<https://leetcode.com/problems/last-moment-before-all-ants-fall-out-of-a-plank/description/>

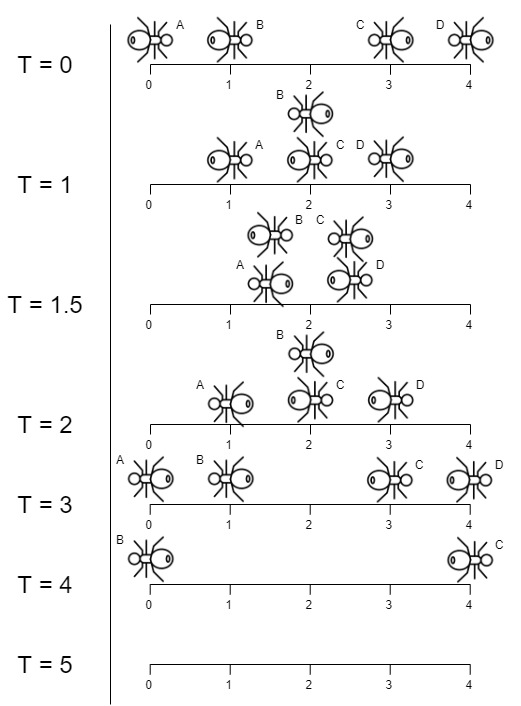
We have a wooden plank of the length n units. Some ants are walking on the plank, each ant moves with a speed of 1 unit per second. Some of the ants move to the left, the other move to the right.

When two ants moving in two different directions meet at some point, they change their directions and continue moving again. Assume changing directions does not take any additional time.

When an ant reaches one end of the plank at a time t, it falls out of the plank immediately.

Given an integer n and two integer arrays left and right, the positions of the ants moving to the left and the right, return the moment when the last ant(s) fall out of the plank.

**Example 1:**



**Input:** n = 4, left = [4,3], right = [0,1]

**Output:** 4

**Explanation:** In the image above:

-The ant at index 0 is named A and going to the right.

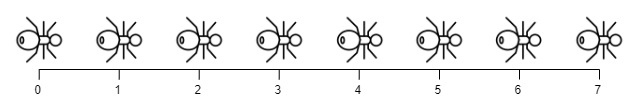
-The ant at index 1 is named B and going to the right.

-The ant at index 3 is named C and going to the left.

-The ant at index 4 is named D and going to the left.

The last moment when an ant was on the plank is t = 4 seconds. After that, it falls immediately out of the plank. (i.e., We can say that at t = 4.0000000001, there are no ants on the plank).

**Example 2:**

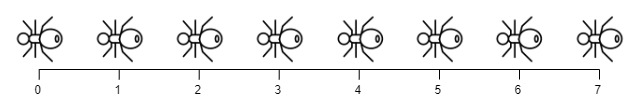


**Input:** n = 7, left = [], right = [0,1,2,3,4,5,6,7]

**Output:** 7

**Explanation:** All ants are going to the right, the ant at index 0 needs 7 seconds to fall.

**Example 3:**



**Input:** n = 7, left = [0,1,2,3,4,5,6,7], right = []

**Output:** 7

**Explanation:** All ants are going to the left, the ant at index 7 needs 7 seconds to fall.

**Constraints:**

1 <= n <= 10^4

0 <= left.length <= n + 1

0 <= left[i] <= n

0 <= right.length <= n + 1

0 <= right[i] <= n

1 <= left.length + right.length <= n + 1

All values of left and right are unique, and each value can appear only in one of the two arrays.

**Attempt 1: 2024-10-23**

**Solution 1: Math (10 min)**

class Solution {

    public int getLastMoment(int n, int[] left, int[] right) {

        int count = 0;

        for(int num : left) {

            count = Math.max(count, num);

        }

        for(int num : right) {

            count = Math.max(count, n - num);

        }

        return count;

    }

}

Time Complexity: O(n)

Space Complexity: O(1)

**Refer to**

<https://leetcode.com/problems/last-moment-before-all-ants-fall-out-of-a-plank/editorial/>

