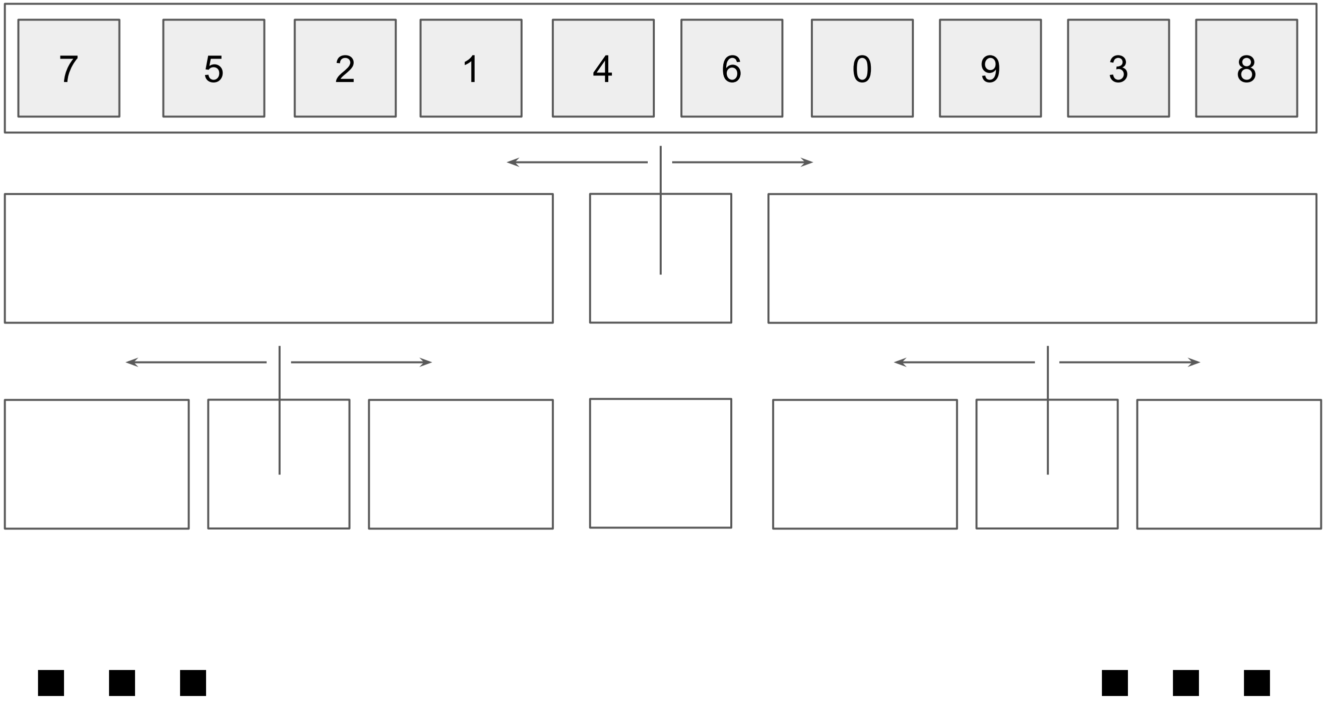
Stable

Merge sort

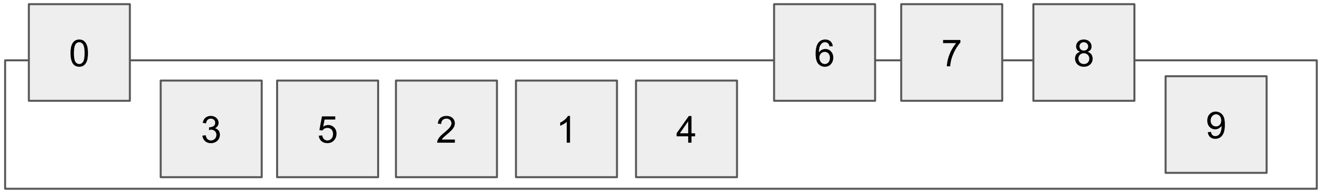
Worst-case: $O(n * log(n))$
Best-case: $O(n * log(n))$
(In-place: No)
(Stable: Yes)

Quicksort

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



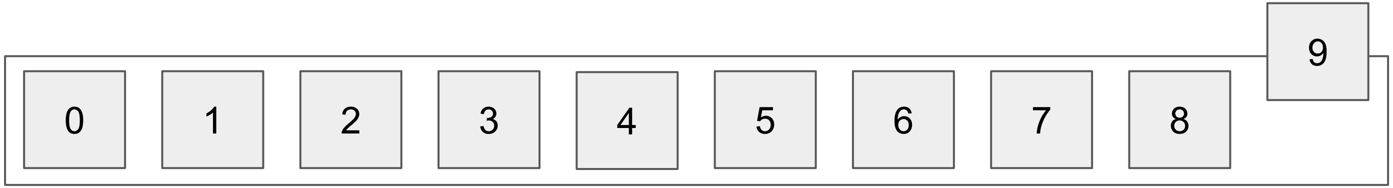
(Notes from the live demo or live coding. Please do NOT assume the code is complete.)



pivot selection

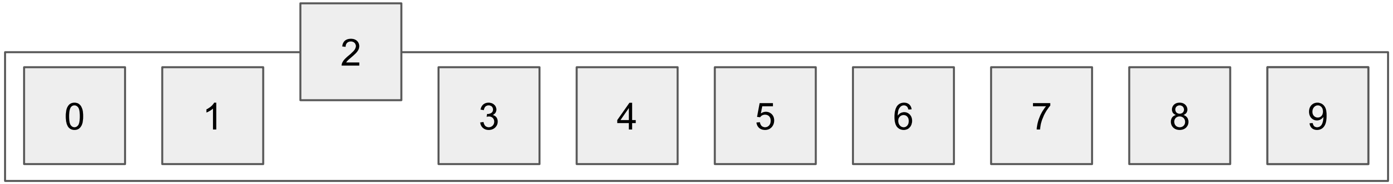
partition phase

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



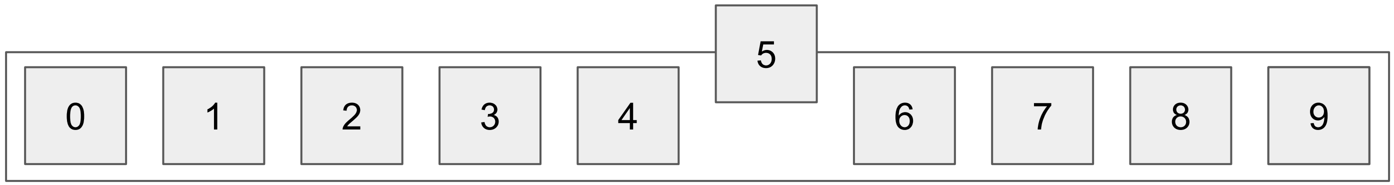
Randomly pick the pivot

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



Median-of-3 partition

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



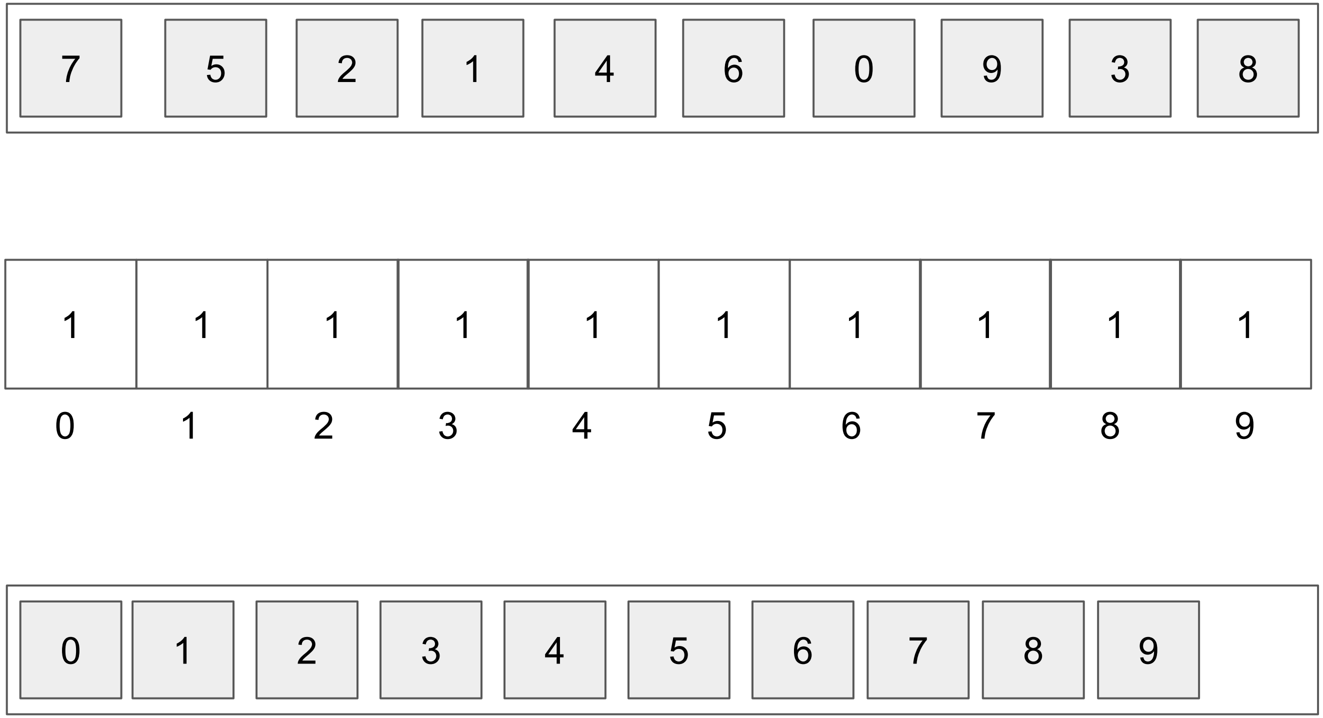
Average case running time

Quicksort

Worst-case: $O(n^2)$
Average-case: $O(n * log(n))$
(In-place: Yes)
(Stable: No (for the version we saw))

Counting sort

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



bounded-universe
(fixed-)

Questions about sorting?

Trees

(Linked structure)
Hierarchical structure

Why trees?

How long does **bool contain(TYPE item)** method of Array or LinkedList class take?

