Code Fellows

Class 13 + DSA 03
Access Control and Trees

seattle-javascript-401n14

Code Challenge 12 Review

Vocab Review!

What is Basic Auth?

What is Bearer Auth?

What is Access Control?

What is a Binary Tree?

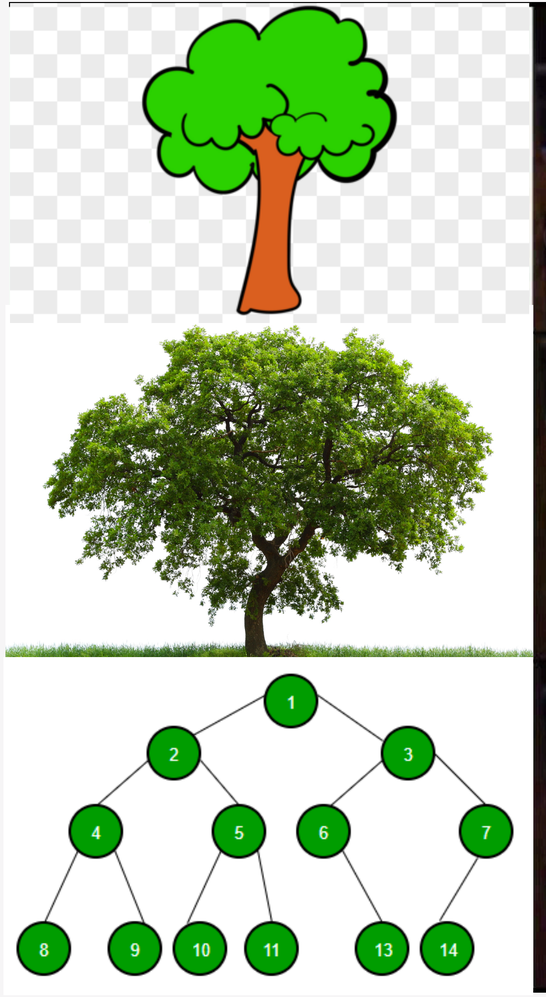
What is a K-ary Tree?

What is Depth-First Traversal?

What is Breadth-First Traversal?

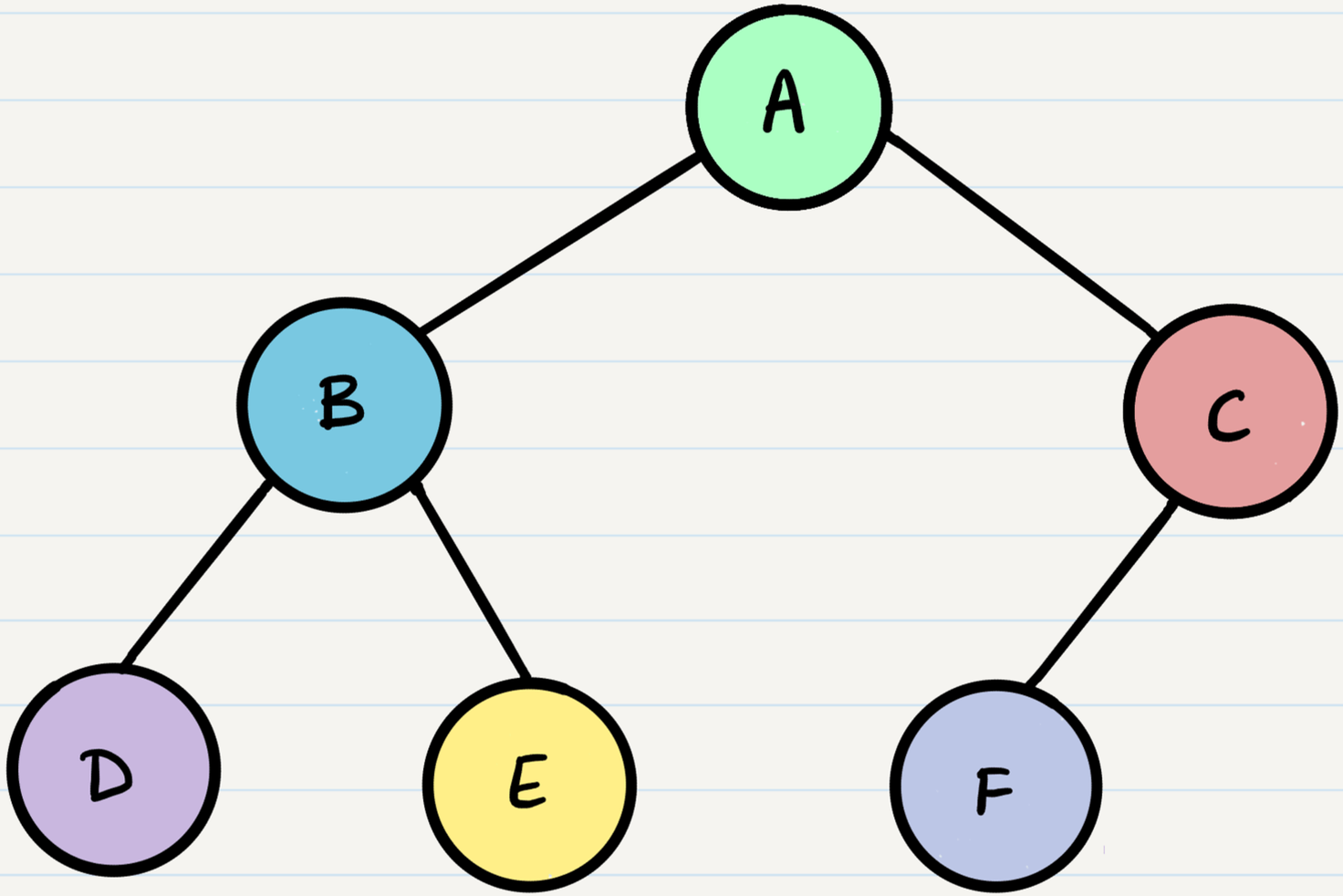
Trees

- The **Tree** data structure consists of parent and children nodes
- A parent can have any number of children, and each child can have its own children as well
- The top of the tree is the **root**
- The child-less ends of the tree are the **leaves**
- We like binary trees the most!