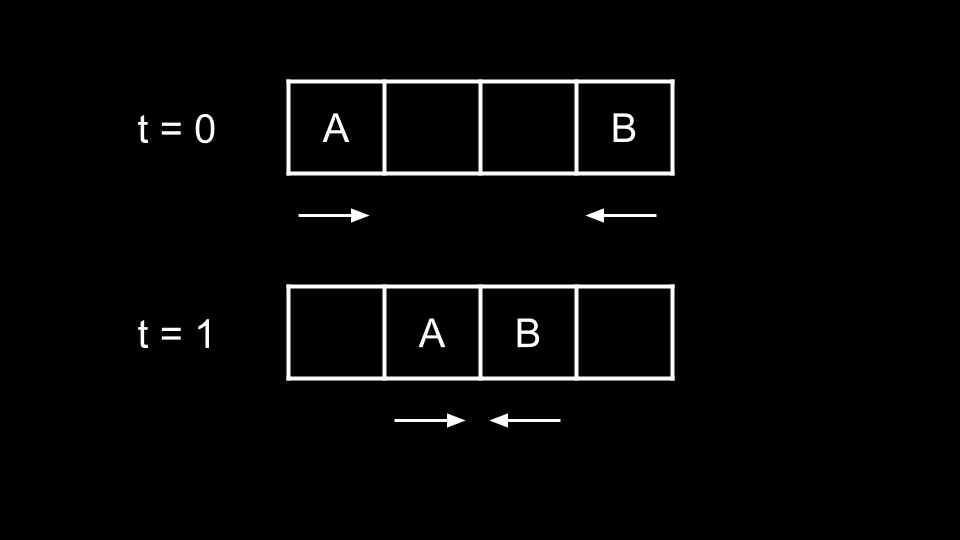
**Approach: Ants Pass Each Other!**

**Intuition**

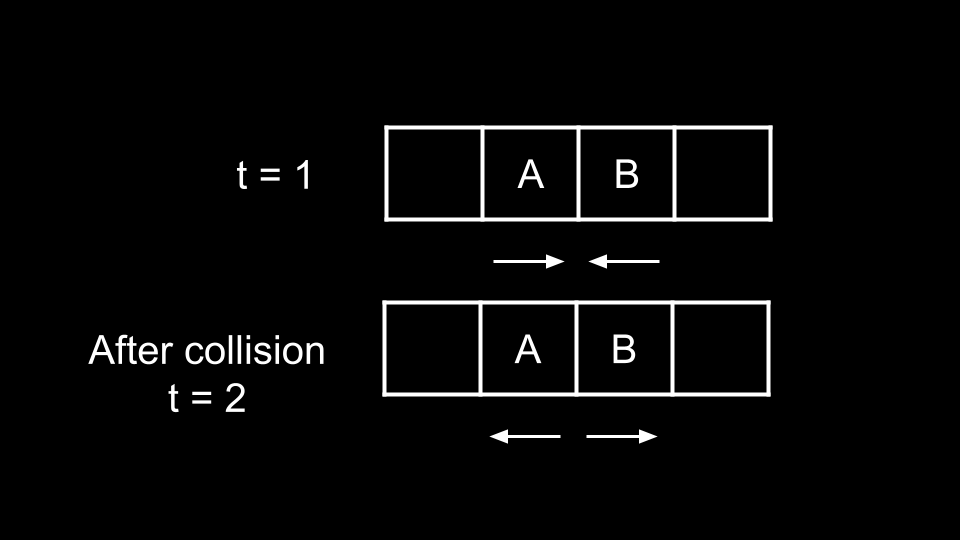
Initially, this problem may seem daunting as there could be many possible collisions between the ants.

However, we can make a few observations that simplify the problem greatly. The first thing to notice is that collisions happen instantaneously. The second thing to notice is that all ants walk at the same speed. The final thing to notice is that the ants eventually fall off the plank, so there won't be any infinite collisions.

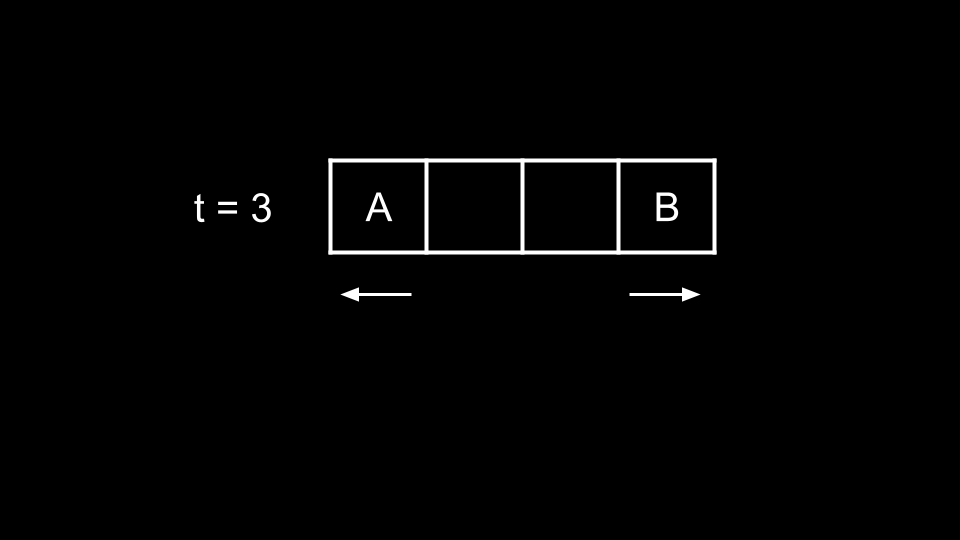
This brings us to the critical observation. Let's say we have an ant A walking right and an ant B walking left, and they are on a collision course.



