2106. Maximum Fruits Harvested After at Most K Steps

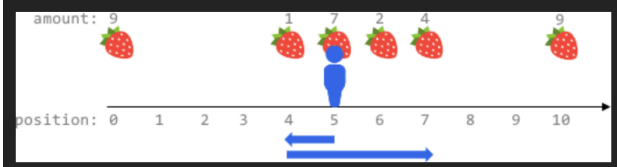
Hard | Topics | Companies | Hint

Fruits are available at some positions on an infinite x-axis. You are given a 2D integer array `fruits` where `fruits[i] = [position_i, amount_i]` depicts `amount_i` fruits at the position `position_i`. `fruits` is already **sorted** by `position_i` in **ascending order**, and each `position_i` is **unique**.

You are also given an integer `startPos` and an integer `k`. Initially, you are at the position `startPos`. From any position, you can either walk to the **left or right**. It takes **one step** to move **one unit** on the x-axis, and you can walk **at most** `k` steps in total. For every position you reach, you harvest all the fruits at that position, and the fruits will disappear from that position.

Return *the **maximum total number** of fruits you can harvest*.

Example 2:




```
Input: fruits = [[0,9],[4,1],[5,7],[6,2],[7,4],[10,9]], startPos = 5, k = 4
Output: 14
Explanation:
You can move at most k = 4 steps, so you cannot reach position 0 nor 10.
The optimal way is to:
- Harvest the 7 fruits at the starting position 5
- Move left to position 4 and harvest 1 fruit
- Move right to position 6 and harvest 2 fruits
- Move right to position 7 and harvest 4 fruits
You moved 1 + 3 = 4 steps and harvested 7 + 1 + 2 + 4 = 14 fruits in total.
```

" Folding sliding window "

i° startPos ≤ left

move rightwards only,

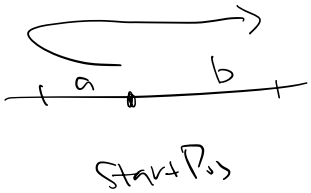
reachable : [startPos , startPos + k]

ii° startPos ≥ right

[startPos - k , startPos]

iii° left < startPos < right

① go left first ② go right first




```python
class Solution:
    def maxTotalFruits(self, fruits: List[List[int]], startPos: int, k: int) -> int:
        n = len(fruits)
        left, right = 0, 0
        res = 0
        ssum = 0

        def step(l, r):
            if fruits[r][0] <= startPos:
                return startPos - fruits[l][0]
            elif fruits[l][0] >= startPos:
                return fruits[right][0] - startPos
            else:
                return min(abs(startPos - fruits[l][0]), abs(startPos - fruits[r][0])) + fruits[r][0] - fruits[l][0]

        while right < n:
            ssum += fruits[right][1]

            while left <= right and step(left, right) > k:
                ssum -= fruits[left][1]
                left += 1

            res = max(res, ssum)
            right += 1

        return res
```