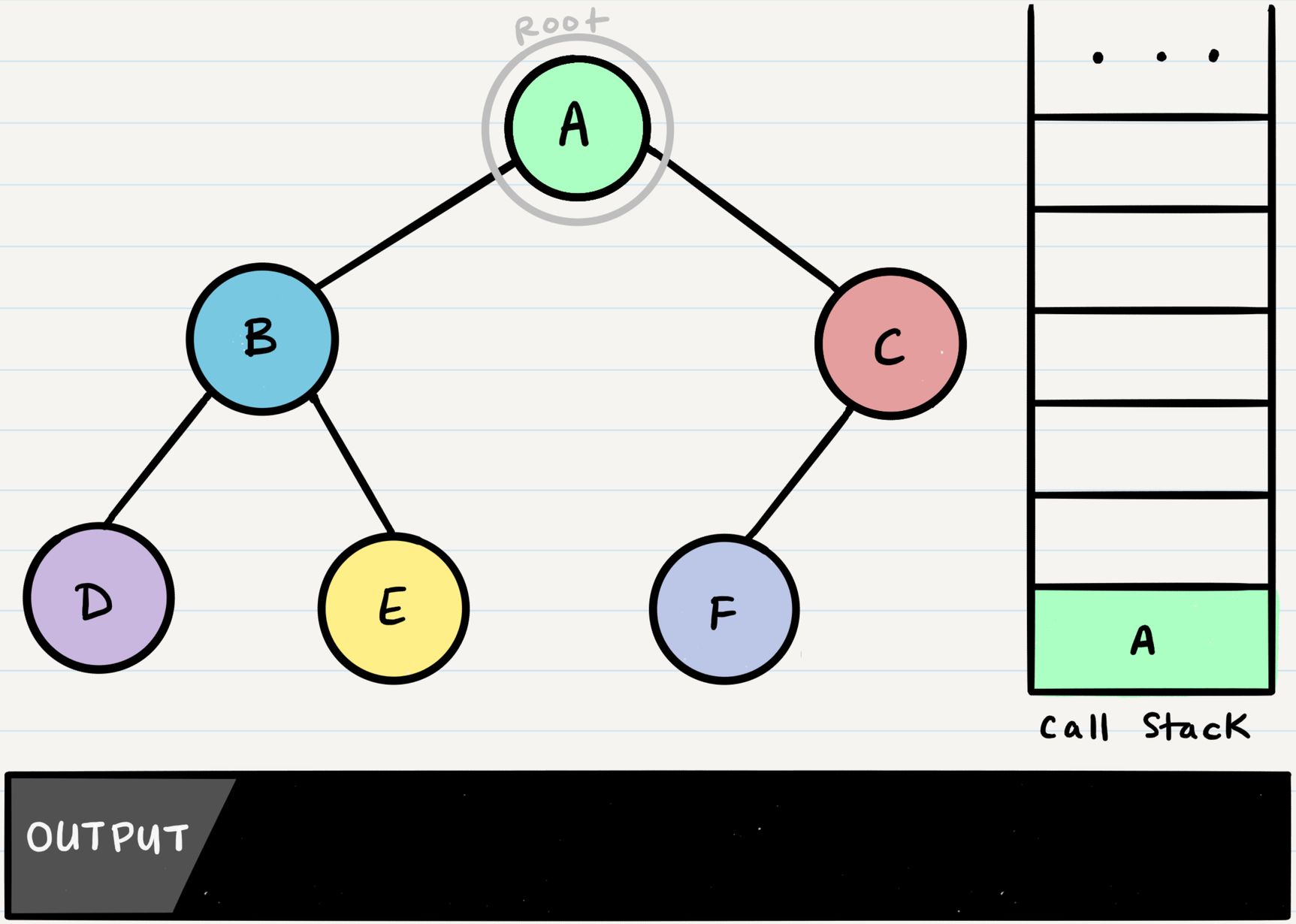


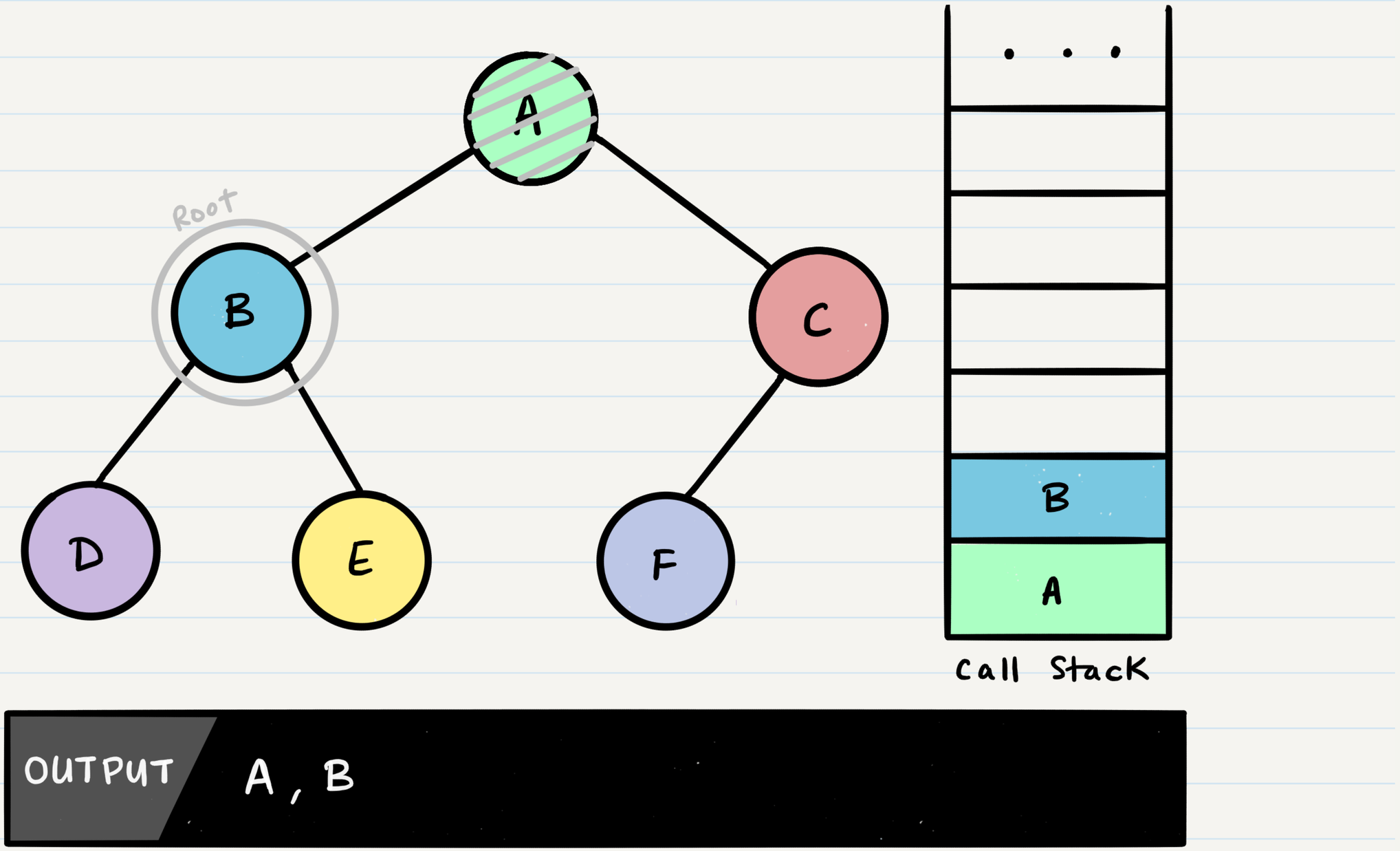
Traversal

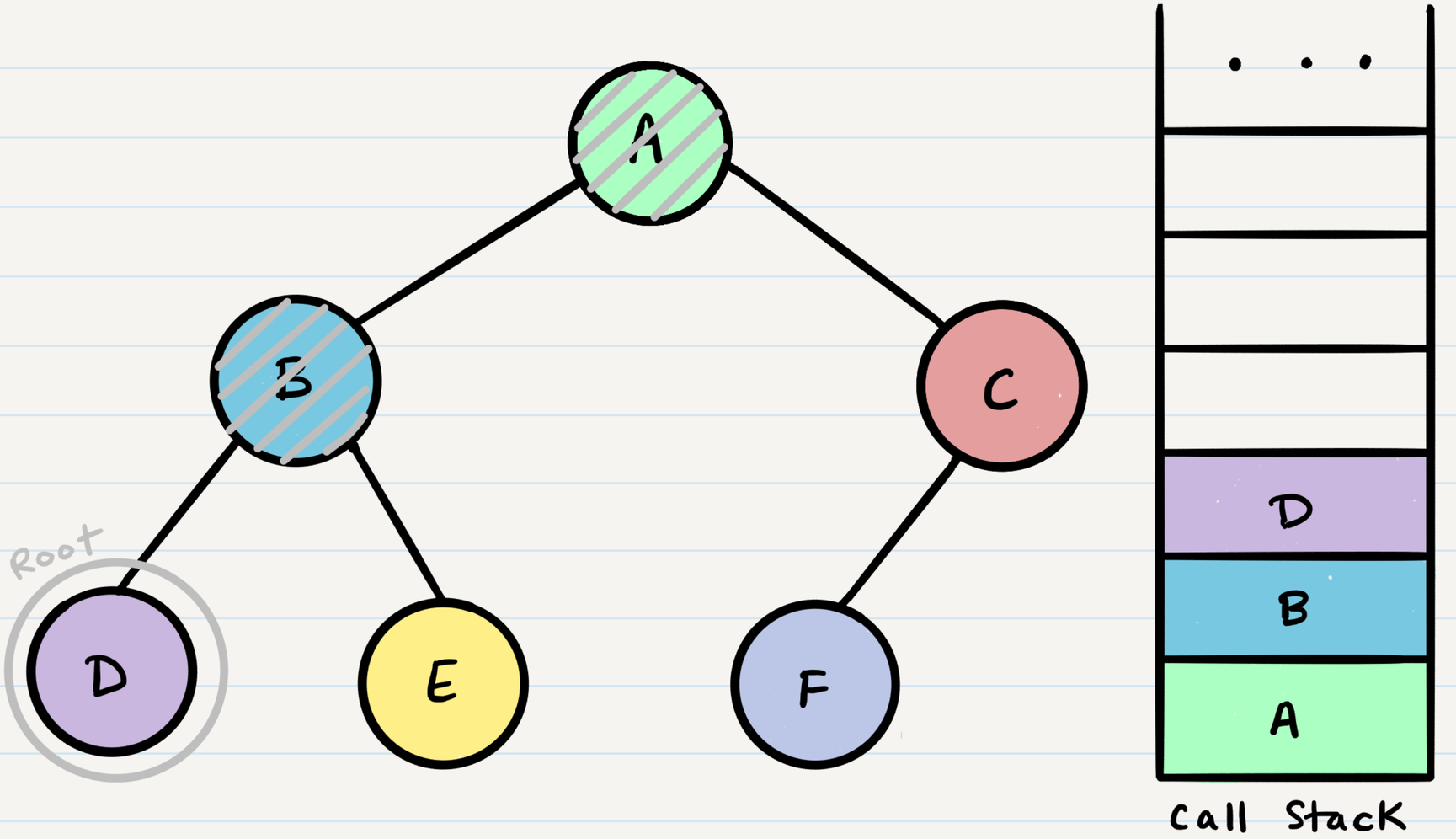
- Trees are powerful because there's multiple ways to **traverse** them
- Depth-first traversal is the most popular, and there are three types!
    - Pre-order, in-order, post-order
- Breadth-first traversal has its own benefits as well


