Newton School of Technology

Join the lecture online on your dashboard.

Newton School of Technology

15

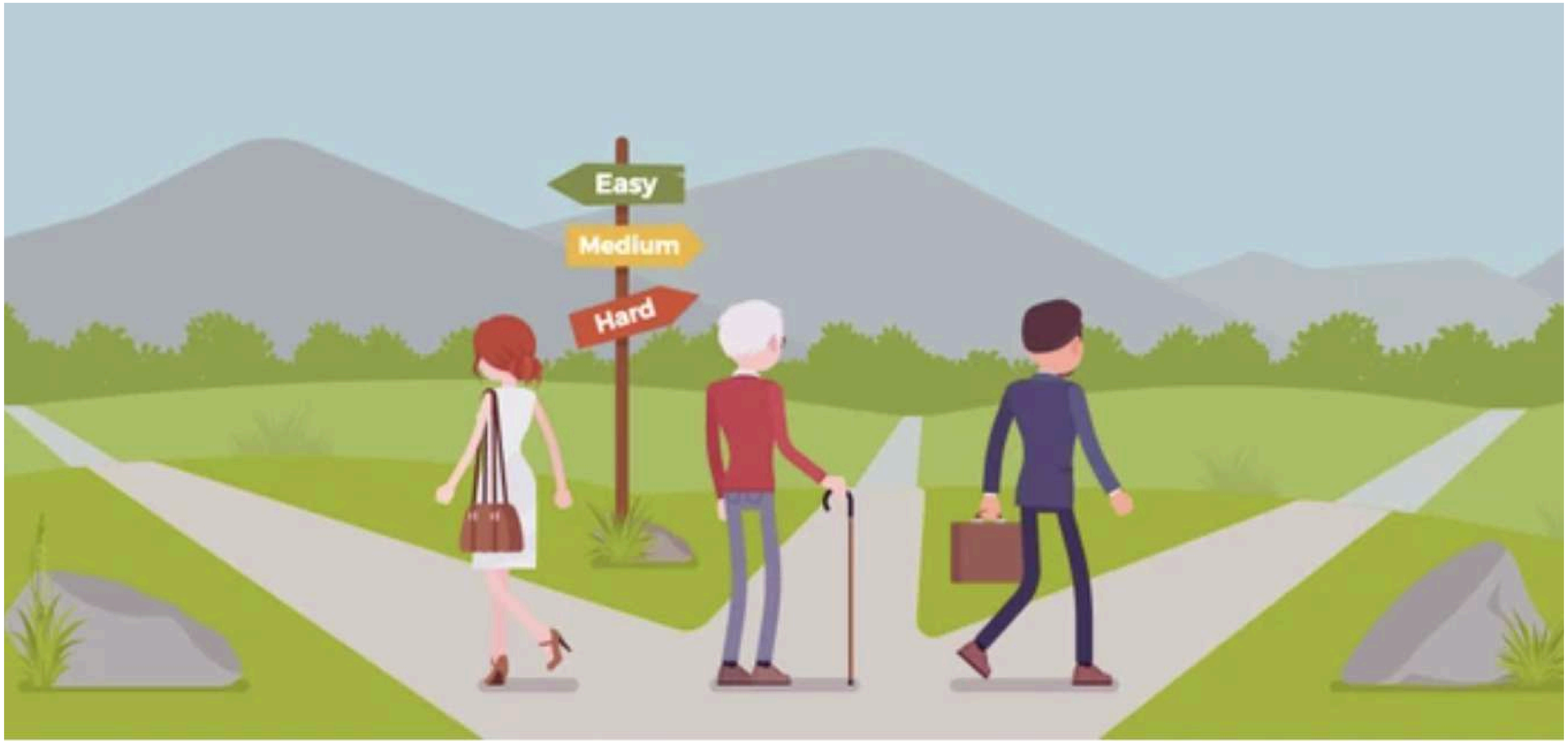
Greedy : 1

by <Instructor>

Analysis and Design of Algorithms

Newton School of Technology

Greedy: The "What's the best I can do right now?" Algorithm



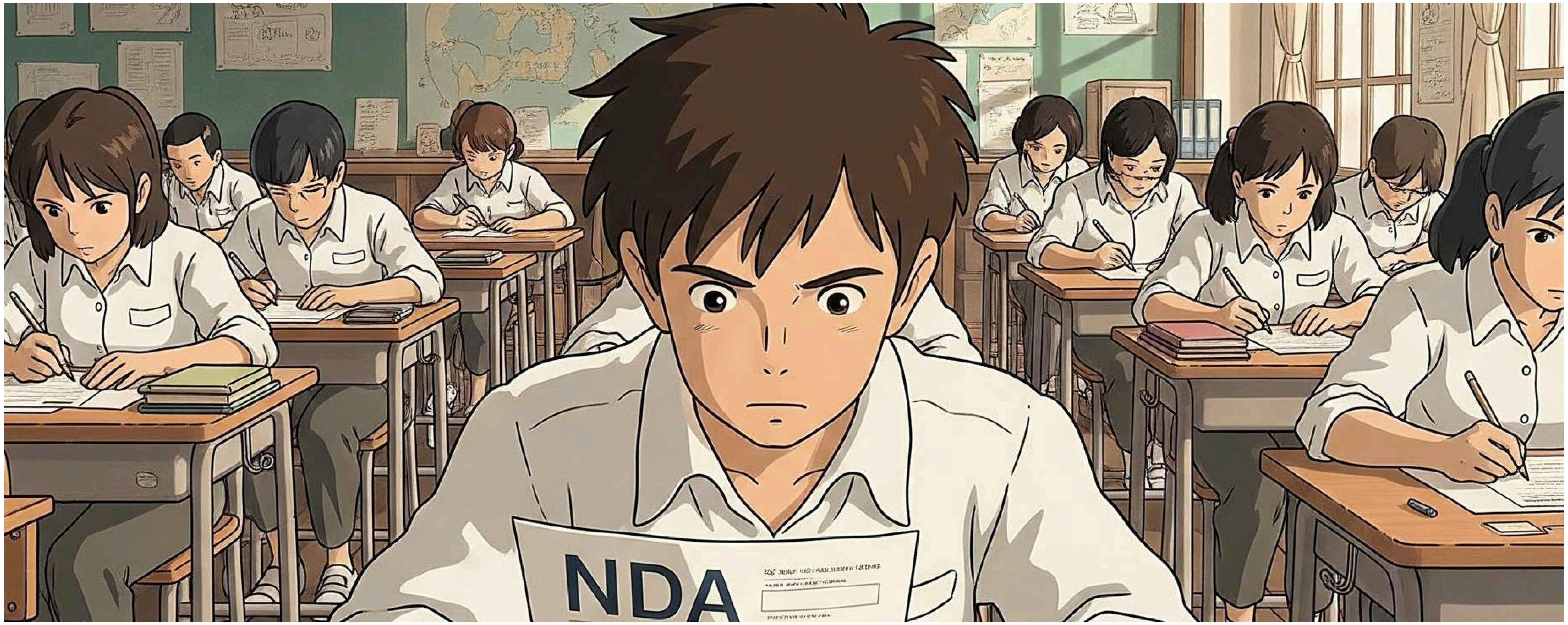
Greedy algorithms are like the NDA selection process :

At every stage, a locally **best decision** is made, with the aim of reaching the best overall outcome (the final merit list).

Newton School of Technology

Step 1 : Written Exam (Maximize Score)

You make the best decision at each question to maximize your score in the limited time.



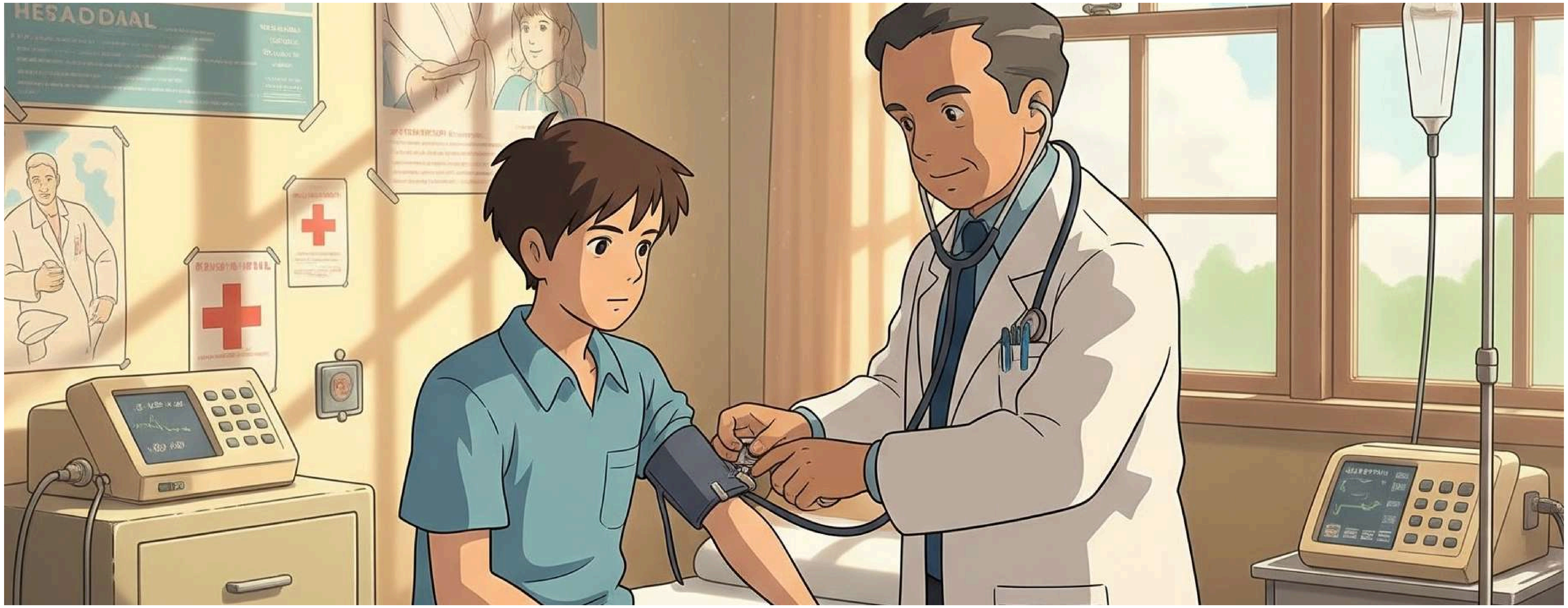
Step 2: SSB Interview (Stage-Wise Elimination):

Each stage is a new, local challenge. You must clear the screening and conference rounds to proceed.

