Binary Trees Pt. 1

AGENDA

- Binary Trees
- Depth-First Search, Breadth-First Search
- Breakout Sessions
- Behavioral Interview Questions and Project Discussions
- ***Please post questions in Slack channel se103-s2-help-jun21***

@lizTheDeveloper - They/Them

I’m @lizTheDeveloper everywhere on the internet!

Currently Director of R&D @ Karat.io : Technical Interviewing As A Service

Taught Computer Science for 10 years

Founded: Hackbright Academy, Hipcamp, Galvanize Web Development Program

👩‍🎤CTO: Hipcamp, Enki.com

🧑‍💻Software Engineer & Manager for 20 years 🙌

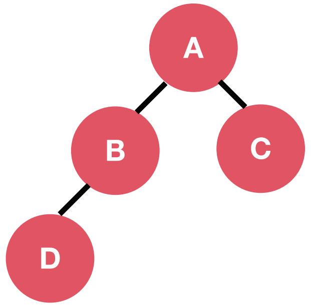
AMA @ the end of class!

Binary Trees

AKA Arboreal Directed Graphs

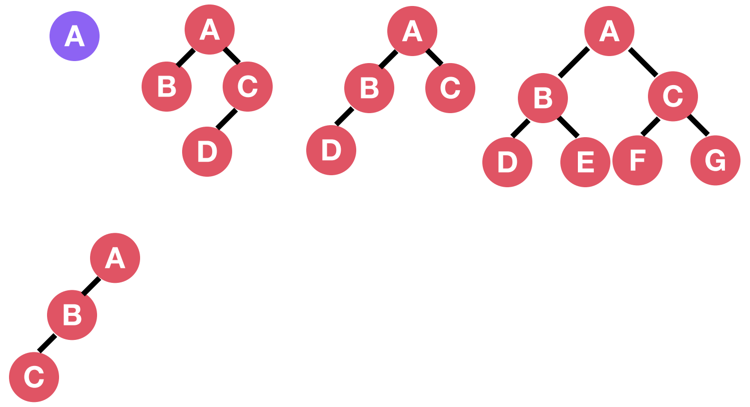
BINARY TREES: REVIEW

- Remember: Trees are a type of Graph!
- Comprised of nodes
- Binary means 2! Has Left and Right Children
- Properties: Binary Search Tree, balanced, complete, perfect etc
- Many Representations



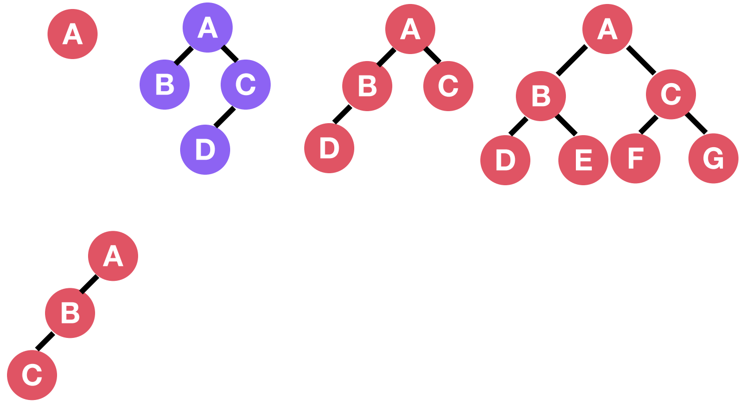
WHAT CAN A TREE LOOK LIKE?

- The appearance of a binary tree can look many different ways
- Nodes can also have duplicate values
- Realizing this is helpful when creating test cases!



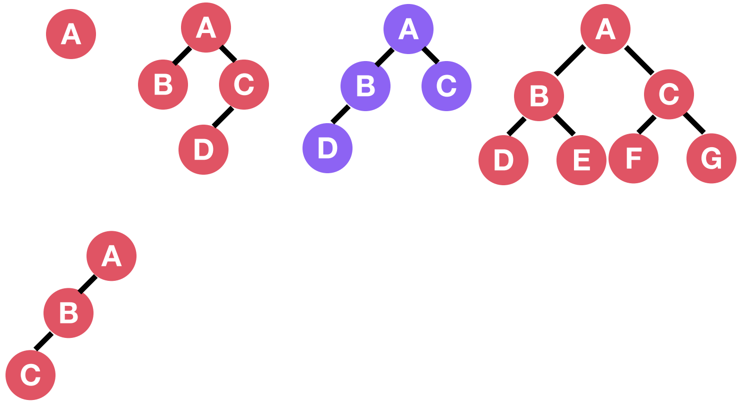
WHAT CAN A TREE LOOK LIKE?