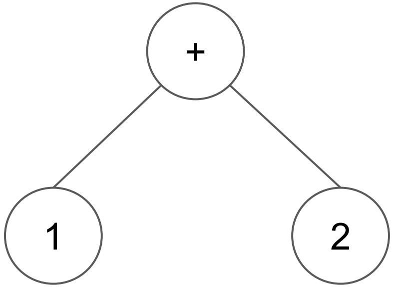
(tree examples)

A sneak peek preview (Comp 105, Programming Languages)

Abstract Syntax Tree (AST)

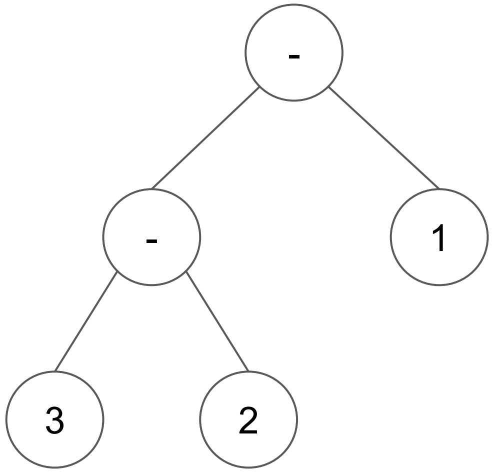
```
1 + 2;
```

```
1 + 2;
```



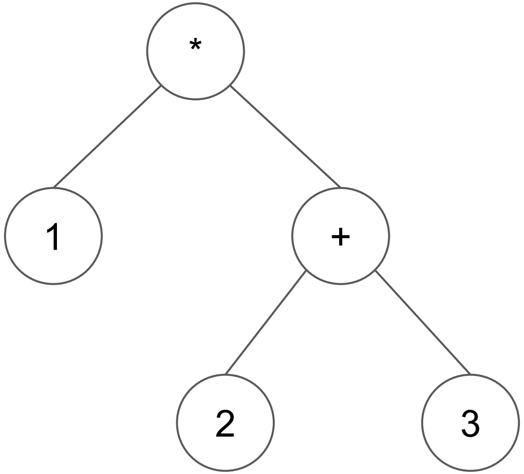
```
3 - 2 - 1;
```

```
3 - 2 - 1;
```



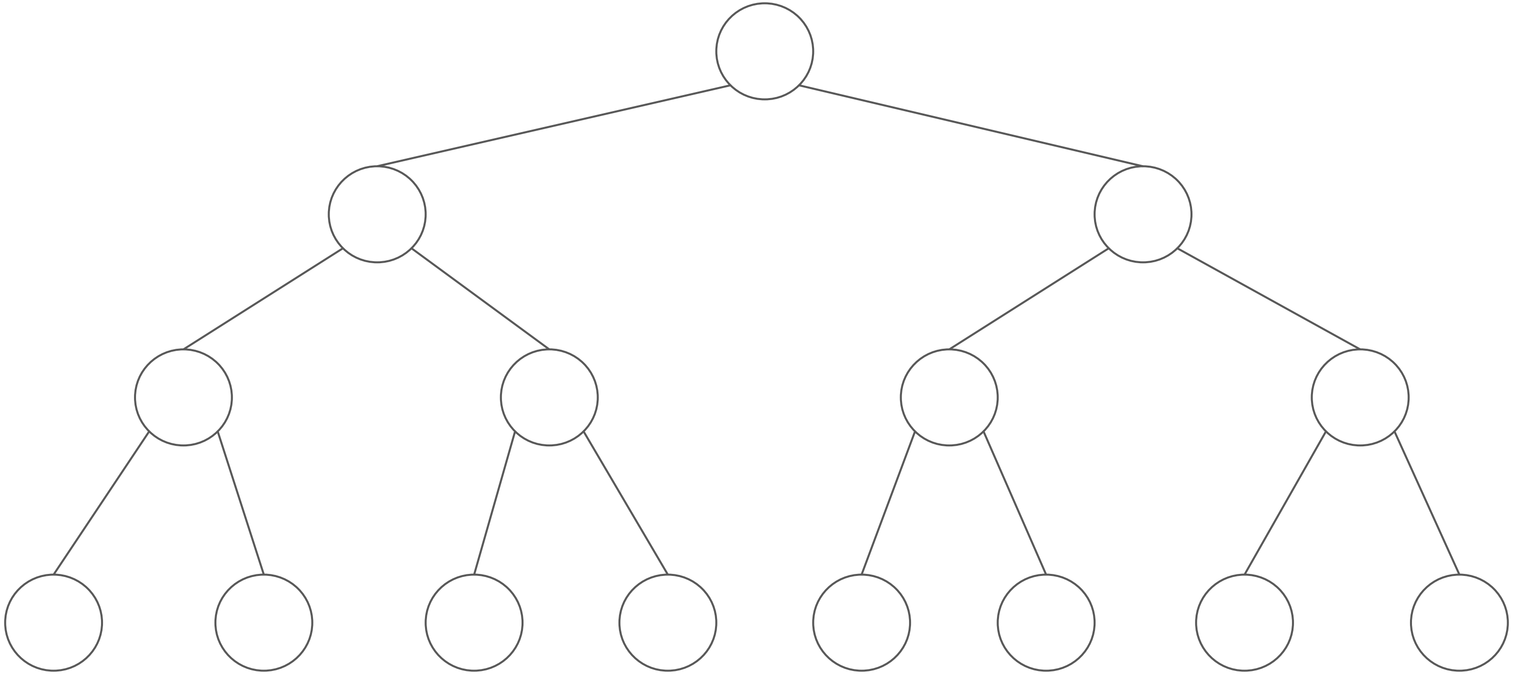
```
1 * (2 + 3);
```

```
1 * (2 + 3);
```