A
B
C
D
E
F

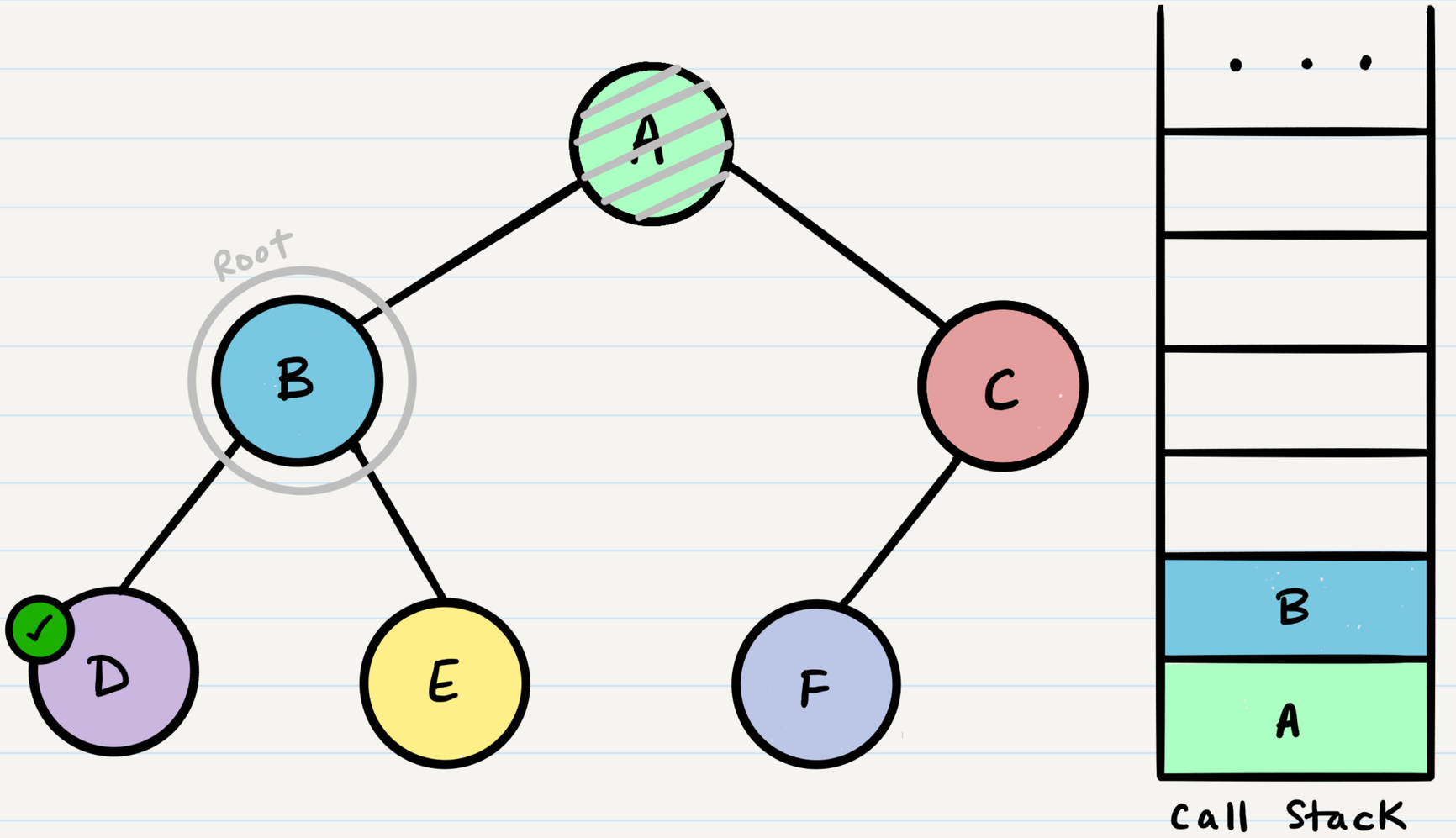
Depth-first pre-order traversal

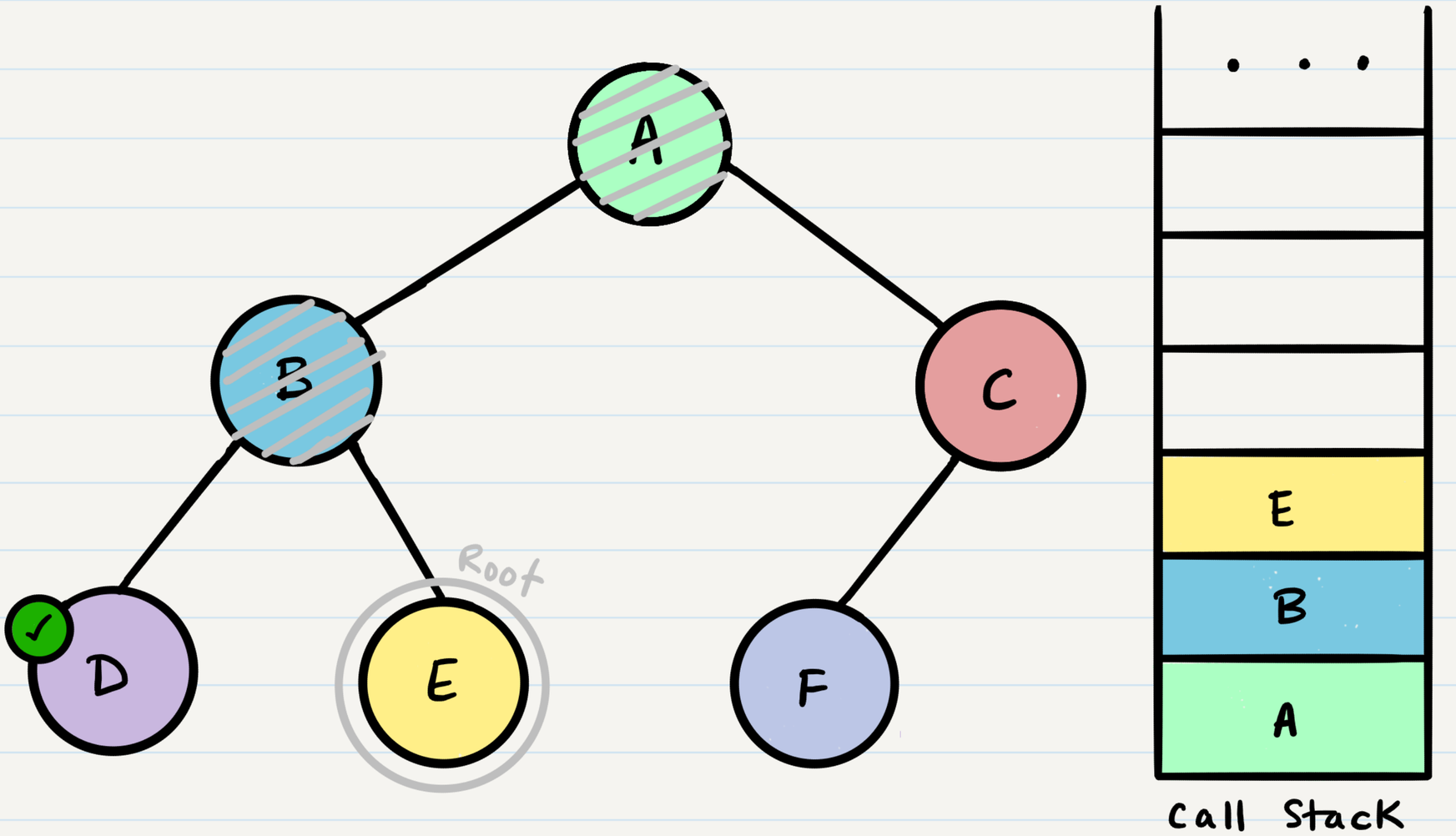

Root
A
B
C
D
E
F
. . .
A
Call Stack
OUTPUT


A
Root
B
C
D
E
F
B
A
Call Stack
OUTPUT
A , B

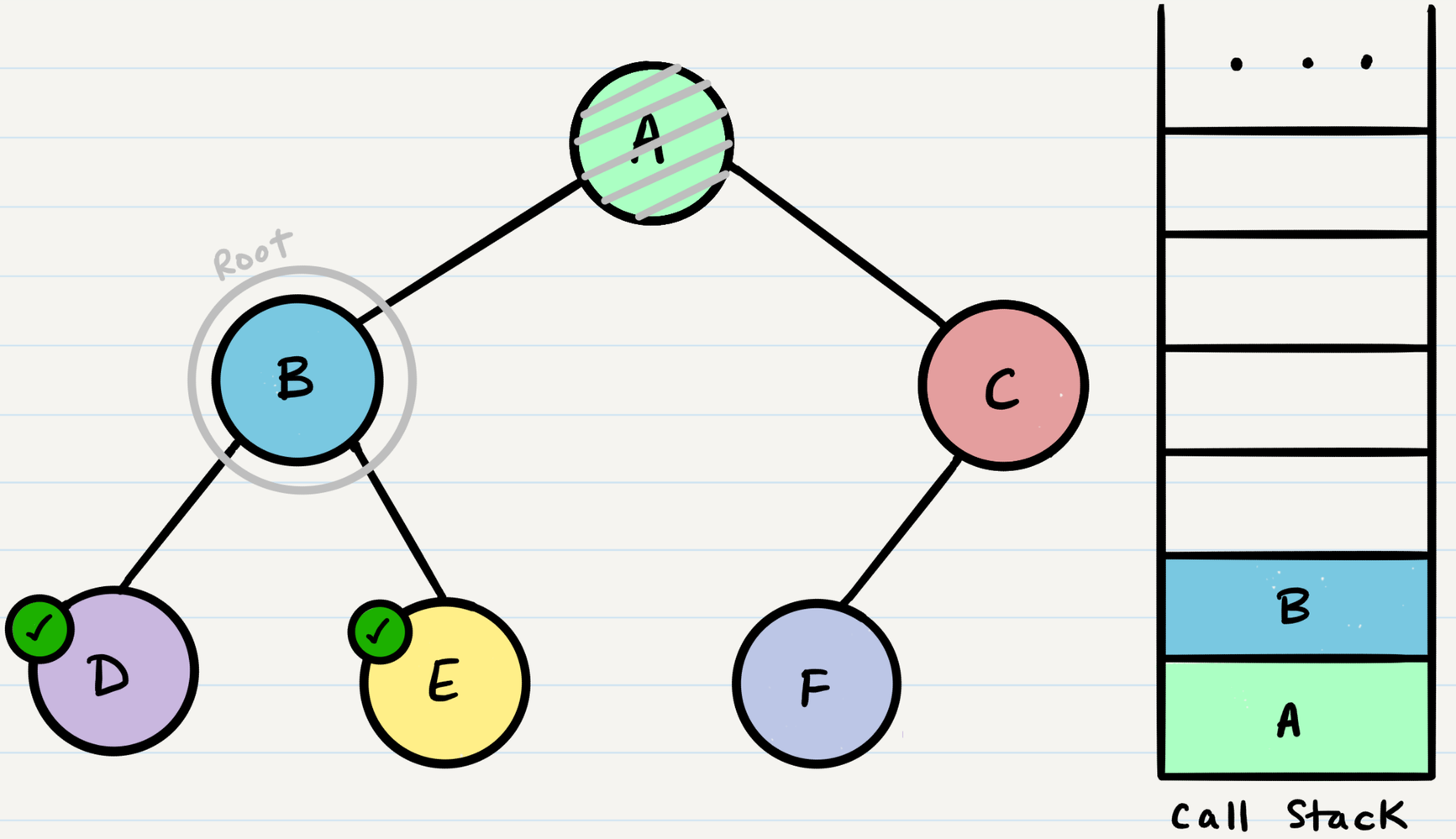


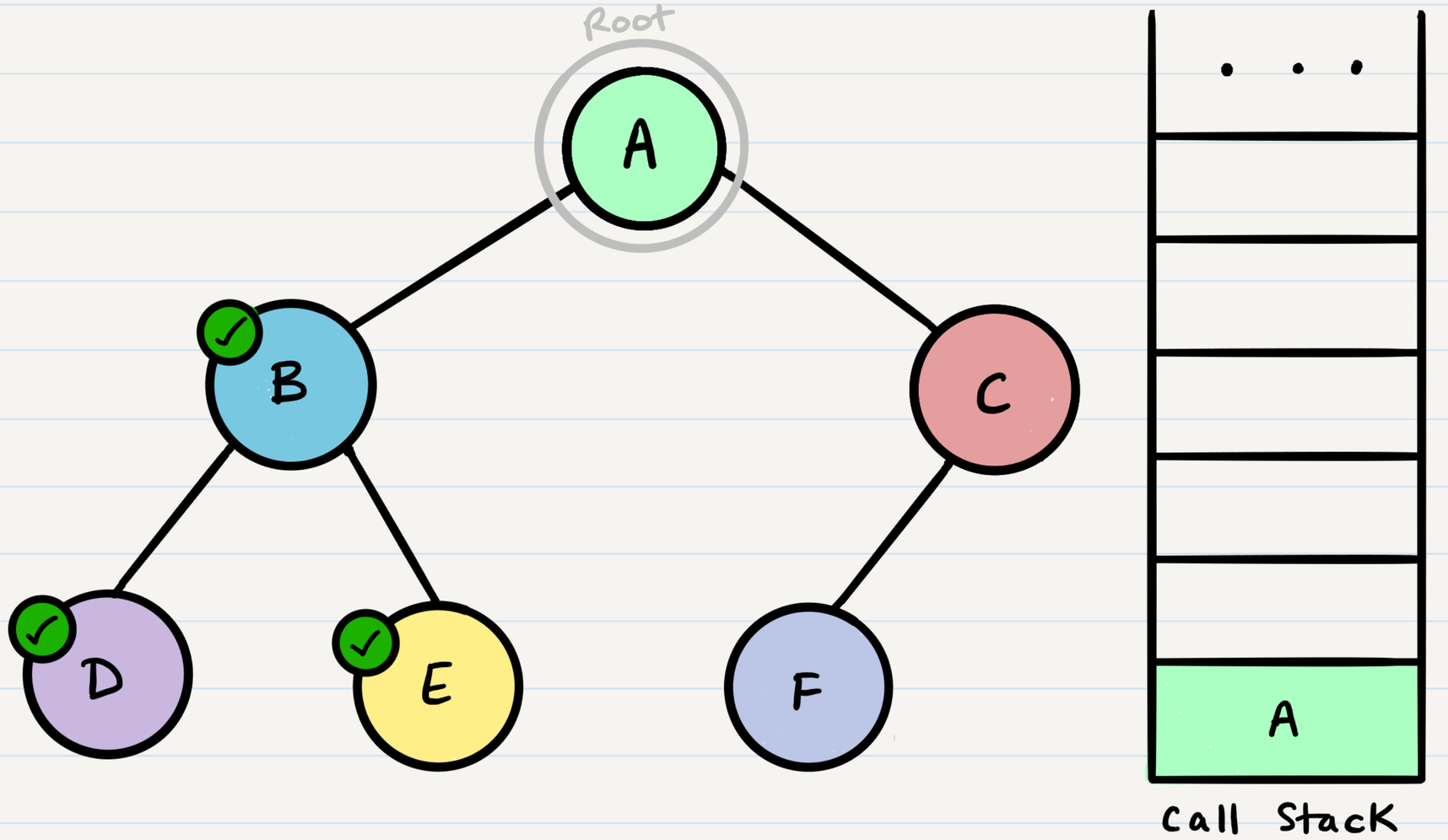
OUTPUT A, B, D


A
Root
B
C
D
E
F
B
A
Call Stack
OUTPUT
A, B, D

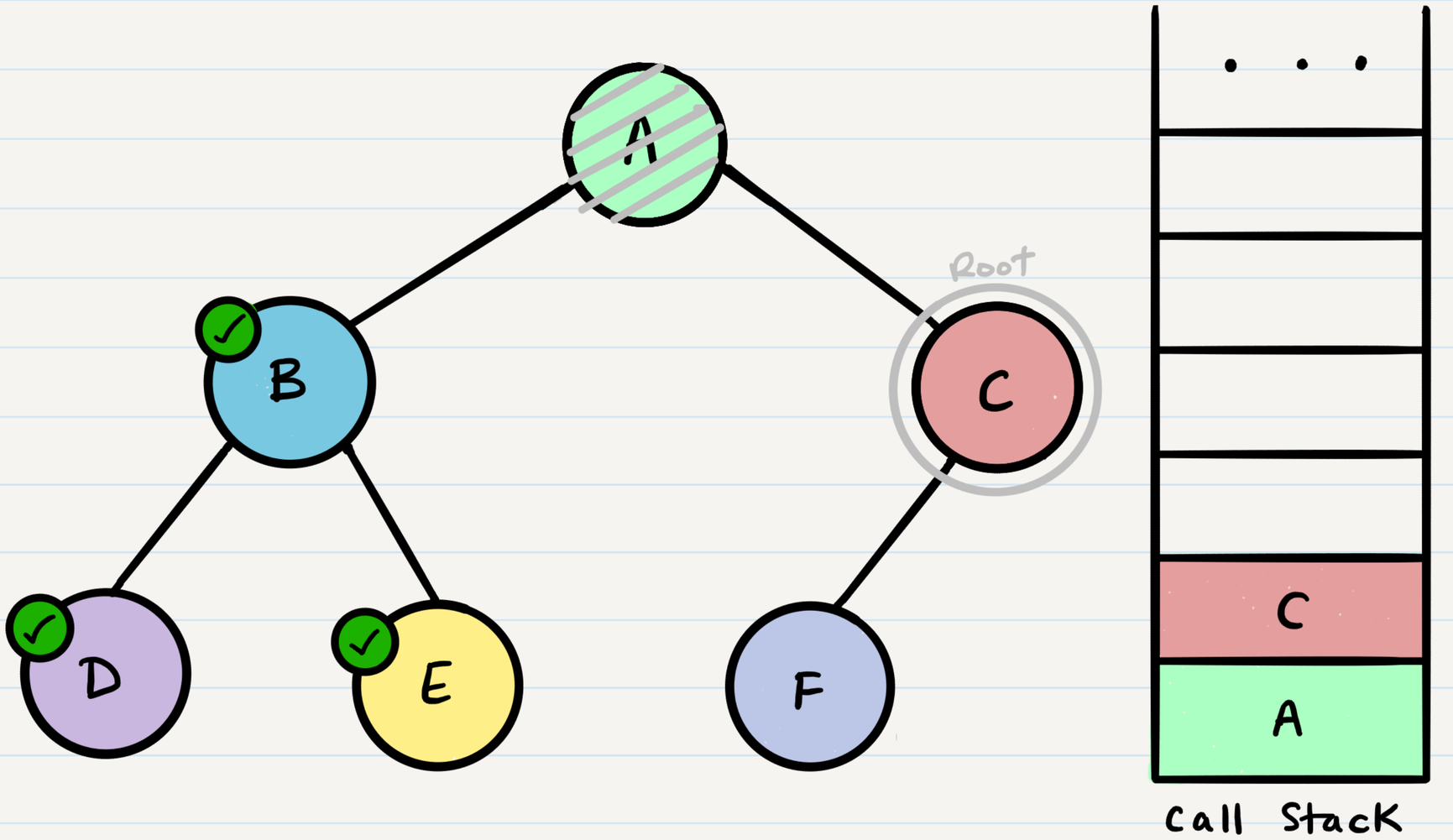


OUTPUT A, B, D, E

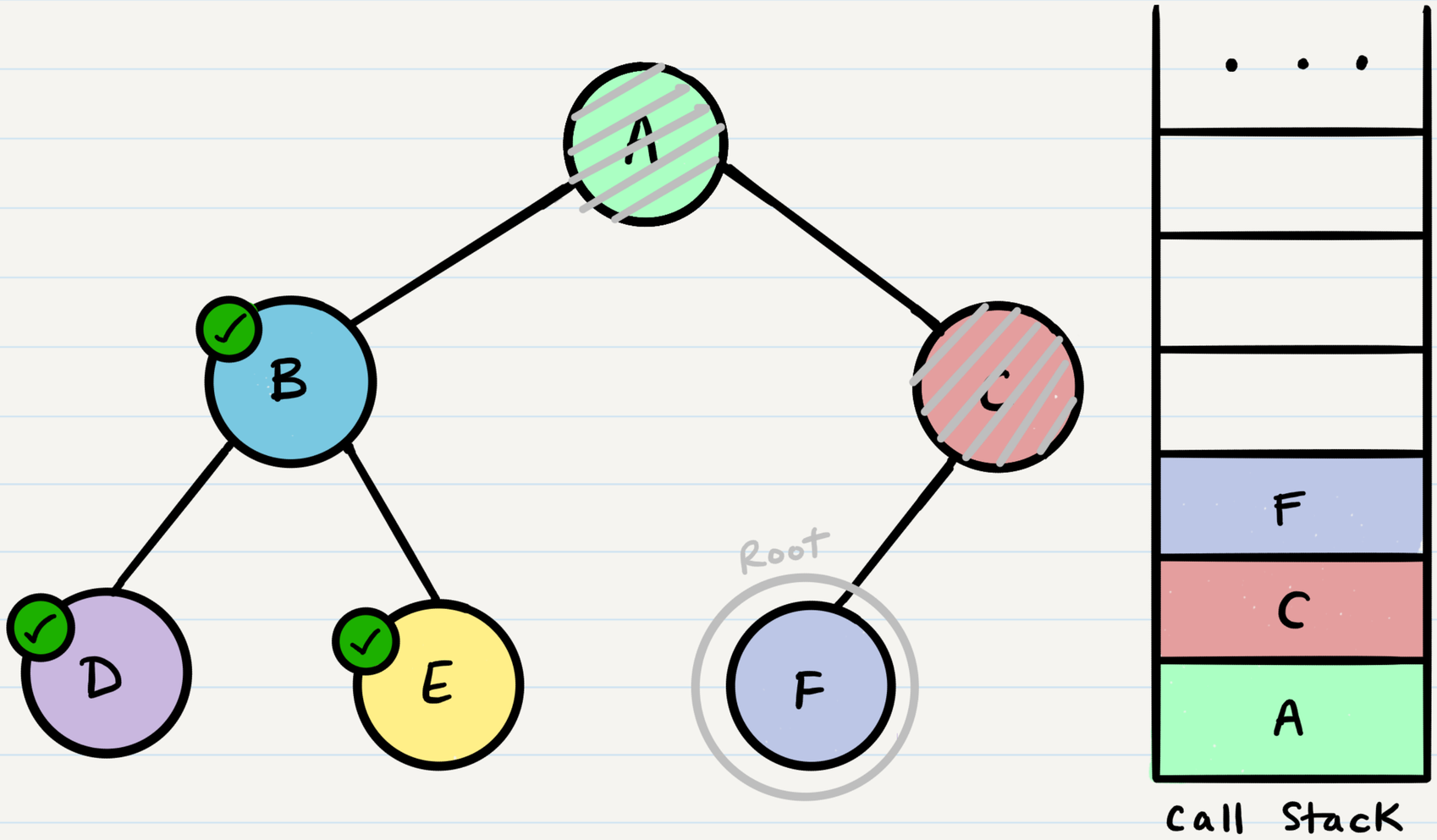

A
Root
B
C
D
E
F
B
A
Call Stack
OUTPUT
A, B, D, E



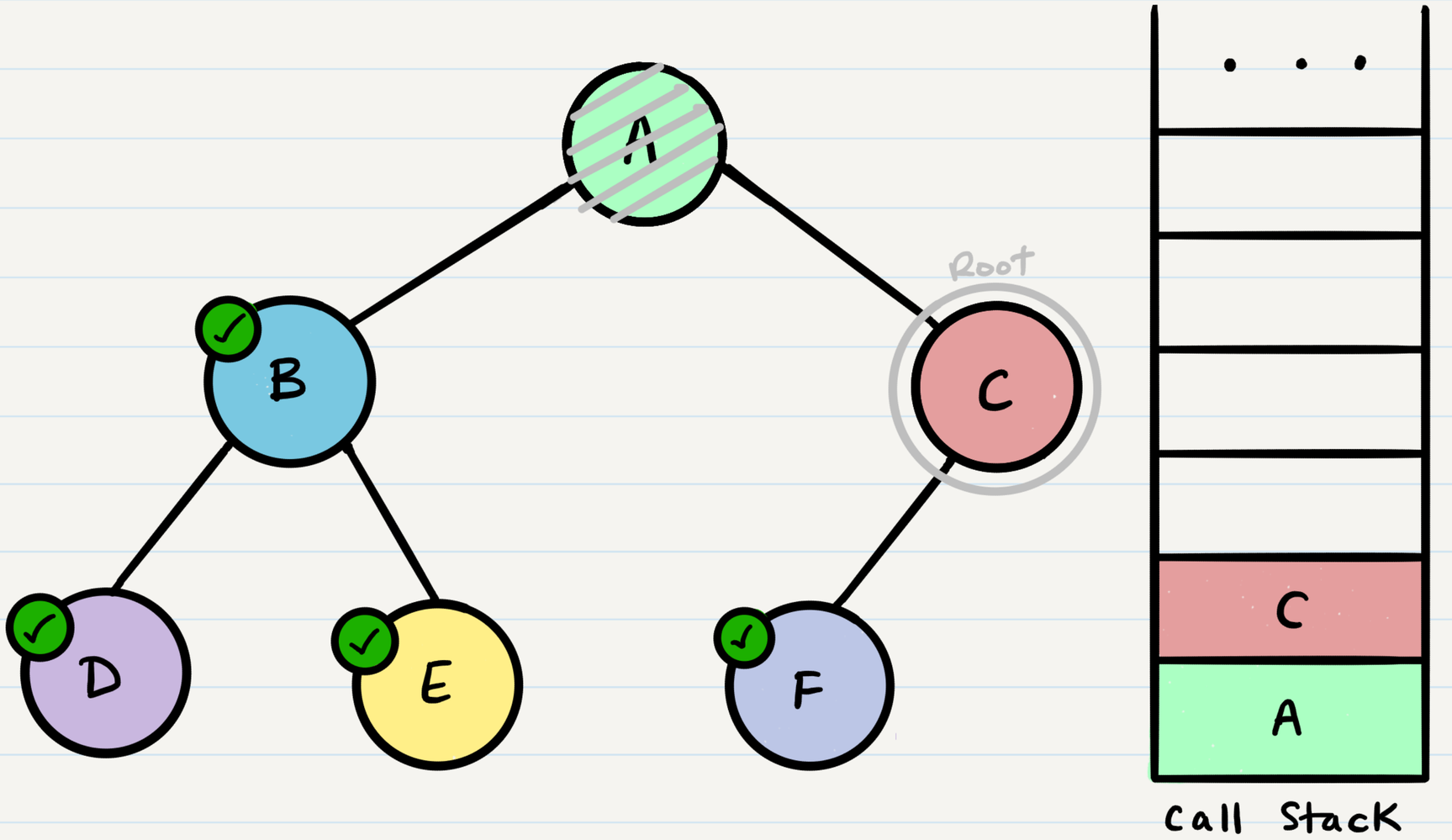
OUTPUT A, B, D, E



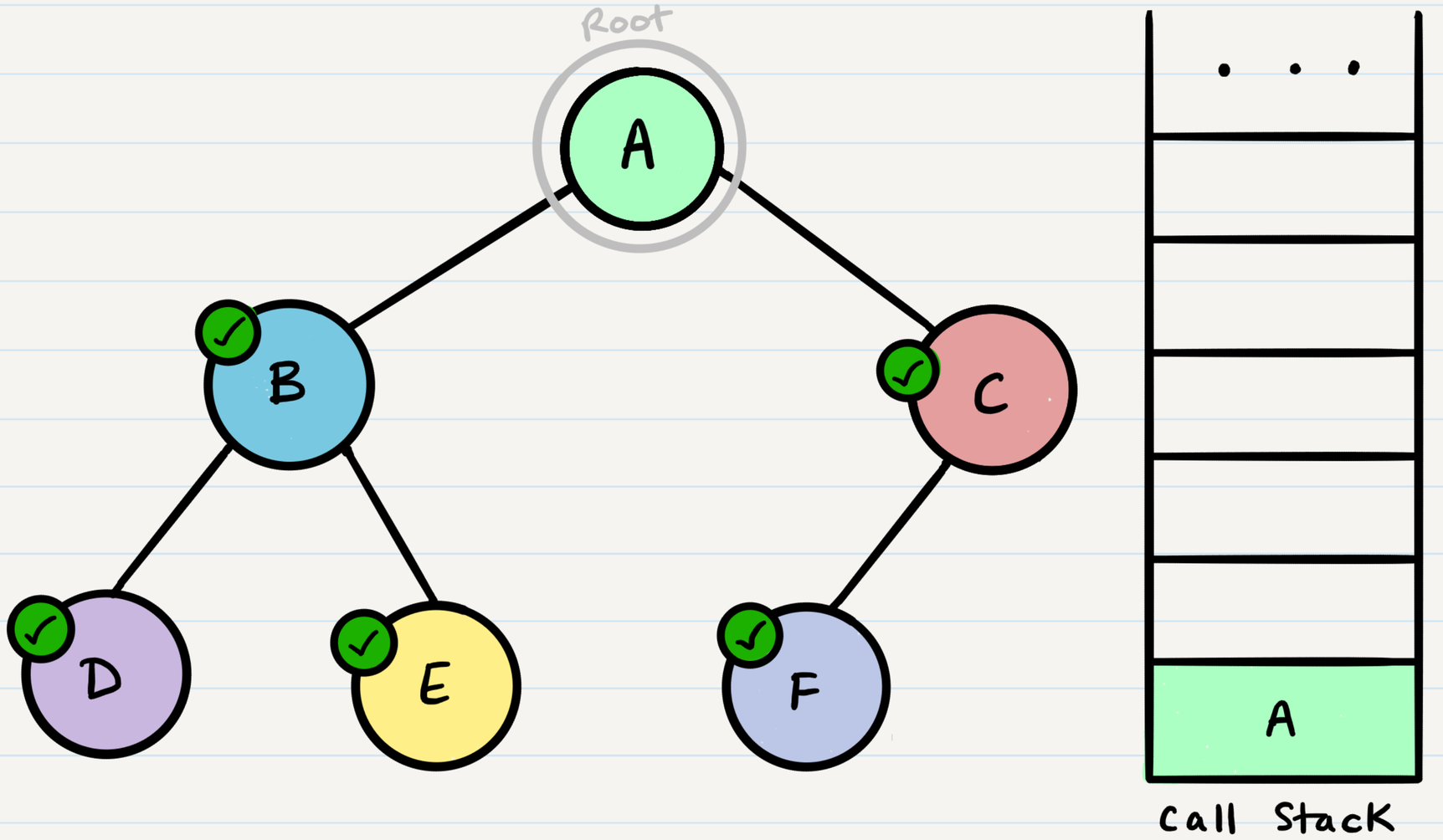
OUTPUT A, B, D, E, C



OUTPUT A, B, D, E, C, F



OUTPUT A, B, D, E, C, F



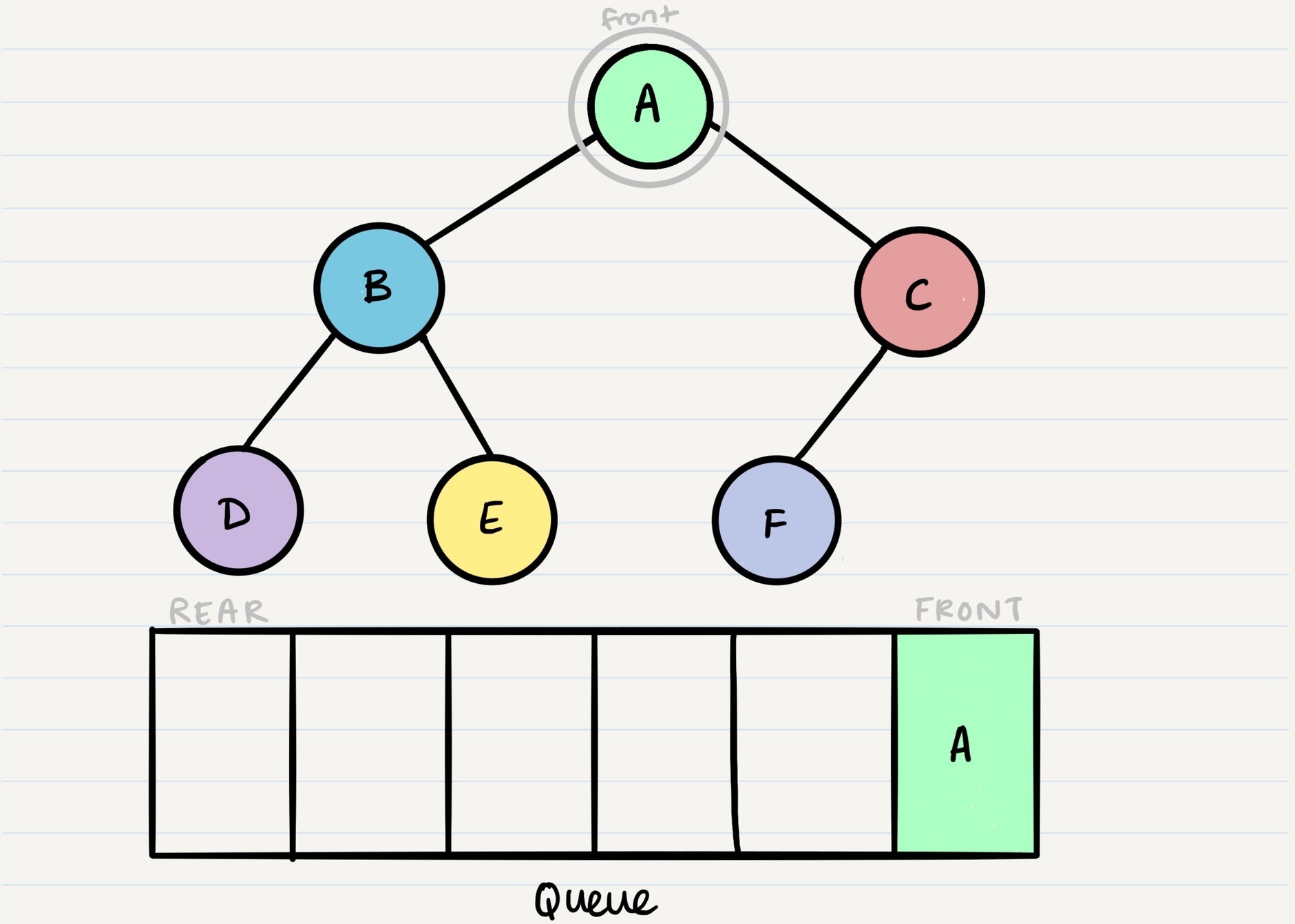
