# Jump Game II

Try to solve the Jump Game II problem.

#### We'll cover the following

^

- Statement
- Examples
- Understand the problem
- Figure it out!
- Try it yourself

#### **Statement**

In a single-player jump game, the player starts at one end of a series of squares, with the goal of reaching the last square.

At each turn, the player can take up to s steps towards the last square, where s is the value of the current square.

For example, if the value of the current square is 3, the player can take either 3 steps, or 2 steps, or 1 step in the direction of the last square. The player cannot move in the opposite direction, that is, away from the last square.

You've been provided with the nums integer array, representing the series of squares.

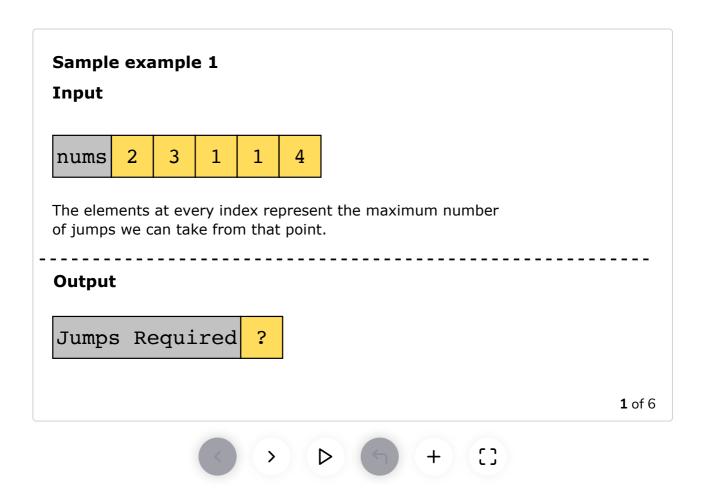
You're initially positioned at the first index of the array. Find the minimum number of jumps needed to reach the last index of the array.

You may assume that you can *always* reach the last index.

### **Constraints:**

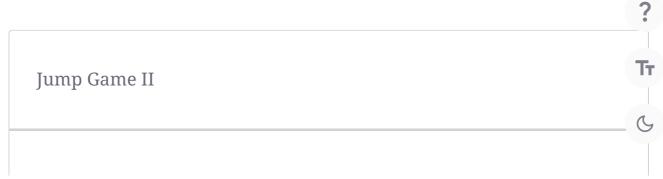
- $1 \leq {\sf nums.length} \leq 10^3$
- $0 \leq {\sf nums[i]} \leq 10^3$

### **Examples**



## Understand the problem

Let's take a moment to make sure you've correctly understood the problem. The quiz below helps you check if you're solving the correct problem:



What is the minimum number of jumps required to reach the end of the array if we start from the first element? array = [1, 2, 3, 4, 5]**A)** 3 **B)** 5 **C)** 1 Question 1 of 2 0 attempted Reset Quiz C

## Figure it out!

We have a game for you to play. Rearrange the logical building blocks to develop a clearer understanding of how to solve this problem.

**Note:** As an additional challenge, we have intentionally hidden the solution to this puzzle.

Drag and drop the cards to rearrange them in the correct sequence.

Traverse the entire nums array. On each  $i^{th}$  iteration, update farthestJump to the max of the current value of farthestJump, and i + nums[i].

Otherwise, do not update either the jumps variable or the currentJump variable, since we haven't yet completed the current jump.

Initialize three variables:

farthestJump, denoting the farthest index we can reach,

currentJump, denoting the end index of our current jump,

and jumps, to store the number of jumps. All three of these variables are set to 0.

At the end of the traversal, the jumps variable will contain the minimum number of jumps required to reach the last index.

If i is equal to currentJump,
we have completed the
current jump and can now
prepare to take the next jump
(if required). So we increment
the jumps variable by 1 and

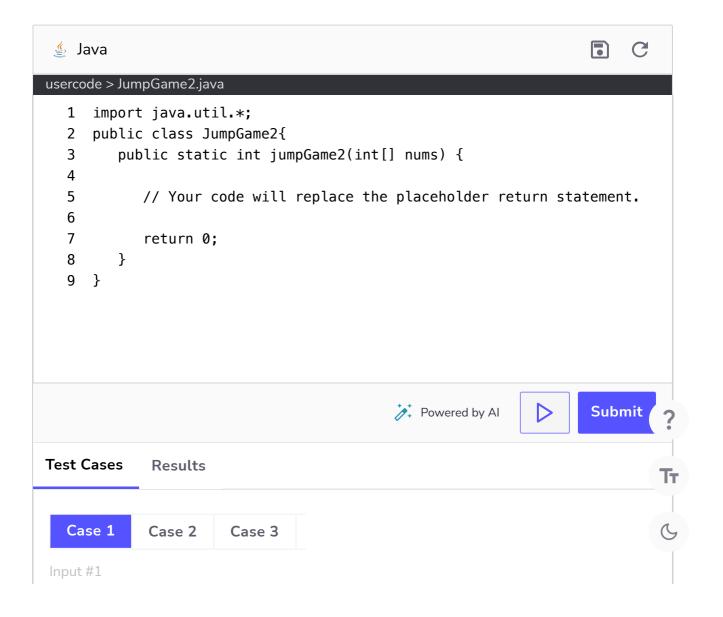






Implement your solution in the following coding playground.

**Note:** We have left the solution to this challenge as an exercise for you. You may try to translate the logic of the solved puzzle into a coded solution.



### Jump Game II



Solution: Minimum Nu...



Backtracking: Introduc...



?



