NYU | TANDON

LeetCode Bootcamp

Presented By: Spriha Jha

Session Outline

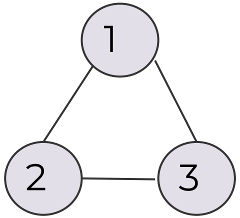
NYU | TANDON

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.

NYU TANDON

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.

NYU | TANDON

PART 02

Problem Sets

Steps to approach the question:

Understand the problem	Code your solution	Manage your time
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.	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.	Don’t forget, you have multiple questions to complete within a said time. Make sure you allocate enough time to carefully consider all problems.

Problem 1: Find the town judge

LeetCode Explore Problems Interview New Contest Discuss Store 0

Description Solution Discuss (999+) Submissions

997. Find the Town Judge

Easy 4012 296 Add to List Share

In a town, there are `n` people labeled from `1` to `n`. There is a rumor that one of these people is secretly the town judge.

If the town judge exists, then:

1. The town judge trusts nobody.
2. Everybody (except for the town judge) trusts the town judge.
3. There is exactly one person that satisfies properties **1** and **2**.

You are given an array `trust` where `trust[i] = [a_i, b_i]` representing that the person labeled `a_i` trusts the person labeled `b_i`.

Return *the label of the town judge if the town judge exists and can be identified, or return* `-1` *otherwise*.

Example 1:

```
Input: n = 2, trust = [[1,2]]
Output: 2
```

Example 2:

```
Input: n = 3, trust = [[1,3],[2,3]]
Output: 3
```

Example 3:

```
Input: n = 3, trust = [[1,3],[2,3],[3,1]]
```

Python3 Autocomplete

```python
class Solution:
    def findJudge(self, n: int, trust: List[List[int]]) -> int:

```

Problems Pick One Prev 37/111 Next Console Contribute Run Code Submit

Approach: One Array

```python
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

LeetCode

Problem List

Description | Editorial | Solutions (2.2K) | Submissions

71. Simplify Path

Medium 3.3K 664

Facebook Amazon Google

Given a string `path`, which is an **absolute path** (starting with a slash `'/'`) to a file or directory in a Unix-style file system, convert it to the simplified **canonical path**.

In a Unix-style file system, a period `'.'` refers to the current directory, a double period `'..'` refers to the directory up a level, and any multiple consecutive slashes (i.e. `'//'`) are treated as a single slash `'/'`. For this problem, any other format of periods such as `'...'` are treated as file/directory names.

The **canonical path** should have the following format:

- The path starts with a single slash `'/'`.
- Any two directories are separated by a single slash `'/'`.
- The path does not end with a trailing `'/'`.
- The path only contains the directories on the path from the root directory to the target file or directory (i.e., no period `'.'` or double period `'..'`)

Return *the simplified **canonical path***.

Example 1:

```
Input: path = "/home/"
Output: "/home"
Explanation: Note that there is no trailing slash after the last directory name.
```

Example 2:

```
Input: path = "/../"
Output: "/"
Explanation: Going one level up from the root directory is a no-op, as the root
level is the highest level you can go.
```

Python3 Auto

```
class Solution:
    def simplifyPath(self, path: str) -> str:

```

Console Run Submit

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

LeetCode | Problem List

Description | Editorial | Solutions (715) | Submissions

950. Reveal Cards In Increasing Order

Medium | 2.3K | 320

Adobe

You are given an integer array `deck`. There is a deck of cards where every card has a unique integer. The integer on the i^{th} card is `deck[i]`.

You can order the deck in any order you want. Initially, all the cards start face down (unrevealed) in one deck.

You will do the following steps repeatedly until all cards are revealed:

Take the top card of the deck, reveal it, and take it out of the deck.

If there are still cards in the deck then put the next top card of the deck at the bottom of the deck.

If there are still unrevealed cards, go back to step 1. Otherwise, stop.

Return *an ordering of the deck that would reveal the cards in increasing order*.

Note that the first entry in the answer is considered to be the top of the deck.

Example 1:

```
Input: deck = [17,13,11,2,3,5,7]
Output: [2,13,3,11,5,17,7]
Explanation:
We get the deck in the order [17,13,11,2,3,5,7] (this order does not matter),
and reorder it.
After reordering, the deck starts as [2,13,3,11,5,17,7], where 2 is the top of
the deck.
We reveal 2, and move 13 to the bottom.  The deck is now [3,11,5,17,7,13].
We reveal 3, and move 11 to the bottom.  The deck is now [5,17,7,13,11].
We reveal 5, and move 17 to the bottom.  The deck is now [7,13,11,17].
We reveal 7, and move 13 to the bottom.  The deck is now [11,17,13].
We reveal 11, and move 17 to the bottom.  The deck is now [13,17].
We reveal 13, and move 17 to the bottom.  The deck is now [17].
We reveal 17.
Since all the cards revealed are in increasing order, the answer is correct.
```

Python3 | Auto

```python
class Solution:
    def deckRevealedIncreasing(self, deck: List[int]) -> List[int]:
```

Console | Run | Submit

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
```

Problem 4: Course Schedule

Description | Guide | Editorial | Solutions | Submissions

207. Course Schedule

Medium | Topics | Companies | Hint

There are a total of `numCourses` courses you have to take, labeled from `0` to `numCourses - 1`. You are given an array `prerequisites` where `prerequisites[i] = [a_i, b_i]` indicates that you **must** take course `b_i` first if you want to take course `a_i`.

- For example, the pair `[0, 1]`, indicates that to take course `0` you have to first take course `1`.

Return `true` if you can finish all courses. Otherwise, return `false`.

Example 1:

```
Input: numCourses = 2, prerequisites = [[1,0]]
Output: true
Explanation: There are a total of 2 courses to take.
To take course 1 you should have finished course 0. So it is possible.
```

Example 2:

```
Input: numCourses = 2, prerequisites = [[1,0],[0,1]]
Output: false
Explanation: There are a total of 2 courses to take.
To take course 1 you should have finished course 0, and to take course 0 you should also have finished course 1. So it is impossible.
```

Constraints:

- `1 <= numCourses <= 2000`
- `0 <= prerequisites.length <= 5000`

15.3K 614 121

Code

Python3 · Auto

```python
class Solution:
    def canFinish(self, numCourses: int, prerequisites: List[List[int]]) -> bool:

```

Saved to cloud Ln 1, Col 1

Testcase | Result

You must run your code first

PART 06

Q/A

Thank you!

Upcoming: Tree, Trie & Heap