Newton School of Technology

Step 3: Medical Test (Pass/Fail):

The decision is based on your current physical state.
There is no going back to change a past condition.



Newton School of Technology

Step 4: Final Merit List (Only Top Scorers):

The final outcome is the result of a series of locally optimal decisions.

Newton School of Technology

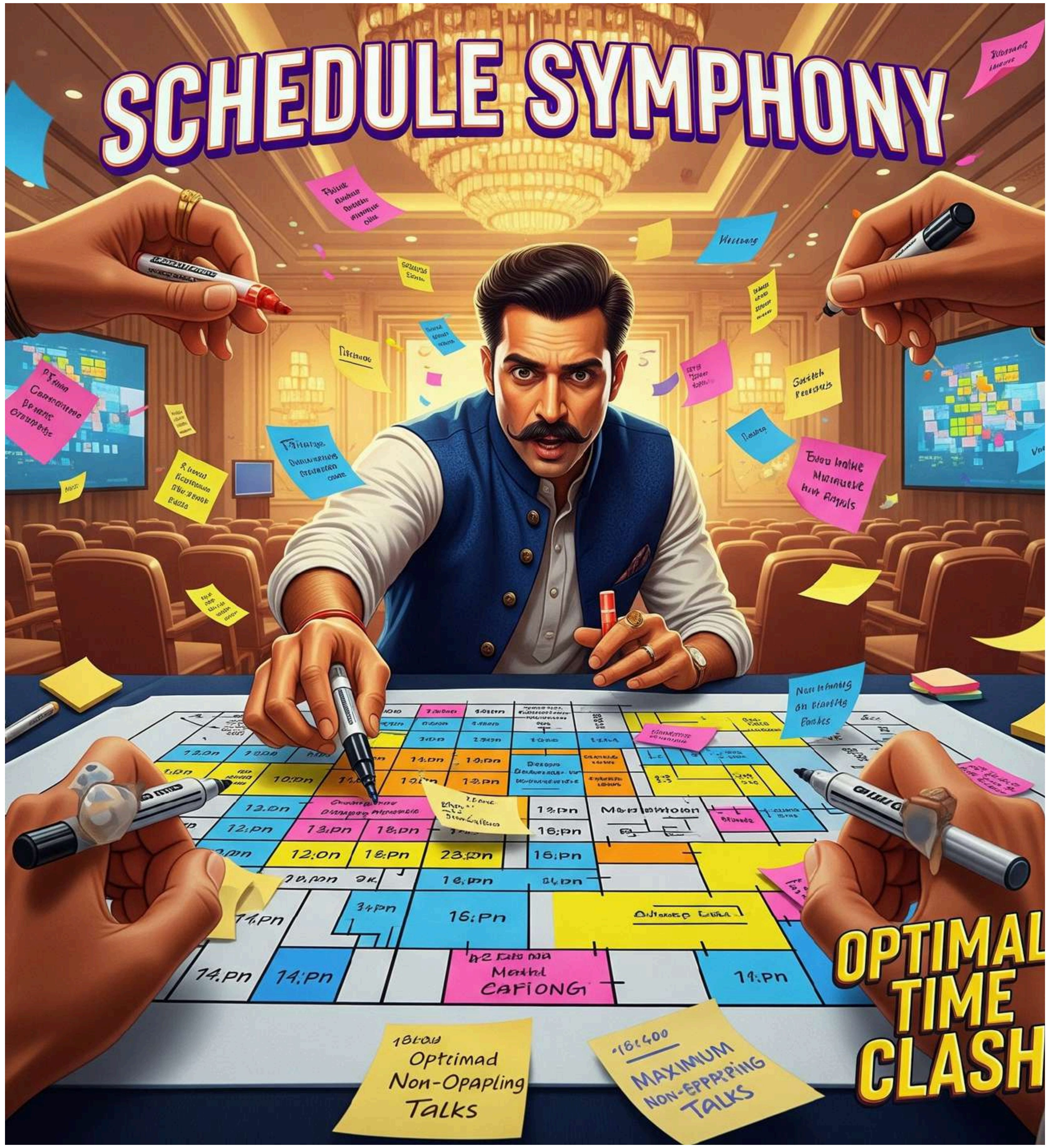
Greedy Algorithm :

An algorithm that follows the problem solving approach of **making the locally optimal choice** at each stage with the hope of finding a **global optimum**.

Newton School of Technology

Non Overlapping Intervals

You have a set of time slots for talks. You need to **remove the minimum** number of talks so that the remaining ones do not overlap.

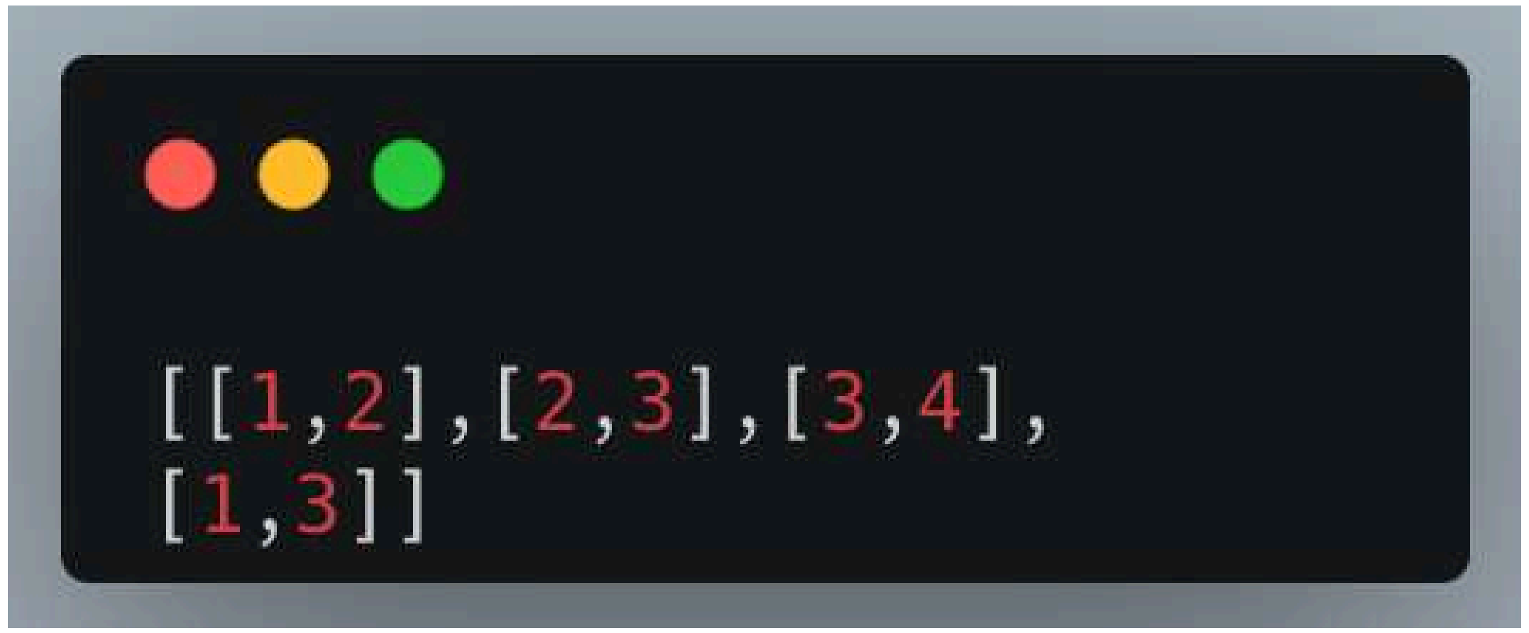


Non Overlapping Intervals

The input for this problem is a **list of time intervals**, represented as a 2D array. Each item in the array is an interval with two numbers:

- The first number is the **start time** (starti)
- The second number is the **end time** (endi)

For e.g **[10, 20] would be a talk that starts at 10:00 and ends at 20:00**



```
[[1,2],[2,3],[3,4],
[1,3]]
```

Non Overlapping Intervals

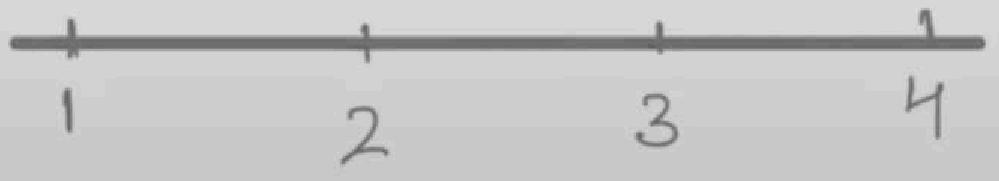
[[1,2],[2,3],
[3,4],[1,3]]

[[1,2],[1,2],
[1,2]]

[[1,2],[2,3]]

