2739. Total Distance Traveled

Easy Math Simulation

Leetcode Link

Problem Description

In this problem, we have a truck with two fuel tanks: a main tank and an additional tank. The main tank's current fuel level is given by the integer `mainTank`, and the additional tank's fuel level is given by `additionalTank`, both measured in liters. The truck's fuel efficiency is fixed at 10 kilometers per liter.

Fuel transfer between tanks works under a specific rule: for every 5 liters consumed from the main tank, 1 liter is transferred from the additional tank to the main tank if there is at least 1 liter of fuel in the additional tank. This transfer is not continuous but occurs instantaneously every time the main tank's fuel level goes down by 5 liters.

We need to calculate the maximum distance the truck can travel given these conditions.

Intuition

The core idea behind the solution is to simulate the truck's fuel consumption while considering the rule for transferring fuel from the additional tank to the main tank.

We keep track of the total distance (`ans`) and the fuel consumption (`cur`) as we decrement fuel from the `mainTank`. For every liter of fuel used, the truck travels 10 km, so with each iteration, we increase the total distance by 10 km.

We need to track every time the main tank uses 5 liters of fuel to simulate the transfer from the additional tank. If the `additionalTank` has at least 1 liter, we transfer 1 liter to the `mainTank`. This is done by checking if `cur` is divisible by 5 and there is fuel in the `additionalTank`. If both conditions are met, we decrement 1 liter from the `additionalTank` and add 1 liter to the `mainTank`.

The loop continues until there is no more fuel in the `mainTank`. At this point, we have traveled the maximum possible distance, and we return the `ans` variable as the result.

Solution Approach

To implement the solution, the `Solution` class defines a method `distanceTraveled` that takes `mainTank` and `additionalTank` as its parameters. This method does not use any complex data structures or algorithms but follows a straightforward iterative process.

Here's a step-by-step breakdown of the approach:

1. Initialize `ans` to `0`, which will keep track of the total distance traveled by the truck.
2. Keep track of liters consumed from the main tank with `cur`, initializing it to `0`.
3. Use a `while` loop to iterate until the `mainTank` is empty (i.e., `mainTank` becomes `0`).
4. Inside the loop, increment `cur` by `1` for each iteration to count the fuel consumption.
5. For every liter of fuel consumed, add `10` to `ans`, which is the distance traveled per liter of fuel.
6. Decrement `1` from `mainTank` since a liter of fuel is used.
7. Check if the value of `cur` is a multiple of `5` and that `additionalTank` has at least `1` liter.
8. If both conditions are true, transfer `1` liter from the `additionalTank` to the `mainTank` by decrementing `1` from `additionalTank` and incrementing `1` to `mainTank`.

The key idea is to simulate the consumption and transfer of fuel from one tank to another while keeping track of the distance traveled every time the fuel is consumed. The loop halts when the main tank is empty, meaning no more distance can be traveled. At this point, the `ans` variable holds the maximum distance the truck can travel, so we return `ans` as the final result.

Note that no additional data structures are needed for this implementation, and the method uses simple arithmetic and conditional checks to achieve the required simulation.

Example Walkthrough

Let's assume the following scenario where `mainTank` has 12 liters of fuel and `additionalTank` has 2 liters of fuel. We want to calculate the maximum distance the truck can travel. Here's how the solution approach is applied in this example:

1. We initialize `ans` to `0` to start counting the total distance traveled, and `cur` to `0` to track the liters consumed from the main tank.
2. We enter the `while` loop since `mainTank` is not empty (contains 12 liters).
3. The truck uses 1 liter of fuel, `cur` is incremented to `1`, and `mainTank` is decremented to `11`. We add `10` km to `ans`, for a total of `10` km traveled.
4. This process repeats, with `cur` incrementing each time and `mainTank` decrementing for every liter used. Also, with every liter, another `10` km is added to `ans`.
5. When `cur` becomes `5`, indicating we have consumed 5 liters, and since `additionalTank` has at least `1` liter, we transfer `1` liter from the `additionalTank` to the `mainTank`. Now, `mainTank` has `7` liters (originally 6 before the transfer) and `additionalTank` has `1` liter left.
6. The loop continues; we increment `cur`, decrement `mainTank` by 1, and add `10` km to `ans` for each liter of fuel.
7. When `cur` reaches `10`, we have consumed another batch of 5 liters from the main tank, but this time the `additionalTank` is empty, so no transfer occurs.
8. The loop finally terminates when `mainTank` reaches `0`, meaning there is no more fuel left to use.

By the end of the process:

- The `ans` value reflects the total distance traveled. After consuming all 12 liters from the `mainTank` and the 1 liter transferred from the `additionalTank`, the truck would have traveled 130 km (12 liters + 1 transferred liter, each multiplied by 10 km per liter).
- The `while` loop has terminated because the `mainTank` is now empty.

