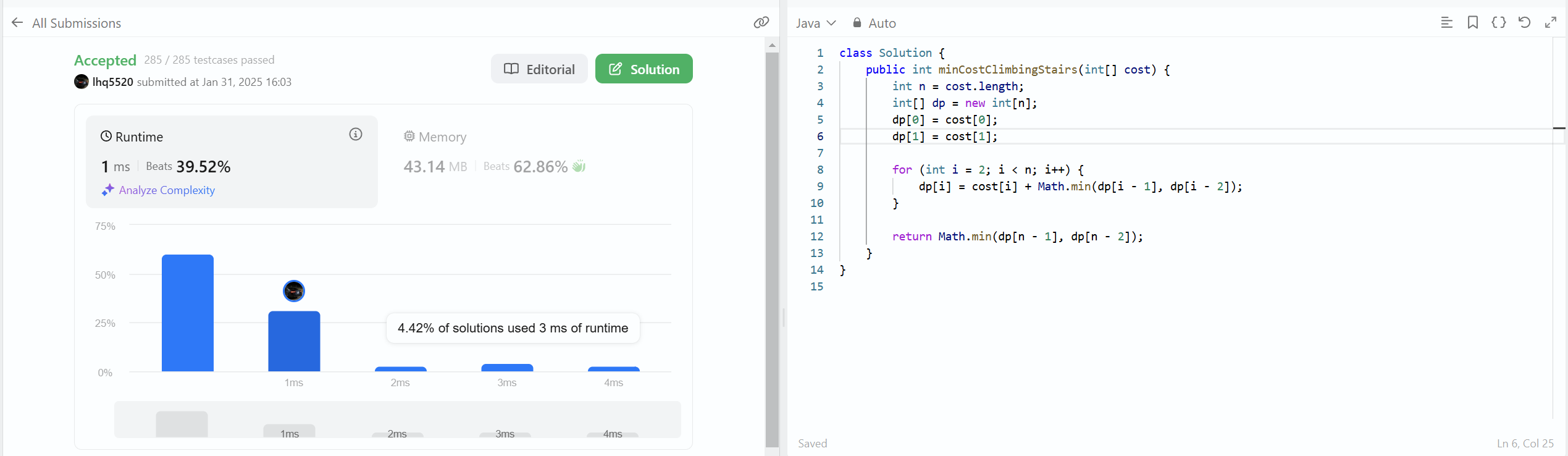
P4

a)

LeetCode 746: Min Cost Climbing Stairs, Easy



LeetCode 1137: N-th Tribonacci Number, Easy

A screenshot of a computer

Description automatically generated

b)

I used a brute-force recursive method when I originally tried to solve the Tribonacci problem. I created a function that recursively called itself for n-1, n-2, and n-3, with base cases for n = 0, n = 1, and n = 2, because the sequence follows a distinct pattern in which each integer is the sum of the preceding three. At first, I felt this method made sense because recursion matched the issue statement exactly, and I assumed it would be a simple method to obtain the right answer. But when I went through the various test cases, I saw that the code was repeatedly recalculating the same values. In order to avoid having to recalculate the findings, this made me think about saving previously calculated results.

I used an array to implement memoization in order to better my strategy. I made sure that every number in the sequence was only calculated once by keeping the outcomes of each function call. I was able to access values instantly rather than recalculating them, which greatly reduced the number of unnecessary calculations. Although this was an improvement, I discovered that even though I only ever used the last three numbers at a time, I was still storing all values up to n in an array.

With this realization, I further improved my strategy by tracking the final three digits in the sequence using an iterative solution with just three variables. This kept the results accurate while doing away with the necessity for an array. The final approach was more memory and compute efficient because it employed a loop rather than recursive logic, which it closely mimicked.

I was able to comprehend how many strategies built upon one another because to this challenge. I gained a thorough grasp of the problem's structure by beginning with a straight recursive solution, and then I worked to optimize it gradually. I learned from the process how crucial it is to know when it makes sense to store intermediate findings and when it is not. It also reaffirmed that there are frequently several methods to address a problem and that trying out various strategies might yield deeper insights into enhancing coding clarity and efficiency.