Non Overlapping Intervals

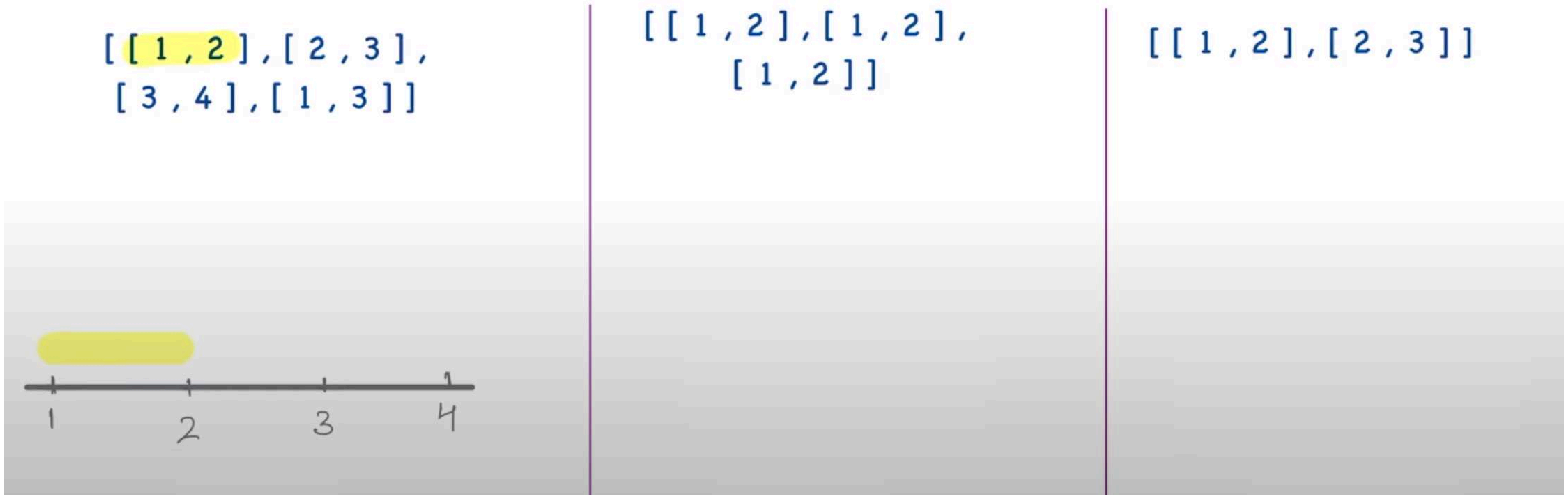
[[1,2],[2,3],
[3,4],[1,3]]



[[1,2],[1,2],
[1,2]]

[[1,2],[2,3]]

Non Overlapping Intervals



[[1,2],[2,3],[3,4],[1,3]]

[[1,2],[1,2],[1,2]]

[[1,2],[2,3]]

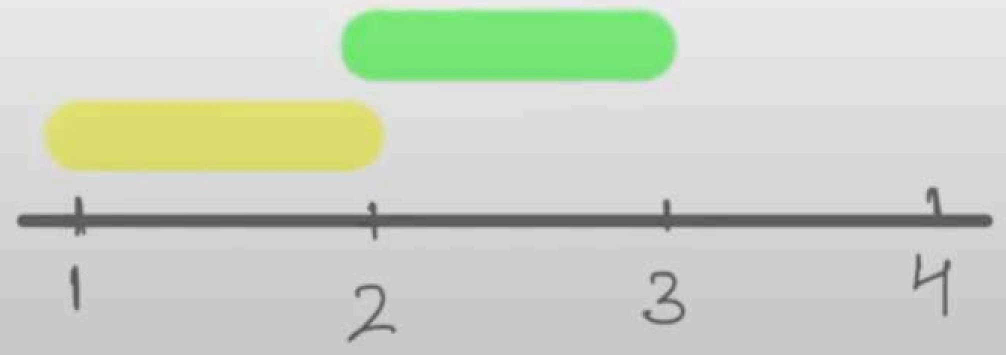
Newton School of Technology

Non Overlapping Intervals

[[1,2],[2,3],
[3,4],[1,3]]

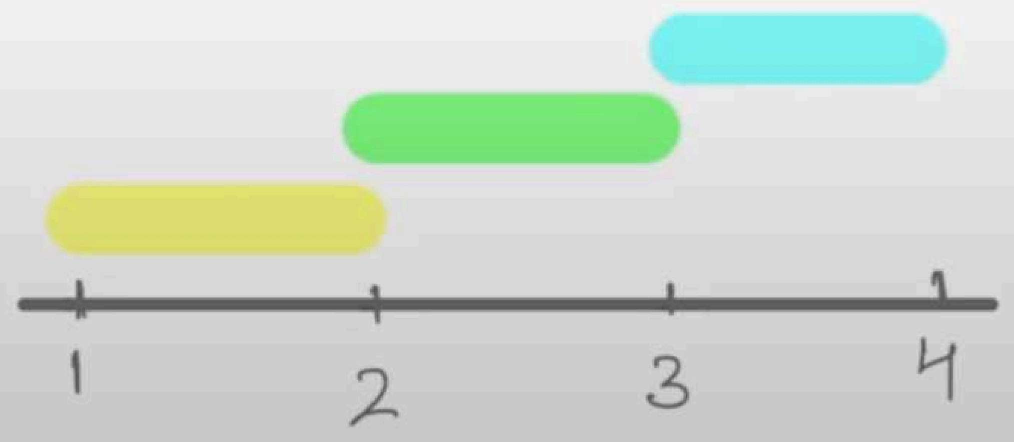
[[1,2],[1,2],
[1,2]]

[[1,2],[2,3]]



Non Overlapping Intervals

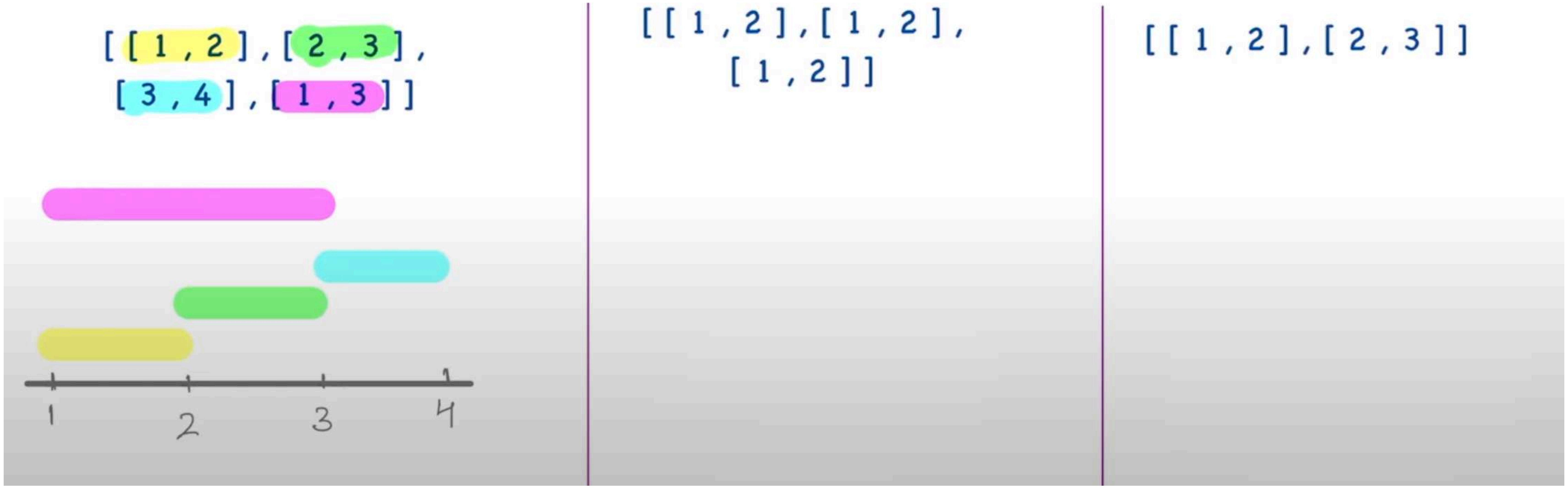
[[1,2],[2,3],
[3,4],[1,3]]



[[1,2],[1,2],
[1,2]]

[[1,2],[2,3]]

Non Overlapping Intervals



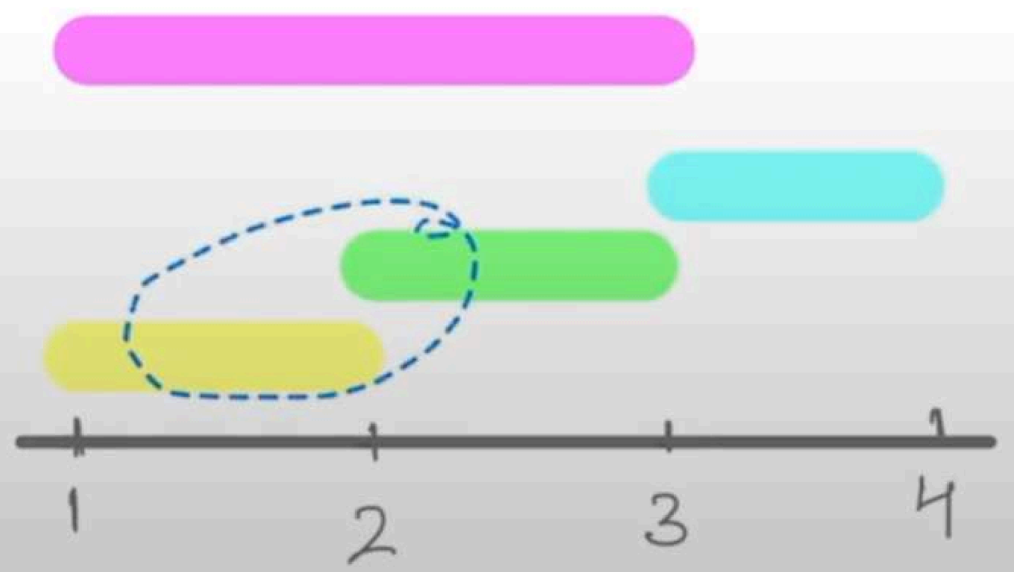
[[1,2],[2,3],
[3,4],[1,3]]

[[1,2],[1,2],
[1,2]]

[[1,2],[2,3]]