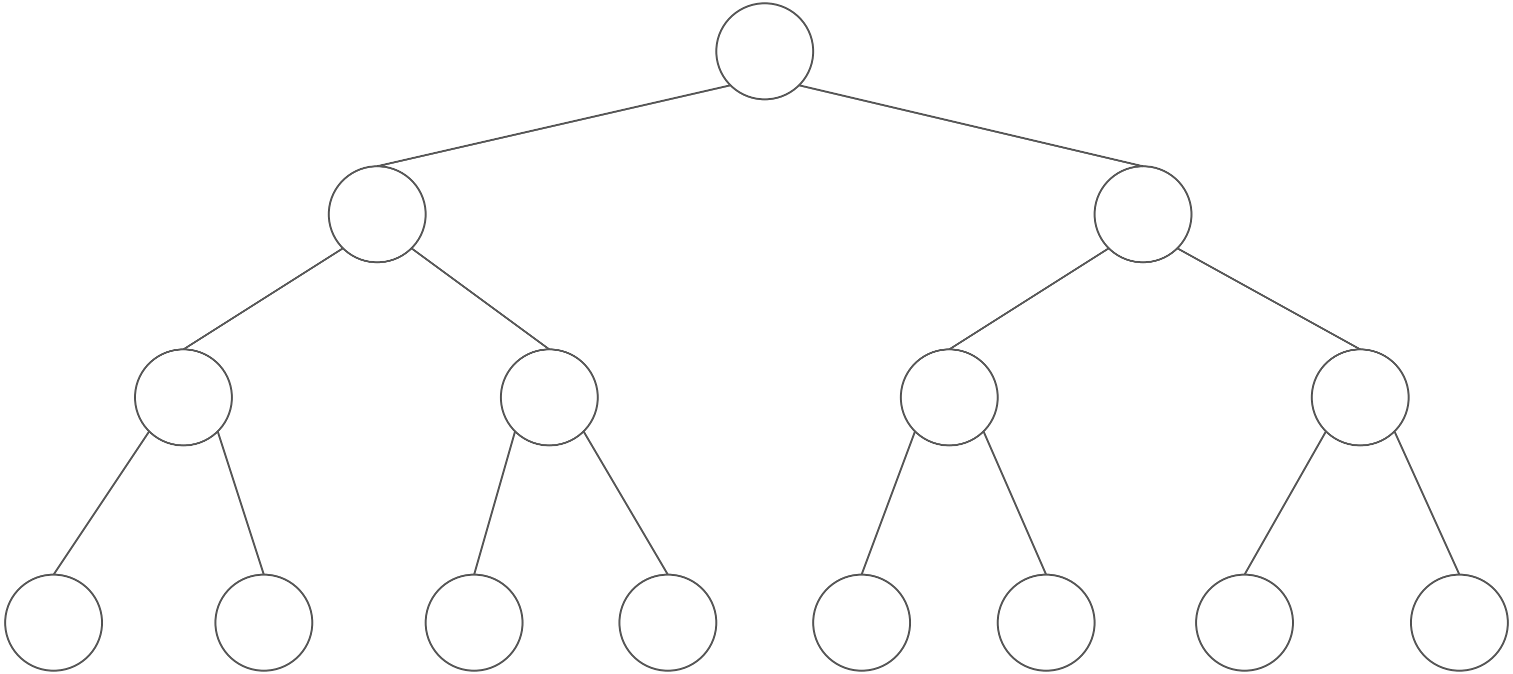


Terminologies

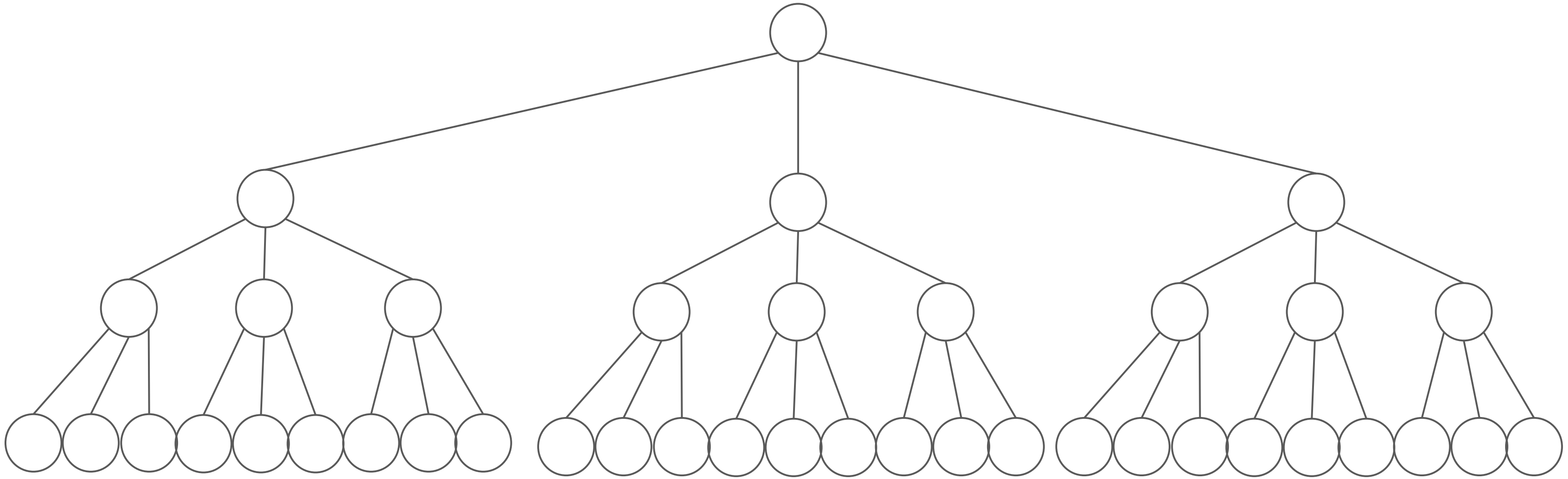
tree



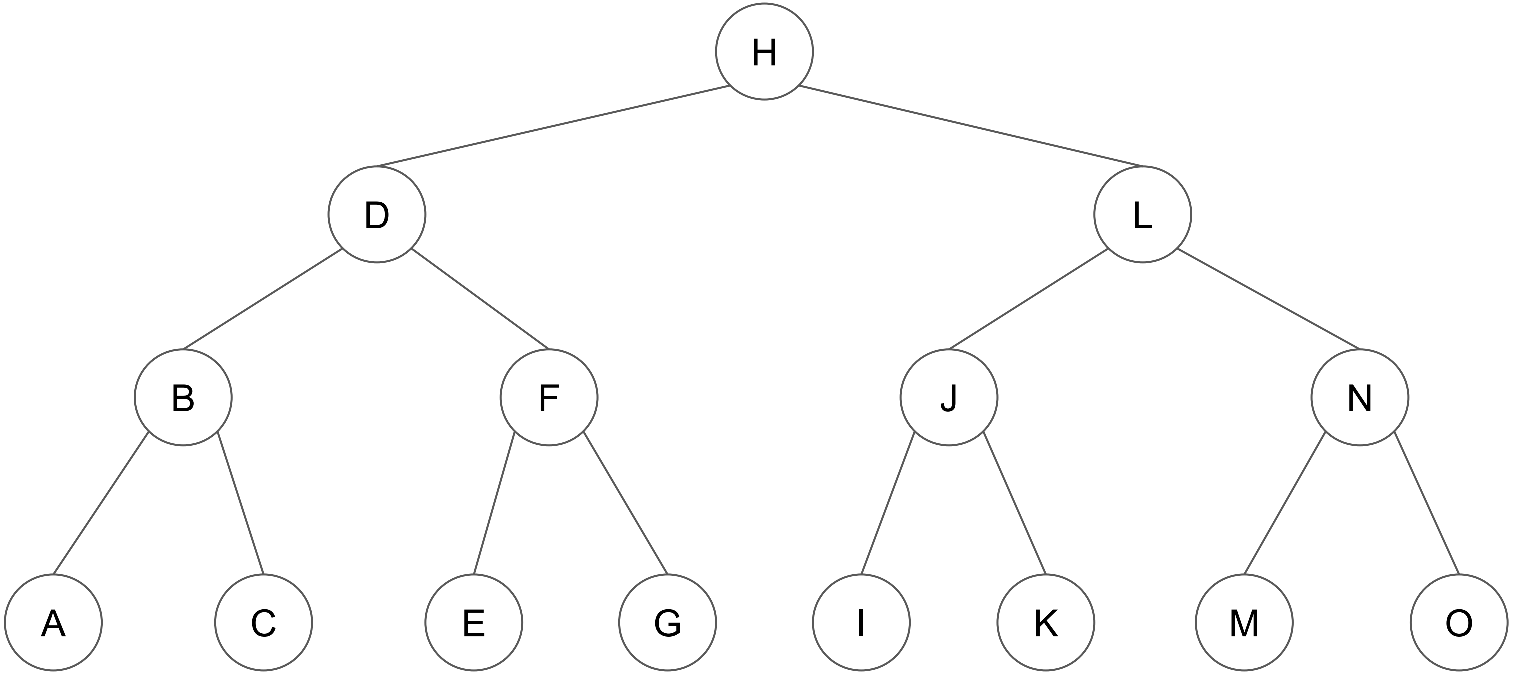
binary tree



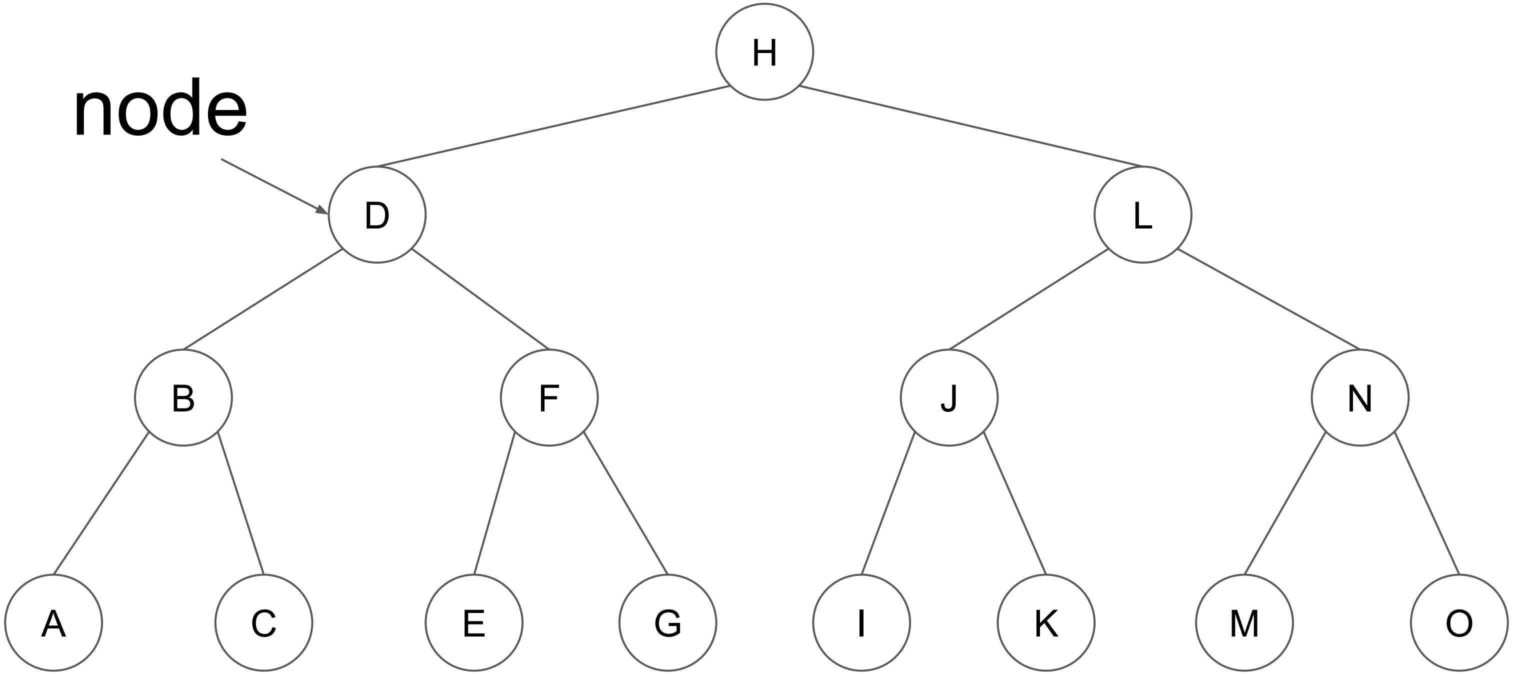
ternary tree



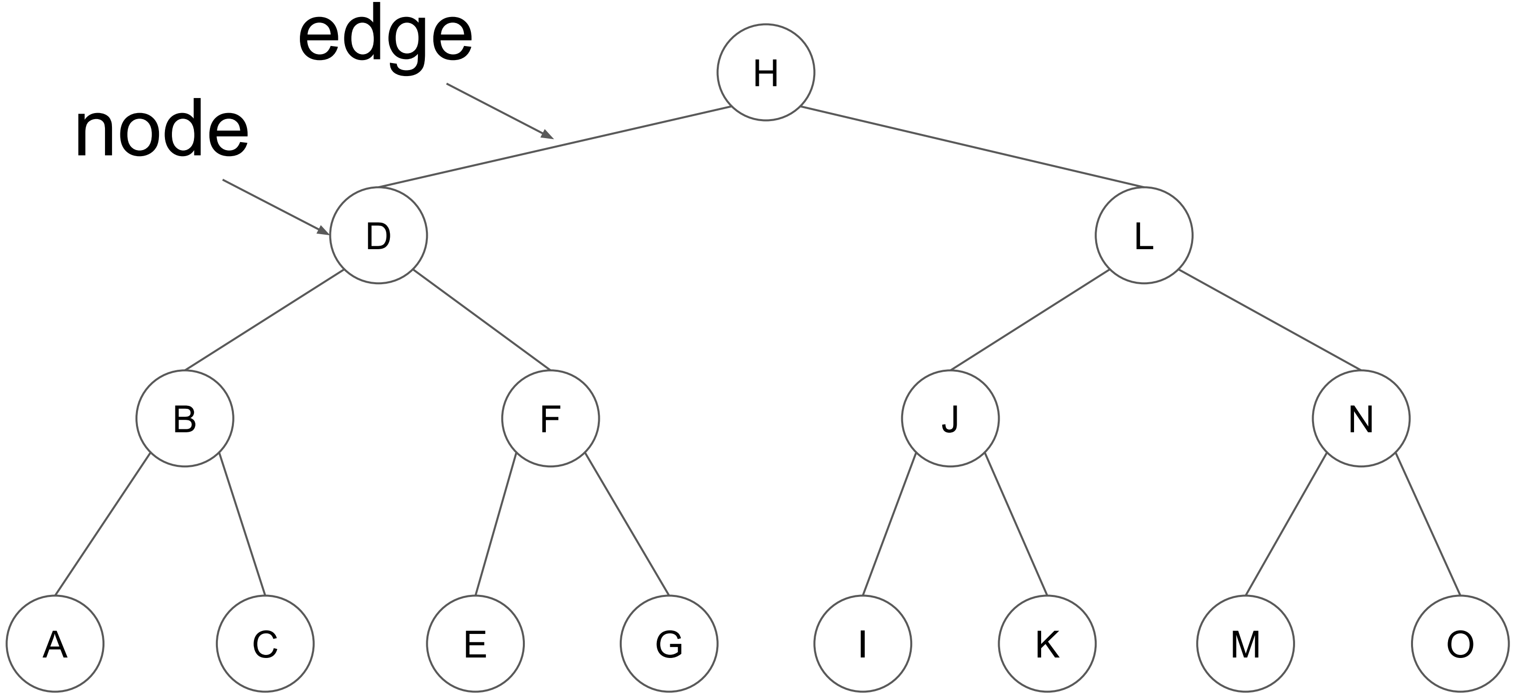