OUTPUT A, B, D, E, C, F

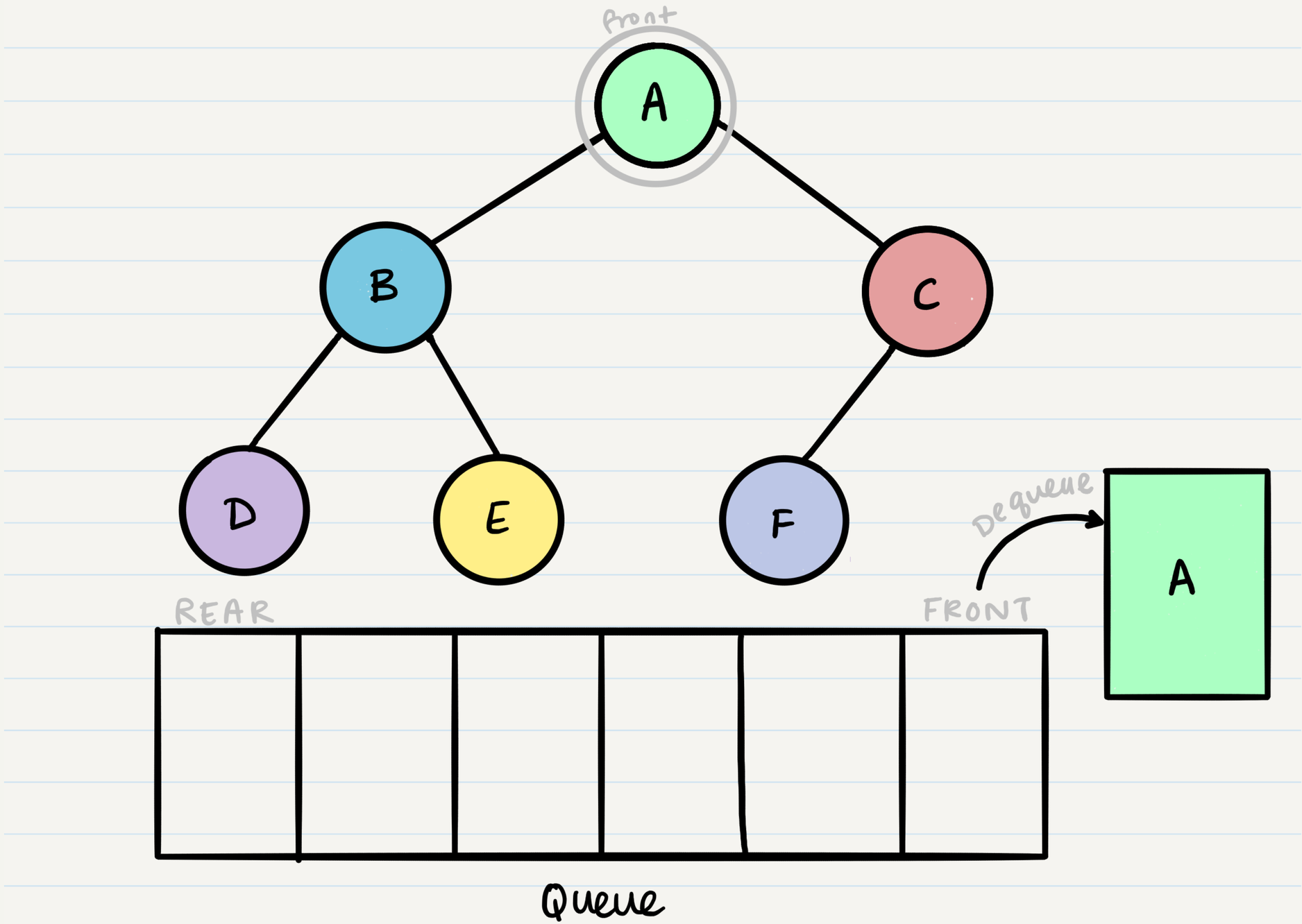
Pre-order: `root >> left >> right`

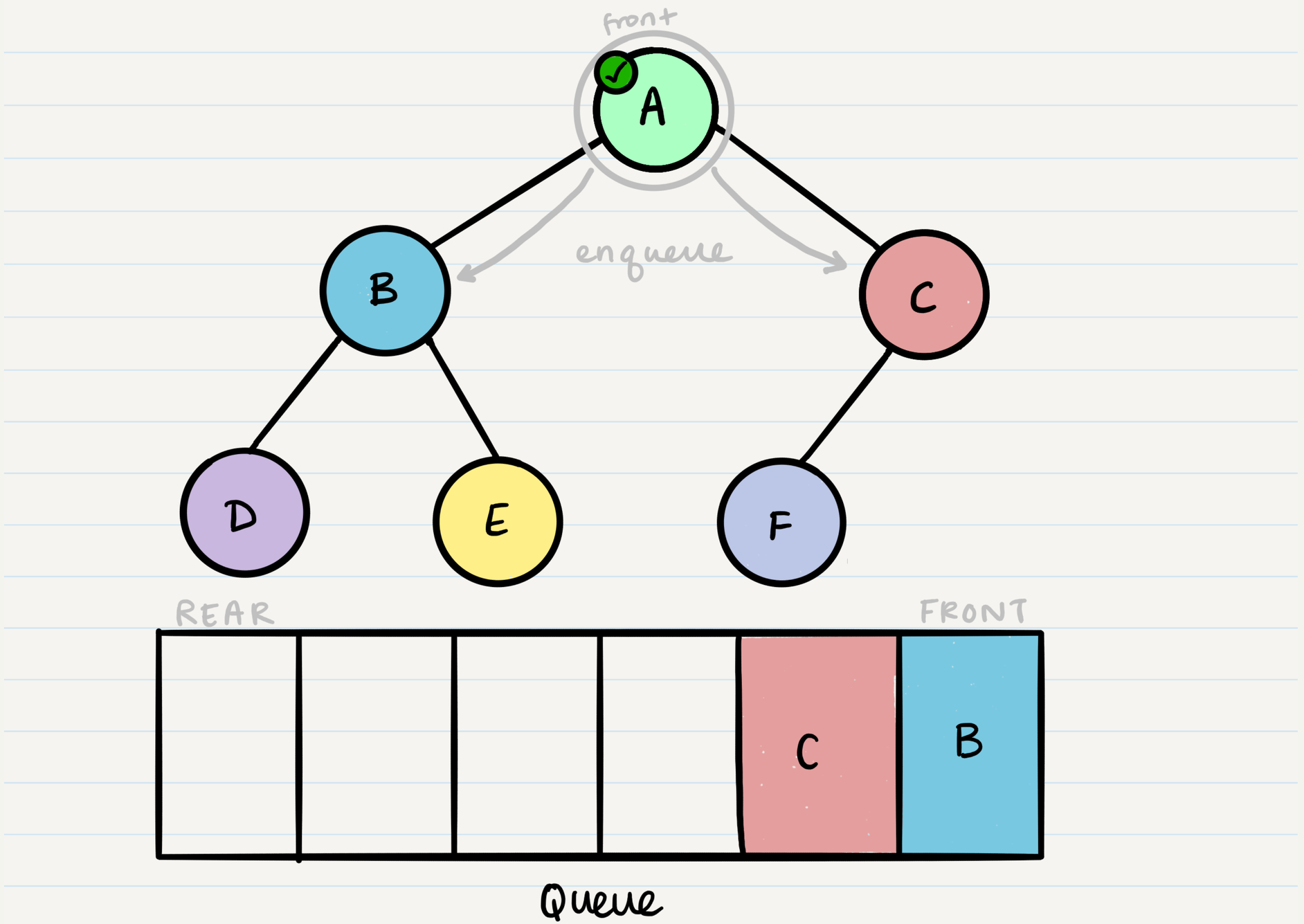
In-order: `left >> root >> right`

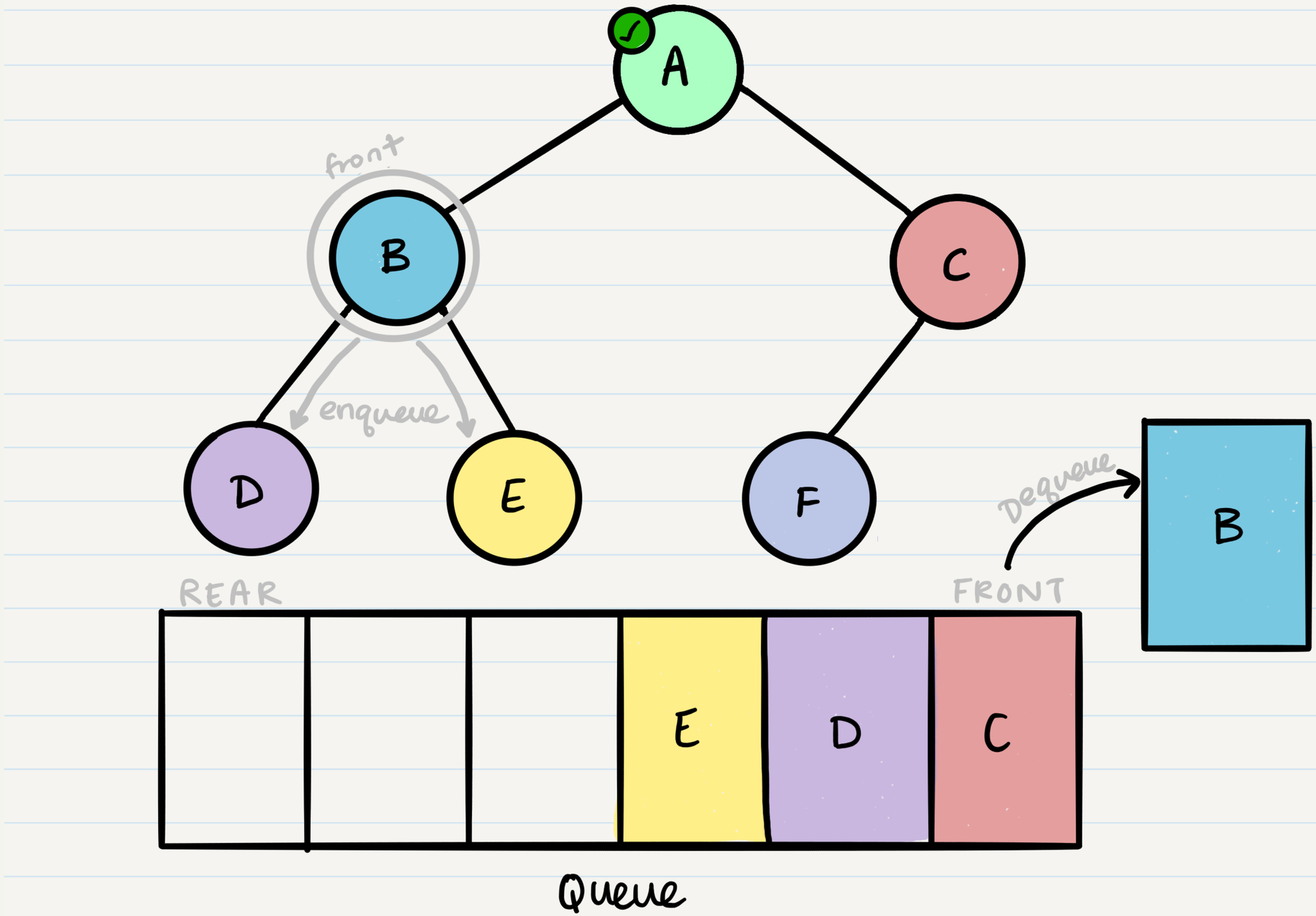
Post-order: `left >> right >> root`

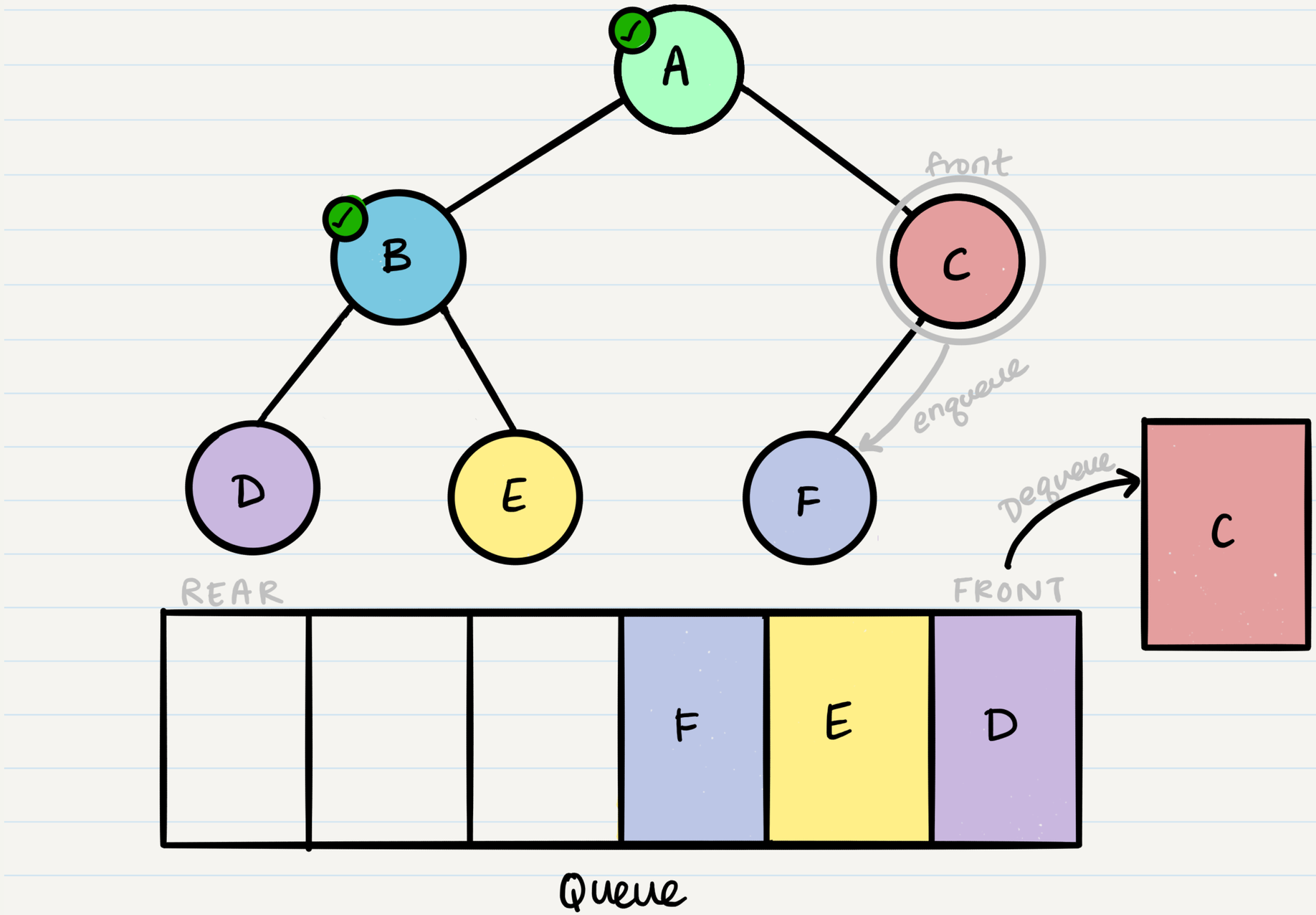
Breadth-first traversal

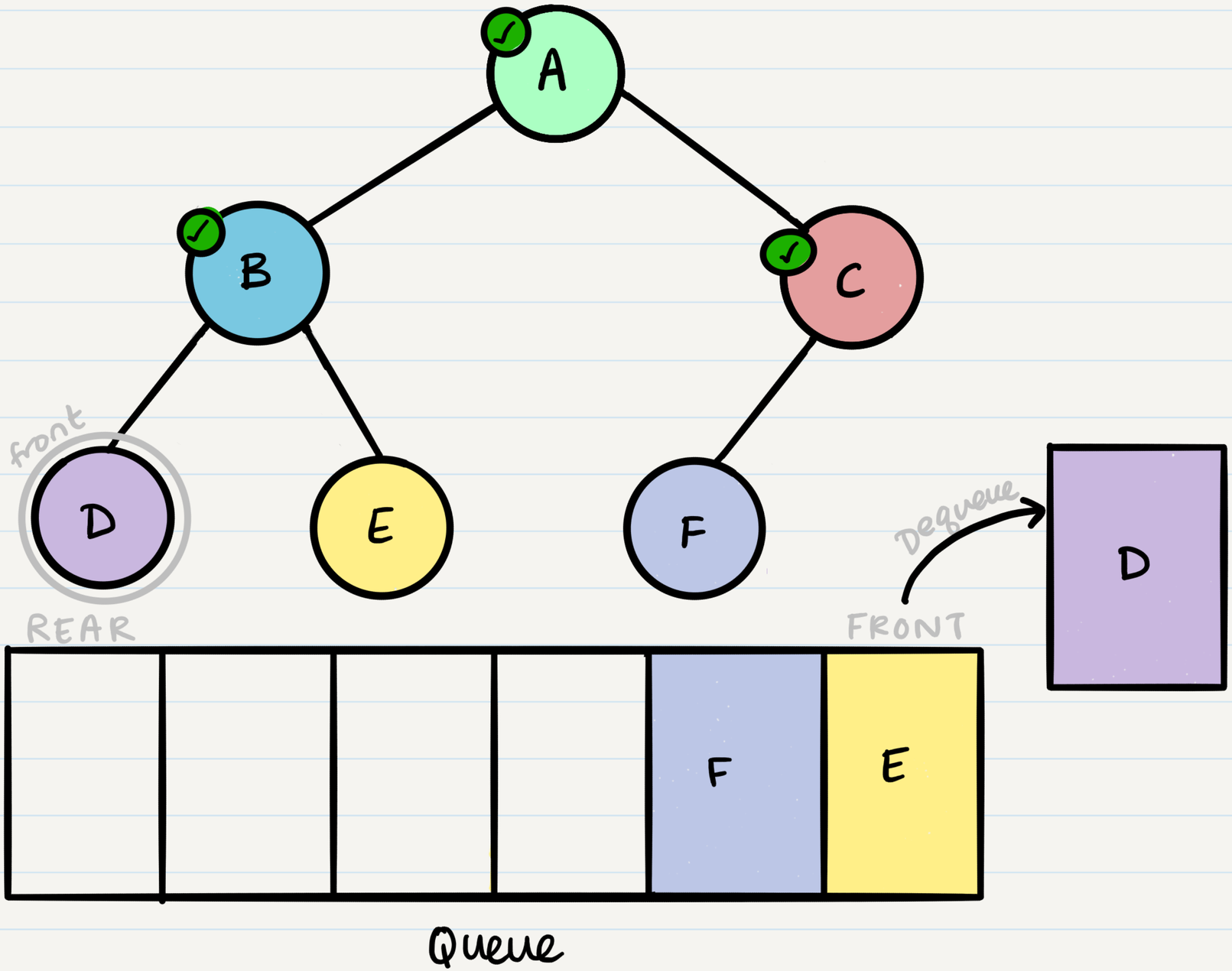

front
A
B
C
D
E
F
REAR
FRONT
A
Queue

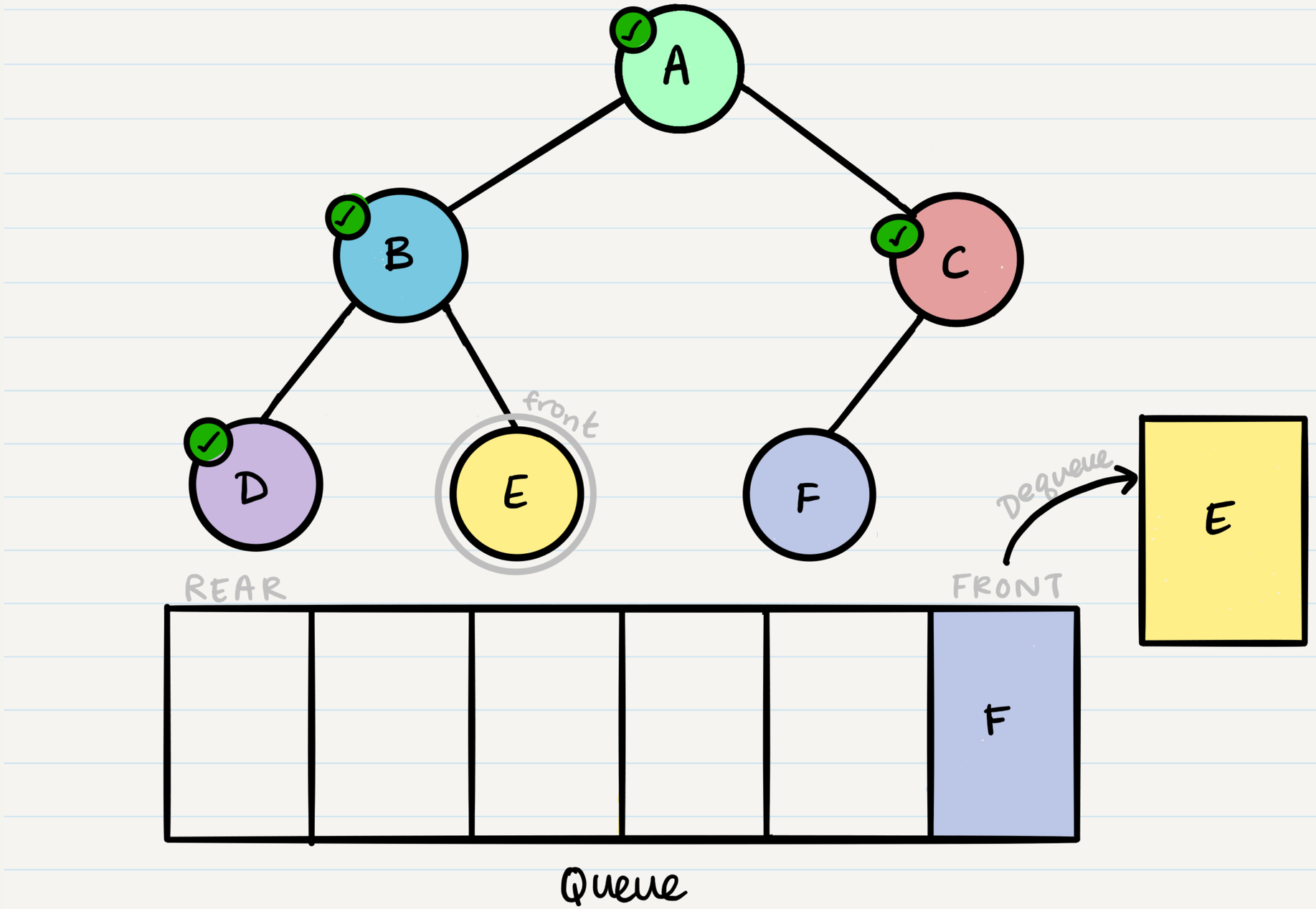

Front
A
B
C
D
E
F
Dequeue
A
REAR
FRONT
Queue

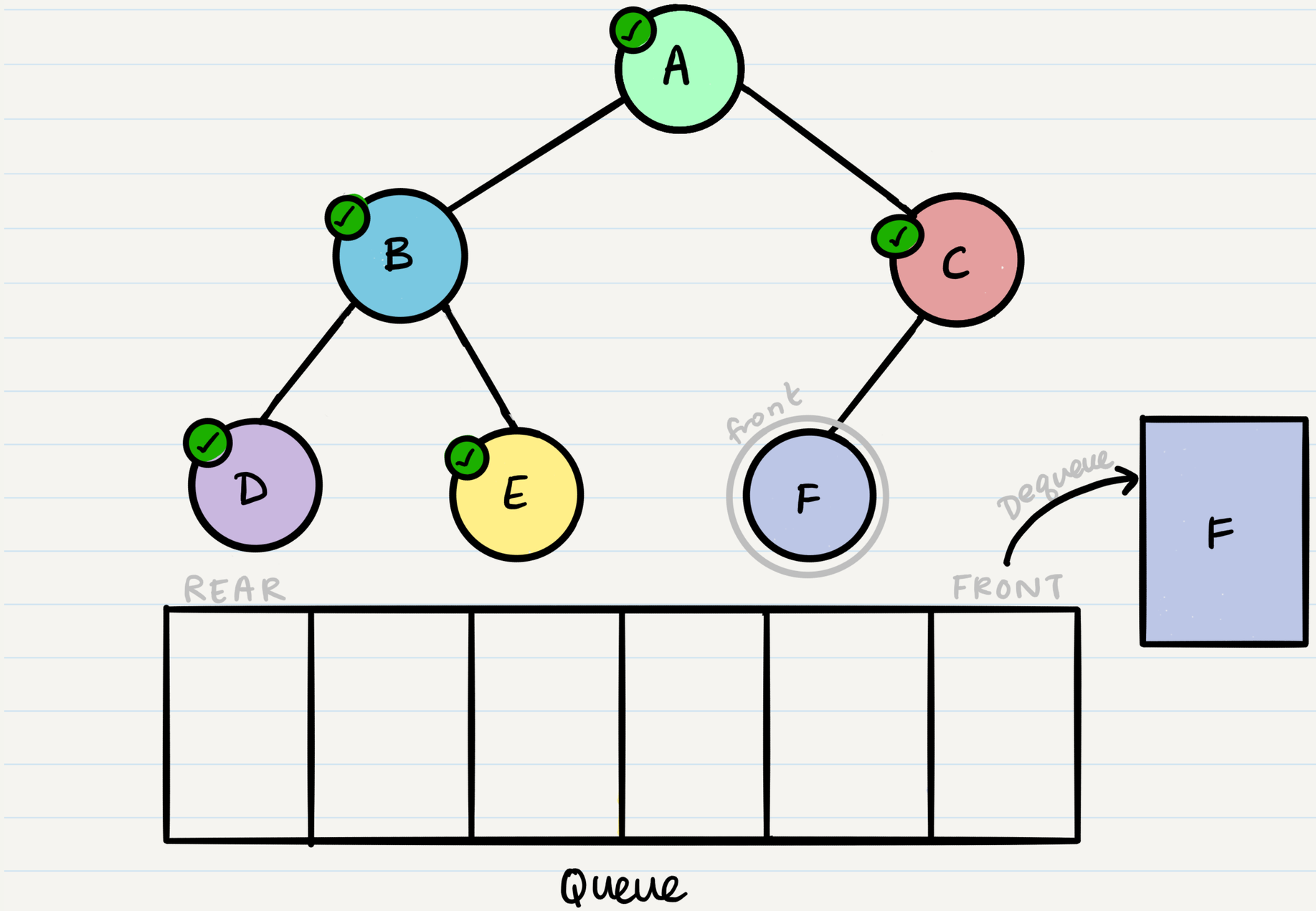

front
A
enqueue
B
C
D
E
F
REAR
FRONT
C
B
Queue


A
front
B
C
enqueue
D
E
F
Dequeue
B
REAR
FRONT
E
D
C
Queue


A
B
C
D
E
F
front
enqueue
Dequeue
C
REAR
FRONT
F
E
D
Queue

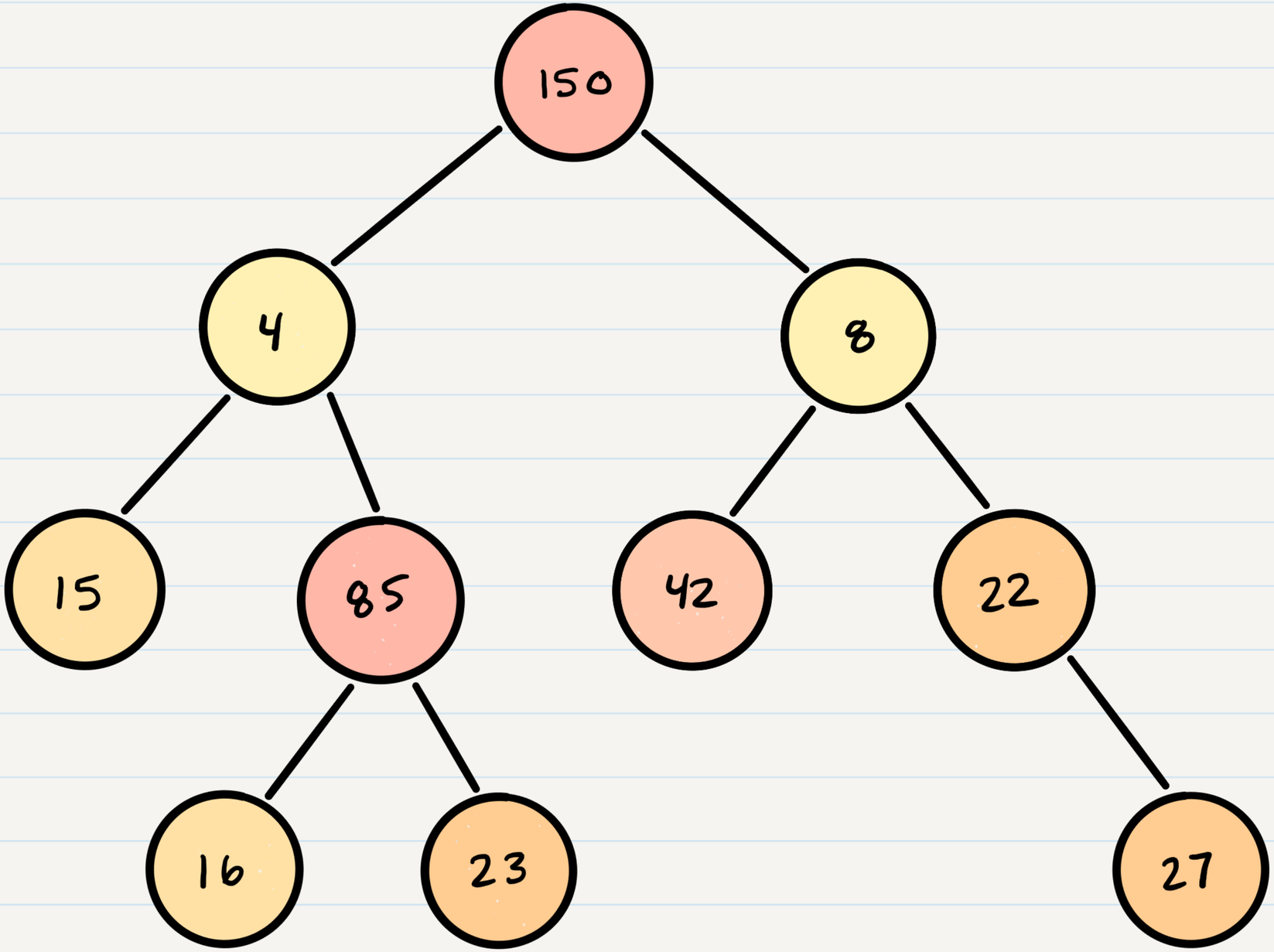

