FINAL WRAP UP!

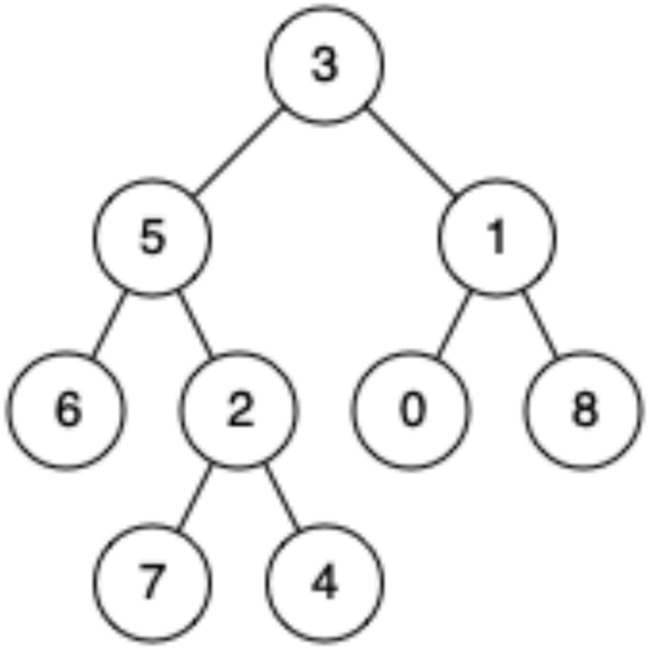
Problem Solving with Computers-II

Preparing for the final exam…

I can deal with pressure, and deadlines.

Problem:

Find the lowest common ancestor of nodes (u, v) in a binary tree

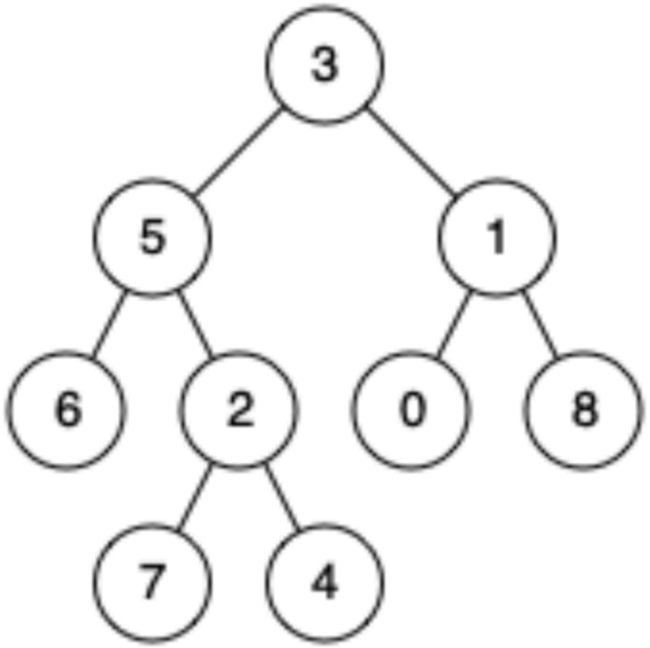


https://leetcode.com/problems/lowest-common-ancestor-of-a-binary-tree/

Approach 1: Turn definitions into an algorithm

Ancestor(u):
any node on a **path** ending in u

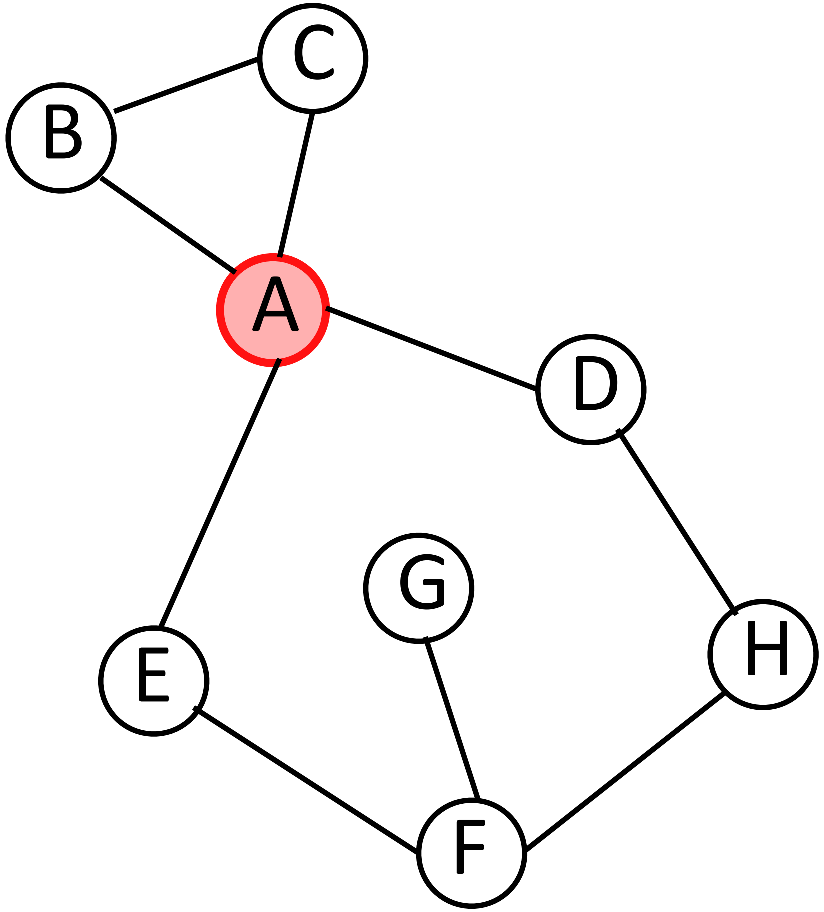
```
LCA(r: root of tree, u, v):
   Find the path from root(r) to u
   Find the path from root(r) to v
   Return common node on both paths
          farthest from the root(r)
```



https://leetcode.com/problems/lowest-common-ancestor-of-a-binary-tree/

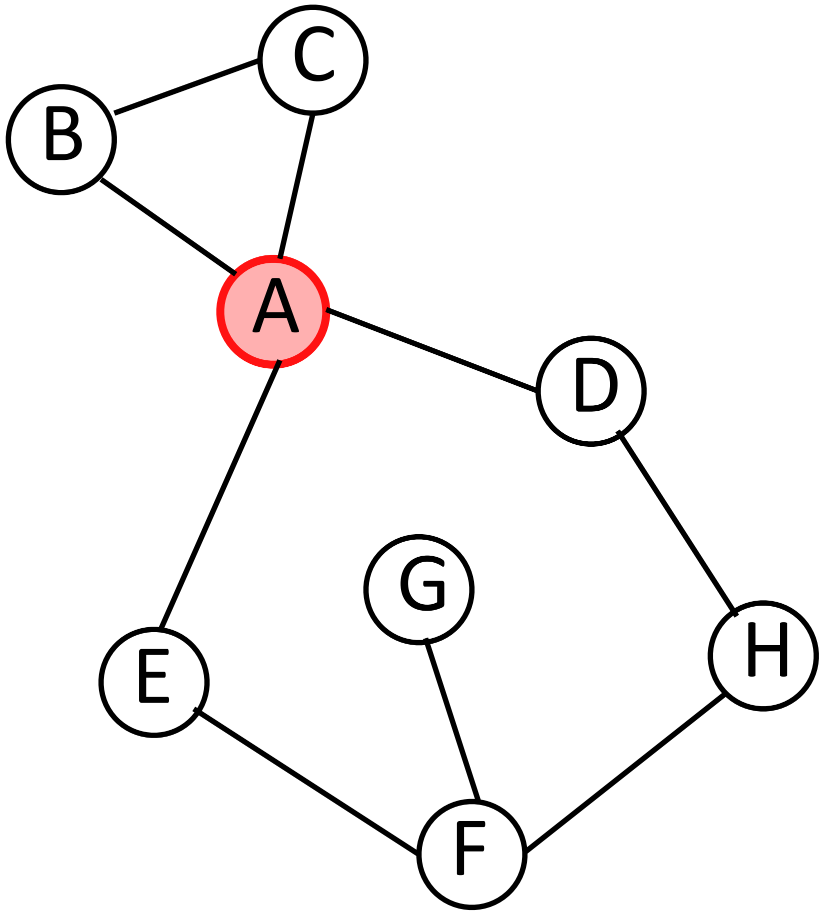
Explore – Depth First on a Graph

```
exploreDFS(v)
   v.visited ← true
   For each edge (v,w)
      If not w.visited
         exploreDFS(w)
```