binary tree



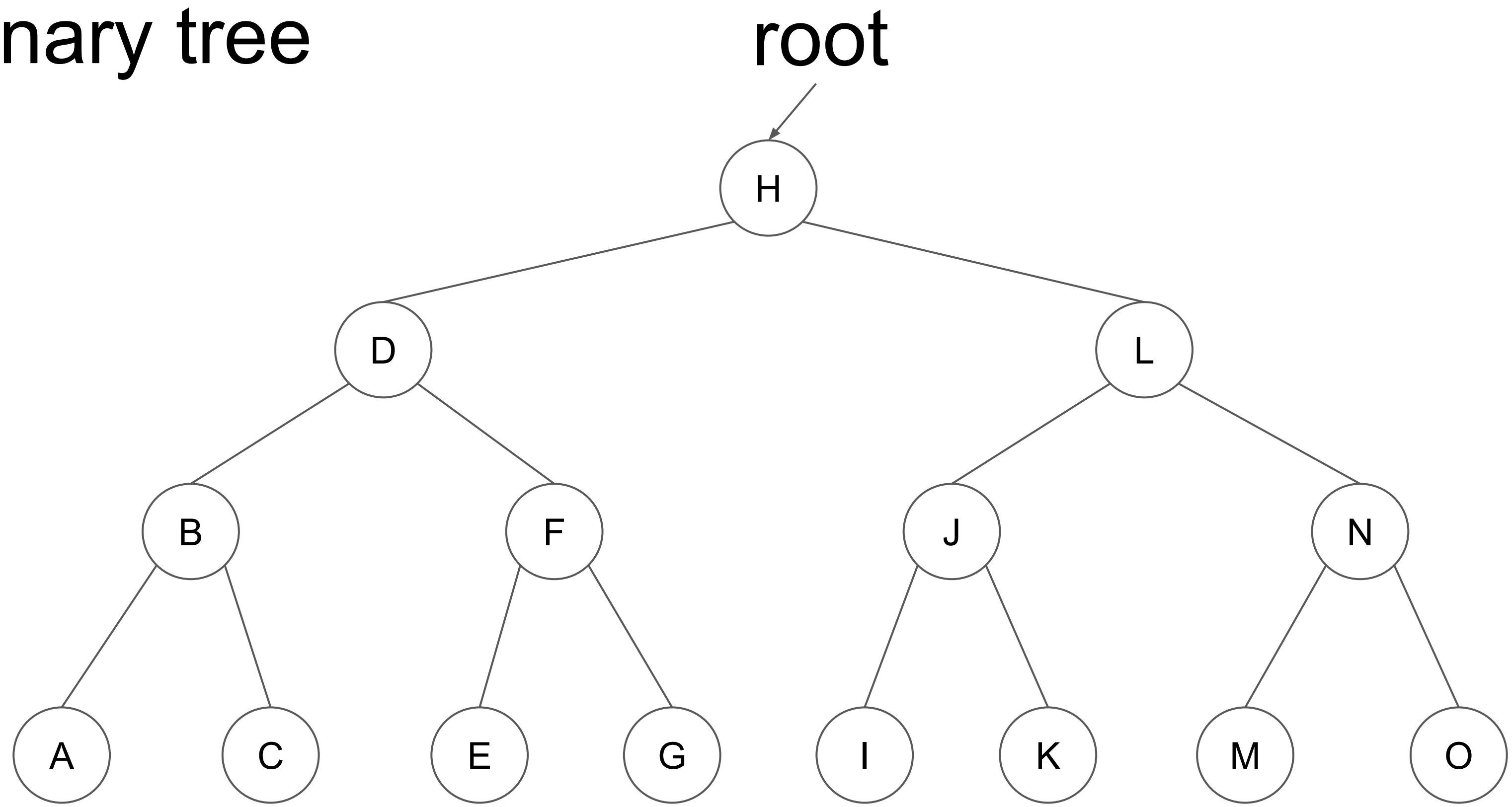
binary tree



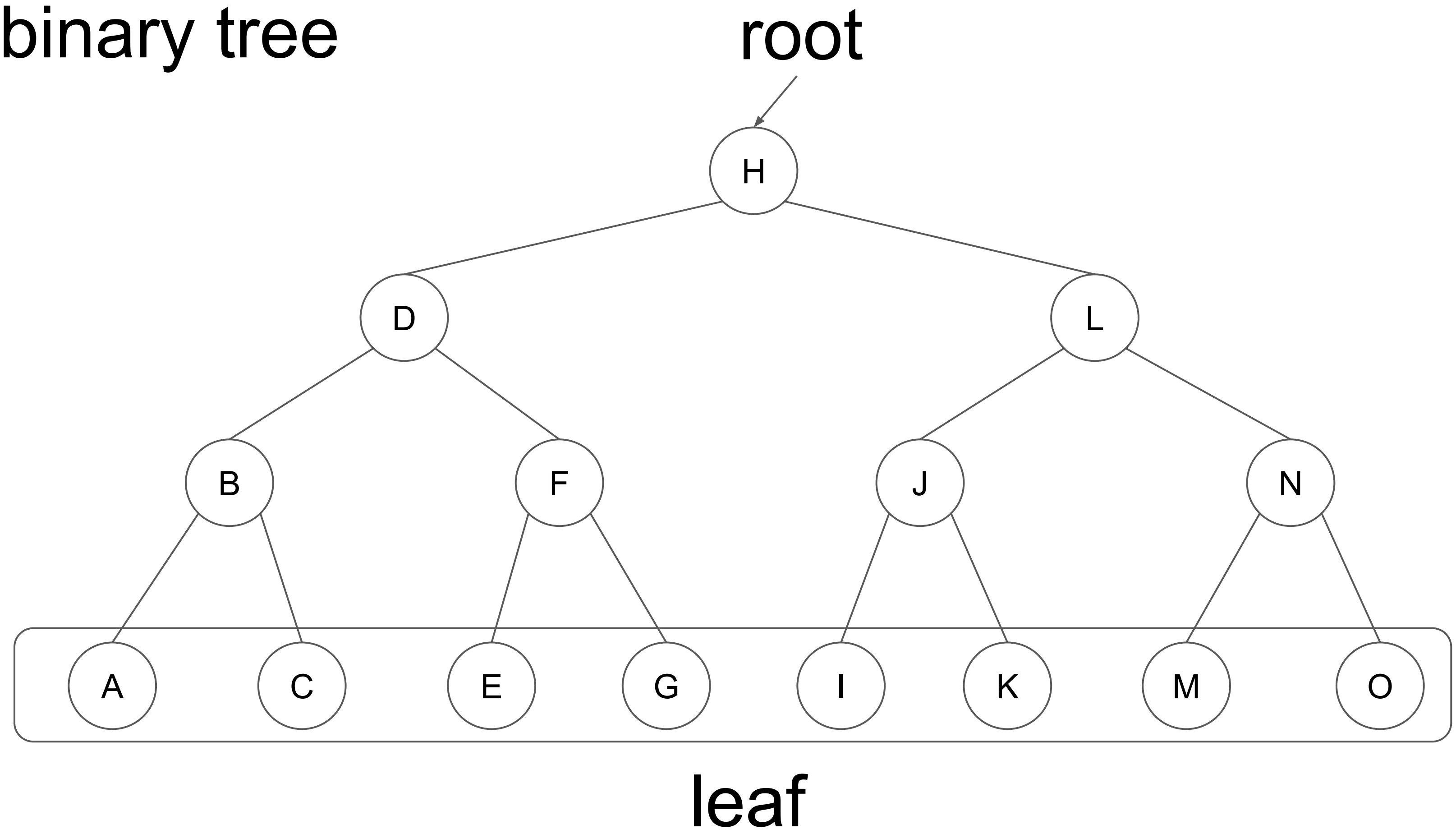
binary tree

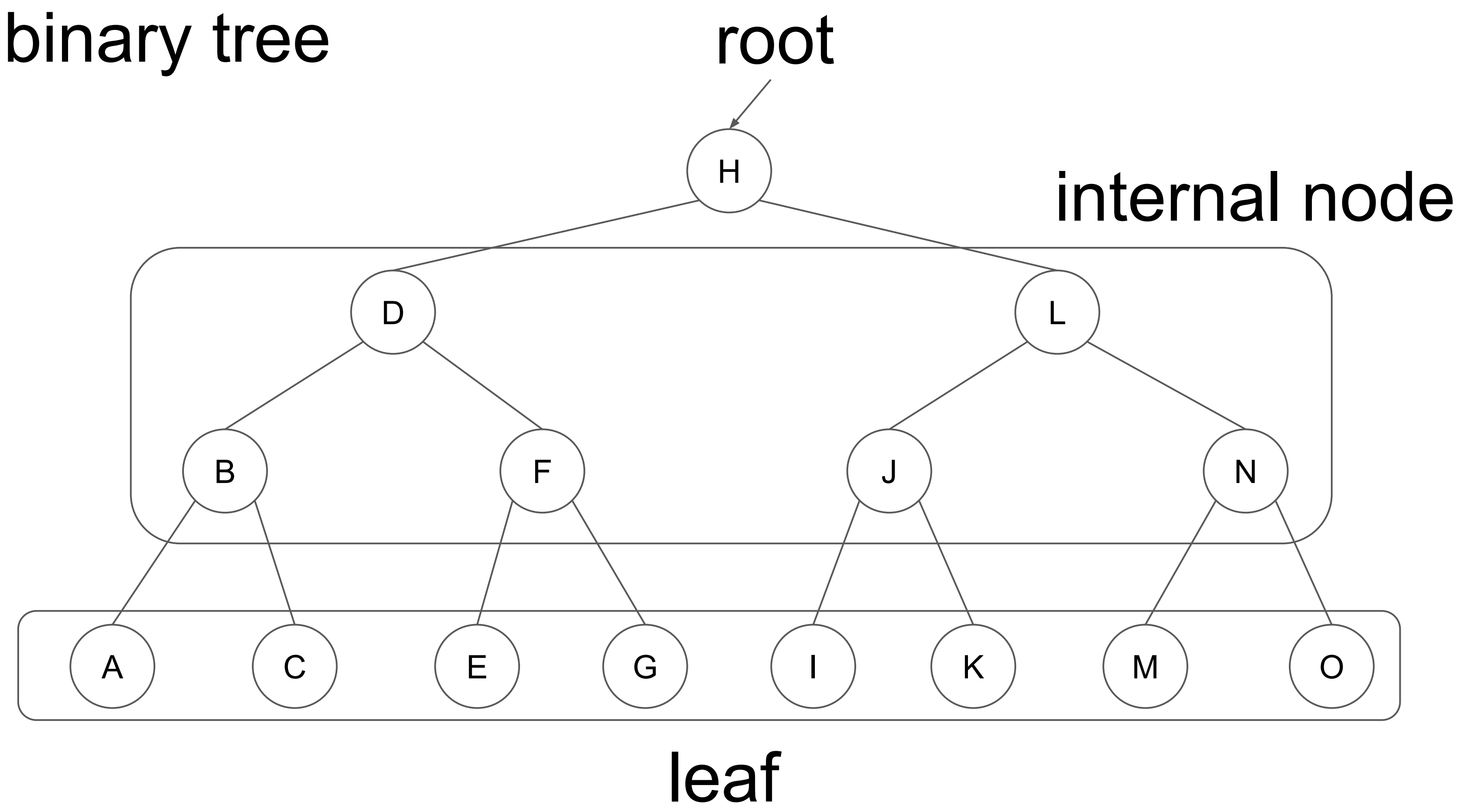


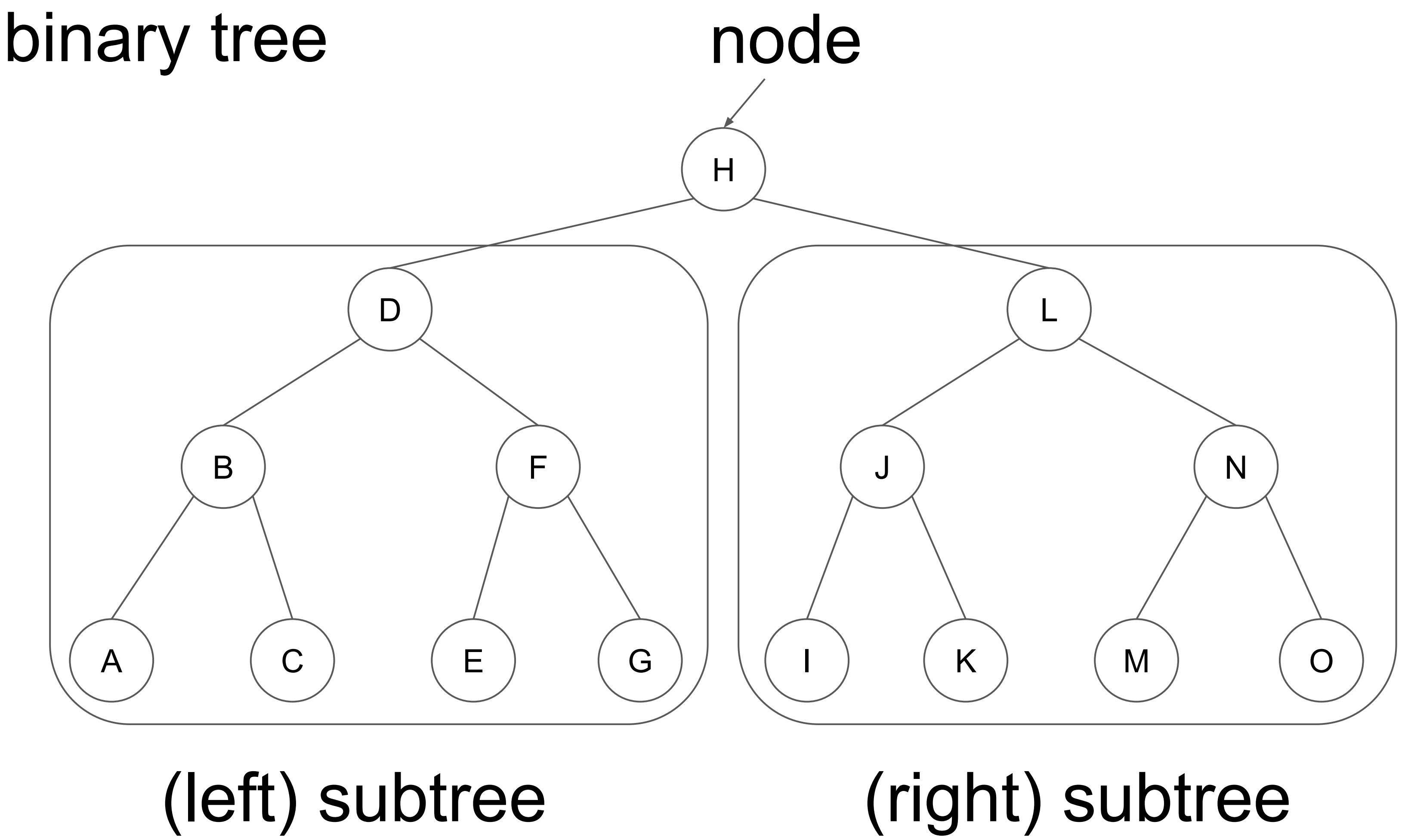
binary tree

