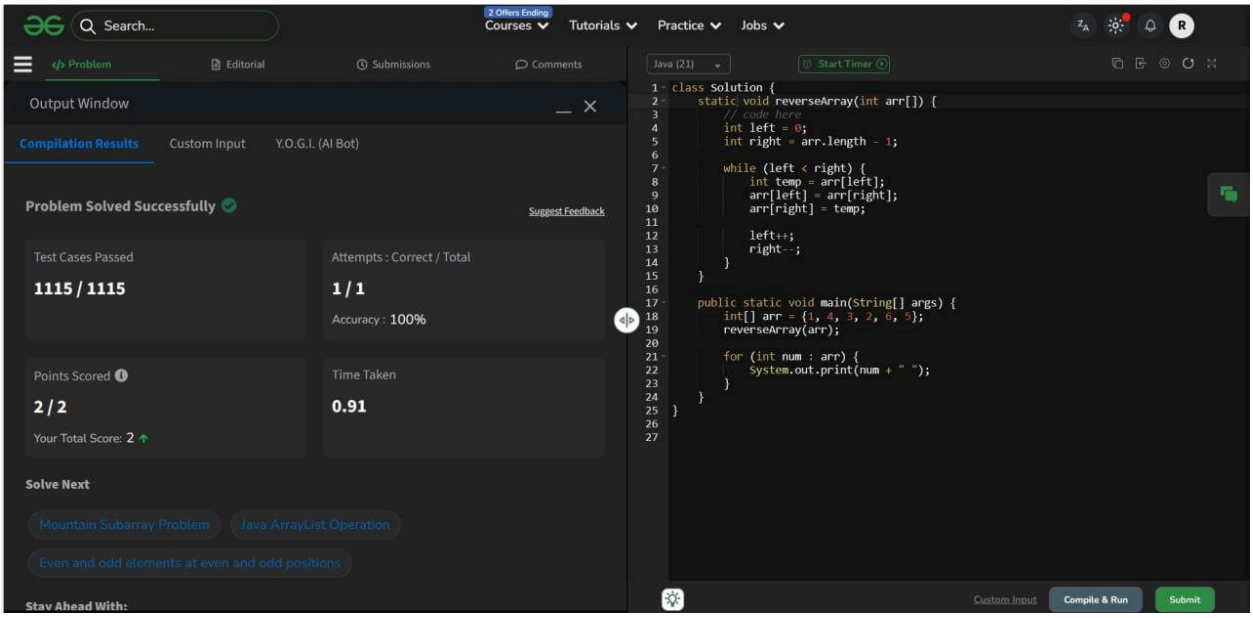
EXPERIMENT LIST FOR PROGRAMMING ABILITY AND LOGIC BUILDING 1

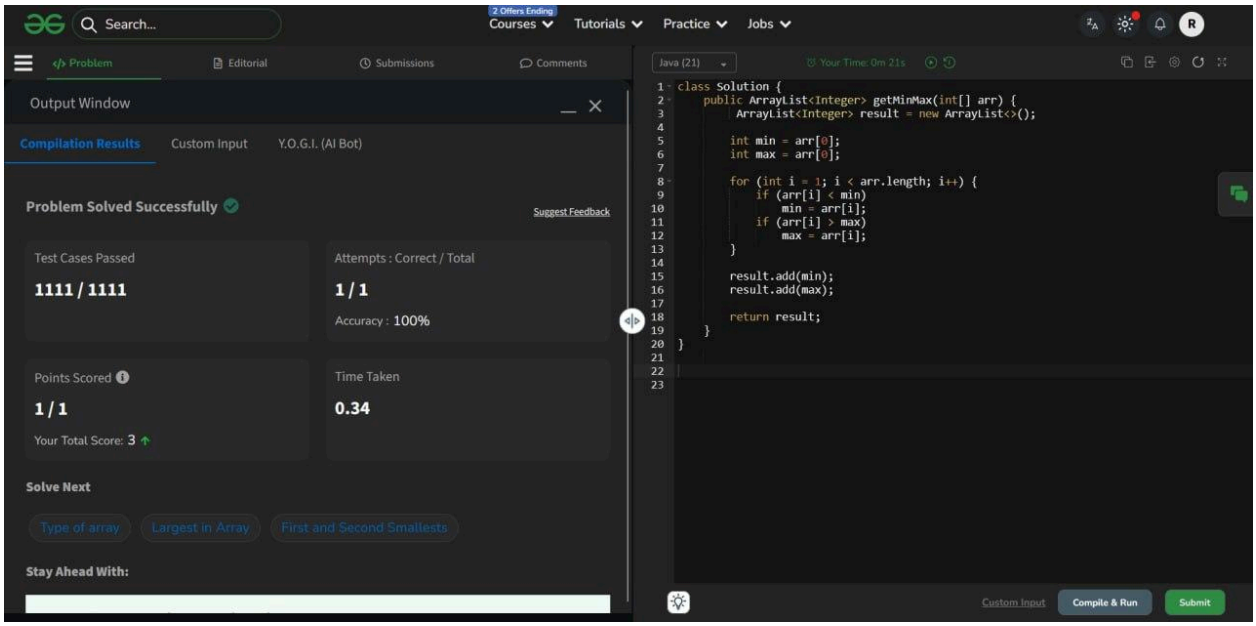
You are given an array of integers arr[]. You have to reverse the given array.
Note: Modify the array in place.
Examples:
Input: arr = [1, 4, 3, 2, 6, 5]
Output: [5, 6, 2, 3, 4, 1]

Approach (Two-Pointer Method)
Use two pointers:
Left → start of the array
right → end of the array
Swap arr[left] and arr[right]
Move left++ and right--
Stop when left >= right



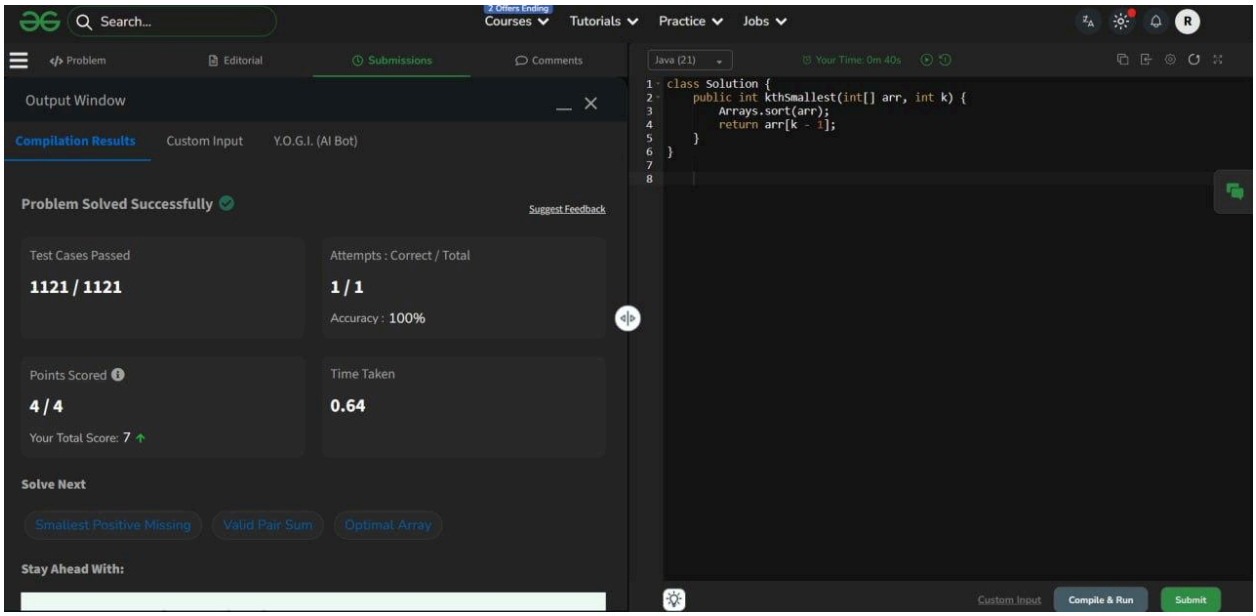
Given an array arr[]. Your task is to find the minimum and maximum elements in the array.
Examples:
Input: arr[] = [1, 4, 3, 5, 8, 6]
Output: [1, 8]
Explanation: minimum and maximum elements of array are 1 and 8.

