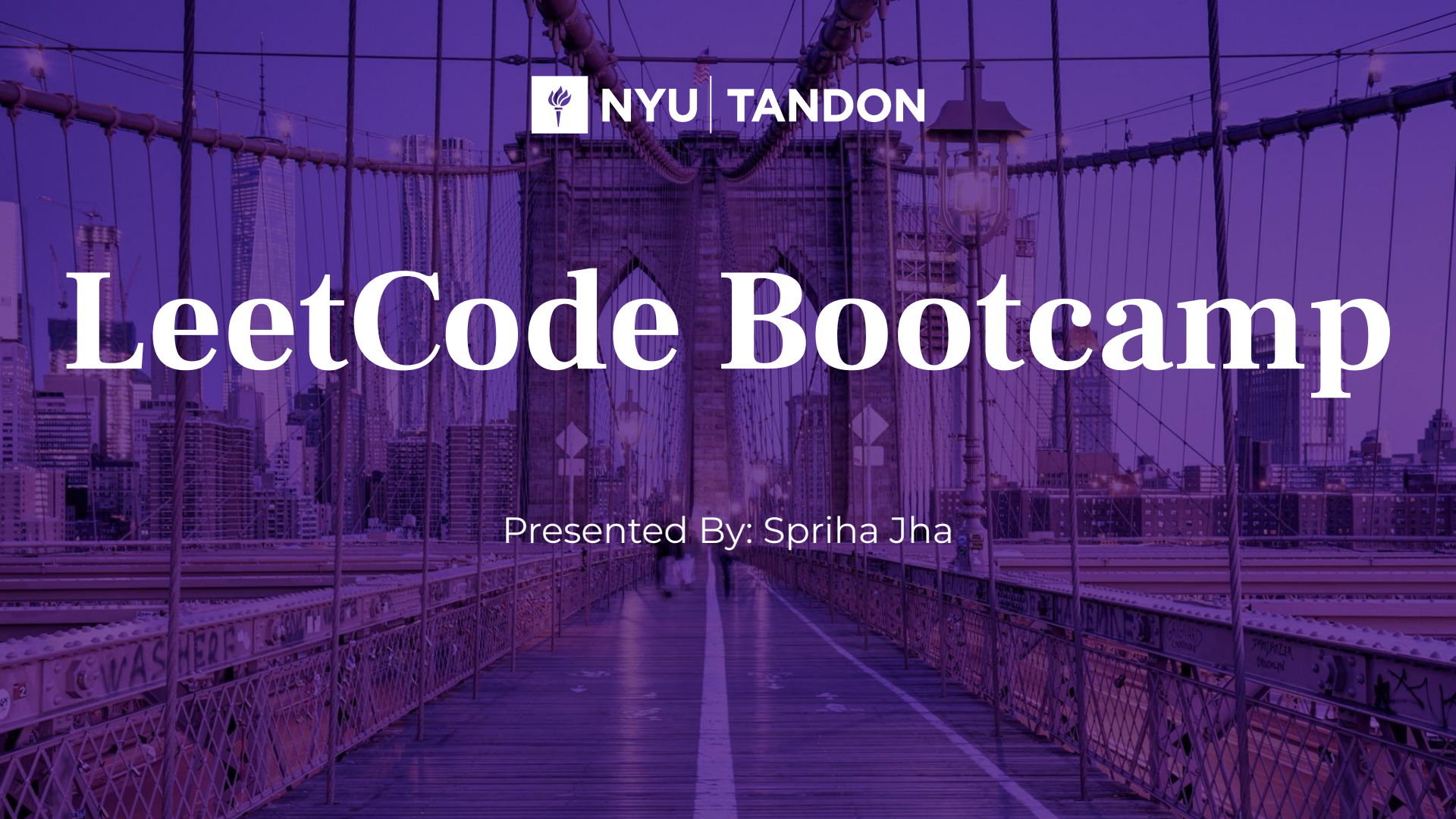

NYU TANDON
LeetCode Bootcamp
Presented By: Spriha Jha

Session Outline

01. Introduction to Graphs, Stacks & Queues

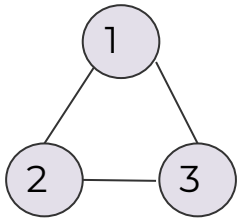
02. Problem Sets

03. Debrief & Q/A

Graph

- A ***graph*** is a data structure with two distinct parts: a finite set of vertices, which are also called nodes, and a finite set of edges, which are references/links/pointers from one vertex to another.
- In directed graphs, the connections between nodes have a direction, and are called ***arcs***; in undirected graphs, the connections have no direction and are called ***edges***.
- Sometimes the nodes or arcs of a graph have weights or costs associated with them, and we are interested in finding the cheapest path.
- The characteristics of graph are strongly tied to what its vertices and edges look like. (Dense/Sparse)