Non Overlapping Intervals

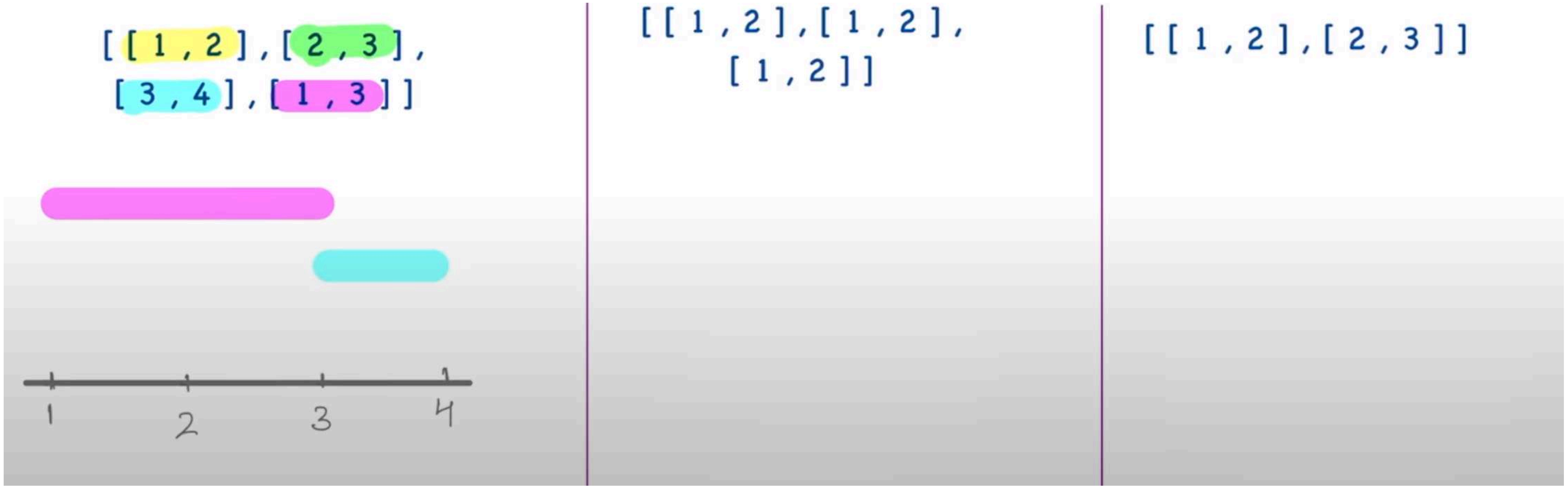
[[1,2],[2,3],
[3,4],[1,3]]



[[1,2],[1,2],
[1,2]]

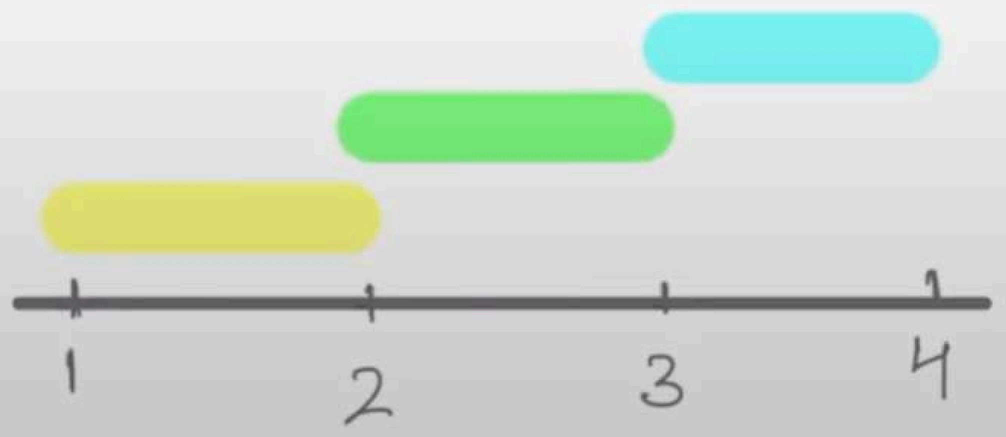
[[1,2],[2,3]]

Non Overlapping Intervals



Non Overlapping Intervals

[[1, 2], [2, 3], [3, 4], [1, 3]]

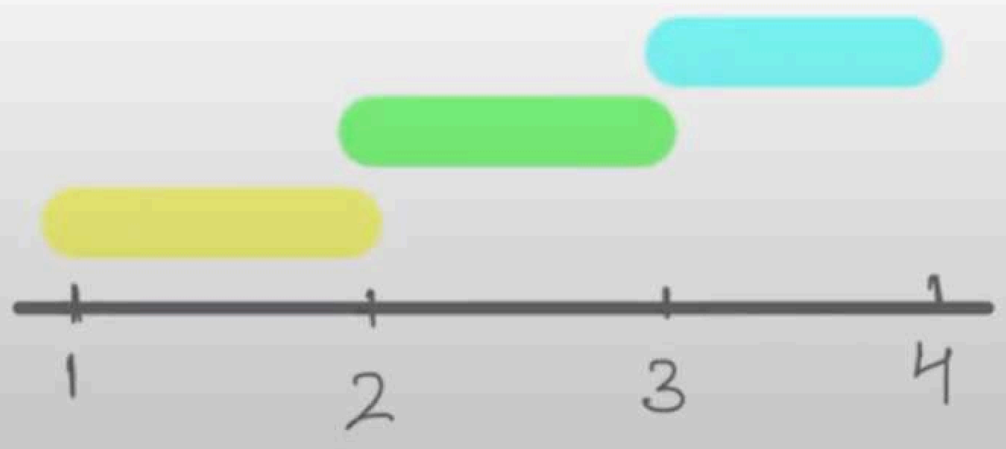


[[1, 2], [1, 2], [1, 2]]

[[1, 2], [2, 3]]

Non Overlapping Intervals

[[1,2],[2,3],
[3,4],[1,3]]

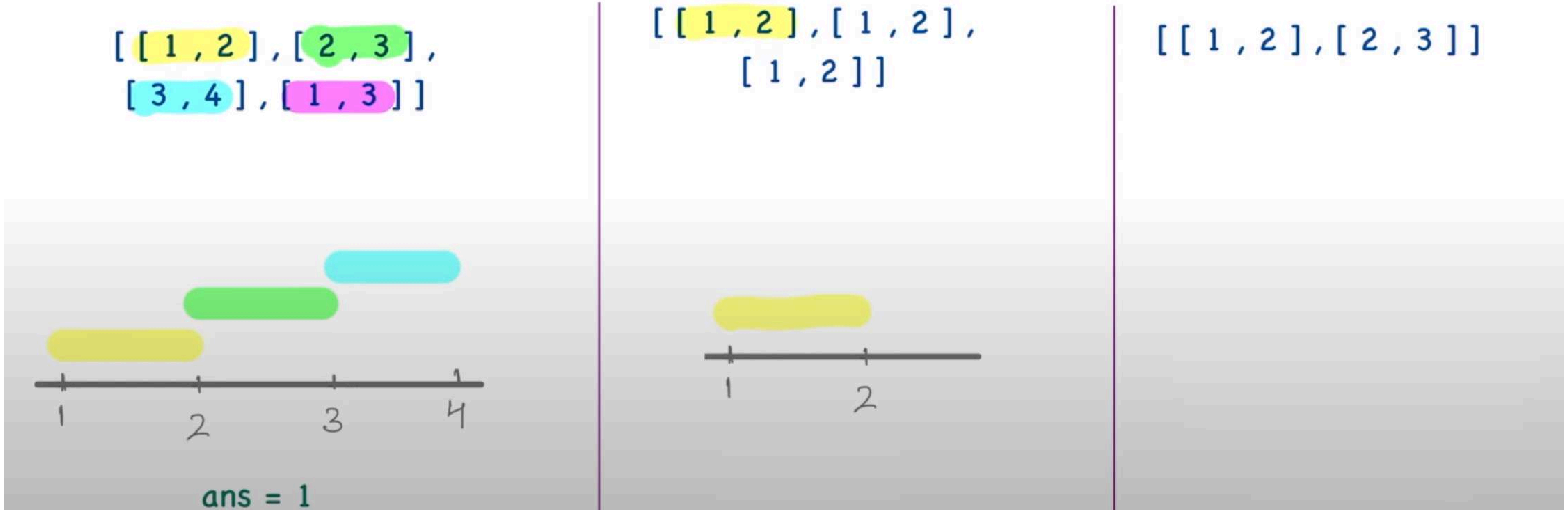


ans = 1

[[1,2],[1,2],
[1,2]]

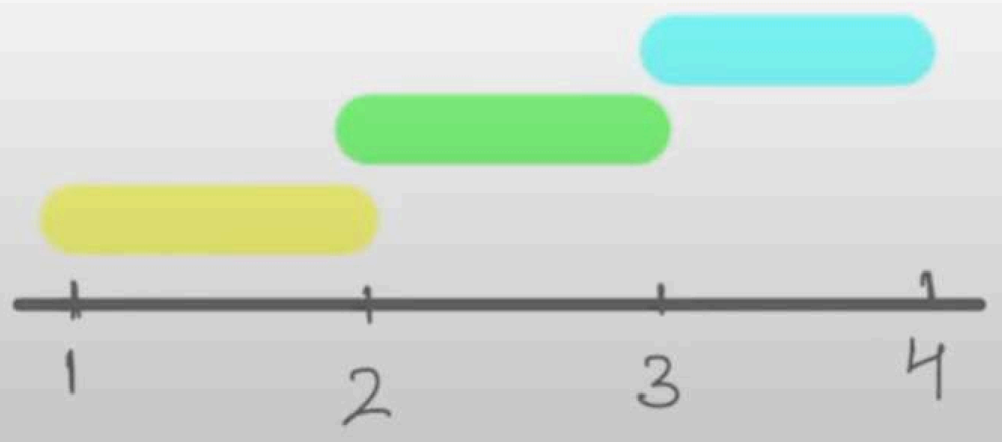
[[1,2],[2,3]]

Non Overlapping Intervals



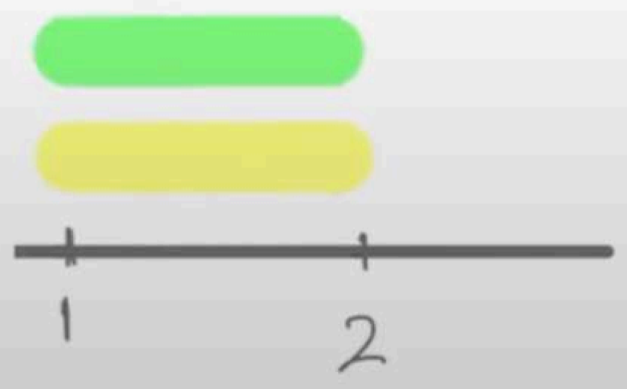
Non Overlapping Intervals

[[1,2],[2,3],
[3,4],[1,3]]