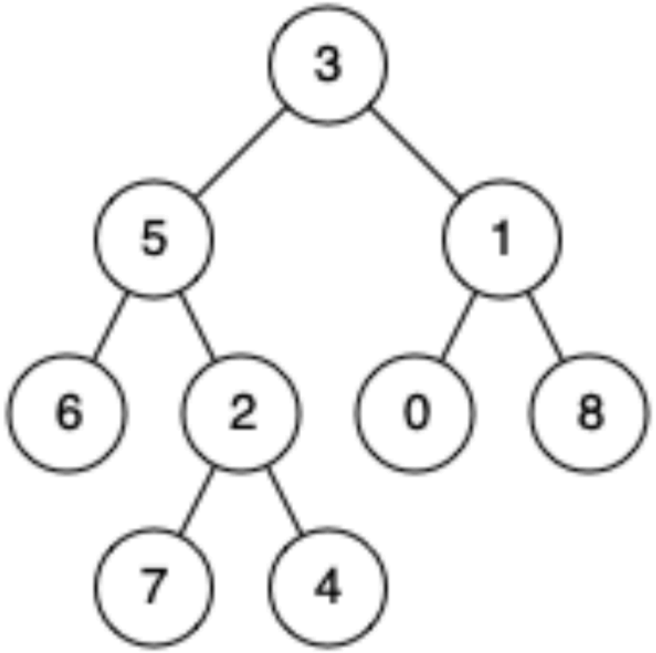
Modify exploreDFS to find paths

```
exploreDFS(v)
  v.visited ← true
  For each edge (v,w)
    If not w.visited
      w.prev ← v
      exploreDFS(w)
```

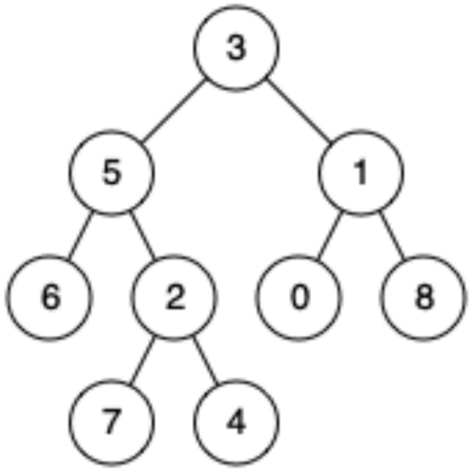


Explore – Depth First on a Tree

```
exploreDFS(r: root node):
 Print r.val
 if(r.left)      exploreDFS(r.left)
 if(r.right)     exploreDFS(r.right)
```

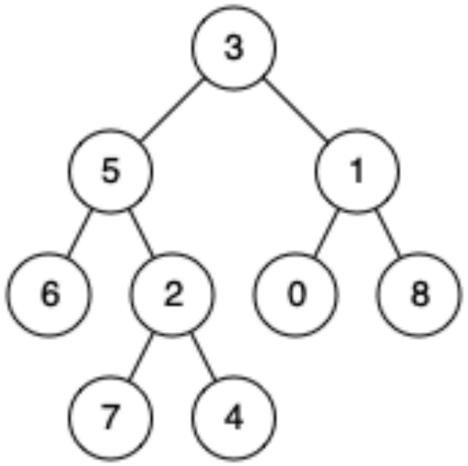


Modify DFS to find node u



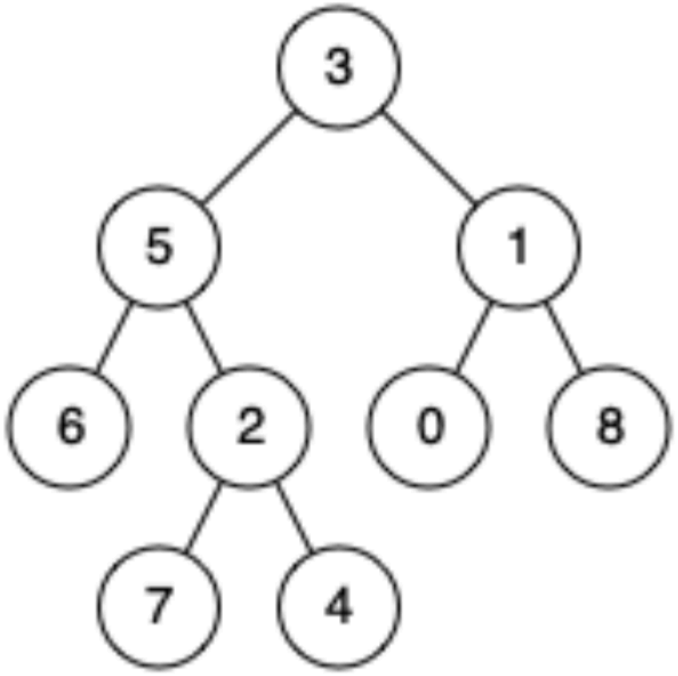

```
DFS_FindNode(r, u, found):



 ______________________________________

 if(r.left ____________) DFS_FindNode(r.left,_____________)

 if(r.right ___________) DFS_FindNode(r.right,____________)
```

