Problem 3522: Calculate Score After Performing Instructions

Problem Information

Difficulty: Medium
Acceptance Rate: 56.55%
Paid Only: No
Tags: Array, Hash Table, String, Simulation

Problem Description

You are given two arrays, `instructions` and `values`, both of size `n`.

You need to simulate a process based on the following rules:

* You start at the first instruction at index `i = 0` with an initial score of 0. * If `instructions[i]` is `"add"`: * Add `values[i]` to your score. * Move to the next instruction `(i + 1)`. * If `instructions[i]` is `"jump"`: * Move to the instruction at index `(i + values[i])` without modifying your score.

The process ends when you either:

* Go out of bounds (i.e., `i < 0 or i >= n`), or * Attempt to revisit an instruction that has been previously executed. The revisited instruction is not executed.

Return your score at the end of the process.

Example 1:

Input: instructions = ["jump","add","add","jump","add","jump"], values = [2,1,3,1,-2,-3]

Output: 1

Explanation:

Simulate the process starting at instruction 0:

* At index 0: Instruction is `"jump"`, move to index `0 + 2 = 2`. * At index 2: Instruction is `"add"`, add `values[2] = 3` to your score and move to index 3. Your score becomes 3. * At index 3: Instruction is `"jump"`, move to index `3 + 1 = 4`. * At index 4: Instruction is `"add"`, add `values[4] = -2` to your score and move to index 5. Your score becomes 1. * At index 5: Instruction is `"jump"`, move to index `5 + (-3) = 2`. * At index 2: Already visited. The process ends.

Example 2:

Input: instructions = ["jump","add","add"], values = [3,1,1]

Output: 0

Explanation:

Simulate the process starting at instruction 0:

* At index 0: Instruction is `"jump"`, move to index `0 + 3 = 3`. * At index 3: Out of bounds. The process ends.

Example 3:

Input: instructions = ["jump"], values = [0]

Output: 0

Explanation:

Simulate the process starting at instruction 0:

* At index 0: Instruction is `"jump"`, move to index `0 + 0 = 0`. * At index 0: Already visited. The process ends.

Constraints:

* `n == instructions.length == values.length` * `1 <= n <= 105` * `instructions[i]` is either `"add"` or `"jump"`. * `-105 <= values[i] <= 105`

Code Snippets

C++:

```
class Solution {
public:
long long calculateScore(vector<string>& instructions, vector<int>& values) {

}
};
```

Java:

```
class Solution {
public long calculateScore(String[] instructions, int[] values) {

}
}
```

Python3:

```
class Solution:
def calculateScore(self, instructions: List[str], values: List[int]) -> int:
```