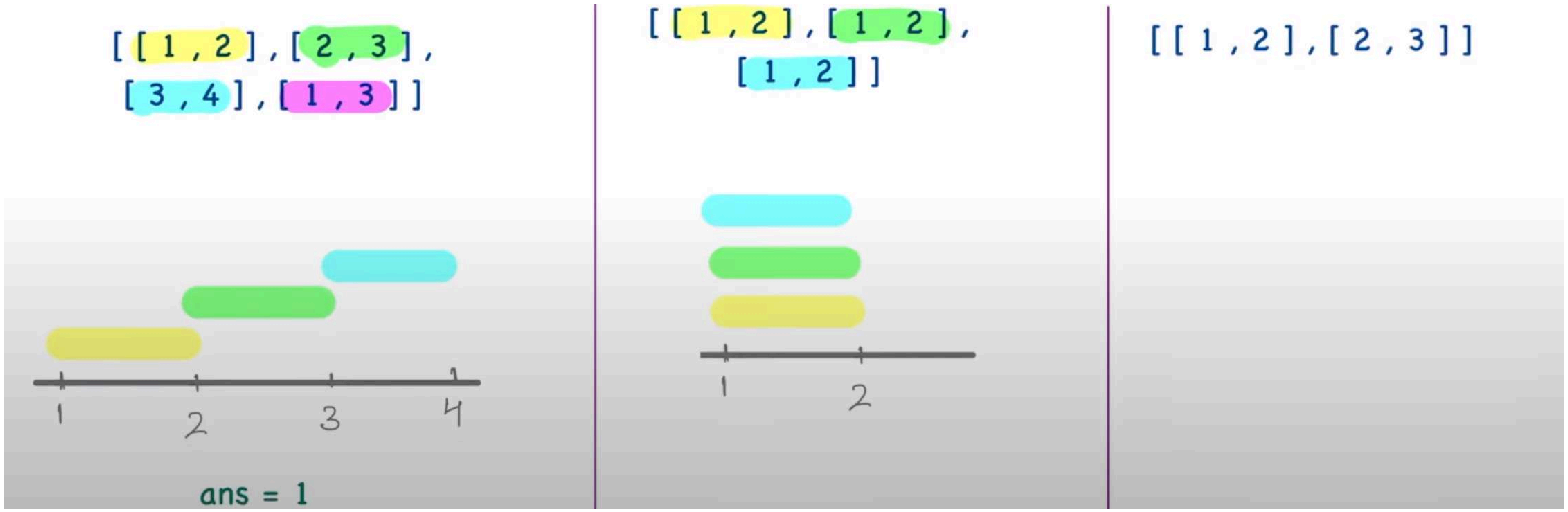
ans = 1

[[1,2],[1,2],
[1,2]]

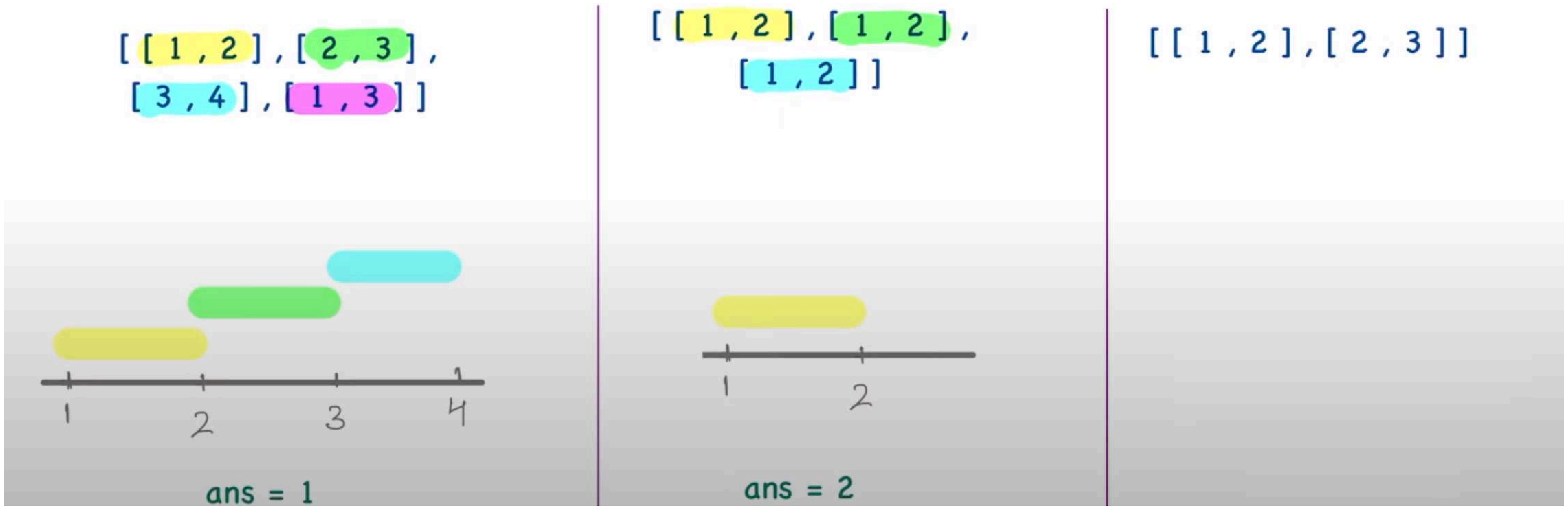


[[1,2],[2,3]]

Non Overlapping Intervals



Non Overlapping Intervals


[[1,2],[2,3],
[3,4],[1,3]]