Approach
Initialize
min = arr[0]
max = arr[0]
Traverse the array from index 1
Update:
if arr[i] < min → update min
if arr[i] > max → update max

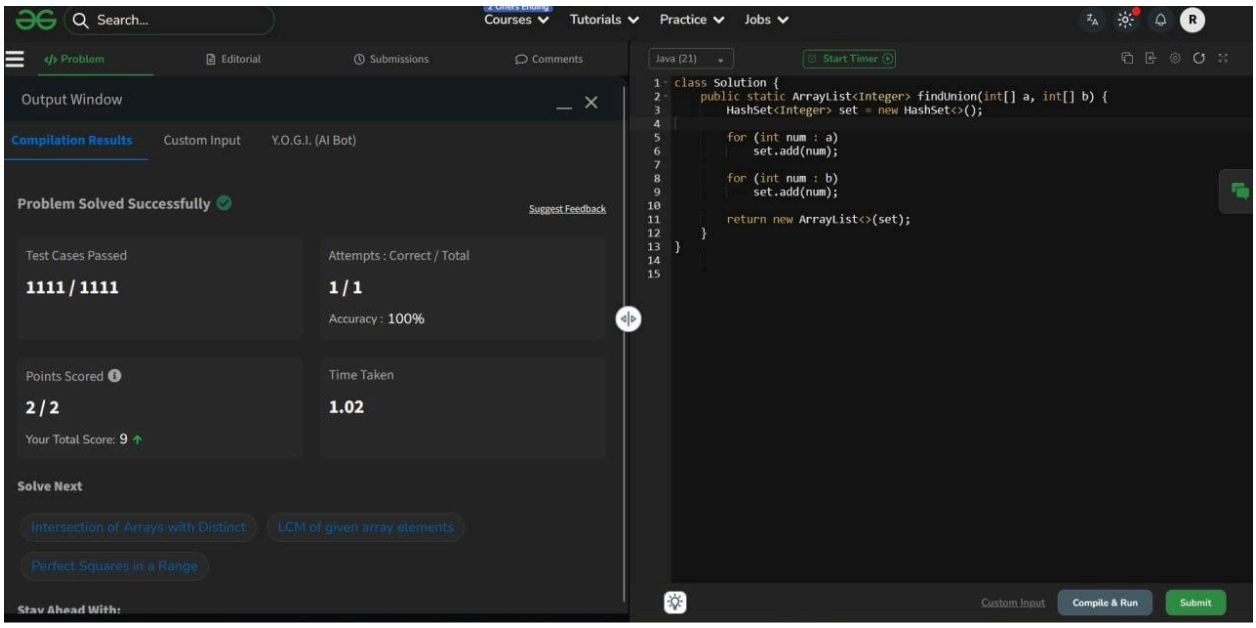


Given an integer array arr[] and an integer k, your task is to find and return the k
th smallest element in the given array.
Note: The kth smallest element is determined based on the sorted order of the array.
Examples :
Input: arr[] = [10, 5, 4, 3, 48, 6, 2, 33, 53, 10], k = 4
Output: 5
Explanation: 4th smallest element in the given array is 5.

Approach
Sort the array → the element at index k-1 is the answer.

