Problem Statement: Candy Distribution

Description

You are given n children standing in a line, each with a rating value represented in an integer array ratings. Your task is to distribute candies to these children based on the following requirements:

1. Each child must receive at least one candy.
2. Children with a higher rating than their neighbors must receive more candies than those neighbors.

Your goal is to determine the minimum number of candies needed to distribute according to the above rules.

Example 1

- **Input:** ratings = [1, 0, 2]
- **Output:** 5
- **Explanation:** The first child can receive 2 candies, the second child 1 candy, and the third child 2 candies. This satisfies the condition that children with higher ratings receive more candies than their neighbors.

Example 2

- **Input:** ratings = [1, 2, 2]
- **Output:** 4
- **Explanation:** The first child receives 1 candy, the second child 2 candies, and the third child 1 candy. Even though the second and third children have the same rating, the conditions are still satisfied.

Constraints

- n == ratings.length
- 1 <= n <= 2 * 10^4
- 0 <= ratings[i] <= 2 * 10^4

Solution Approach

The solution involves two main passes through the ratings array to ensure the candy distribution rules are met.

Steps

1. **Initialize Candies Array:**

   - Create an array candies of the same length as ratings, initialized with 1 for each child since each child must have at least one candy.

2. **Left-to-Right Pass:**

   - Iterate through the ratings array from left to right.
   - For each child, if their rating is higher than the previous child's rating, give them one more candy than the previous child.

3. **Right-to-Left Pass:**

- Iterate through the ratings array from right to left.
- For each child, if their rating is higher than the next child's rating, give them one more candy than the next child, but ensure not to reduce the number of candies already assigned if it was higher during the left-to-right pass.

4. Sum the Candies:

- Calculate the total number of candies by summing up the values in the candies array.

Example Walkthrough

Example 1:

ratings = [1, 0, 2]

Initial candies array: [1, 1, 1]

Left-to-Right Pass:

At i = 1, ratings[1] (0) is not greater than ratings[0] (1), so no change.

At i = 2, ratings[2] (2) is greater than ratings[1] (0), so candies[2] = candies[1] + 1 = 2.

Result after Left-to-Right: [1, 1, 2]

Right-to-Left Pass:

At i = 1, ratings[1] (0) is not greater than ratings[2] (2), so no change.

At i = 0, ratings[0] (1) is greater than ratings[1] (0), so candies[0] = max(candies[0], candies[1] + 1) = 2.

Result after Right-to-Left: [2, 1, 2]

Total Candies:

Sum of candies array: 2 + 1 + 2 = 5

Example 2:

ratings = [1, 2, 2]

Initial candies array: [1, 1, 1]

Left-to-Right Pass:

At i = 1, ratings[1] (2) is greater than ratings[0] (1), so candies[1] = candies[0] + 1 = 2.

At i = 2, ratings[2] (2) is not greater than ratings[1] (2), so no change.

Result after Left-to-Right: [1, 2, 1]

Right-to-Left Pass:

At i = 1, ratings[1] (2) is greater than ratings[2] (2), so no change.

At i = 0, ratings[0] (1) is not greater than ratings[1] (2), so no change.

Result after Right-to-Left: [1, 2, 1]

Total Candies:

Sum of candies array: 1 + 2 + 1 = 4

By following these steps, you can determine the minimum number of candies needed to distribute to the children in a way that satisfies the given conditions.