Modify DFS to find the path to node u




```
DFS_FindPath(r, u, found, path):

____________________________________________

  if(r.val == u.val) {found ← true; return;}

  if(r.left ___________) DFS_FindPath(r.left,______________)

  if(r.right __________) DFS_FindPath(r.right,_____________)

______________________________________
```

Approach 2: Divide and Conquer

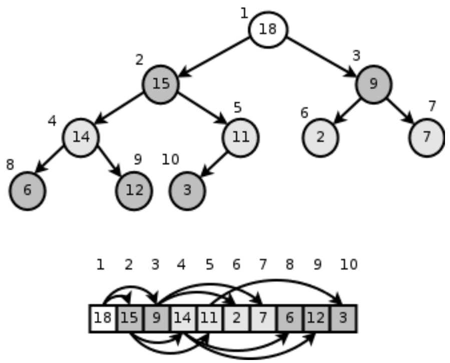


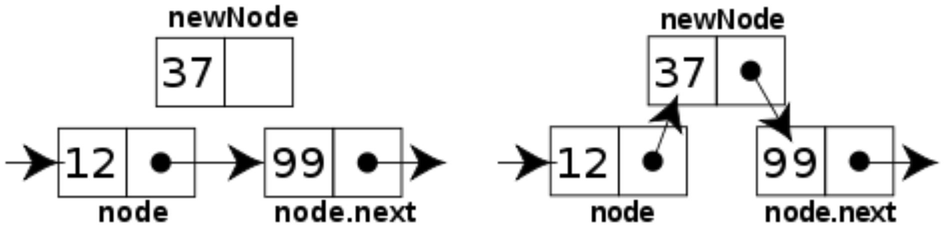

```
LCA(r: root of tree, u, v):
  If(r.left && u,v exist in r.left)
     Return LCA(r.left, u, v)
  If(r.right && u,v exist in r.right)
     Return LCA(r.right, u, v)
  If(u or v exists in r.left && u or v exists in r.right):
     Return ___________________________
  If(u or v exists in r.left || u or v exists in r.right ):
    If(r.val == u.val ||r.val== v.val) Return___________________
```

https://leetcode.com/problems/lowest-common-ancestor-of-a-binary-tree/

Learning goals

- Design and implement **larger programs** that **run fast**
- Organize **data** in programs using **data structures**
- **Analyze** the **complexity** of your programs
- Understand what goes on **under the hood of programs**





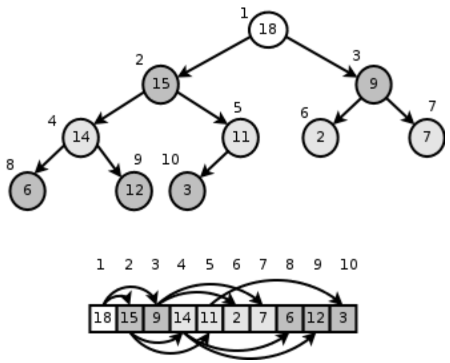
INSERTION-SORT(A)

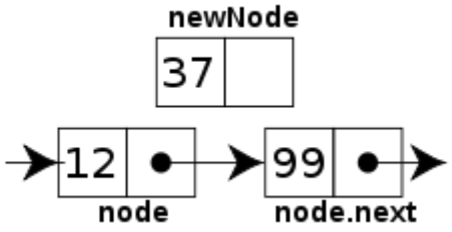
	cost	times
for $j = 2$ **to** $A.length$	c_1	n
$key = A[j]$	c_2	$n-1$
// Insert $A[j]$ into the sorted sequence $A[1..j-1]$.	0	$n-1$
$i = j-1$	c_4	$n-1$
while $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
$A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n} (t_j - 1)$
$i = i-1$	c_7	$\sum_{j=2}^{n} (t_j - 1)$
$A[i+1] = key$	c_8	$n-1$

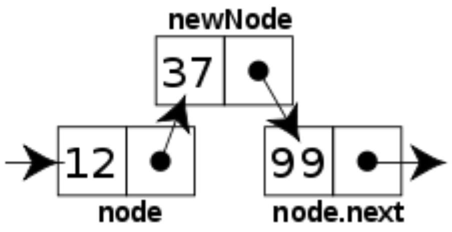
Data Structures and C++ **Complexity Analysis**