- The appearance of a binary tree can look many different ways
- Nodes can also have duplicate values
- Realizing this is helpful when creating test cases!



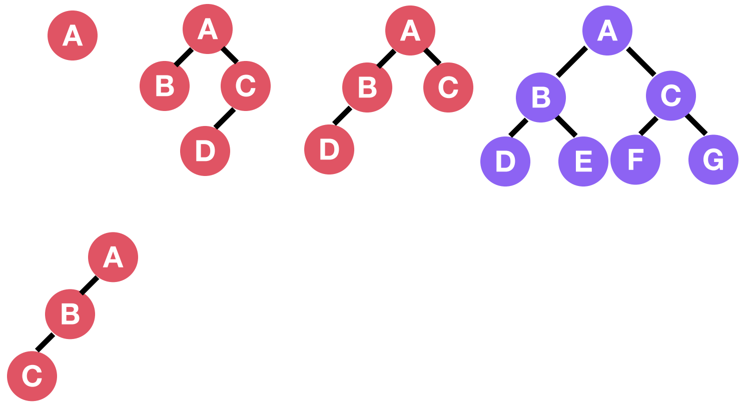
WHAT CAN A TREE LOOK LIKE?

- The appearance of a binary tree can look many different ways
- Nodes can also have duplicate values
- Realizing this is helpful when creating test cases!



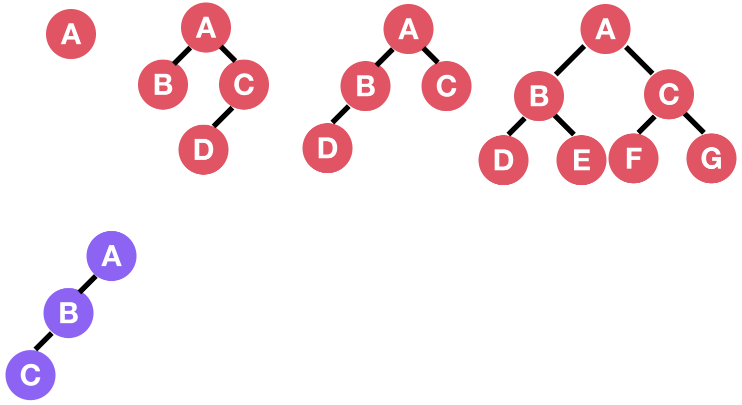
WHAT CAN A TREE LOOK LIKE?

- The appearance of a binary tree can look many different ways
- Nodes can also have duplicate values
- Realizing this is helpful when creating test cases!

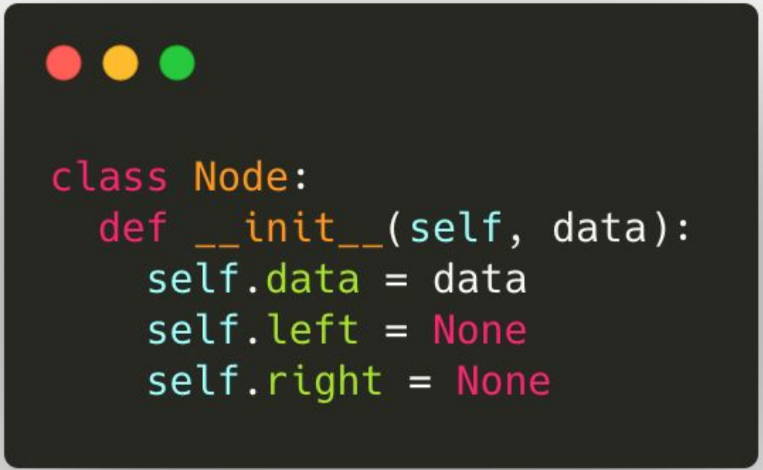


WHAT CAN A TREE LOOK LIKE?

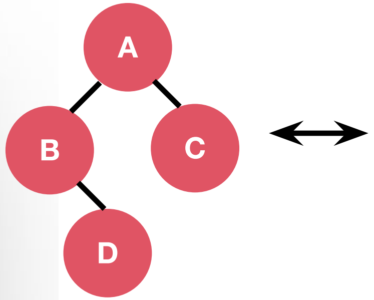
- The appearance of a binary tree can look many different ways
- Nodes can also have duplicate values
- Realizing this is helpful when creating test cases!