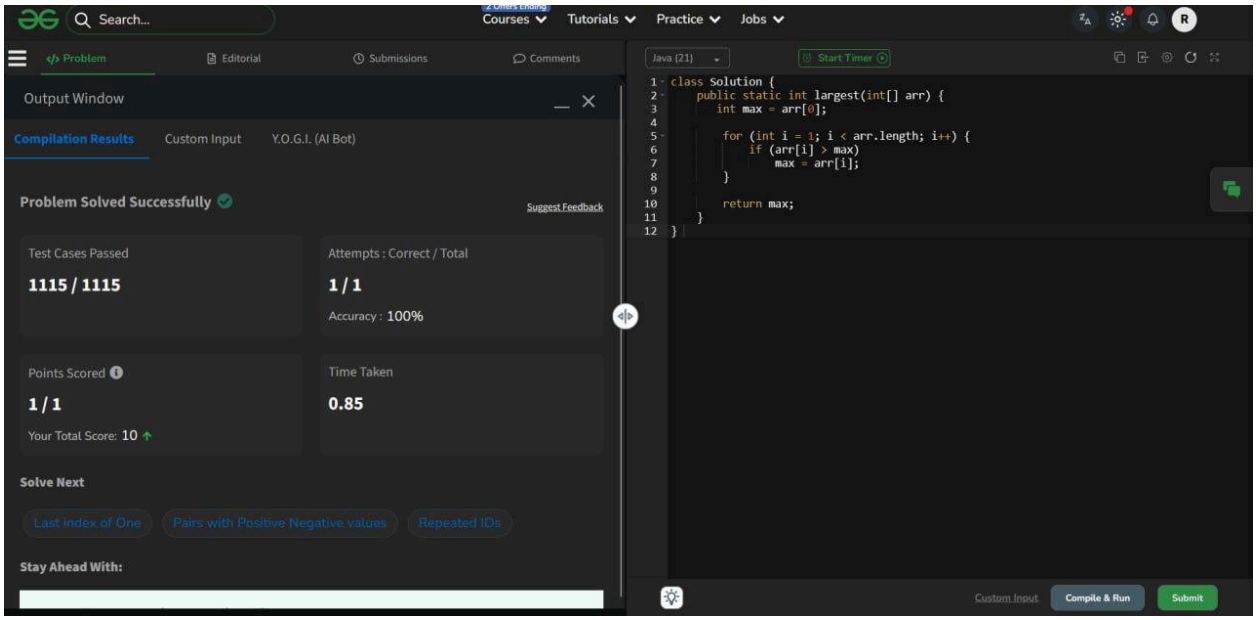


You are given two arrays a[] and b[], return the Union of both the arrays in any order.
The Union of two arrays is a collection of all distinct elements present in either of the arrays. If an element appears more than once in one or both arrays, it should be included only once in the result.
Note: Elements of a[] and b[] are not necessarily distinct.
Note that, You can return the Union in any order but the driver code will print the result in sorted order only.
Examples:
Input: a[] = [1, 2, 3, 2, 1], b[] = [3, 2, 2, 3, 3, 2]
Output: [1, 2, 3]
Explanation: Union set of both the arrays will be 1, 2 and 3.