Resources for the Final Exam

- Office hours will be offered until Wed of Finals Week
- Code from lectures: https://github.com/ucsb-cs24-w24/cs24-w24-lectures
- Practice Problems and Labs: https://ucsb-cs24.github.io/w24/
- Past Exams: Available on Canvas
- Tool to visualize data structures: https://visualgo.net/







INSERTION-SORT(A)	*cost*	*times*
for $j = 2$ **to** $A.length$	c_1	n
$key = A[j]$	c_2	$n-1$
// Insert $A[j]$ into the sorted sequence $A[1..j-1]$.	0	$n-1$
$i = j - 1$	c_4	$n-1$
while $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
$A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n} (t_j - 1)$
$i = i - 1$	c_7	$\sum_{j=2}^{n} (t_j - 1)$
$A[i+1] = key$	c_8	$n-1$

Data Structures and C++ **Complexity Analysis**

Break: Please take a moment to fill the course evaluations!

PROBLEM SOLVING II

Student-FO

https://go.blueja.io/tJ9l2Cs-j068oUfOzQn2jA

To access the evaluation, scan this QR code with your mobile phone.

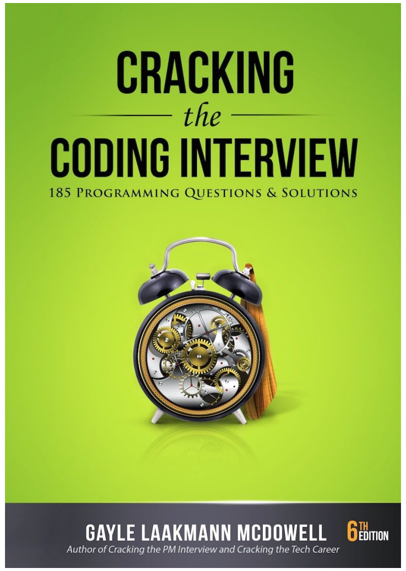
PROBLEM SOLVING II

Student-FO

https://go.blueja.io/tJ9l2Cs-j068oUfOzQn2jA

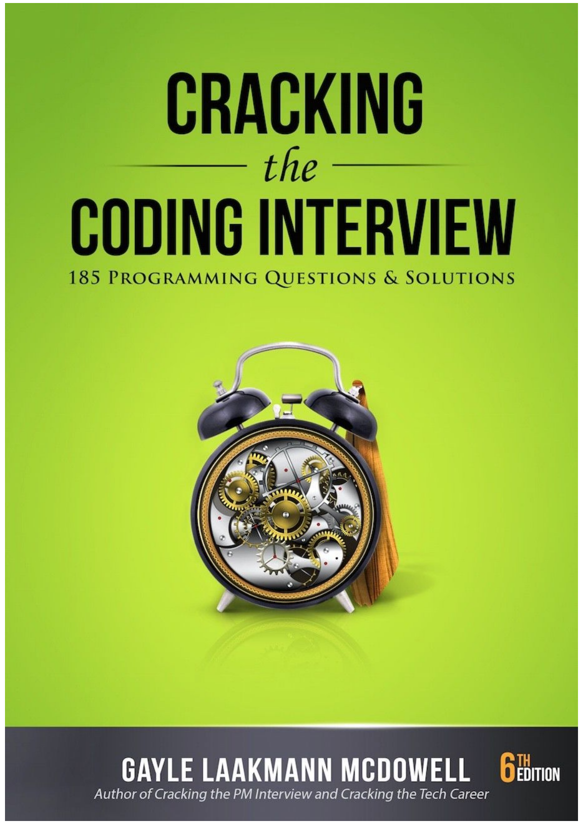
To access the evaluation, scan this QR code with your mobile phone.

Tips for Technical Interviews



1. Listen carefully
2. Draw an example
3. State the brute force or a partially correct solution
   - then work to get at a better solution
4. Optimize:
   - Make time-space tradeoffs to optimize runtime
   - Precompute information: Reorganize the data e.g. by sorting
5. Solidify your understanding of your algo before diving into writing code.
6. Start coding!

Interview practice!



Write a ADT called minStack that provides the following methods

- push() // inserts an element to the "top" of the minStack
- pop() // removes the last element that was pushed on the stack
- top () // returns the last element that was pushed on the stack
- min() // returns the minimum value of the elements stored so far

Practice the interview tips:

- Draw/solve a small example! (2 min)
  - Think of the most straightforward approach (1 min)
  - Evaluate its performance (1 min)
  - Think of another approach and evaluate it (5 min)
    - Can you trade off space/memory for better runtime?
- Pick the most promising approach and start coding! (10 min)

Thank you and all the best !