REPRESENTING NODES

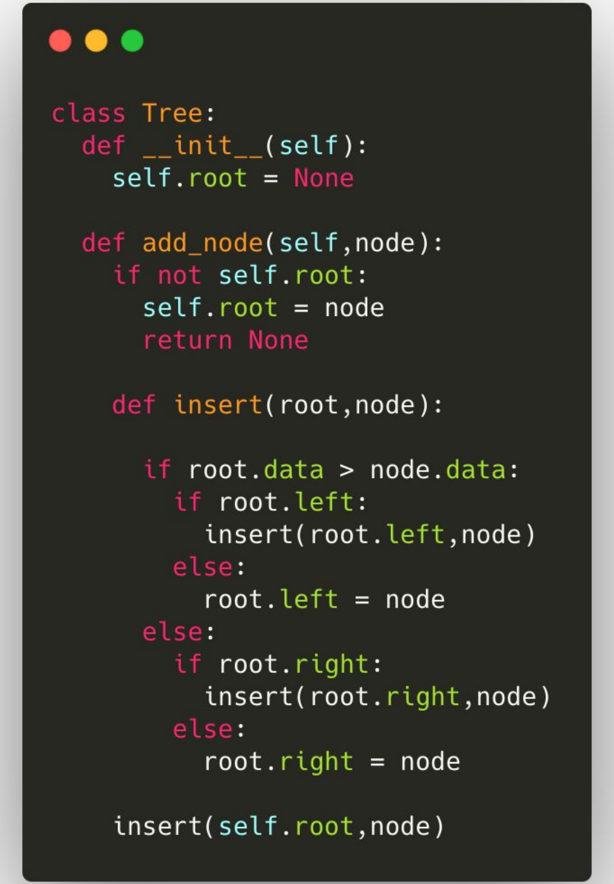


```
class Node:
  def __init__(self, data):
    self.data = data
    self.left = None
    self.right = None
```

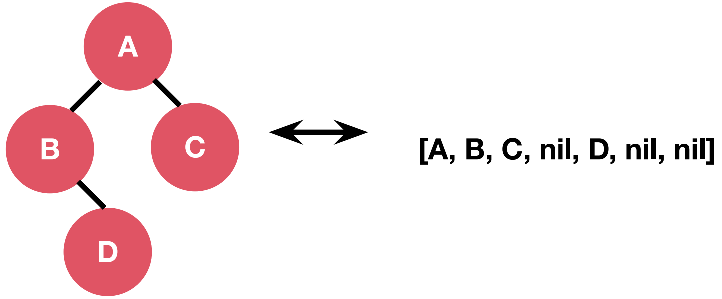


[A, B, C, nil, D, nil, nil]

REPRESENTING TREES & INSERTING A NODE

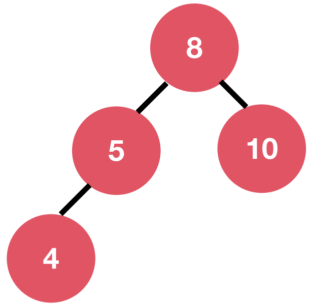


```
class Tree:
  def __init__(self):
    self.root = None

  def add_node(self,node):
    if not self.root:
      self.root = node
      return None

    def insert(root,node):

      if root.data > node.data:
        if root.left:
          insert(root.left,node)
        else:
          root.left = node
      else:
        if root.right:
          insert(root.right,node)
        else:
          root.right = node

    insert(self.root,node)
```



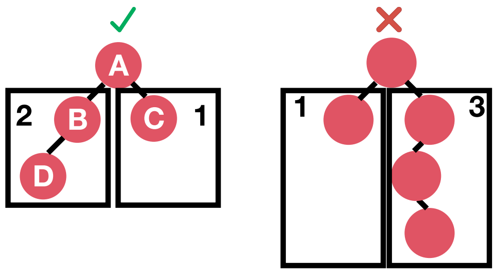
BINARY SEARCH TREES (BST)

- Left subtree contains children <= root
- Right subtree contains children > root
- Lookup is O(h) where h is the *height* of the tree



BALANCED VS UNBALANCED BINARY TREES

"Balanced": ***Height of left and right subtree of every node differs at most by 1***