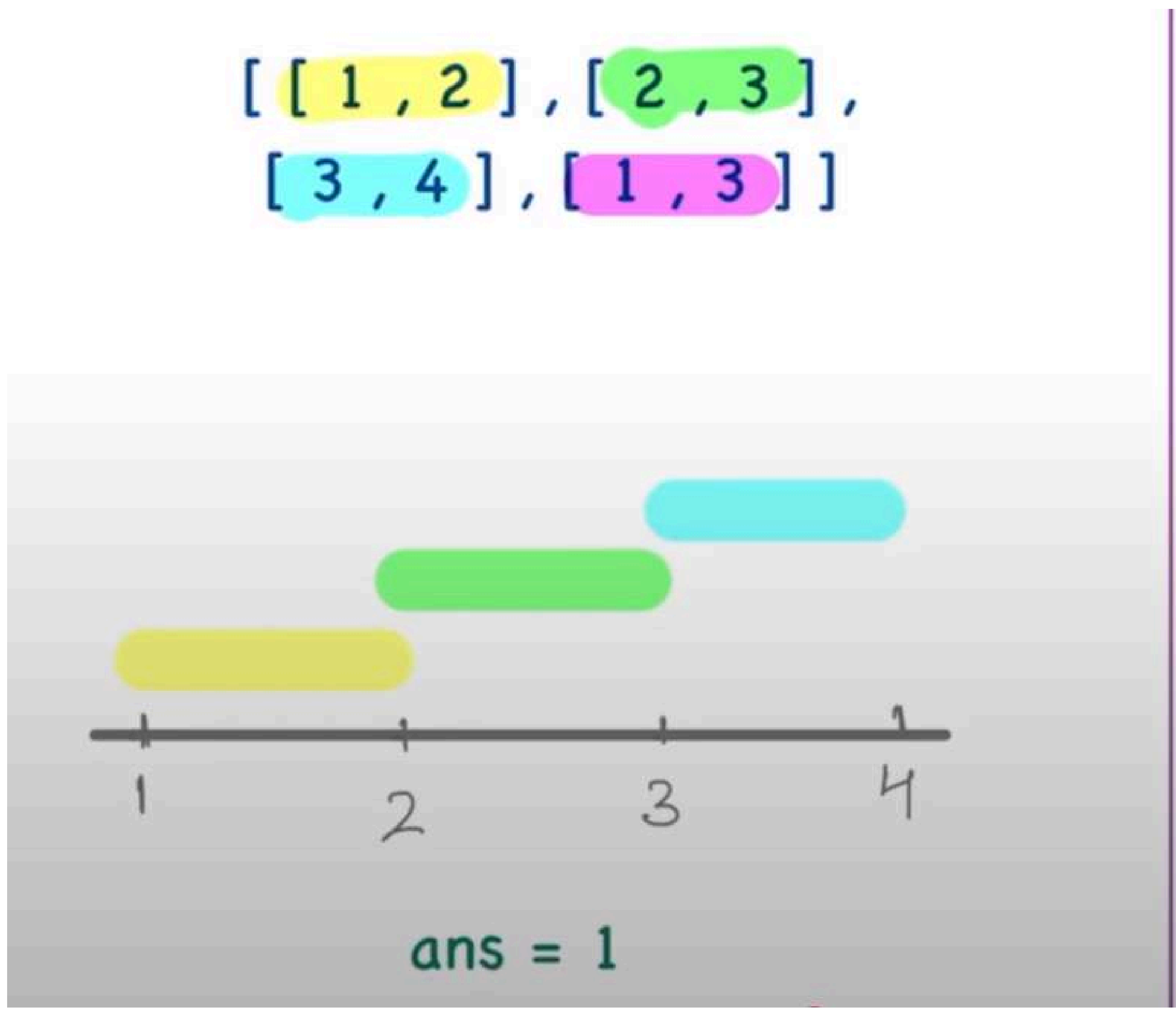
ans = 1

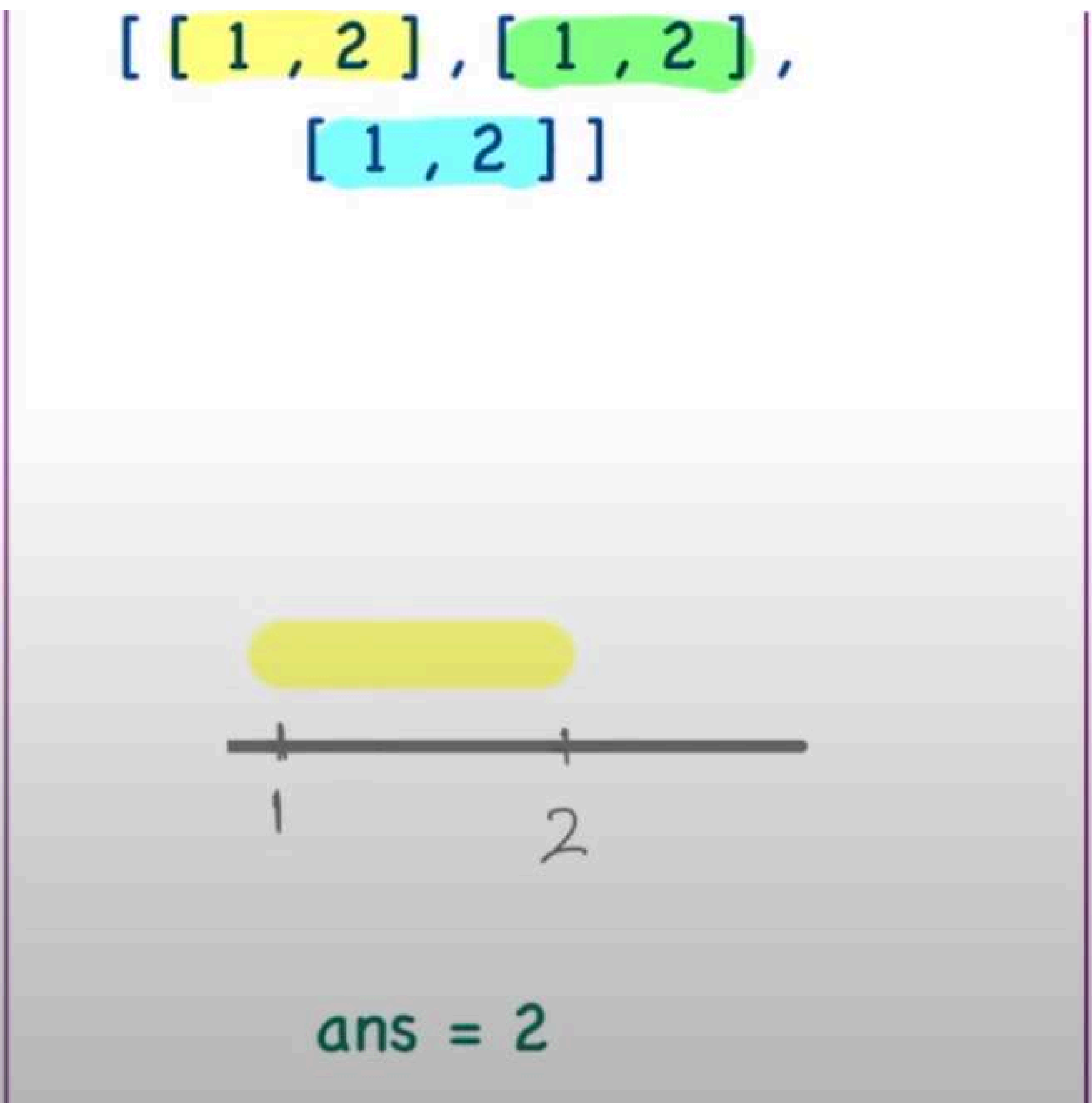
[[1,2],[1,2],
[1,2]]

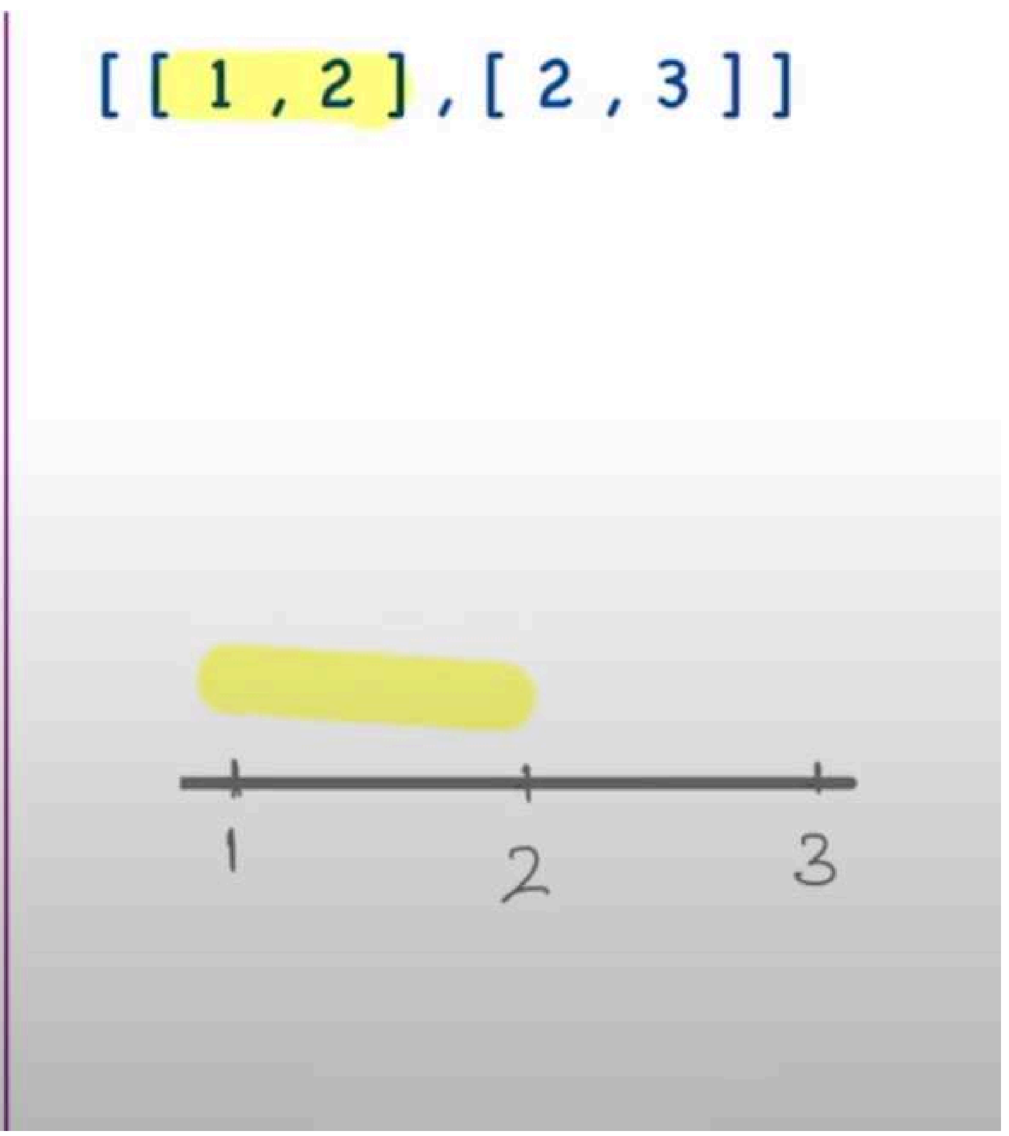
ans = 2

[[1,2],[2,3]]