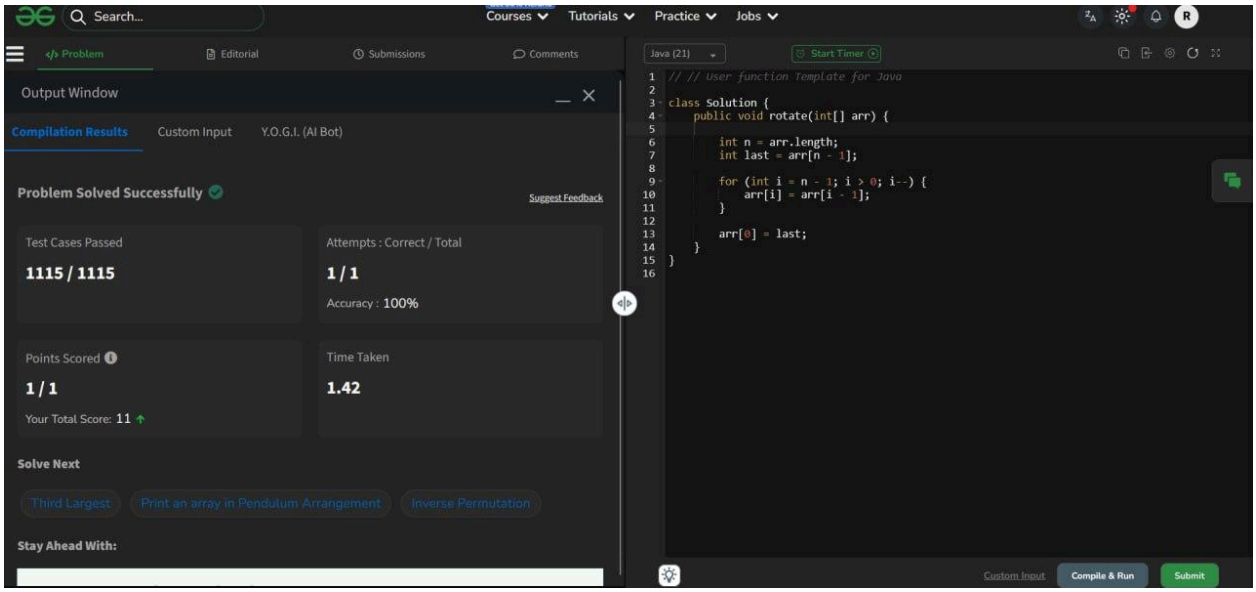
Given an array arr[]. The task is to find the largest element and return it.
Examples:
Input: arr[] = [1, 8, 7, 56, 90]
Output: 90
Explanation: The largest element of the given array is 90.

Approach
Assume the first element is the largest
Traverse the array
Update the maximum whenever a bigger value is found



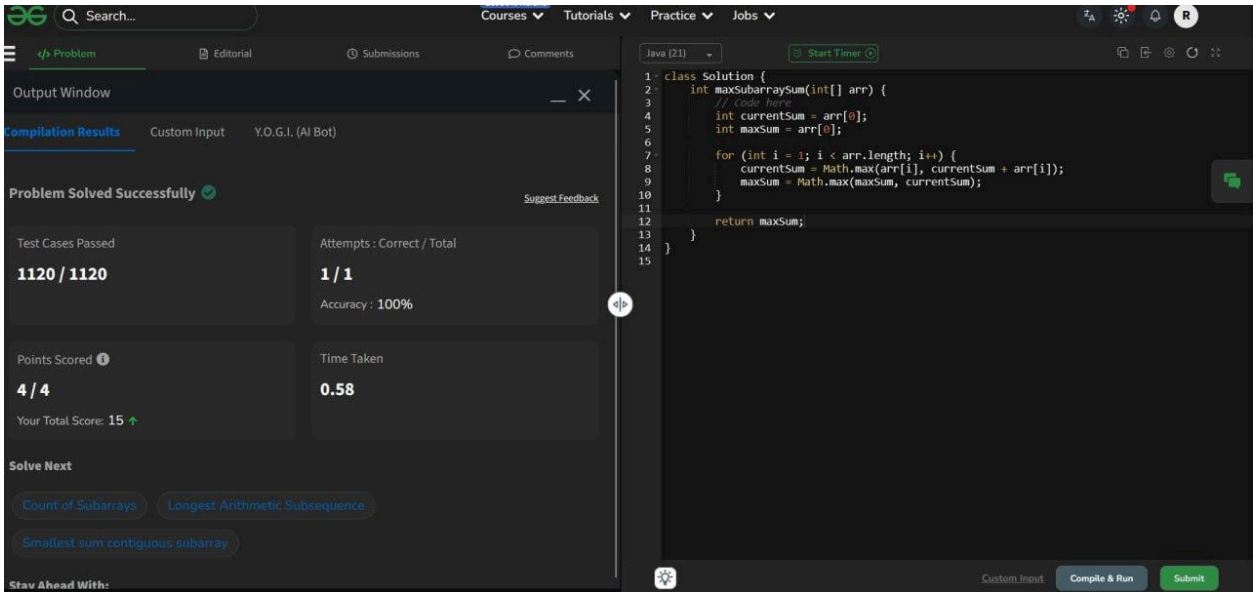
Given an array arr, rotate the array by one position in clockwise direction.
Examples:
Input: arr[] = [1, 2, 3, 4, 5]
Output: [5, 1, 2, 3, 4]
Explanation: If we rotate arr by one position in clockwise 5 come to the front and remaining those are shifted to the end.