Height = The max distance of any node from the root

https://xlinux.nist.gov/dads/HTML/height.html

TIME/SPACE COMPLEXITY

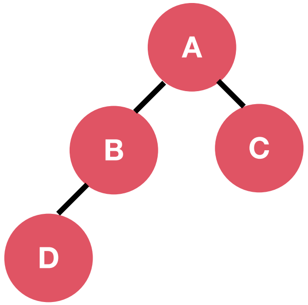
- Time complexities usually differ depending on:
  - If the tree is a Binary Search Tree or not
  - If the tree is balanced/unbalanced
- **Remember your time and space complexities, interviewers will ask!**
- Always preface your assumptions
  - *"If this BST was balanced/unbalanced, the runtime would be..."*

	Non-balanced BST	Balanced BST
Get	O(h)	O(logn)
Insert	O(h)	O(logn)
Remove	O(h)	O(logn)

COMPLETE BINARY TREES

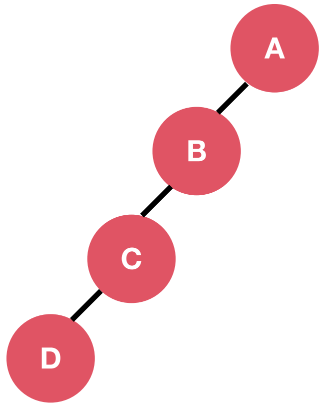
"Complete": ***Every level is filled, except the last level***

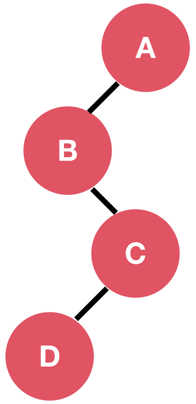
- Last level has all nodes as far left as possible
- A complete binary tree is also balanced



DEGENERATE BINARY TREES

- Also called a *pathological* tree
- Every node has at most 1 child
- This is usually why runtime/space complexity is O(n) for trees (especially BSTs)
- Is caused by particular orders of insertion

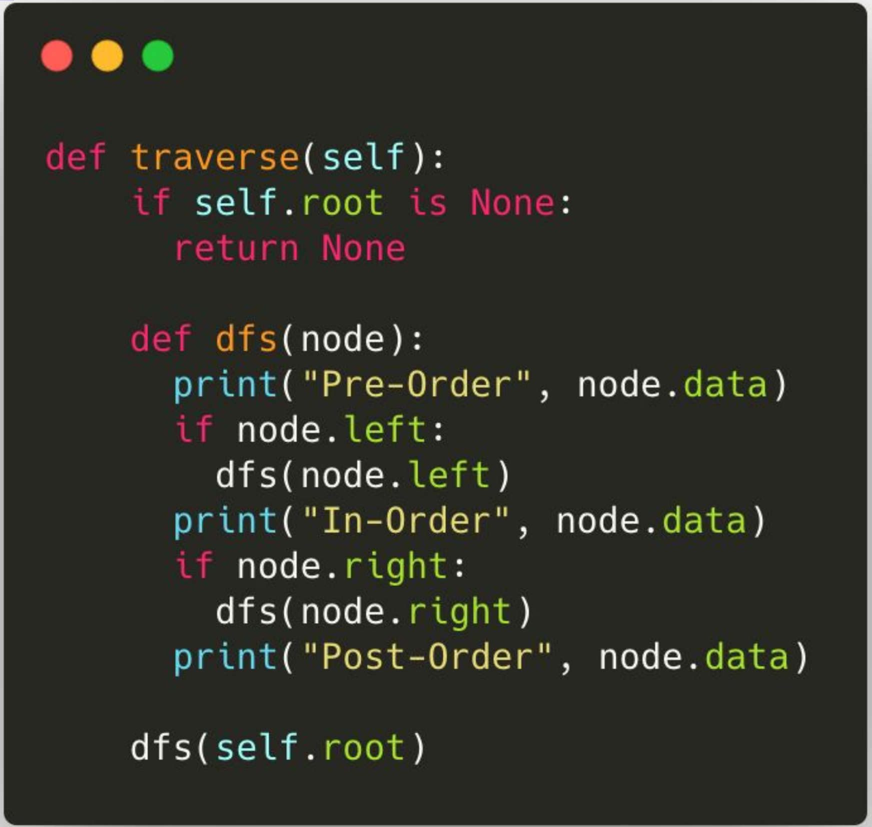




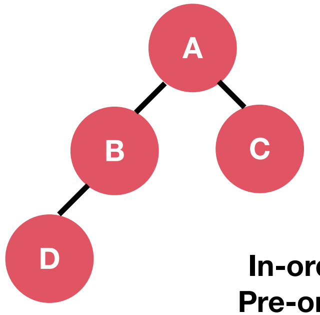
TREE TRAVERSALS

- Binary Tree problems require you to traverse the tree and manipulate/get values from it
- Two ways to traverse:
  - Depth-first search (DFS)
    - In-order, Pre-order, Post-order
  - Breadth-first search (BFS)
    - Level-order
- *Hint: Doing these traversals should be muscle memory!*