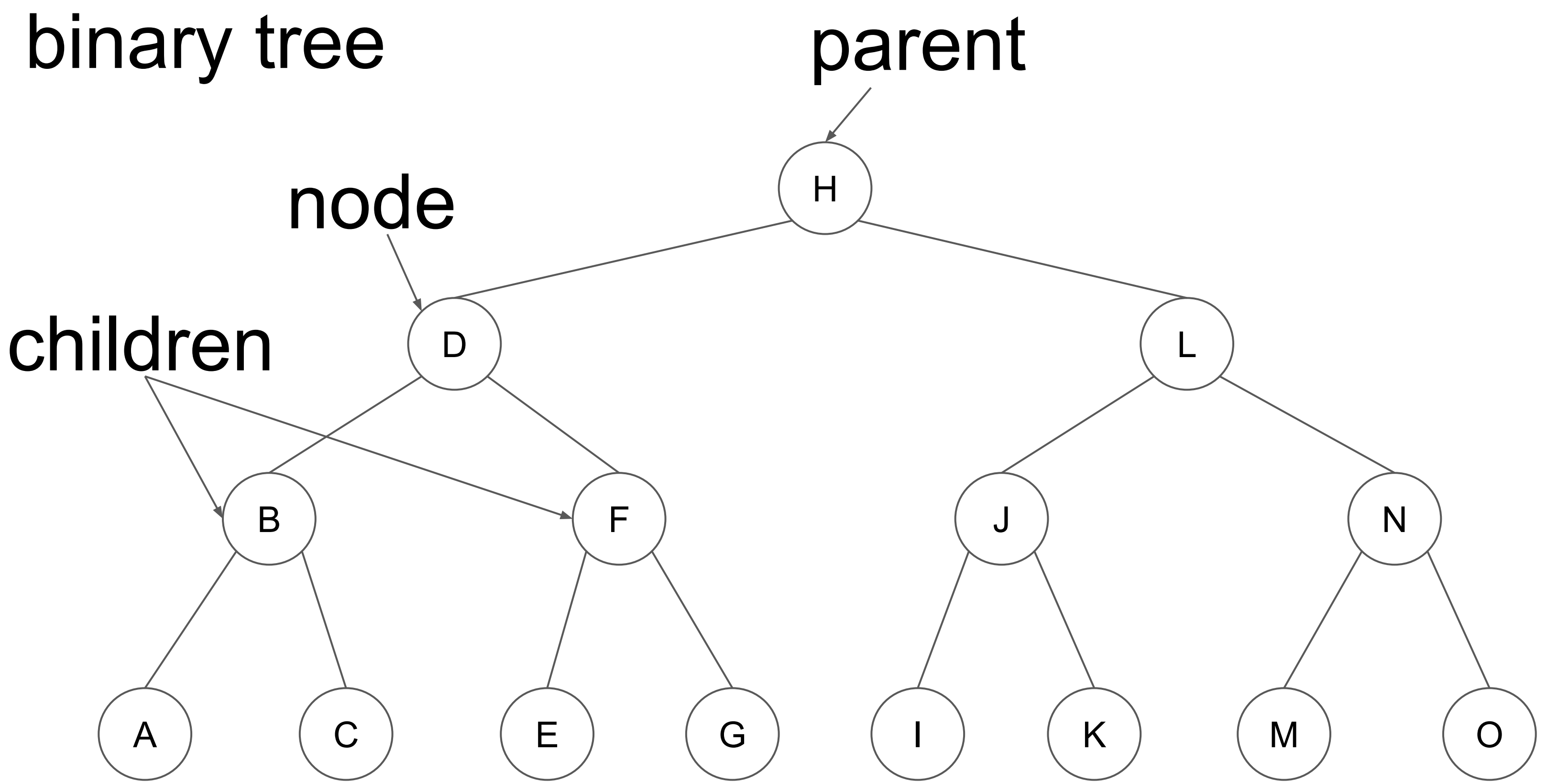


binary tree

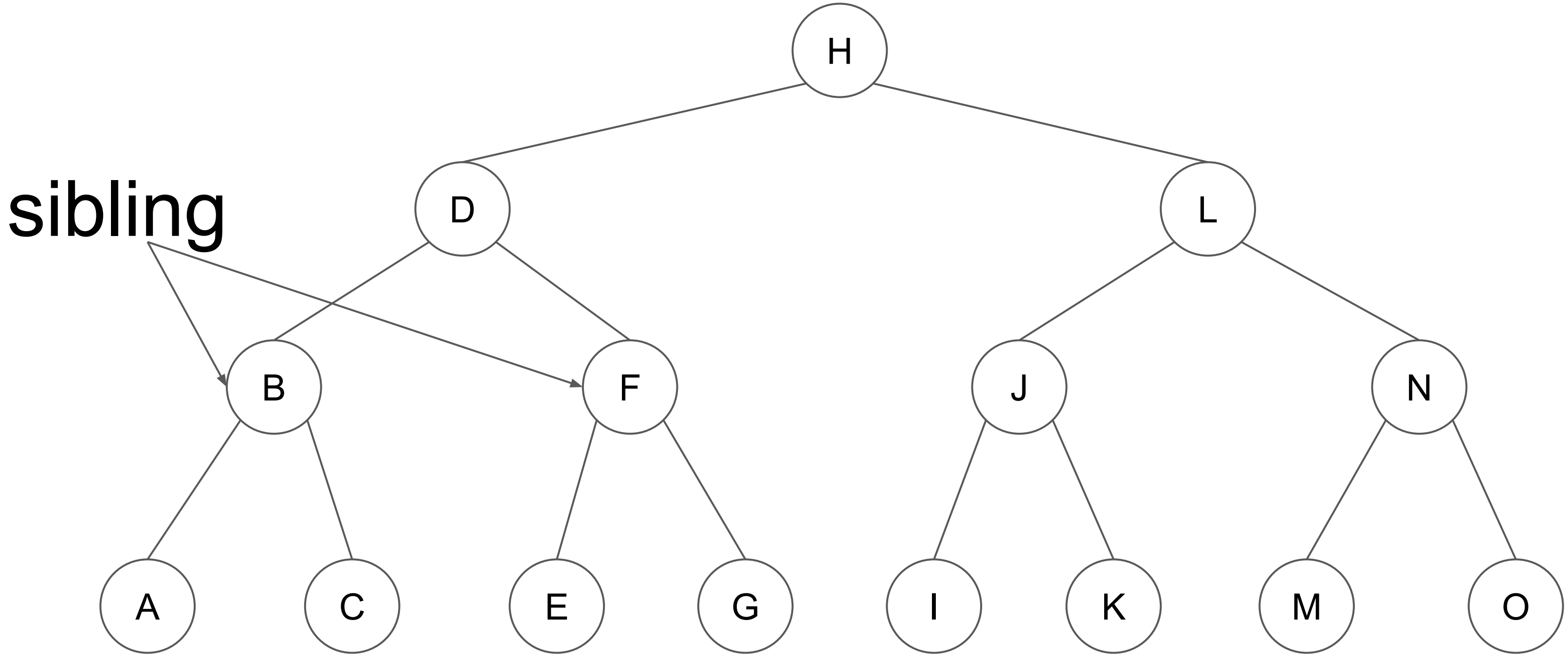



binary tree
root
H
internal node
D
L
B
F
J
N
A
C
E
G
I
K
M
O
leaf

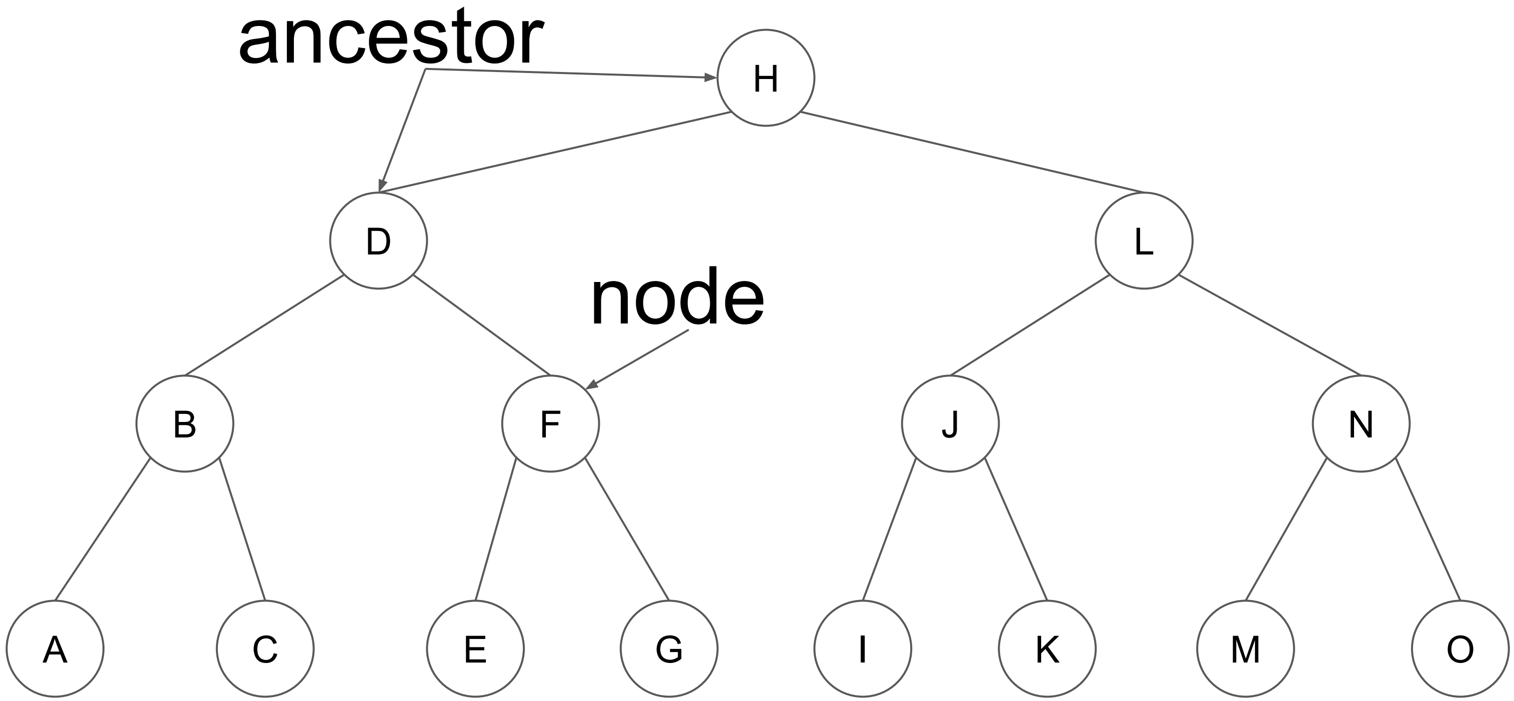

binary tree
node
H
D
L
B
F
J
N
A
C
E