Graph Representation



- As a list (or array) is called an ***edge list***, and is a representation of all the edges (*|E|*) in the graph. [[1, 2],[2, 3],[3, 1]]
- Or, an ***adjacency matrix*** is a matrix representation of exactly *which* nodes in a graph contain edges between them. [[0, 1, 1],[1, 0, 1],[1, 1, 0]]
- The matrix is kind of like a lookup table: once we've determined the two nodes that we want to find an edge between, we look at the value at the intersection of those two nodes.
- The values in the adjacency matrix are like boolean flag indicators; they are either present or not present. If the value is 1, that means that there is an edge between the two nodes; if the value is 0, that means an edge does not exist between them.
- We will have a value of 0 down the diagonal, since most graphs that we're dealing with won't be referential.

Graph Representation

Adjacency Matrix			
	0	**1**	**2**
0	0	1	1
1	1	0	1
2	1	1	0

Adjacency List		
0:	1	2
1:	0	2
2:	0	1

Edge List		
0:	0	1
1:	0	2
2:	1	2

- ***Adjacency list*** hybrid between an *edge list* and an *adjacency matrix*.
- Each vertex is given an index in its list, and has all of its neighboring vertices stored as an linked list (which could also be an array), adjacent to it.
- We can see that, because of the *structure* of an adjacency list, it's very easy to determine all the neighbors of one particular vertex.
- The ***degree*** of a vertex is the number of edges that it has, which is also known as the number of neighboring nodes that it has.

PART 02

Problem Sets

Steps to approach the question:

Understand the problem

Take time to carefully read through the problem from start to finish is critical in finding the correct and complete solution to the problem in hand.

Code your solution

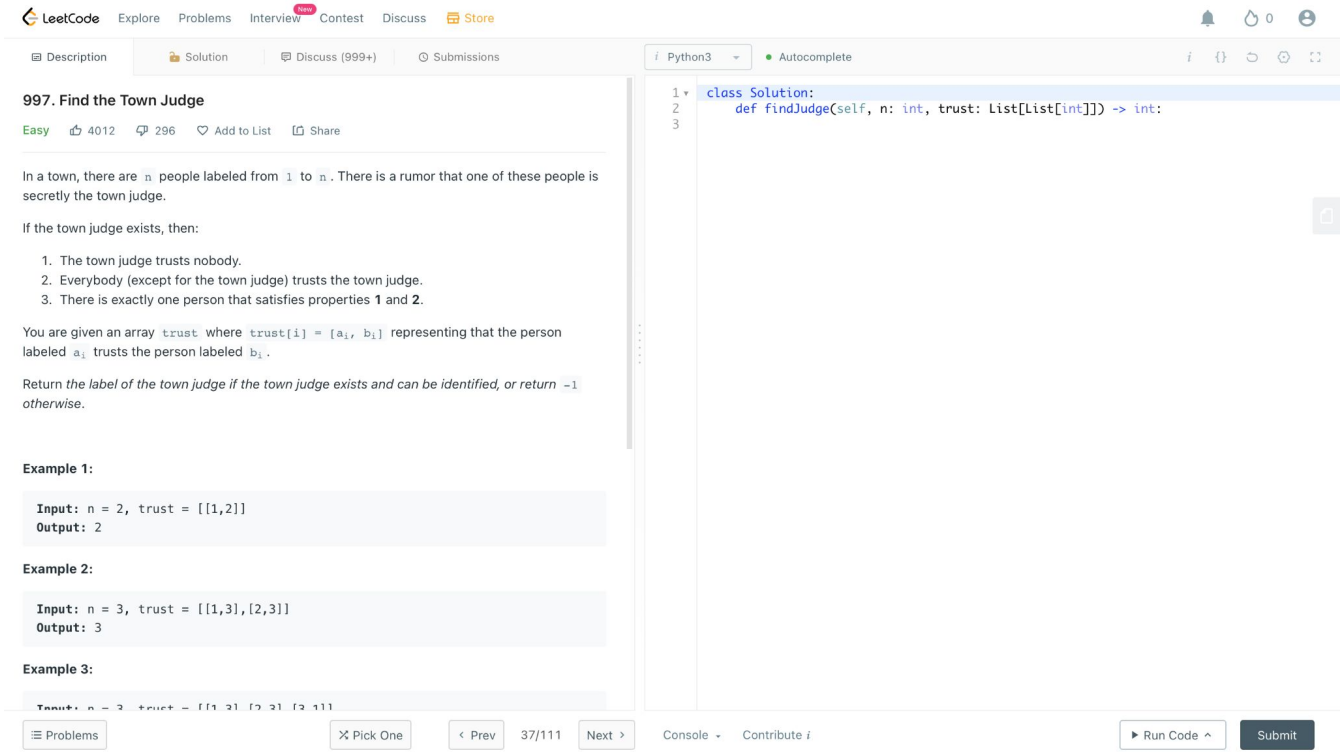
Map out your solution before you write any code. Avoid too much time trying to find the perfect solution. Validate your solution early and often.

