Fishing Problem - Solution and Explanation

Problem Statement

You are given two arrays: 'fish' and 'baits'.

Each element in 'fish' represents the size of a fish in the pond.

Each element in 'baits' represents the size of a bait you have.

Rules:

- A fish can only be caught if the bait size is strictly smaller than the fish size.
- Each bait can be used up to 3 times before it is depleted.
- Once a fish is caught, it is removed from the pond and cannot be caught again.
- The goal is to maximize the number of fish caught using the given baits.

Constraints and Rules

- Fish and baits are given as two separate arrays.
- Fish sizes and bait sizes are positive integers.
- Sorting is necessary to optimize the process.
- The solution should run efficiently in O(N log N + M log M).

Approach & Explanation

1. Sort both 'fish' and 'baits' in descending order.
2. Use two pointers: one for the fish array and one for the bait array.
3. Try to catch the largest available fish using the largest available bait.
4. Track the usage of each bait using a dictionary.
5. Move to the next bait once it has been used 3 times.
6. Continue until all baits are used up or all fish are caught.

Python Solution

```
def solution(fish, baits):
    fish.sort(reverse=True)  # Sort fish in descending order
    baits.sort(reverse=True)  # Sort baits in descending order

    fish_index, bait_index, caught_fish = 0, 0, 0
    bait_usage = {}  # Dictionary to track bait usage

    while fish_index < len(fish) and bait_index < len(baits):
        if baits[bait_index] < fish[fish_index]:  # Check if bait can catch fish
            caught_fish += 1
            fish_index += 1
            bait_usage[bait_index] = bait_usage.get(bait_index, 0) + 1

            if bait_usage[bait_index] == 3:  # If bait used 3 times, move to next bait
                bait_index += 1
        else:
            fish_index += 1  # Skip fish if bait is too large

    return caught_fish
```

Time and Space Complexity

Time Complexity:

- Sorting fish and baits takes O(N log N + M log M), where N is the number of fish and M is the number of baits.
- The while loop runs in O(N + M) since each fish and bait is processed once.
- Overall, the time complexity is O(N log N + M log M).

Space Complexity:

- We use O(1) extra space since sorting is done in place.
- The dictionary tracking bait usage takes O(M) space in the worst case.
- Overall, the space complexity is O(M).

Example Runs & Edge Cases

Example 1:

Input: fish = [1, 2, 3], baits = [1]

Output: 2

Explanation: The bait (1) can catch fish (3) and (2), but not (1).

Example 2:

Input: fish = [2, 2, 3, 4], baits = [1]

Output: 3

Explanation: The bait (1) catches fish (4), (3), and (2).

Edge Cases:

- No baits available -> Output: 0
- All fish too small -> Output: 0
- More baits than needed -> The extra baits are ignored.