G
I
K
M
O
(left) subtree
(right) subtree


binary tree
parent
node
children
H
D
L
B
F
J
N
A
C
E
G
I
K
M
O

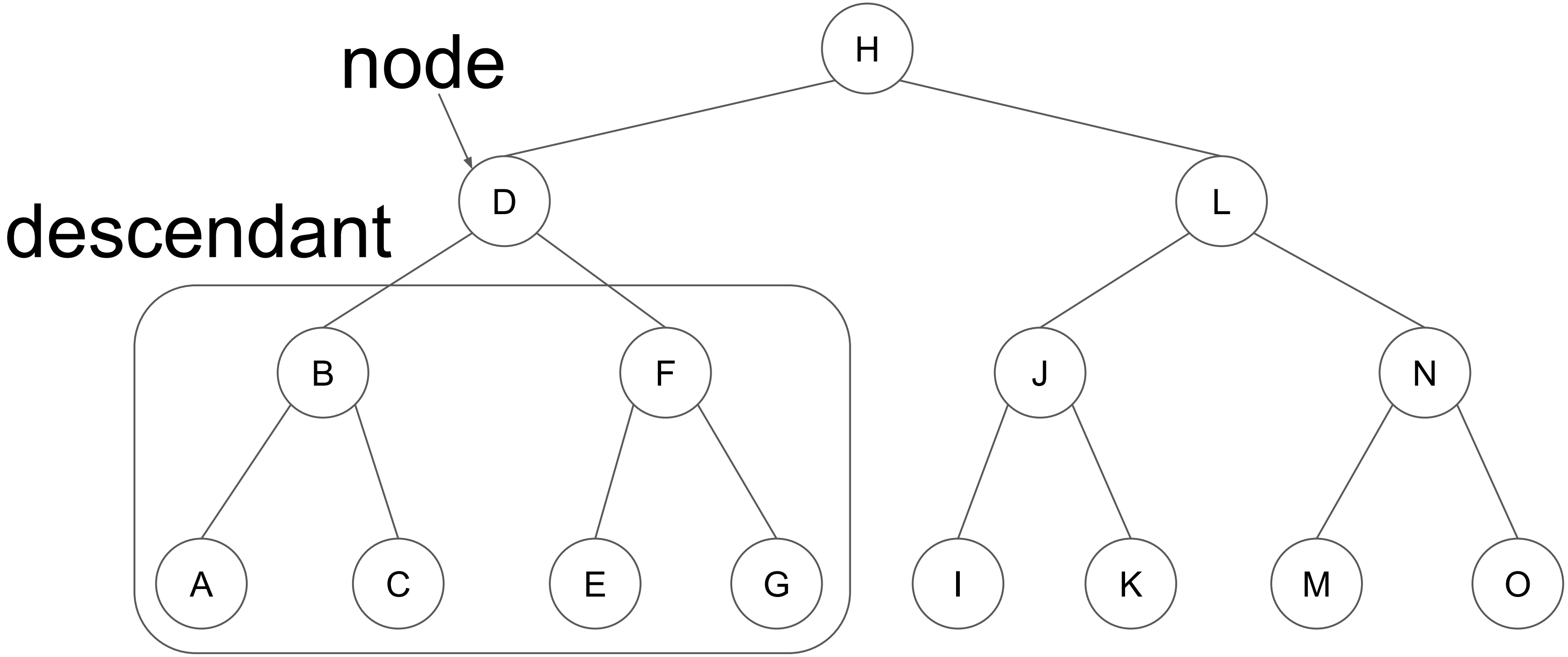
binary tree



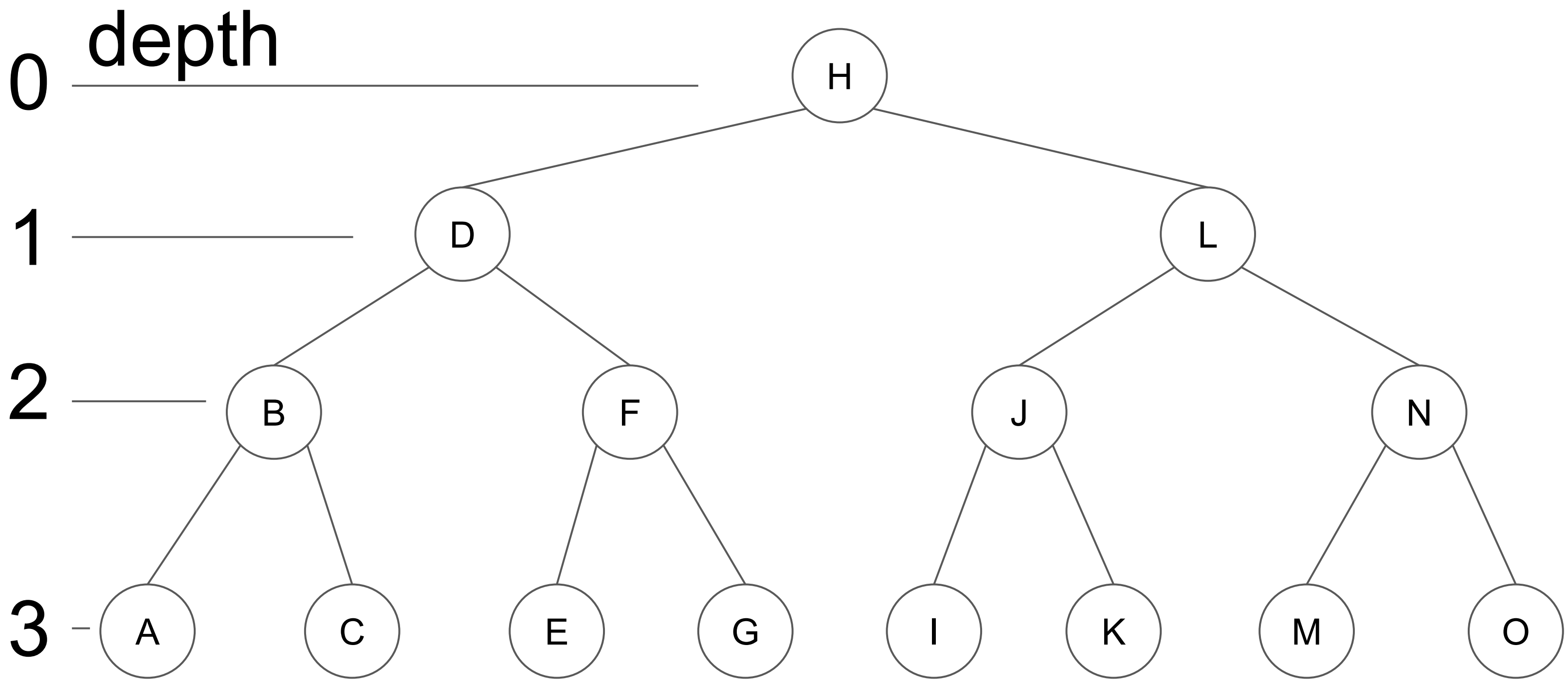
binary tree



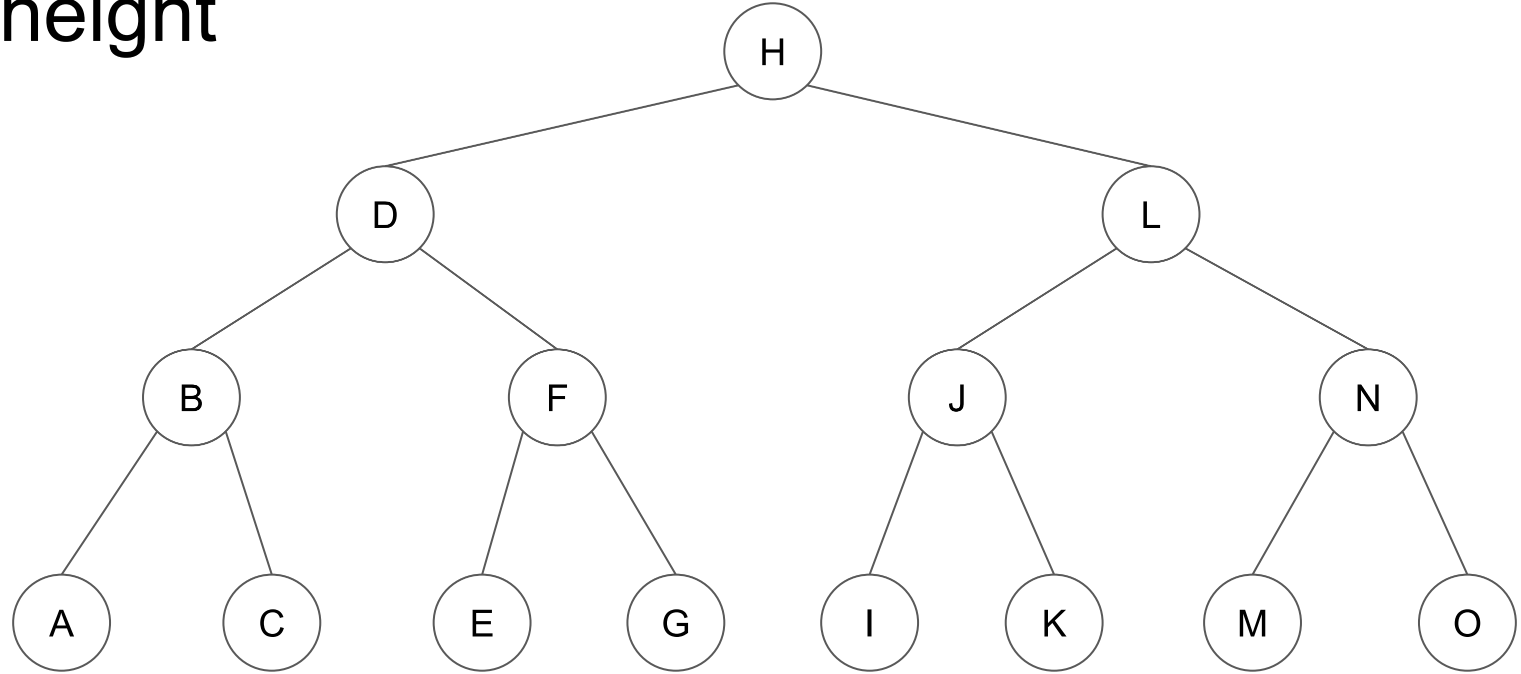