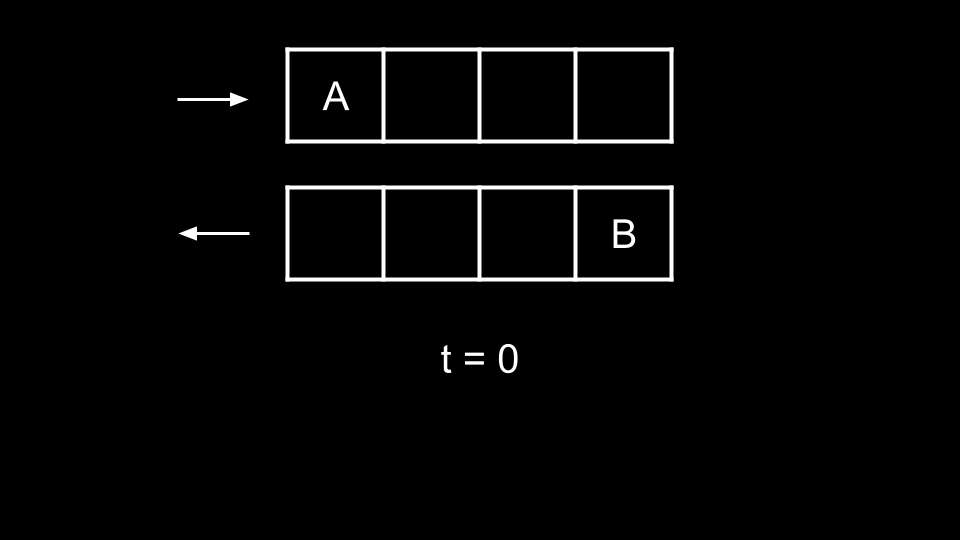
At t = 1, the ants are about to collide. At t = 2, the ants try to walk forward and collide, thus swapping directions.

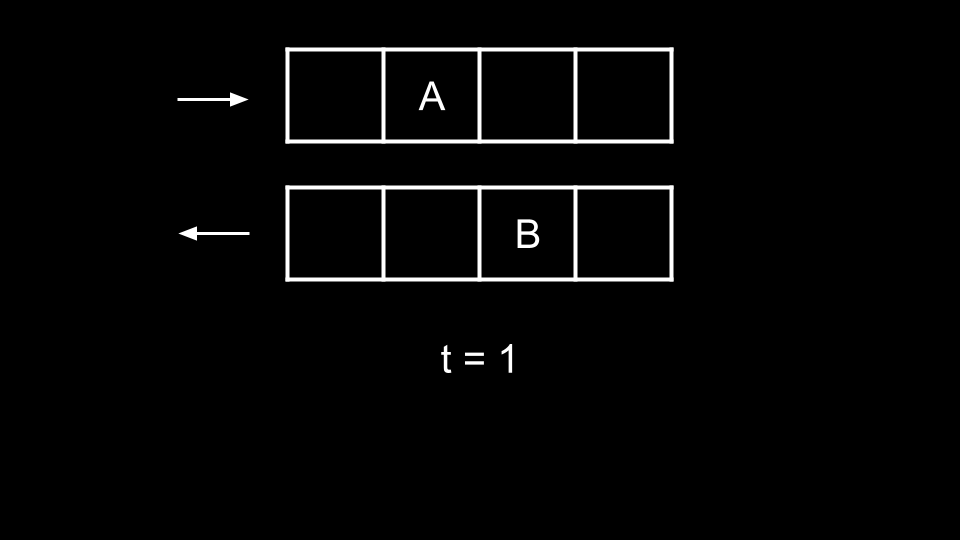


At t = 3, they reach the end of the plank and fall off.