A
B
C
front
D
E
F
REAR
Dequeue
FRONT
D
F
E
Queue

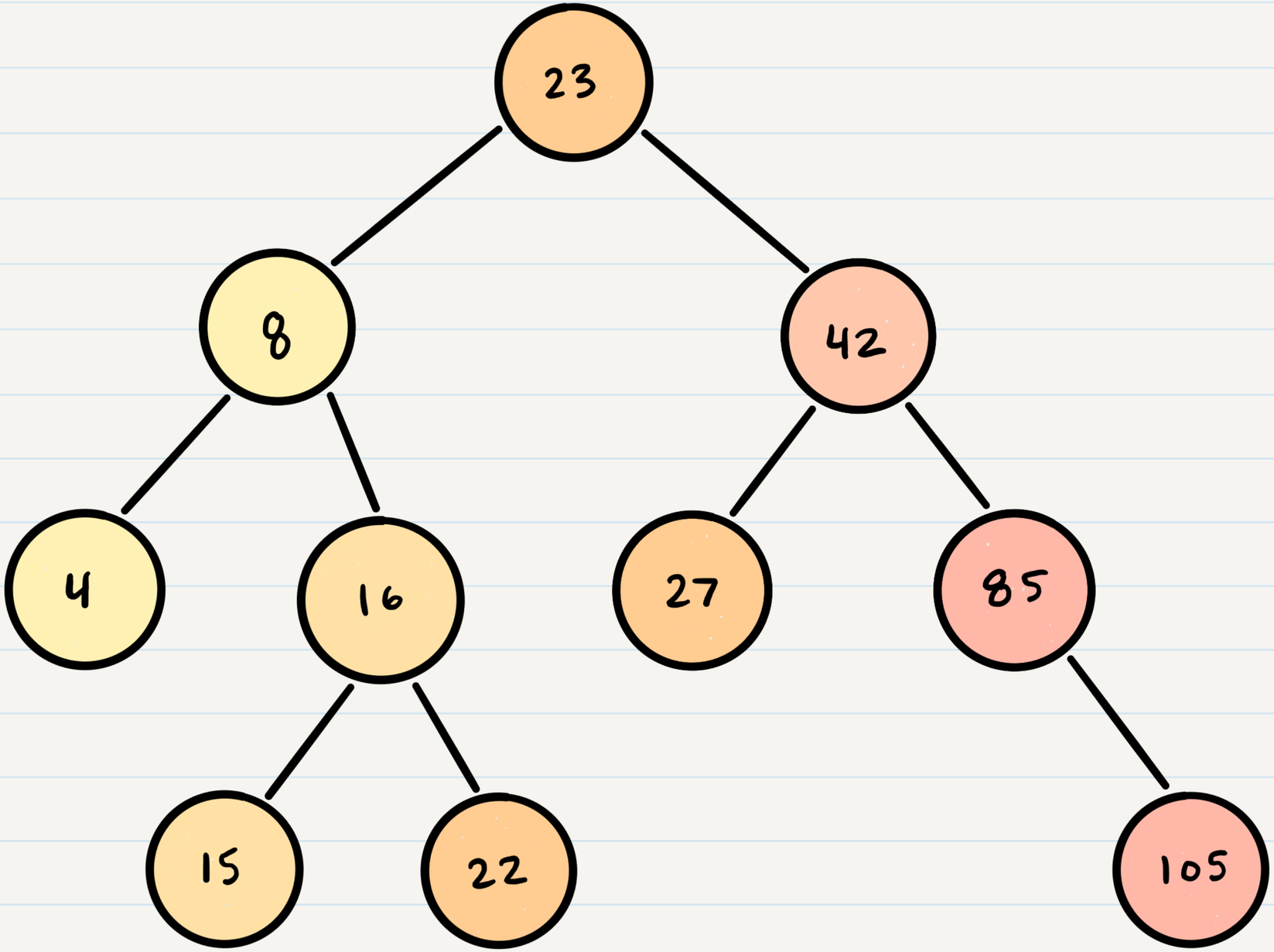

A
B
C
D
front
E
F
Dequeue
E
REAR
FRONT
F
Queue

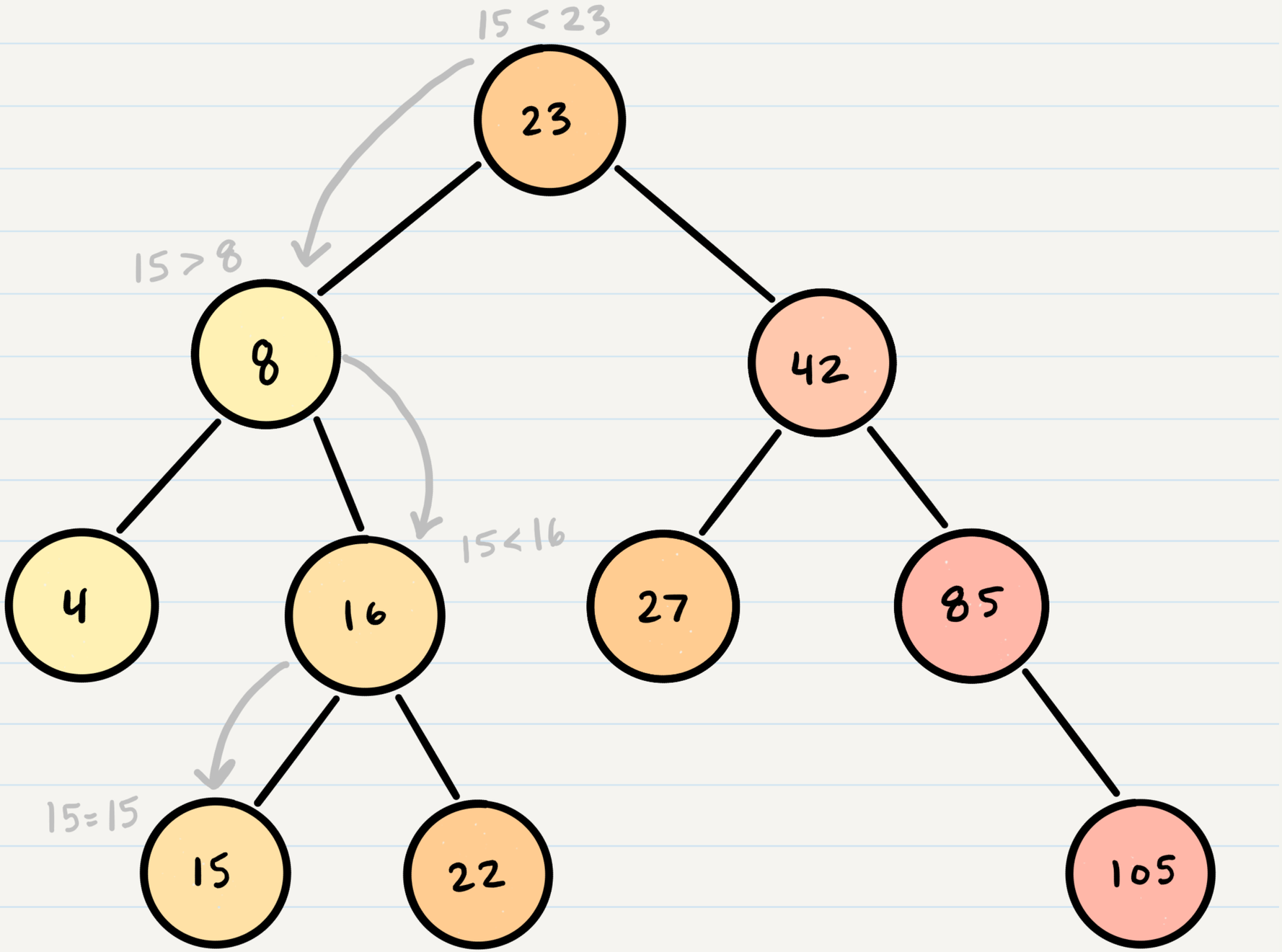

A
B
C
D
E
F
front
Dequeue
F
REAR
FRONT
Queue

Balanced and Binary Search Trees

- Time complexities for traversing a trees is better when the tree is **balanced**
- A balanced tree has the same number of left and right **descendants**
- Balanced Trees have two nodes per parent where possible
- A **binary search tree** is a sorted binary tree
- For each parent node, the left nodes are less than the value of the parent, and the right nodes are more than the value of the parent
- We can find things quicker because we can cut out half of the tree every layer