binary tree



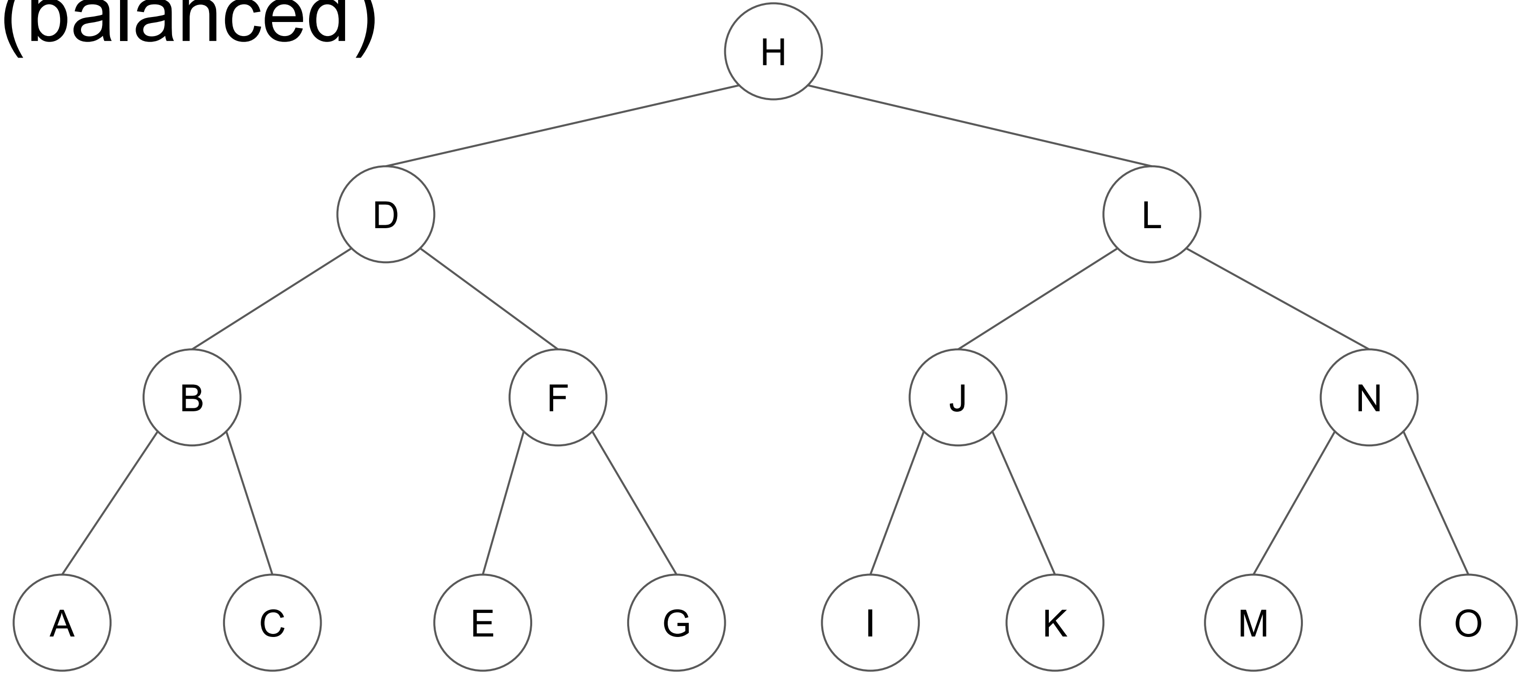
binary tree



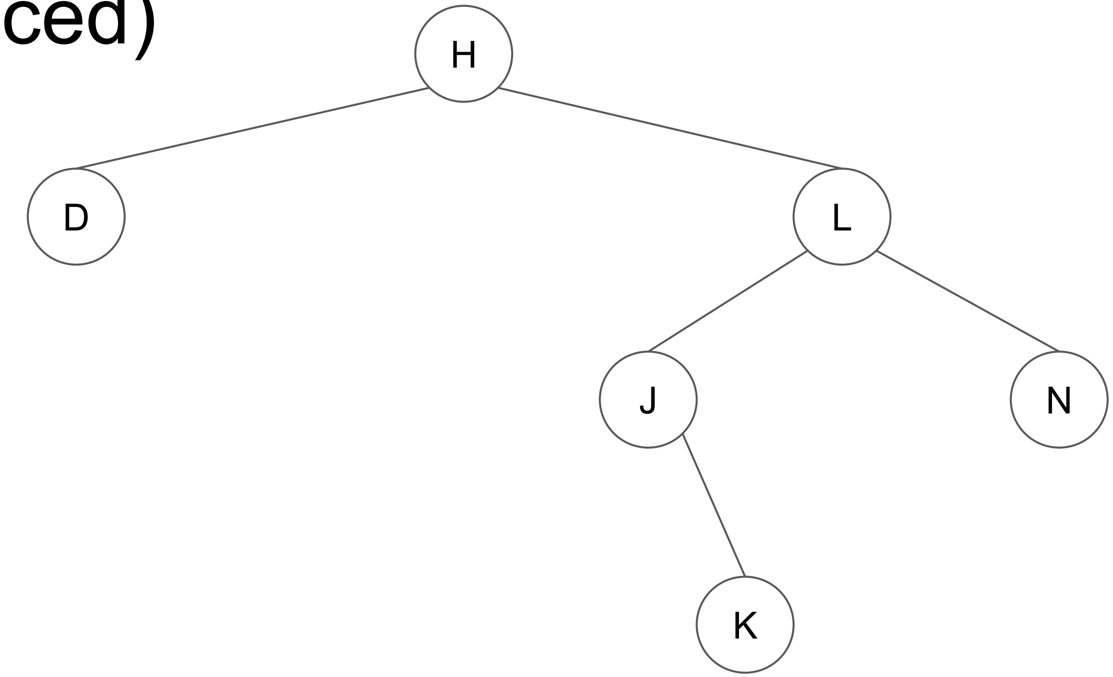
binary tree height



binary tree (balanced)

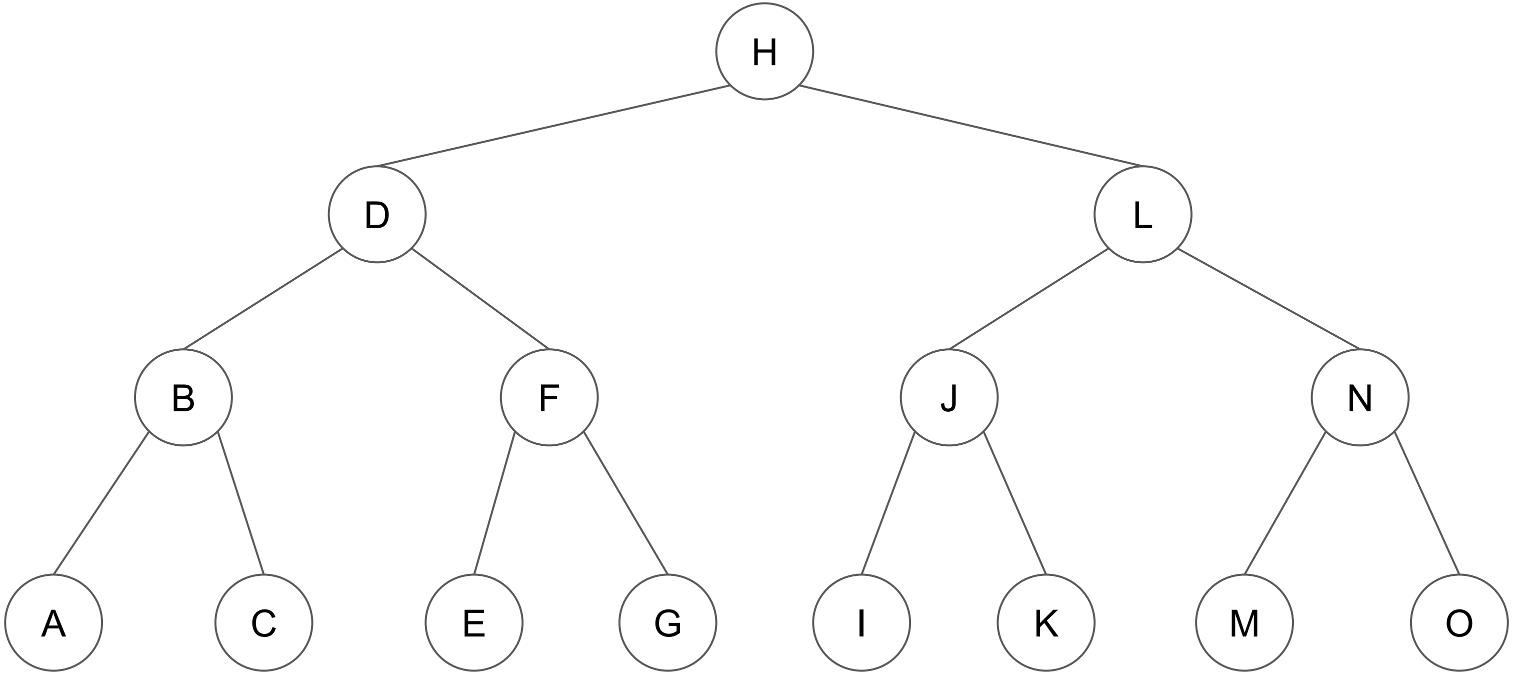


binary tree
(unbalanced)