DEPTH-FIRST SEARCH (DFS)



```
def traverse(self):
    if self.root is None:
      return None

    def dfs(node):
      print("Pre-Order", node.data)
      if node.left:
        dfs(node.left)
      print("In-Order", node.data)
      if node.right:
        dfs(node.right)
      print("Post-Order", node.data)

    dfs(self.root)
```



In-order: [D, B, A, C]
Pre-order: [A, B, D, C]
Post-order: [D, B, C, A]

In-order (left, current, right)

- *Hint: In-order = Left to right*
- *This traversal in a BST gives you the sorted order*
- Reverse In-order (right, current, left)

Pre-order (current, left, right)

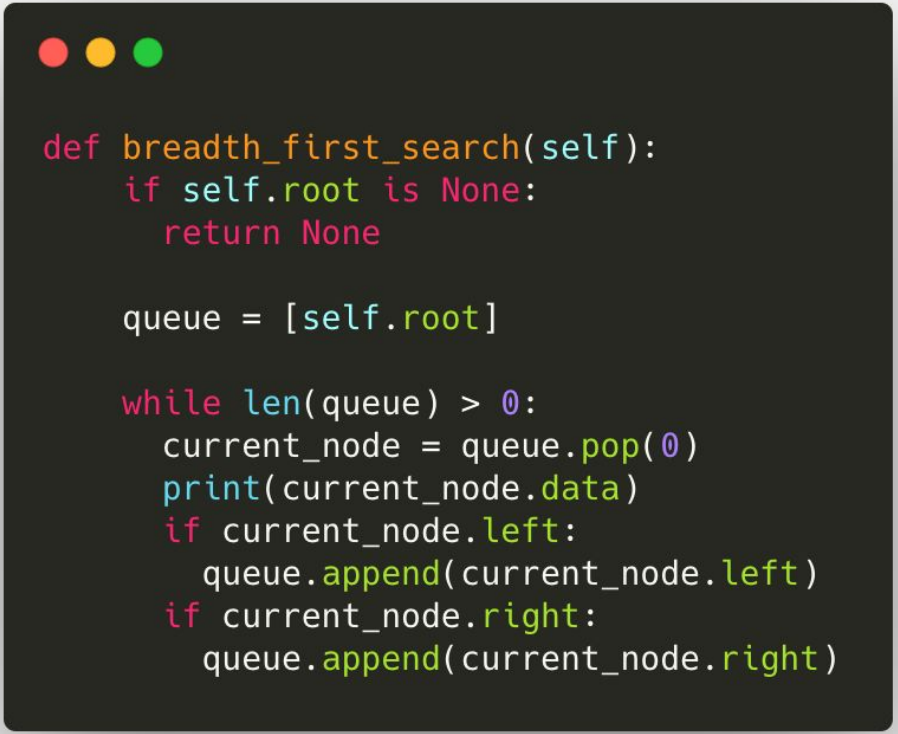
- *Hint: Pre = Before*
- Reverse Pre-order (current, right, left)

Post-order (left, right, current)

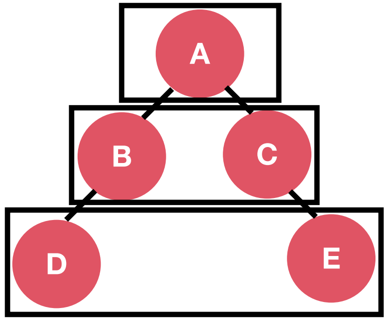
- *Hint: Post = After*
- Reverse Post-order (right, left, current)

- *Keywords: max, deepest, longest*

BREADTH-FIRST SEARCH (BFS)



```python
def breadth_first_search(self):
    if self.root is None:
      return None

    queue = [self.root]

    while len(queue) > 0:
      current_node = queue.pop(0)
      print(current_node.data)
      if current_node.left:
        queue.append(current_node.left)
      if current_node.right:
        queue.append(current_node.right)
```



Level-order: [A, B, C, D, E]

- Traverse the tree in a level-order fashion
- Can be implemented using a queue
- *Keywords: level, row, closest, minimum, width, diameter*

BINARY TREES AND UMPIRE

- Understand
  - Does the tree have any special properties?
  - What are some cases I need to watch out for?
- Match
  - What type of traversal should I do?
- Plan
  - Should I use recursion/iterative? What's my base/recursive case? What should I do w/ the nodes/subtrees?