150
4
8
15
85
42
22
16
23
27


23
8
42
4
16
27
85
15
22
105


15 < 23
23
15 > 8
8
42
4
16
15 < 16
27
85
15 = 15
15
22
105

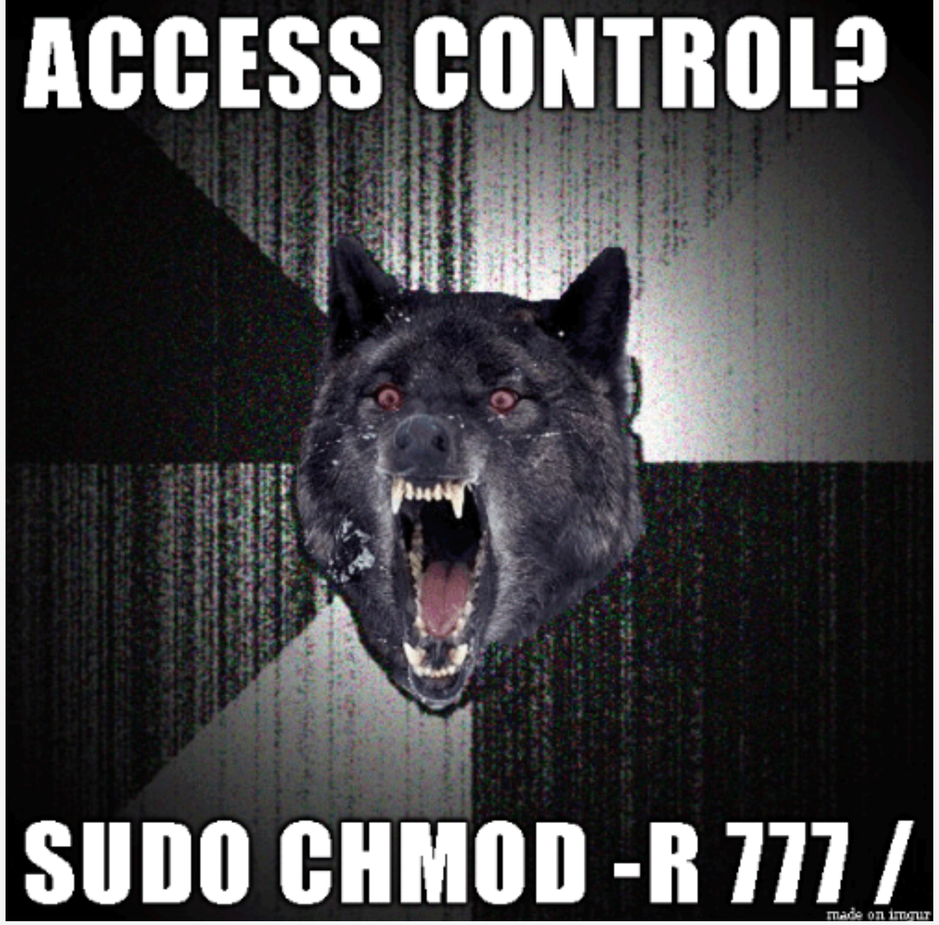
Now back to Access Control

Access Control

- We've touched upon this idea already (remember books and roles?)
- **Access control** is the idea that every items a system should have a set of access restrictions and allowances
    - In old computers this was file read/write privileges
- Multiple types of access control:
    - Mandatory - A central authority prescribes levels of clearance, and those levels gain data access
    - Discretionary - Data owners (admins) decide who can access their data
    - **Role Based** - Each user has a role, and that role defines access