Approach
Save the last element
Shift all elements one step to the right
Put the saved element at index 0



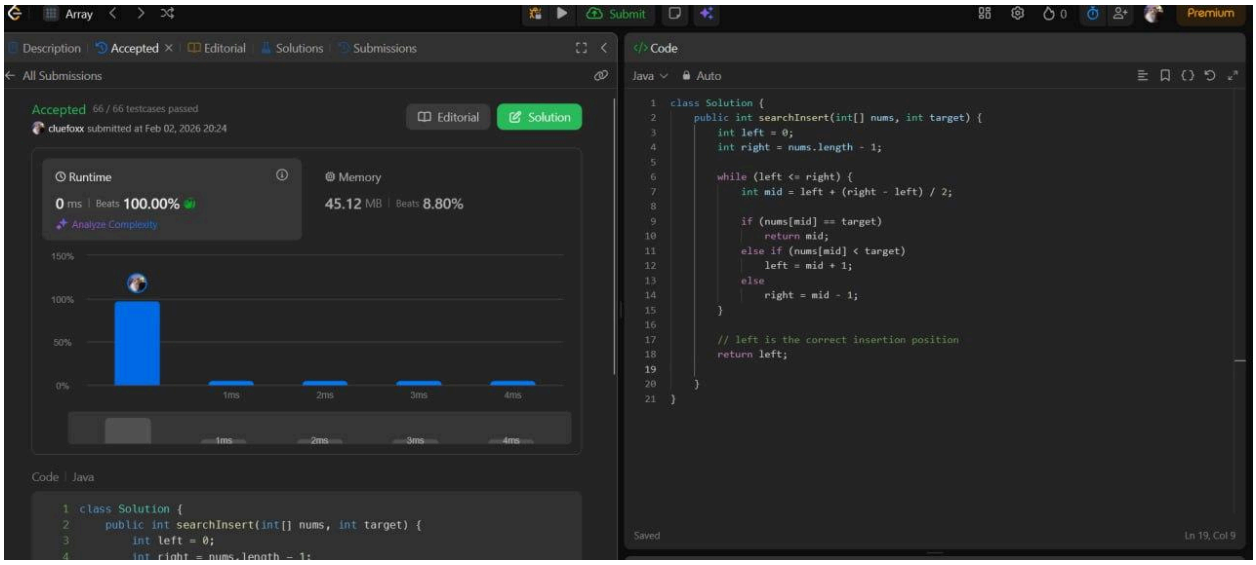
You are given an integer array arr[]. You need to find the maximum sum of a subarray (containing at least one element) in the array arr[].
Note : A subarray is a continuous part of an array.
Examples:
Input: arr[] = [2, 3, -8, 7, -1, 2, 3]
Output: 11
Explanation: The subarray [7, -1, 2, 3] has the largest sum 11.