Manage your time

Don’t forget, you have multiple questions to complete within a said time. Make sure you allocate enough time to carefully consider all problems.

PART 02

Problem 1: Find the town judge



NYU TANDON

Approach: One Array

```
def findJudge(self, N: int, trust: List[List[int]]) -> int:

    if len(trust) < N - 1:
        return -1

    trust_scores = [0] * (N + 1)

    for a, b in trust:
        trust_scores[a] -= 1
        trust_scores[b] += 1

    for i, score in enumerate(trust_scores[1:], 1):
        if score == N - 1:
            return i
    return -1
```

Complexity Analysis

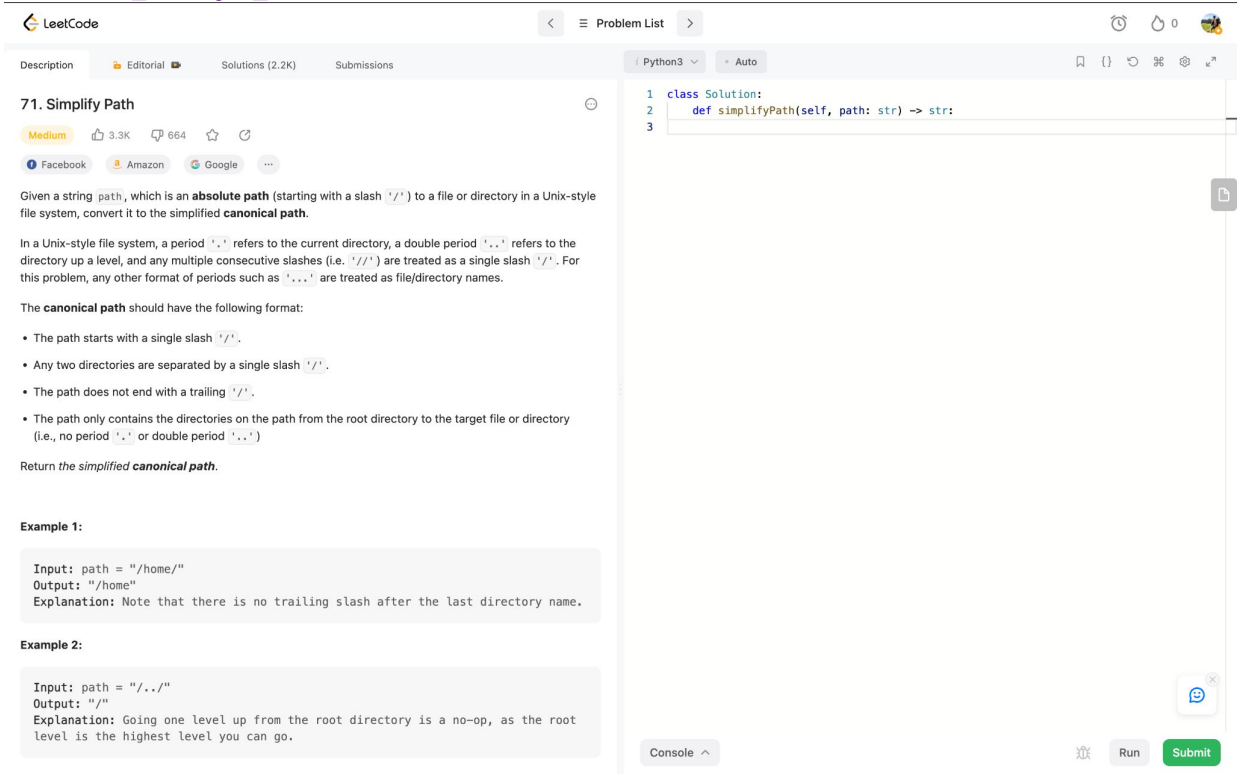
N is the *number of people*, and *E* is the *number of edges* (trust relationships).

Time complexity : $O(E)$.

Space complexity : $O(N)$.

