Day 2: Interval Cover Lab

Overview

In this in-class lab session, you will practice implementing Interval scheduling problems that can be solved with a greedy approach, and learn to reflect greedy algorithms as approaches instead of rules or policies.

Starter Code: `interval.py`

Context

A small community in Minnesota is experiencing drought. Not surprisingly, the town relies heavily on agriculture. The mayor asks you to help set up a drip irrigation system.

Part A: Set up irrigation system

The town has n + 1 garden plots on a straight line (from 0 to n) and each has a pipe within. However, those pipes are not identical, so they have different watering capacity. The watering capacity of the i-th pipe is stored `ranges[i]`. In other words, the i-th pipe can water an area from `[i - ranges[i], i + ranges[i]]`

We want the irrigation system to cover all garden plots! However, opening a pipe costs time, maintenance, and electricity, so we want as few pipes open as possible. If this is impossible to set up, the helper function should let us know.

- **Task**: Find minimum pipes that cover the entire garden [0,n]
  indicate with return -1 if impossible to find

General Approach: We need to cover the interval [0, n]. we can start with the first interval and out of all intervals that intersect with it we choose the one that covers the farthest point to the right.

- Step 1: Make the range of each pipe an Inverval
    - Left end: `L = max (0, i - ranges[i])`
    - Right end: `R = (n, i + ranges[i])`

- Step 2: Sort Intervals by start point
    - You may use `intervals.sort`

- Step 3: Greedily Swap Intervals
    - Keep track of rightmost point currently watered in `watered`
    - While `watered < n`

        Among all intervals with `L ≤ n`, **choose the one with largest R**

        Update `watered` to that R

Part B: Who gets watered first? (Collaborate)

For this part, you will need to find a partner to collaborate and finish.

Now as someone designed the algorithm, the mayor invites you to the town meeting, where the mayor and other community members are deciding whether to adopt your algorithm. They've got some questions for you to answer.

- **Task**: as a pair, talk through those questions and we will share your thoughts later!

- Question 1: Did your algorithm treat every garden the same? Or, should we be treating every garden the same, even though there are more or fewer resources?

- Question 2: Who is benefiting from your current system? In other words, what can people do to be beneficial under your rules?

- Question 3: What factors have you been ignoring? Think about who pays for the water? How much energy has been used in this system? Are there better solutions?

That is a stressful meeting! Don't worry, the community really wants you to learn and think about assumptions you made in designing algorithms. After this reflection, you will definitely become a better algorithm developer and perhaps help them write a better irrigation system!

Python

```python
while watered <= n:
        farthest = max(right of all intervals starting ≤ watered)
        if no such interval: return -1
        count += 1
        if farthest ≥ n: return count
        watered = farthest + 1
```