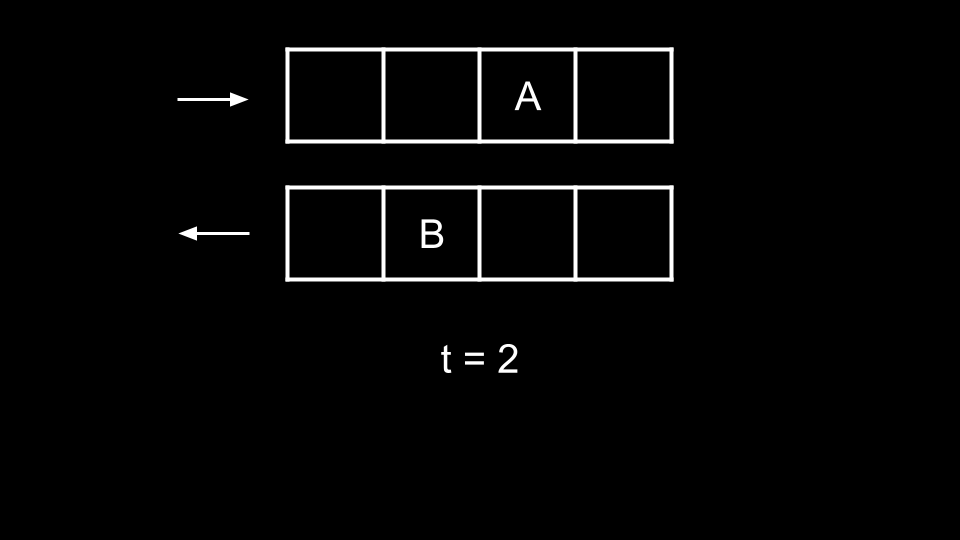


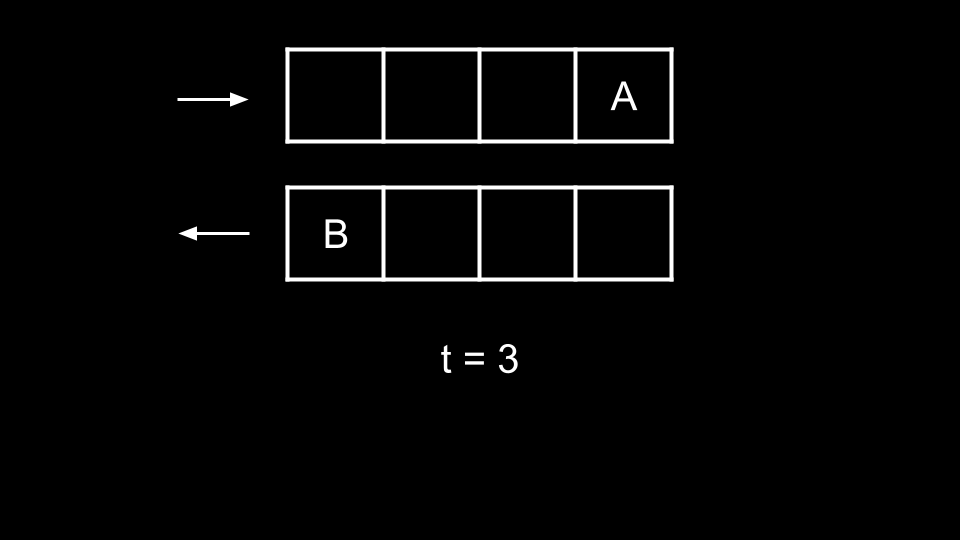
Now, let's consider a new scenario with the same ants A and B. Imagine if all the ants walking left were on one plank, and all the ants walking right were on a different plank.





At t = 2 in the original scenario, the ants collide and swap directions. In the new scenario, the ants will simply pass each other.





The two scenarios are actually equivalent! That is, the collisions are completely irrelevant. Why?

When the ants collide, they do not change position because their attempt at moving forward is blocked. What we mean here is that at t = 2, A is at index 1 and tries to walk to the right. However, it collides into B and stays at index 1. The same can be said for B remaining at position 2.

However, the ant that they collided with is at the position that they **would have been** at had there not been any collision. The ant they collided with also now has their original velocity (since their velocities swapped after the collision).

Because all the ants here are the same, we previously referred to them as A and B for better distinction. They have no differences in reality. Thus two ants colliding according to the rules and simply passing through each other are two entirely identical scenarios. If the ant they collided with has their original velocity and is at the same position they would have been at had there not been any collision (and vice-versa), did the collision really change anything? No.

Thus, we can consider the ants walking right simply passing through those walking left. So what will be our answer?

An ant walking left from position num will take num time to fall off the plank.

An ant walking right from position num will take n - num time to fall off the plank.

We simply take the maximum of all these times.

**Algorithm**

Initialize ans = 0.

Iterate over left. For each num:

Update ans with num if it is larger.

Iterate over right. For each num:

Update ans with n - num if it is larger.

Return ans.

**Implementation**

class Solution {

public int getLastMoment(int n, int[] left, int[] right) {

int ans = 0;

for (int num : left) {

ans = Math.max(ans, num);

}

for (int num : right) {

ans = Math.max(ans, n - num);

}

return ans;

}

}

**Refer to**

[L2211.Count Collisions on a Road](note://WEB54fbc019f7cda043b32ce15a1ff3bf88)