    - Rule Based - Each item has some rules about how and when it can be accessed
    - Attribute Based - Data and users have attributes, and access is dynamically decided based on those attributes

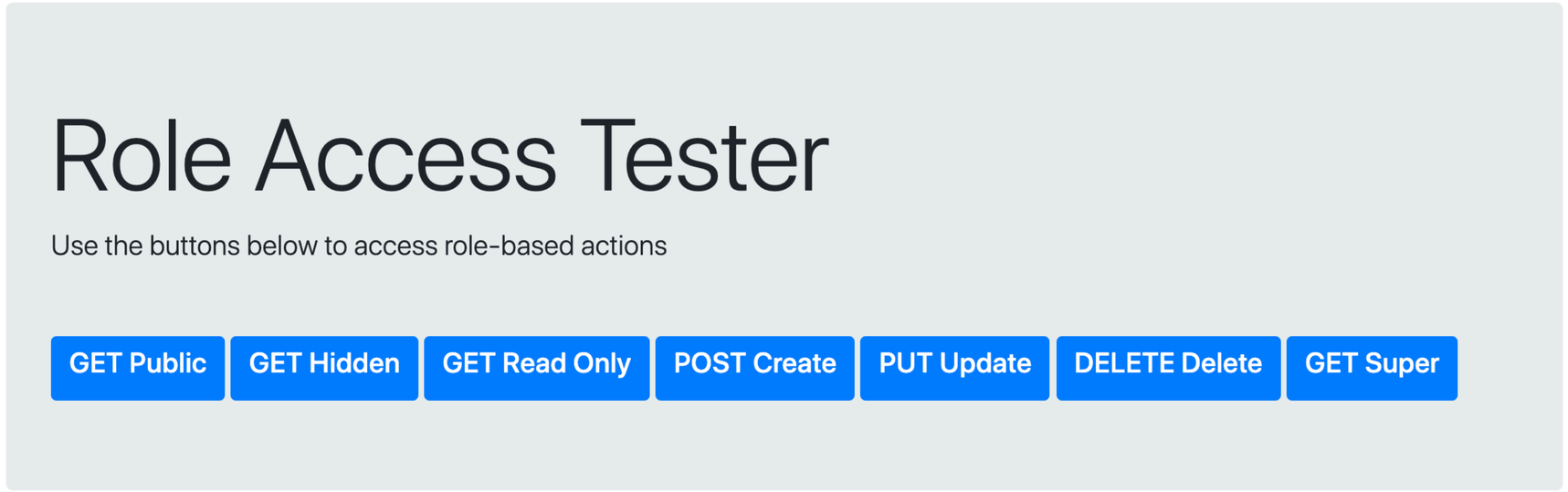
Role Based Access Control

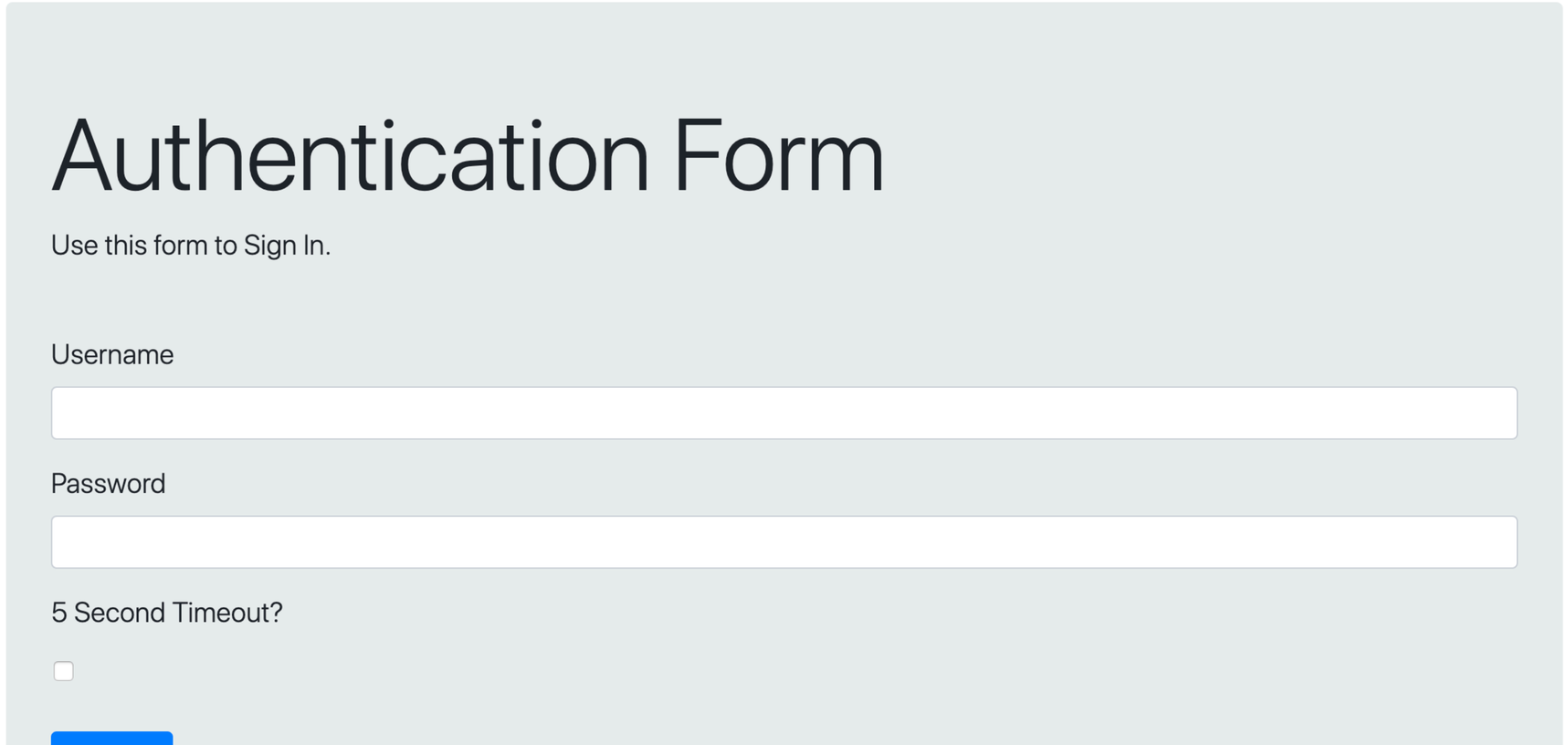
- One of the most common
- Usually, users have a **role** field - like our users had “admin”, “editor” and “user”
- There is a **separate model** that defines the roles in more details
    - Assigns **capabilities** to each role
    - Admins have the capability to read, write, update, delete, move, upload, etc



Role Capabilities

- In the role *model*, each role usually contains its name and a collection of capabilities
- These are usually defined as an array of strings
- The capabilities can be as detailed or surface level as the app designer wants; however a standard generic set is usually:
    - Create
    - Read
    - Update
    - Delete





Lab

Role Based Capabilities

Let's review the lab for today, and try to implement pieces of it!

What's Next:

- Due by Midnight tonight:
  - Learning Journal Class 13
  - Partner Power Hour Report #3
- Due by Midnight Sunday:
  - **Feedback Week 7**
  - **Career Coaching - Personal Pitch**
  - **Career Coaching - Star Methodology**
- Due by Midnight Monday:
  - **Code Challenge 13**
- Due by 6:30pm Tuesday:
  - **Lab 13**
- Next Class:
  - **Class 14 - API Server Revisited**

Questions?