Approach (Kadane's Algorithm)
Keep two variables:
currentSum → max sum ending at current index
maxSum → overall maximum found so far
At each step:
Either start new subarray from current element
Or extend previous subarray

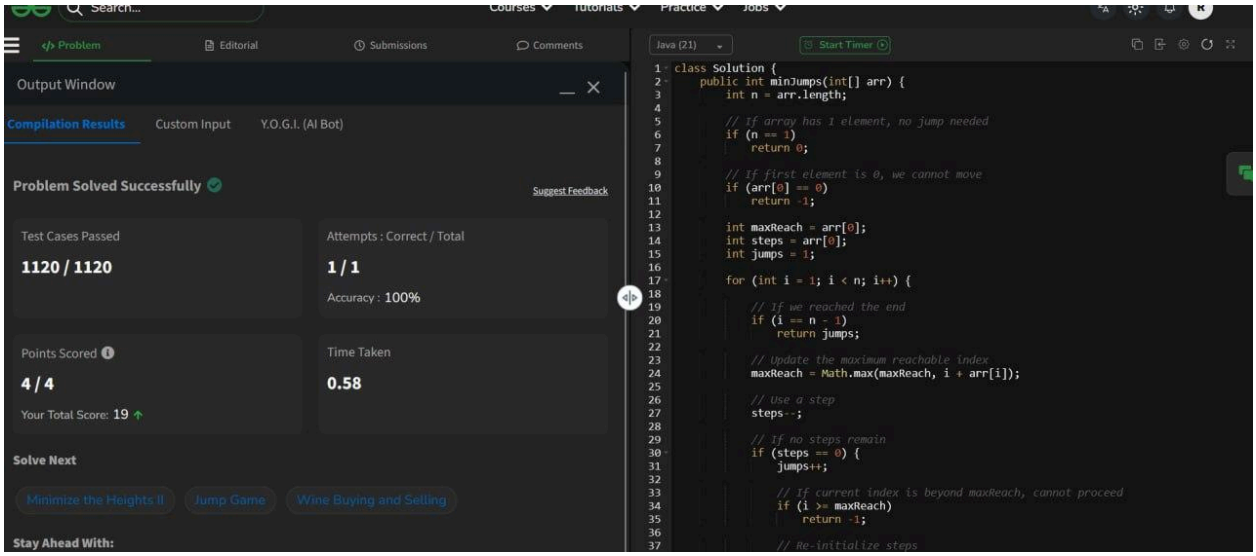


Given a sorted array of distinct integers and a target value, return the index if the target is found. If not, return the index where it would be if it were inserted in order.
You must write an algorithm with O(log n) runtime complexity.
Example 1:
Input: nums = [1,3,5,6], target = 5
Output: 2

Approach (Binary Search)
Since the array is sorted and distinct
Use binary search:
If target is found → return its index
If not found → return the position where it should be inserted



You are given an array arr[] of non-negative numbers. Each number tells you the maximum number of steps you can jump forward from that position.
For example:
• If arr[i] = 3, you can jump to index i + 1, i + 2, or i + 3 from position i.
• If arr[i] = 0, you cannot jump forward from that position.
Your task is to find the minimum number of jumps needed to move from the first position in the array to the last position.
Note: Return -1 if you can't reach the end of the array.
Examples :
Input: arr[] = [1, 3, 5, 8, 9, 2, 6, 7, 6, 8, 9]
Output: 3
Explanation: First jump from 1st element to 2nd element with value 3. From here we jump to 5th element with value 9, and from here we will jump to the last.



Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target.
You may assume that each input would have exactly one solution, and you may not use the same element twice.
You can return the answer in any order.
Example 1:
Input: nums = [2,7,11,15], target = 9
Output: [0,1]
Explanation: Because nums[0] + nums[1] == 9, we return [0, 1].

Approach (HashMap)
Traverse the array once
For each number:
Compute complement = target - nums[i]
If complement already exists in the map → solution found
Otherwise, store the current number with its index