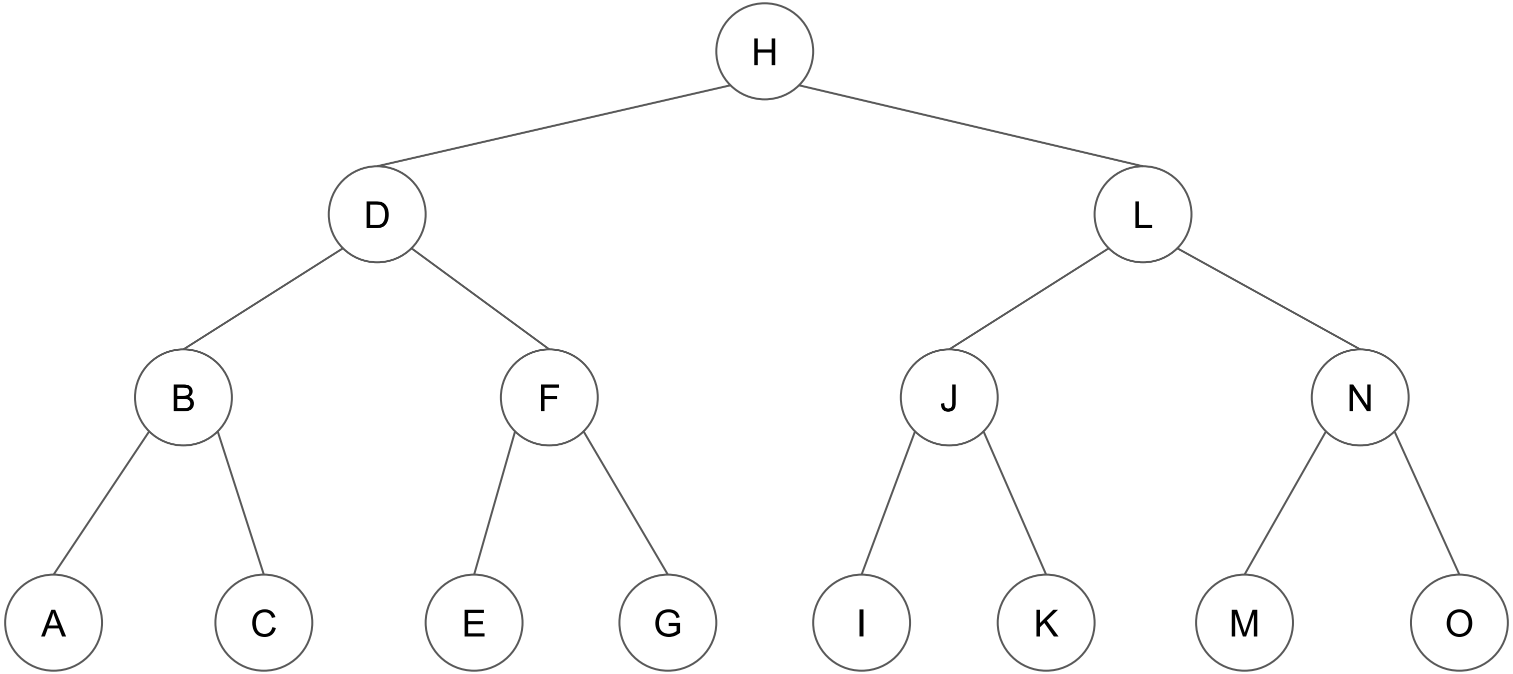


Tree traversals

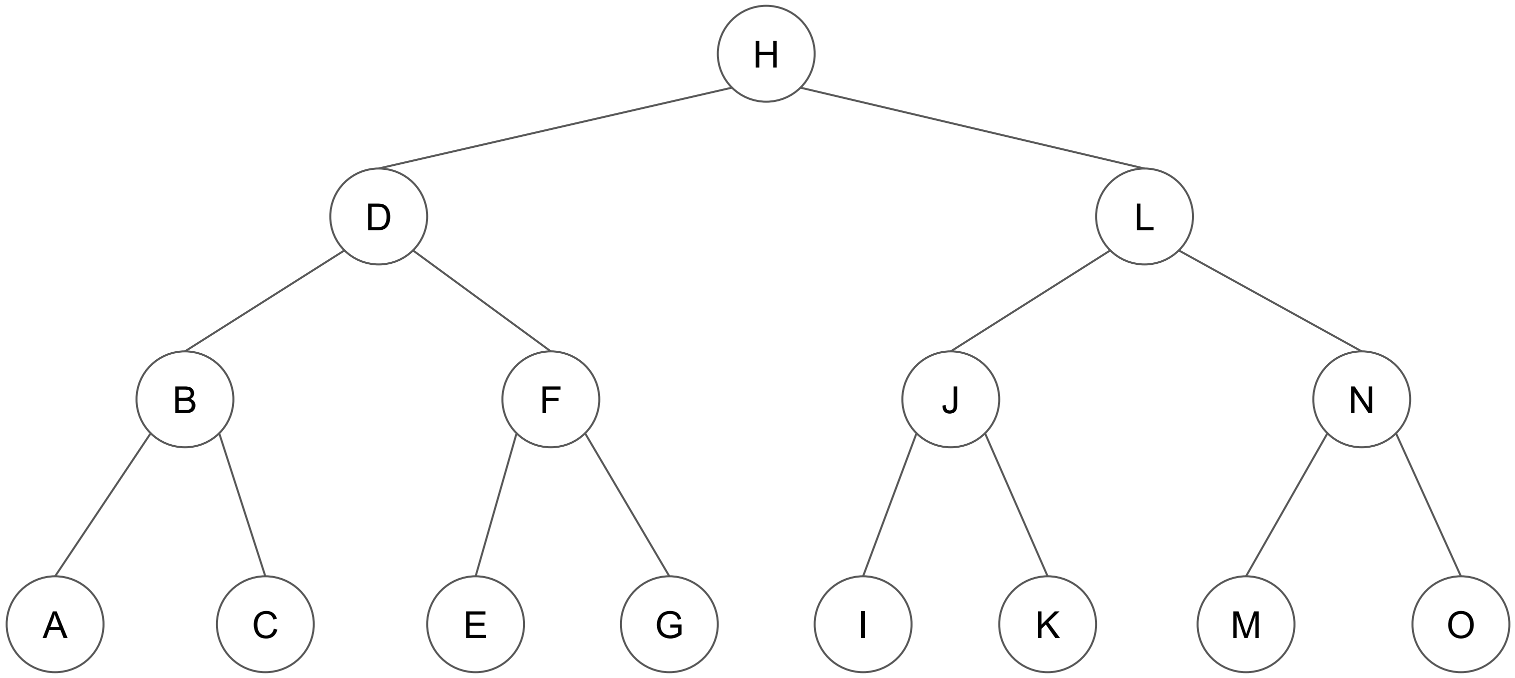
Pre-order traversal


H
D
L
B
F
J
N
A
C
E
G
I
K
M
O

In-order traversal

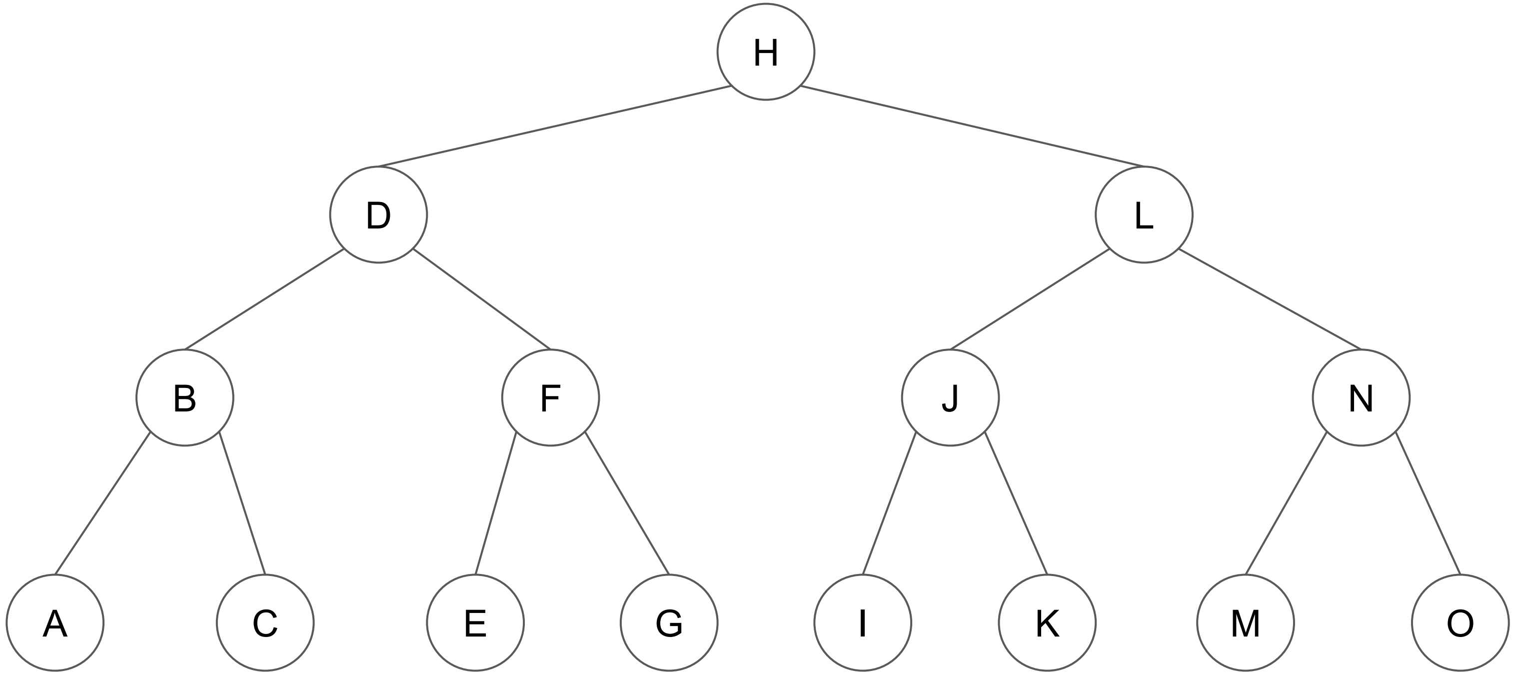

H
D
L
B
F
J
N
A
C
E
G
I
K
M
O

Post-order traversal


H
D
L
B
F
J
N
A
C
E
G
I
K
M
O

Depth-First Traversal

Level-order traversal


H
D
L
B
F
J
N
A
C
E
G
I
K
M
O

Breadth-First Traversal

In-Class Activity

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:

- running time complexity of sorting algorithms
- worst-case, best-case, average-case
- characteristics of input arrays
- in-place
- stable
- pivot selection, randomly, median-of-3
- counting sort
- bounded-universe
- tree
- hierarchical structure
- abstract syntax tree (AST)
- binary tree, ternary tree
- node, edge, root, leaf, internal node, subtree, children, sibling, ancestor, descendant, depth, height, balanced, unbalanced
- pre-order, in-order, post-order (, level-order) tree traversal

To the lab!