Breakout Session

BREAKOUT SESSIONS

- We'll go into breakout rooms for 1 hour
- For each problem, do them individually using UMPIRE
- Someone volunteer and present their solution using UMPIRE
- Others give feedback/discuss
- Rinse and repeat for the other problem

BREAKOUT SESSIONS RECAP

- Min Depth of Tree - using BFS, enqueueing other data in the queue
- Pruning Binary Tree - using recursive post-order traversal

WHAT TYPE OF DFS TRAVERSAL SHOULD I DO?

- For some problems, it doesn’t matter as long as you traverse all the nodes
- In-order
  - Going through the sorted order (ascending/descending) in a BST
- Pre-order
  - Processing the current node and passing down an updated value down to the children
- Post-order
  - Processing children first before figuring out what to do with the current node (e.g. pruning binary tree)

Behavioral Interview Questions

WHAT ARE BEHAVIORAL INTERVIEW QUESTIONS?

Tell me about a time when you…

Had a conflict with a coworker or teammate?

Disagreed with your manager about how to get something done?

Missed a deadline, and how you handled it?

Communicated bad news successfully?

Had to set a boundary with a teammate?

HOW TO PREPARE: STAR TECHNIQUE

STAR Technique:

Situation

Task

Action

Result

HOW TO PREPARE: STAR TECHNIQUE

Situation: I was an engineer at Square; I was a student at Cal State Monterey Bay; etc

Task

Action

Result

HOW TO PREPARE: STAR TECHNIQUE

Situation:

“I was an engineer at Square”; “I was a student at Cal State Monterey Bay”; etc. **College experience counts!**

“I had a thesis supervisor”, “The manager of the intern team”, etc

Give details about where, what your team was, what you were responsible for. If another person is involved, explain their role as well.

Task:

“We were trying to fix TCP Incasting”

“Our old JSP web server needed replacing”

“We were moving to React 16”

“It was an open-source project and I had a few issues assigned to me”

“This was a college team project, we were building a fantasy racing league”

“I was building a website for my uncle”

College experiences count! Family experiences count!

HOW TO PREPARE: STAR TECHNIQUE

Task:

“My supervisor wanted me to skip writing tests, even though we already had 0 test coverage.”

“One of my teammates dropped out and my other teammate decided to write all the code alone over a weekend”

“My thesis supervisor thought my project was too focused on gaming, and they didn’t think it was a good career path for me”

Describe the conflict or problem where your behavior defined what happened next. Show you understand.

HOW TO PREPARE: STAR TECHNIQUE

Action: How You Handled It

“I spent the extra time to write the tests, but I made sure to get the main functionality in on time.”

“I reviewed all the code my teammate wrote, then sat down with them to plan the next phase and decide what to rewrite. We also talked about how to deal with frustrating teammates.”

“I found an engineer at my favorite company who mentioned being open to mentorship, and sent them a message to get some outside support.”

HOW TO PREPARE: STAR TECHNIQUE

Result: What happened next & how well did your approach work?

“My supervisor was pleased we had both the functionality and the test coverage- it inspired the other interns to complete the test harness and our time-to-release went way down because we could rely on the tests.”

“My teammate realized being a lone hero wasn’t enough to finish the project, and learned about communication. We finished the project with a 3-day extension and it was a great learning experience.”

“I got the support I needed to finish my project on my own and even landed an internship at my favorite gaming company!”

HOW TO PREPARE: STAR TECHNIQUE

What happens if the situation didn’t go well?

- Own your own mistakes and share what you learned
- Show your own growth and experiences
- Accepting Responsibility === Growth Mindset 🧠💪

This will NOT tank your interview! It signals strength to accept responsibility and share a growth mindset!

Additional Resources

- STAR Technique
- 30 Practice Behavioral Questions
- Growth Mindset in Interviews
- UMPIRE Walkthrough for Reverse-Level-Order Traversal

Questions?