- Each person gains 1 "point" for each person they are trusted by, and loses 1 "point" for each person they trust. Then at the end, the town judge, if they exist, must be the person with N - 1 "points".
- Therefore, for a person to maximize their "score", they should be trusted by as many people as possible, and trust as few people as possible.
- In graph theory, we say the **outdegree** of a vertex (person) is the number of directed edges going out of it. For this graph, the outdegree of the vertex represents the number of other people that person trusts.
- Likewise, we say that the **indegree** of a vertex (person) is the number of directed edges going *into* it. So here, it represents the number of people *trusted by* that person.
- The maximum indegree is N - 1. This represents everybody trusting the person (except for themselves, they cannot trust themselves). The minimum indegree is 0. This represents not trusting anybody. Therefore, the maximum value for indegree - outdegree is **(N - 1) - 0 = N - 1**. These values also happen to be the definition of the town judge!

Problem 2: Simplify path



Approach: Stacks

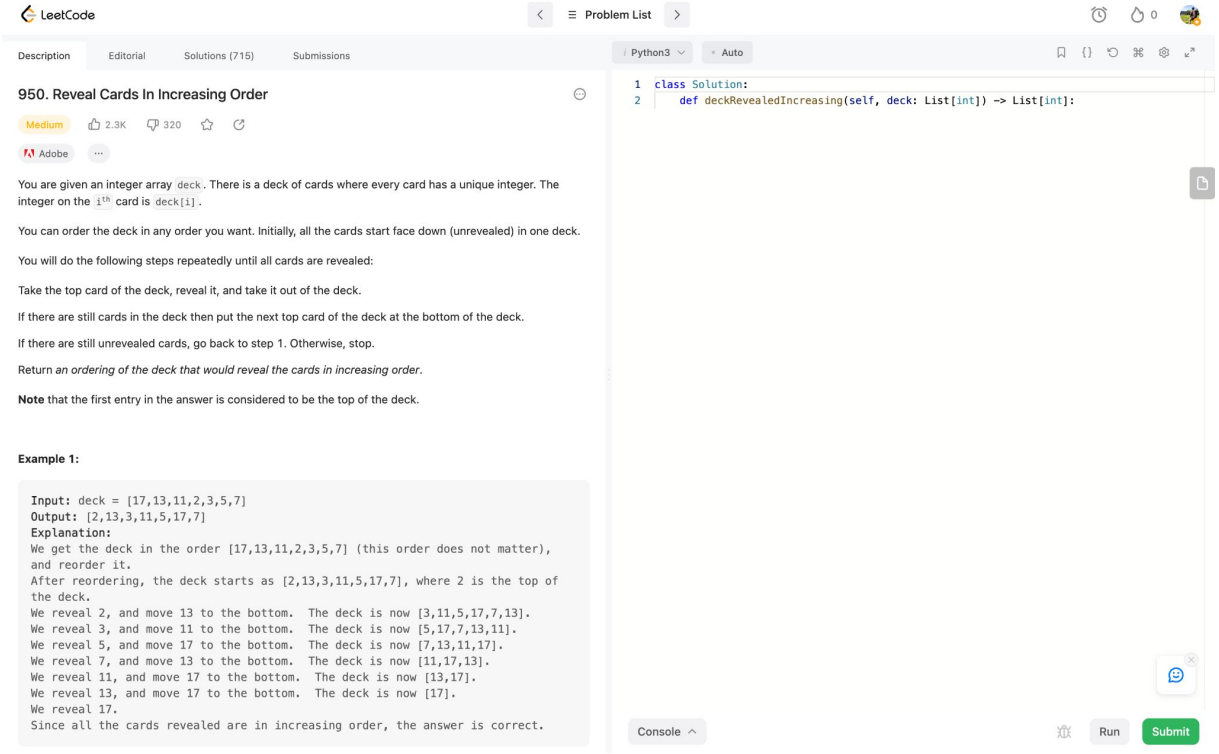
```
def simplifyPath(self, path: str) -> str:
    # Initialize a stack
    stack = []

    # Split the input string on "/" as the delimiter and process each portion one by one
    for portion in path.split("/"):

        # If the current component is a "..", then we pop an entry from the stack if it's non-empty
        if portion == "..":
            if stack:
                stack.pop()
        elif portion == "." or not portion:
            # A no-op for a "." or an empty string
            continue
        else:
            # Finally, a legitimate directory name, so we add it to our stack
            stack.append(portion)

    # Stitch together all the directory names together
    final_str = "/" + "/".join(stack)
    return final_str
```

Problem 3: Reveal cards in increasing order



Approach: Queue

```
def deckRevealedIncreasing(self, deck: List[int]) -> List[int]:
    N = len(deck)
    index = collections.deque(range(N))
    ans = [None] * N

    for card in sorted(deck):
        ans[index.popleft()] = card
        if index:
            index.append(index.popleft())

    return ans
```

PART 06

Q/A


NYU TANDON
Thank you!
Upcoming: Tree, Trie & Heap