3440. Reschedule Meetings for Maximum Free Time II

Medium | Topics | Companies | Hint

You are given an integer `eventTime` denoting the duration of an event. You are also given two integer arrays `startTime` and `endTime`, each of length `n`.

These represent the start and end times of `n` **non-overlapping** meetings that occur during the event between time `t = 0` and time `t = eventTime`, where the `i^th` meeting occurs during the time `[startTime[i], endTime[i]]`.

You can reschedule **at most** one meeting by moving its start time while maintaining the **same duration**, such that the meetings remain non-overlapping, to **maximize** the **longest** *continuous period of free time* during the event.

Return the **maximum** amount of free time possible after rearranging the meetings.

Note that the meetings can **not** be rescheduled to a time outside the event and they should remain non-overlapping.

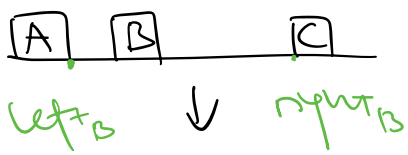
Note: *In this version*, it is **valid** for the relative ordering of the meetings to change after rescheduling one meeting.

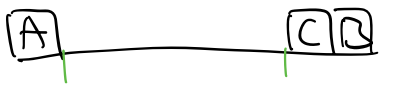
1° find gaps

2° find the sum of two consecutive gap, check

① if there's another gap to fit the meeting in between the gap, or ② the shifting of the meeting also create the max gap achievable.

①





for B

length = right_B - left_B

②

length = right_B - left_B - len(B)

↳ How to find it's case ① or ② ?

Sol: maintain an array g. to check if there's large enough gap for the curr meeting.

* find all gaps first, then compare with max gap is not efficient.

Better: iterate from head to tail, then backwards.

```python
class Solution:
    def maxFreeTime(self, eventTime: int, startTime: List[int], endTime: List[int]) -> int:
        n = len(startTime)
        gap = [False] * n
        t1, t2 = 0, 0

        for i in range(n):
            if endTime[i] - startTime[i] <= t1:
                gap[i] = True
            t1 = max(t1, startTime[i] - (0 if i == 0 else endTime[i-1]))

            if endTime[n-i-1] - startTime[n-i-1] <= t2:
                gap[n-i-1] = True
            t2 = max(t2, (eventTime if i == 0 else startTime[n-i]) - endTime[n-i-1])

        res = 0
        for i in range(n):
            left = 0 if i == 0 else endTime[i-1]
            right = eventTime if i == n-1 else startTime[i+1]

            if gap[i]:
                res = max(res, right - left)
            else:
                res = max(res, right - left - (endTime[i] - startTime[i]))

        return res
```

iterate in two directions

make current optimal choice