Solution: N-th Tribonacci Number

Let's solve the N-th Tribonacci Number problem using the Dynamic Programming pattern.

We'll cover the following

Statement

Given a number `n`, calculate the corresponding Tribonacci number. The Tribonacci sequence T_n is defined as:

$$T_0 = 0,\ T_1 = 1,\ T_2 = 1, \text{and}\ \ T_n = T_{n-1} + T_{n-2} + T_{n-3}\ \ \text{for}\ n >= 3$$

The input number, `n`, is a non-negative integer.

Constraints:

- $0 \leq$ `n` ≤ 37
- The answer is guaranteed to fit within a 32-bit integer, i.e., answer $\leq 2^{31} - 1$

Solution

So far, you've probably brainstormed some approaches and have an idea of how to solve this problem. Let's explore some of these approaches and figure out which to follow based on considerations such as time complexity and implementation constraints.

Naive approach

The first thing that comes to mind after reading the question is to use the same approach used for solving the Fibonacci sequence: recursion. The recursive approach works and will return us the correct answer here. We'll first initialize three numbers as 0, 1, and 1. If our required Tribonacci number is 1 or 2, we'll return 1 in the base case. Otherwise, we'll call the function recursively three times to compute the next number's sum, as the tribonacci number is the sum of the previous three numbers. We'll repeat this process until we hit the base case.

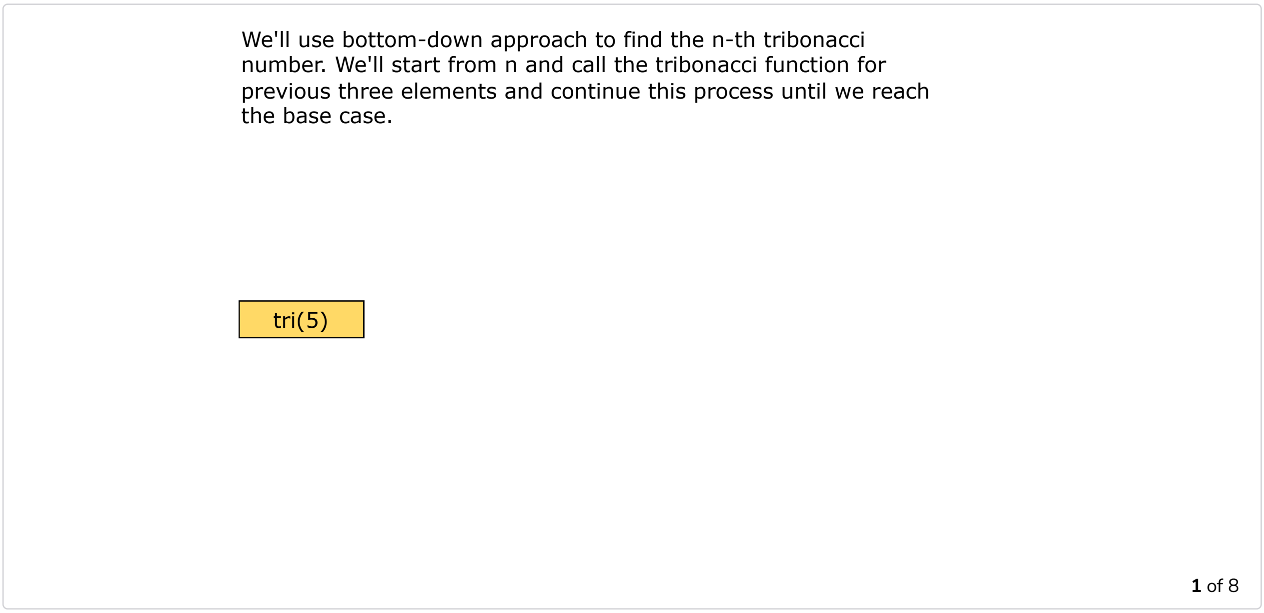
Computing the nth tribonacci number this way takes $O(3^n)$ time and the space complexity of $O(n)$.

Optimized approach using dynamic programming

We can reduce the time complexity if we use dynamic programming here. We can store the values of the elements and then reuse them to find the next element. When the specific element is needed, we just return the already stored value. This way, we don't need the array of size n because at any given time we just need the value to be computed and the three values preceding it.

Here is how we'll implement this algorithm:

- Initialize the first three variables as 0, 1, and 1.
- If `n` is less than 3, then return 1.
- Otherwise, run a loop `n−2` times and perform the following steps:
    - Replace the first variable with the second variable.
    - Replace the second variable with the third variable.
    - Replace the third variable with the sum of these three variables.



We'll use bottom-down approach to find the n-th tribonacci number. We'll start from n and call the tribonacci function for previous three elements and continue this process until we reach the base case.

Let's look at the code for this solution below:

Java

```java
class Tribonacci{
    public static int findTribonacci(int n){
        // if n is less than 3, then it's the base case
        if (n < 3)
            return n == 0 ? 0 : 1;
        // initializing first three numbers
        int firstNum = 0, secondNum = 1, thirdNum = 1;
        int temp = 0;
        // loop proceeds for n-2 times
        for (int i = 3; i <= n; i++) {
            // temp stores sum of last three computed Tribonacci numbers
            temp = firstNum + secondNum + thirdNum;
            // replaces firstNum by secondNum
            firstNum = secondNum;
            // replaces secondNum by thirdNum
            secondNum = thirdNum;
            // replaces thirdNum by sum of last three
            // computed Tribonacci numbers, which are stored in temp
            thirdNum = temp;
        }
        // returns nth Tribonacci number
        return thirdNum;
    }

```

```
        int index = 0;
        for (int input : inputList) {
```

N-th Tribonacci Number

Solution summary

To recap, the solution to this problem can be divided into the following parts:

- Initialize three variables as 0, 1, and 1, respectively.
- If n is less than 3, then return 1.
- Otherwise, compute the next number by adding the last three numbers until the required number is obtained.

Time complexity

The time complexity of this solution is $O(n)$, where n is the number of elements present in the array.

Space complexity

The space complexity of this solution is $O(1)$, as we keep track of a constant number of variables at a time.

Back
N-th Tribonacci Number

Next
Partition Equal Subset...

✓ Mark as Completed