So, by plugging in the actual liters and applying the approach outlined in the Problem Description, we've concluded that the truck can travel a maximum distance of 130 kilometers with the given amounts of fuel in the main and additional tanks.

Python Solution

```python
class Solution:
    def distanceTraveled(self, main_tank: int, additional_tank: int) -> int:
        # Initialize variables to track the distance and the current step count
        distance = current_step = 0

        # Continue the loop as long as there is fuel in the main tank
        while main_tank:
            current_step += 1  # Increment the step count
            distance += 10      # Increase the distance traveled by 10 (assumed unit of distance per unit of fuel)
            main_tank -= 1      # Decrease the main tank fuel by 1 unit

            # Check if a unit from the additional tank should be transferred to the main tank
            if current_step % 5 == 0 and additional_tank:
                additional_tank -= 1  # Remove 1 unit from the additional tank
                main_tank += 1        # Add 1 unit to the main tank (refueling from the additional tank)

        # Return the total distance traveled
        return distance

```

Java Solution

```java
class Solution {
    public int distanceTraveled(int mainTank, int additionalTank) {
        int distance = 0;  // Initialize the total distance travelled to 0
        int moves = 0;     // Counter to keep track of moves made

        // Loop until the main tank is empty
        while (mainTank > 0) {
            moves++;                 // Increment moves count
            distance += 10;          // Increase distance by 10 for each move
            mainTank--;              // Use one unit of fuel from the main tank

            // Every 5 moves, if there is fuel in the additional tank, transfer it to the main tank
            if (moves % 5 == 0 && additionalTank > 0) {
                additionalTank--;    // Use one unit of fuel from the additional tank
                mainTank++;          // Add one unit of fuel to the main tank
            }
        }

        return distance; // Return the total distance travelled
    }
}

```

C++ Solution

```cpp
class Solution {
public:
    // Function to calculate the distance traveled given the amount of fuel in the main tank and the additional tank
    int distanceTraveled(int mainTank, int additionalTank) {
        int distance = 0; // Initialize distance traveled
        int steps = 0;    // Initialize steps taken

        // Loop runs as long as there is fuel in the main tank
        while (mainTank > 0) {
            steps++;            // Increment steps
            distance += 10;     // Increase distance by 10 for each step
            mainTank--;         // Decrease main tank fuel by 1

            // Every 5 steps, if there is fuel in the additional tank, transfer 1 unit to the main tank
            if (steps % 5 == 0 && additionalTank > 0) {
                additionalTank--; // Use 1 unit of fuel from the additional tank
                mainTank++;       // Add 1 unit of fuel to the main tank
            }
        }

        // Return the total distance traveled
        return distance;
    }
};

```

Typescript Solution

```typescript
// Function to calculate the distance traveled given the amount of fuel in the main tank and the additional tank
function distanceTraveled(mainTank: number, additionalTank: number): number {
    let distance: number = 0; // Variable to store the total distance traveled
    let steps: number = 0;    // Variable to count the number of steps taken

    // Loop runs as long as there is fuel in the main tank
    while (mainTank > 0) {
        steps++;            // Increment the step count by 1
        distance += 10;     // Increase distance by 10 for each step
        mainTank--;         // Decrement the main tank fuel by 1

        // On every 5th step:
        if (steps % 5 === 0 && additionalTank > 0) {
            additionalTank--; // Decrease the additional tank fuel by 1
            mainTank++;       // Transfer 1 unit of fuel to the main tank
        }
    }

    // Return the calculated distance traveled
    return distance;
}

```

Time and Space Complexity

Time Complexity

The time complexity of the code is primarily determined by the `while` loop that runs as long as `mainTank` is not empty. For each iteration in the loop, we perform a constant amount of work: increment `cur`, decrement `mainTank`, add 10 to `ans`, and conditionally transfer fuel from the `additionalTank` to `mainTank`. The loop could run for as many as `mainTank` iterations, and in the worst case, we do not add any fuel from `additionalTank` to `mainTank`. Thus, the worst-case time complexity is `O(mainTank)`.

However, it's worth noting that every 5th iteration, if `additionalTank` has fuel, `mainTank` is incremented, which can happen up to `additionalTank` times. This effectively adds extra iterations to the loop, but since this action only happens after every 5th decrement of `mainTank`, the additional loop iterations are bounded by `additionalTank / 5`. Therefore, the precise worst-case time complexity of the code would be `O(mainTank + additionalTank/5)`.

Space Complexity

The space complexity of the code is `O(1)` because there is only a fixed number of variables used (`ans`, `cur`, `mainTank`, `additionalTank`), and no additional space is required that grows with the size of the input.