Aggressive Cows Problem CodeStudio

You are given an array arr consisting of n integers, which represent the positions of stalls. You are also given an integer k, which denotes the number of aggressive cows. Your task is to assign stalls to k cows such that the minimum distance between any two cows is maximized. Your goal is to determine the maximum possible minimum distance between the cows and then output that value.

Input: 1 2 3, k: 2

output: 2

In this case, we have three stalls, and we need to assign 2 cows. To maximize the minimum distance between the cows, we can assign the cows to the stalls at positions 1 and 3, resulting in a minimum distance of 2 between them.

Input: 0 3 4 7 10 9, k:4

output: 3

Here, we have six stalls, and we need to assign 4 cows. To maximize the minimum distance, we can assign the cows to the stalls at positions 0, 4, 7, and 10. The minimum distance between any two cows is 3.

Input: 2 3 1 4 6, k:2

output: 5

In this case, we have five stalls, and we need to assign 2 cows. To maximize the minimum distance, we can assign the cows to the stalls at positions 1 and 6. The minimum distance between them is 5.

Approach 1: Using Binary Search Algorithm to find the maximum minimum distance of aggressive cows.

The isPossibleSolution function checks if a given allocation is a possible solution by iterating through the stalls and checking if the minimum distance between the cows is at least equal to the given mid value. It returns true if it's possible to assign all aggressive cows within the mid value, and false otherwise.

The aggressiveCows function uses a binary search algorithm to find the maximum possible minimum distance between the aggressive cows. Here are the steps it follows:

Sort the stalls in ascending order using sort(stalls.begin(), stalls.end()).

Initialize the lower (low) and upper (high) bounds for the binary search. The lower bound (low) is set to 0, and the upper bound (high) is set to the difference between the last and first elements of the sorted stalls array: stalls[stalls.size() - 1] - stalls[0].

Enter the binary search loop: while low <= high.

Calculate the mid value as the average of low and high: mid = low + (high - low) / 2.

Check if the current mid value represents a possible solution by calling isPossibleSolution(stalls, aggressiveCows, mid). If it returns true, update the answer (ans) to the current mid value and look for larger distances by moving the low pointer: low = mid + 1.

If the isPossibleSolution call returns false, update the high pointer to look for smaller distances: high = mid - 1.

Repeat steps 4-6 until low is no longer less than or equal to high.

Return the answer (ans), which represents the maximum possible minimum distance between the aggressive cows.

Time Complexity:

The time complexity of this approach is dominated by the binary search algorithm, which has a time complexity of O(logN), where N is the number of elements in the stalls array. Additionally, sorting the stalls array takes O(NlogN) time. Therefore, the overall time complexity is O(NlogN).

Space Complexity:

The space complexity of the code is O(1) since it uses a constant amount of additional space, regardless of the size of the input array.

Input: 1 2 3, k: 2

output: 2

In this case, we have three stalls, and we need to assign 2 cows. The initial lower bound (low) is 0, and the upper bound (high) is 3 - 1 = 2. The binary search starts by setting mid to 1. The isPossibleSolution function is called with mid = 1. Since the minimum distance between any two cows is 1, it is not possible to assign all aggressive cows within this distance. Therefore, the high pointer is updated to mid - 1 = 0. The next iteration sets mid to 0, and the isPossibleSolution call returns true since we can assign the cows at positions 1 and 3, resulting in a minimum distance of 2. The answer is updated to 0, and the low pointer is updated to mid + 1 = 1. Since low is no longer less than or equal to high, the binary search terminates, and the maximum possible minimum distance is 2.

Input: 0 3 4 7 10 9, k:4

output: 3

In this case, we have six stalls, and we need to assign 4 cows. The initial lower bound (low) is 0, and the upper bound (high) is 10 - 0 = 10. The binary search proceeds as follows:

Iteration 1: mid = 5. The isPossibleSolution call returns false, so high becomes mid - 1 = 4.

Iteration 2: mid = 2. The isPossibleSolution call returns false, so high becomes mid - 1 = 1.

Iteration 3: mid = 0. The isPossibleSolution call returns false, so high becomes mid - 1 = -1.

Iteration 4: low is no longer less than or equal to high, so the binary search terminates. The maximum possible minimum distance is 3.

Input: 2 3 1 4 6, k:2

output: 5

In this case, we have five stalls, and we need to assign 2 cows. The initial lower bound (low) is 0, and the upper bound (high) is 6 - 1 = 5. The binary search proceeds as follows:

Iteration 1: mid = 2. The isPossibleSolution call returns false, so high becomes mid - 1 = 1.

Iteration 2: mid = 1. The isPossibleSolution call returns false, so high becomes mid - 1 = 0.

Iteration 3: low is no longer less than or equal to high, so the binary search terminates. The maximum possible minimum distance is 5.