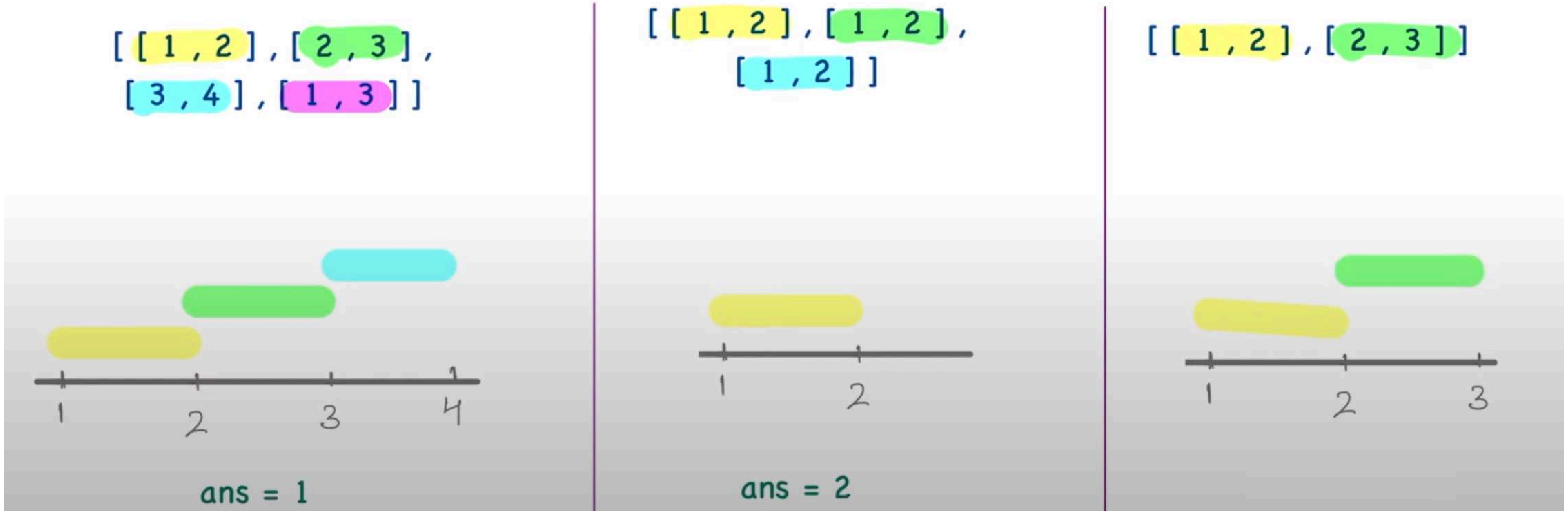
Non Overlapping Intervals



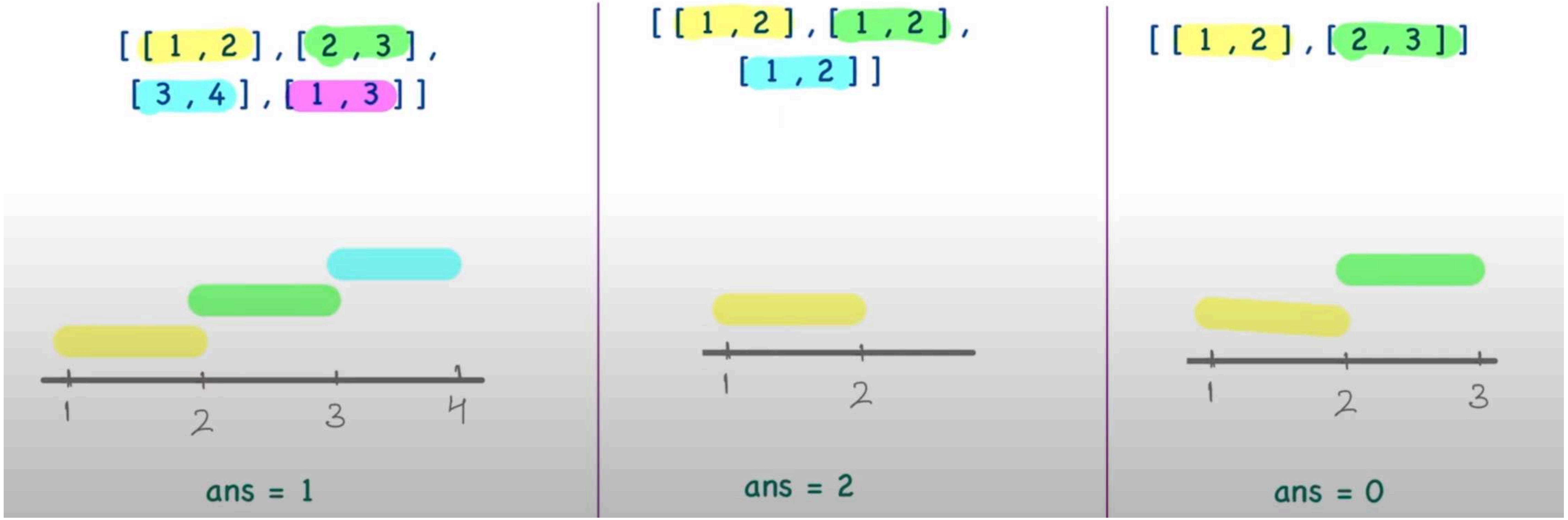


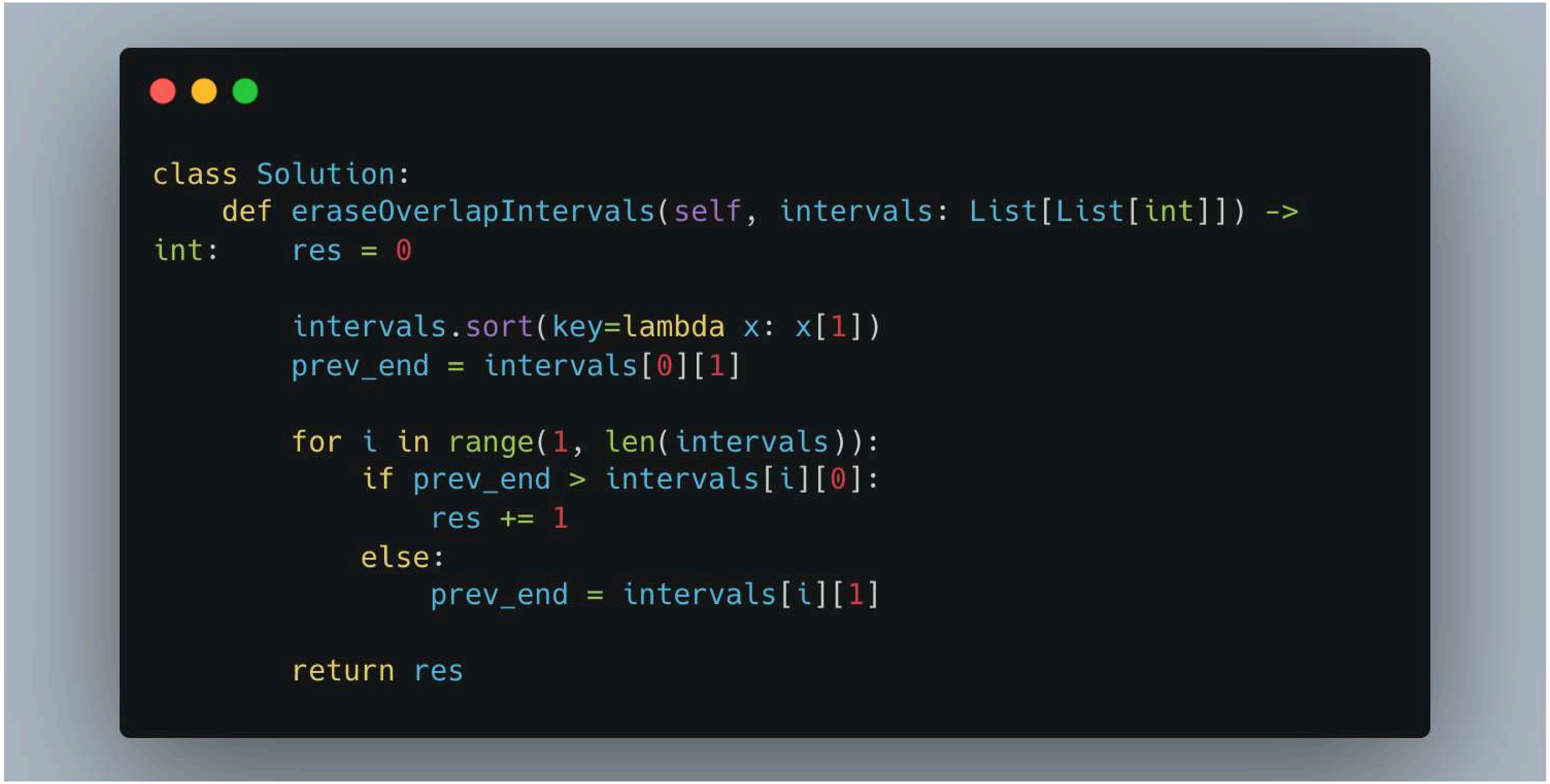


Non Overlapping Intervals



Non Overlapping Intervals





```
class Solution:
    def eraseOverlapIntervals(self, intervals: List[List[int]]) ->
int:    res = 0

        intervals.sort(key=lambda x: x[1])
        prev_end = intervals[0][1]

        for i in range(1, len(intervals)):
            if prev_end > intervals[i][0]:
                res += 1
            else:
                prev_end = intervals[i][1]

        return res
```

Coin Change problem

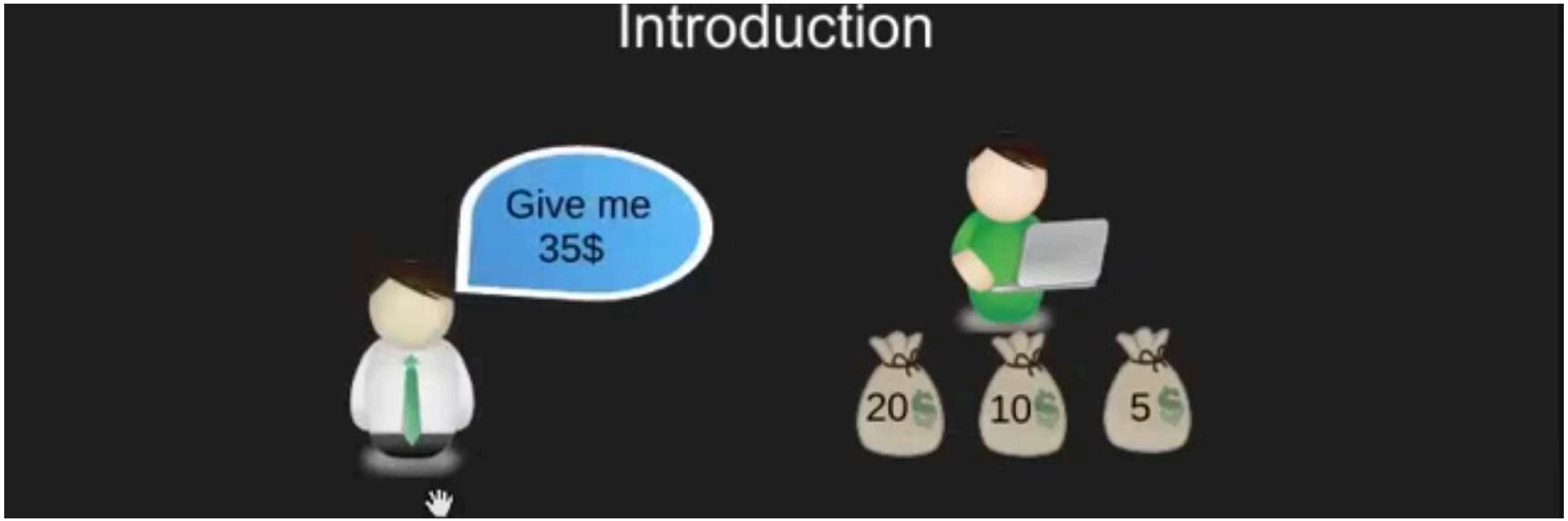
Suppose you’re a Bank with infinite supply of **20,10 & 5 dollar coin**.

Now, some other bank in need comes to you ask for money!

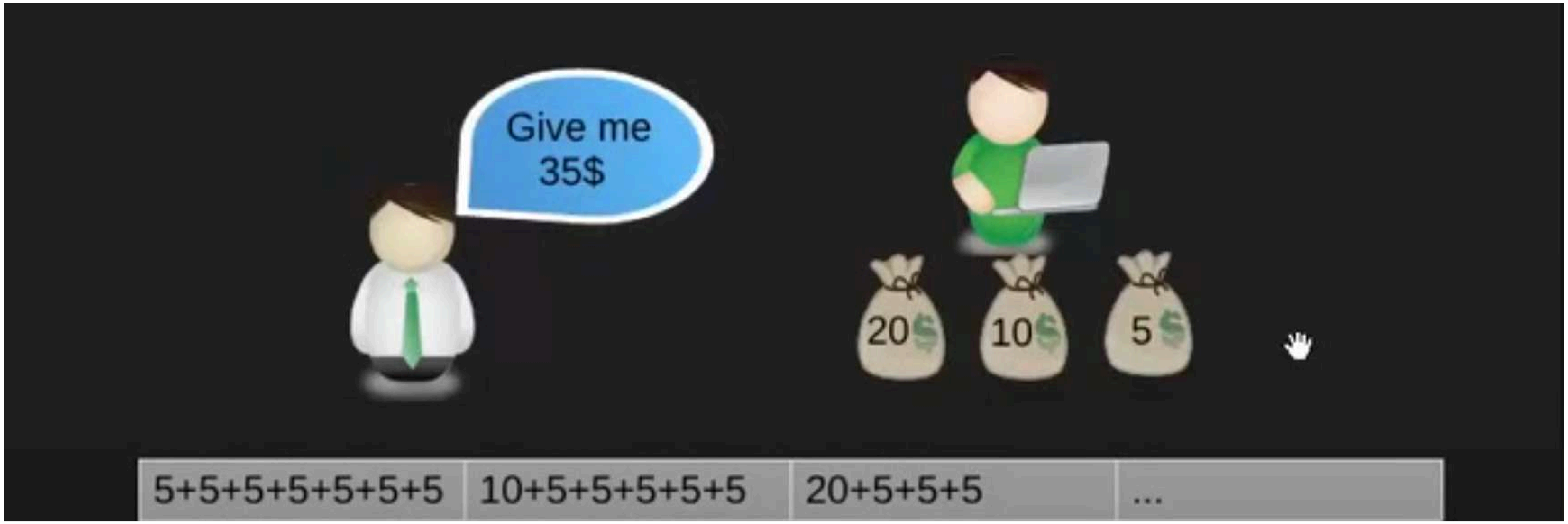
You’re ready to lend money but you want to do it in a **way with minimum number of coins**!

Coin Change problem

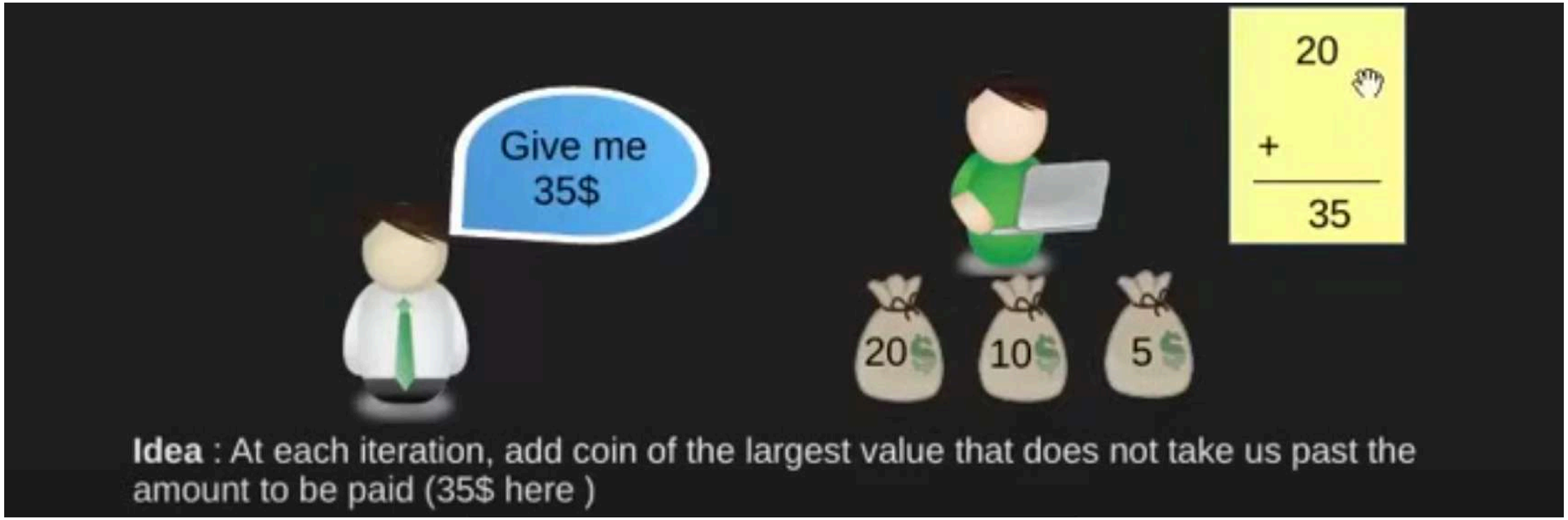
Coin Change problem



Coin Change problem

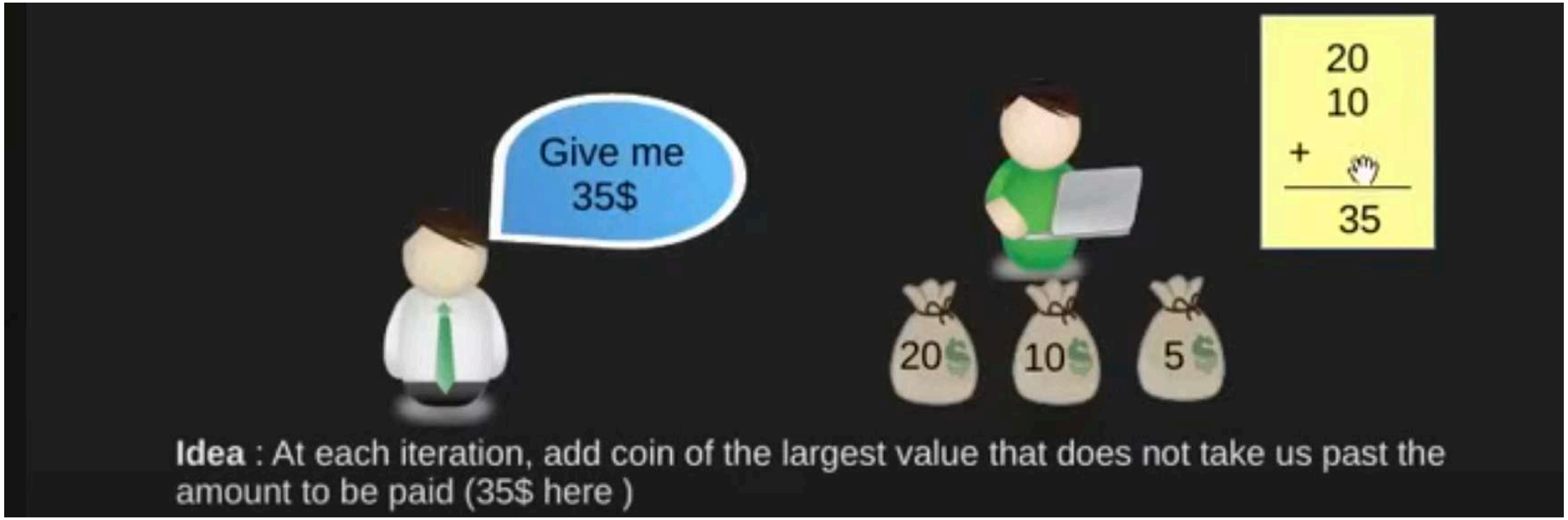


Coin Change problem



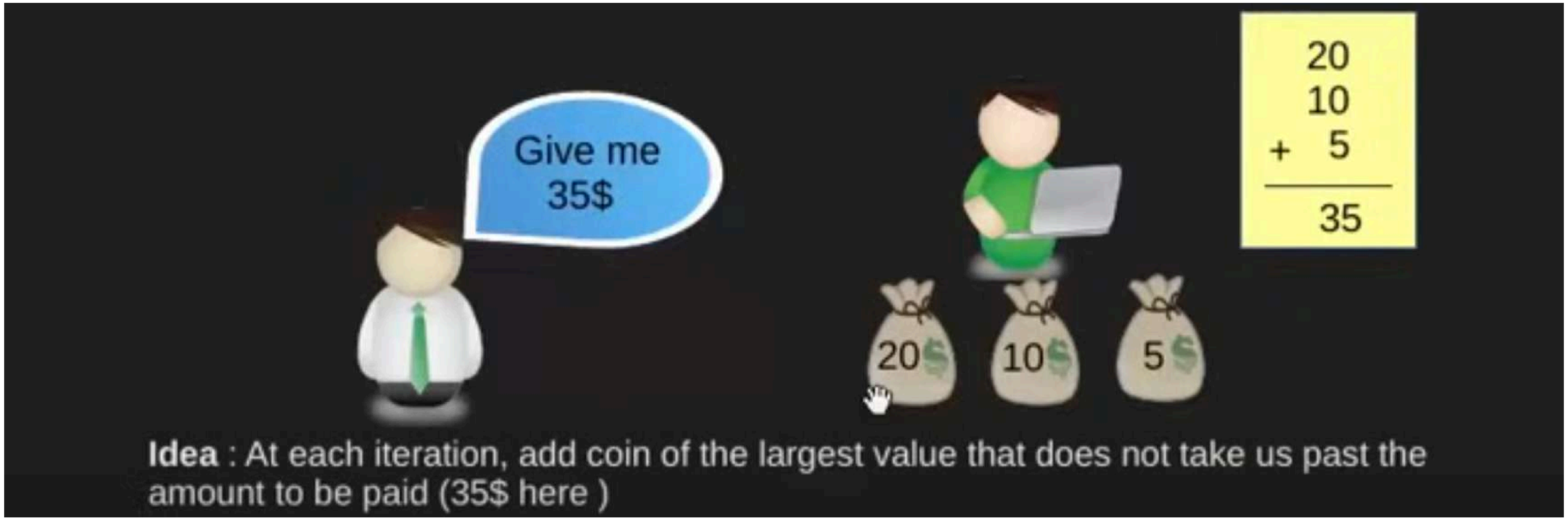
Idea : At each iteration, add coin of the largest value that does not take us past the amount to be paid (35$ here)

Coin Change problem



Idea : At each iteration, add coin of the largest value that does not take us past the amount to be paid (35$ here)

Coin Change problem



Idea : At each iteration, add coin of the largest value that does not take us past the amount to be paid (35$ here)

What if we need to introduce a central bank to cater the needs of diff countries?

Now, we'll be standing with coins of diff denominations and sequential order is not there! What will happen now?

Think!

“Greedy is greedy – it grabs the biggest coin first. But sometimes, thinking ahead saves more.”

Newton School of Technology

Please fill the feedback form.