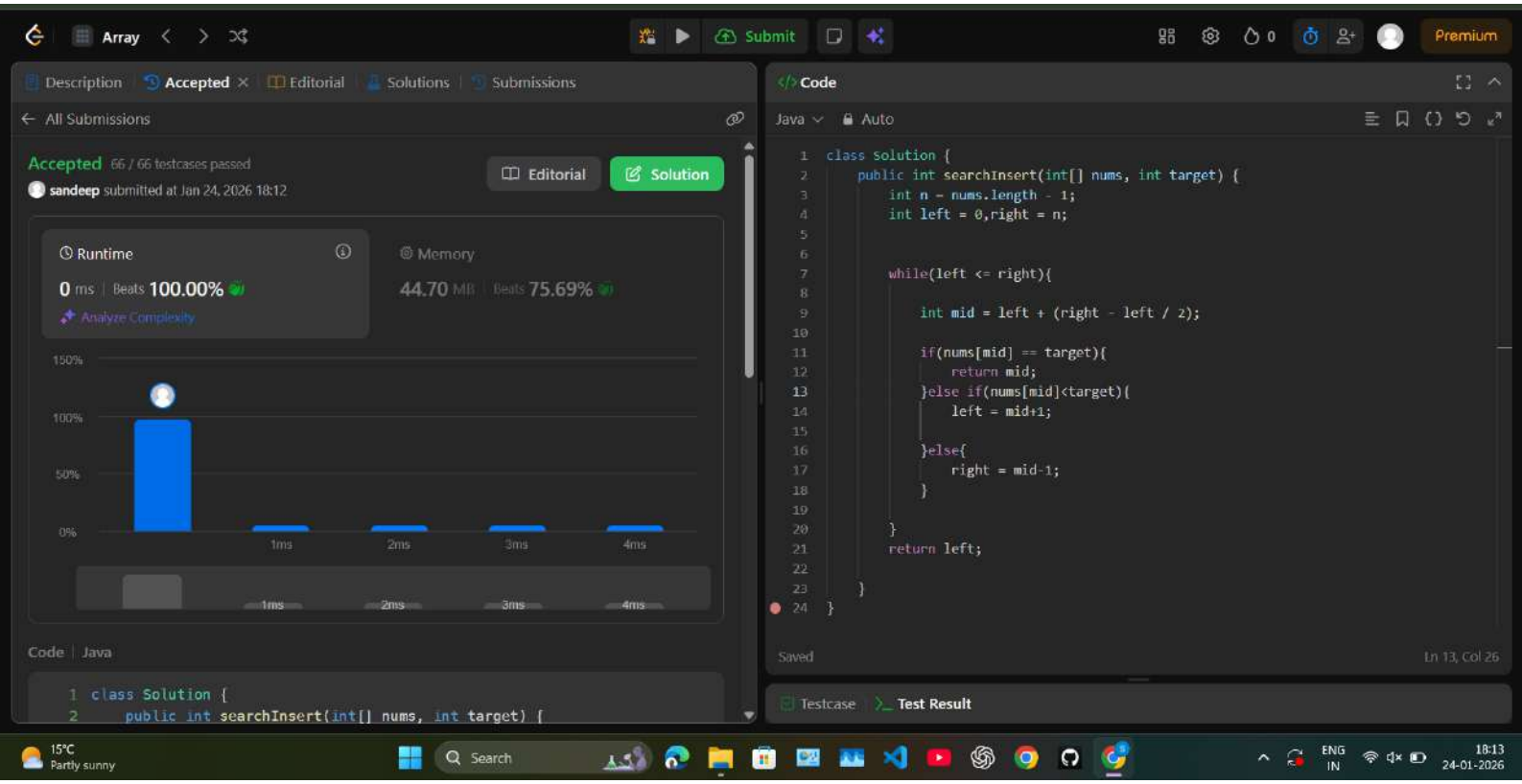
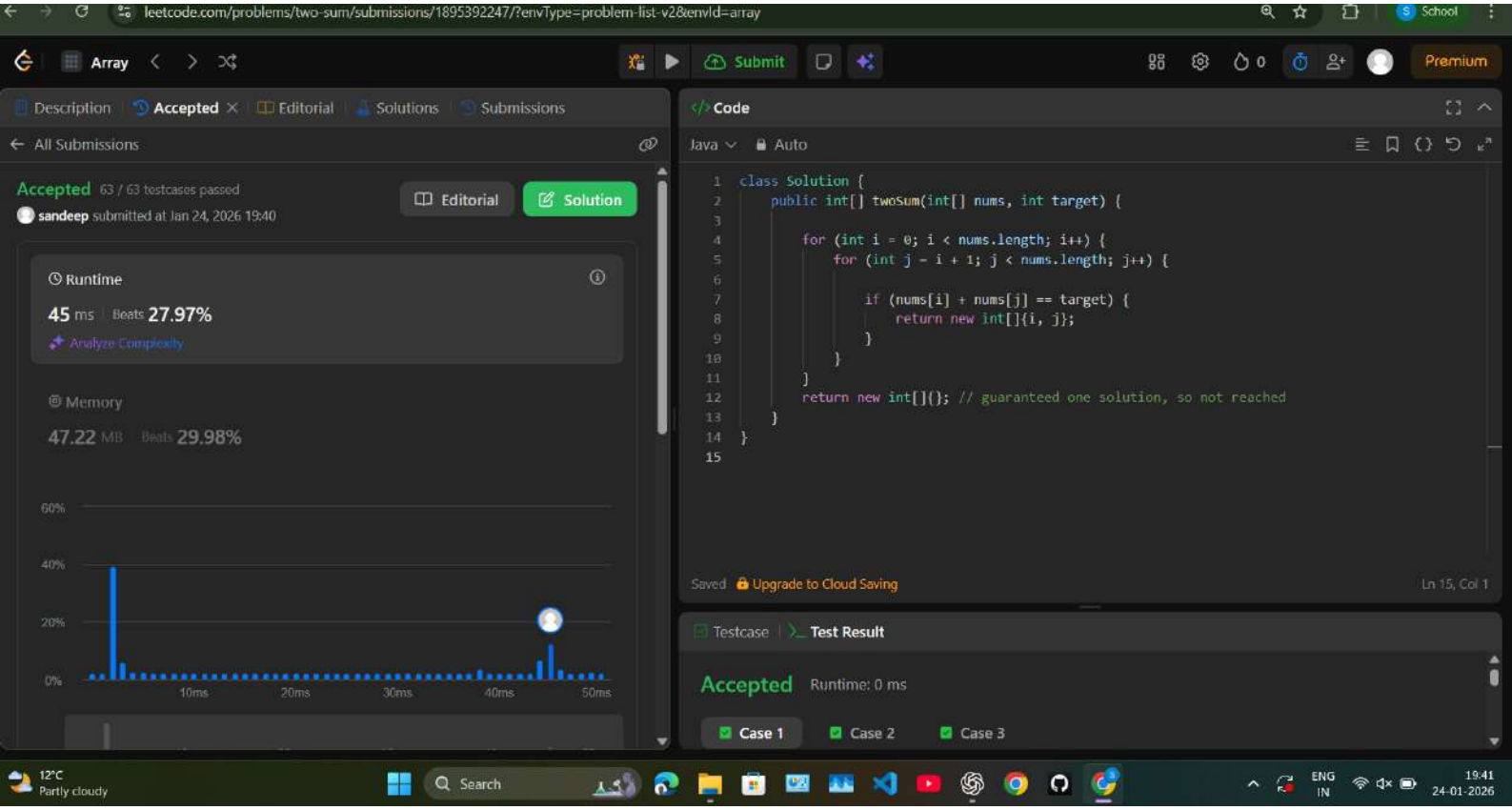
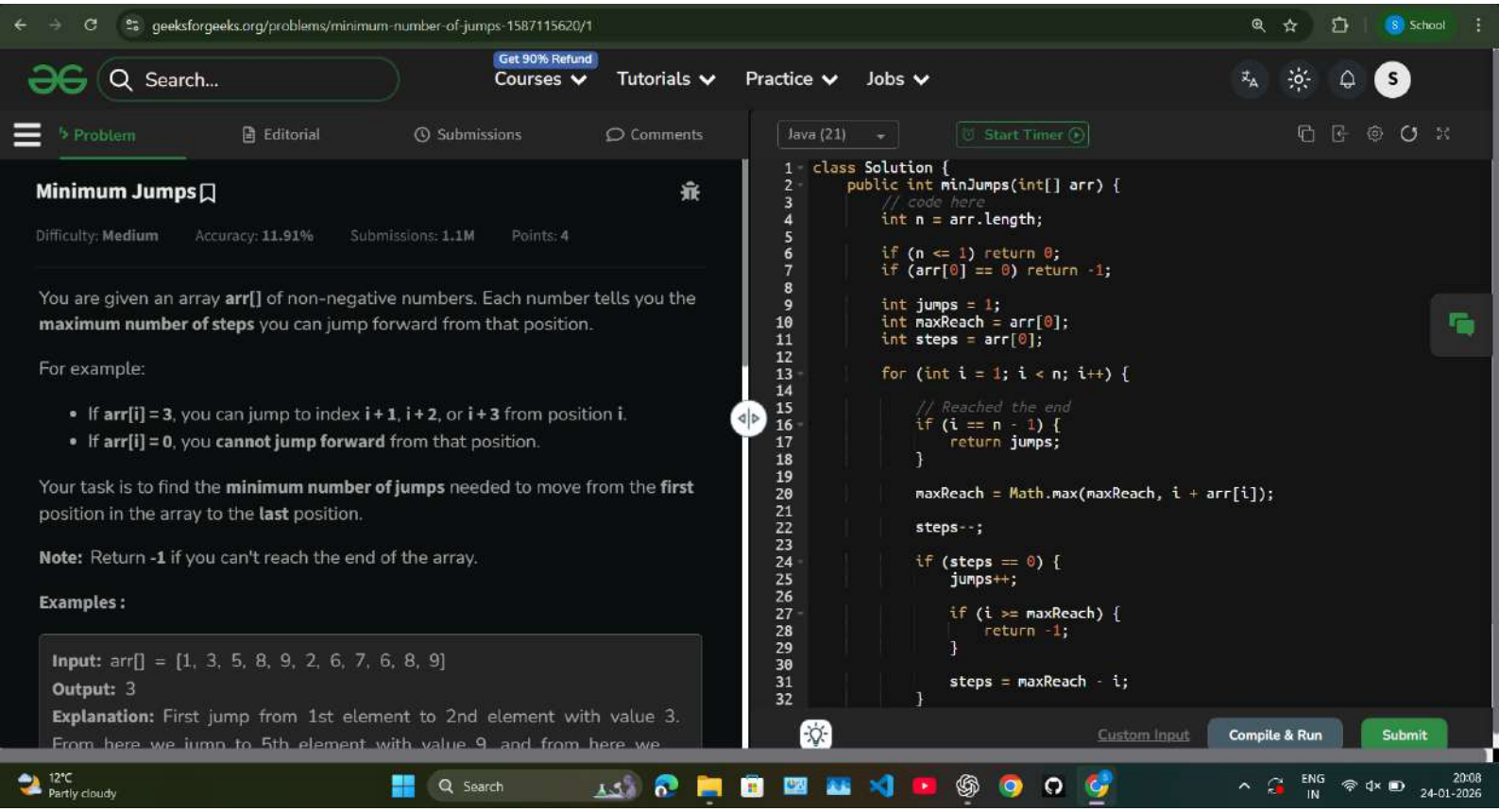

Array
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 66 / 66 testcases passed
sandeep submitted at Jan 24, 2026 18:12
Editorial
Solution
Runtime
0 ms | Beats 100.00%
Analyze Complexity
Memory
44.70 MB | Beats 75.69%
150%
100%
50%
0%
1ms
2ms
3ms
4ms
Code | Java
class Solution {
public int searchInsert(int[] nums, int target) {
Code
Java
Auto
class Solution {
public int searchInsert(int[] nums, int target) {
int n = nums.length - 1;
int left = 0,right = n;
while(left <= right){
int mid = left + (right - left / 2);
if(nums[mid] == target){
return mid;
}else if(nums[mid]<target){
left = mid+1;
}else{
right = mid-1;
}
}
return left;
}
}
Saved
Ln 13, Col 26
Testcase
Test Result
15°C
Partly sunny
Search
ENG
IN
18:13
24-01-2026


leetcode.com/problems/two-sum/submissions/1895392247/?envType=problem-list-v2&envId=array
Array
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Accepted 63 / 63 testcases passed
sandeep submitted at Jan 24, 2026 19:40
Editorial
Solution
Runtime
45 ms | Beats 27.97%
Analyze Complexity
Memory
47.22 MB | Beats 29.98%
60%
40%
20%
0%
10ms
20ms
30ms
40ms
50ms
Code
Java
Auto
class Solution {
public int[] twoSum(int[] nums, int target) {
for (int i = 0; i < nums.length; i++) {
for (int j = i + 1; j < nums.length; j++) {
if (nums[i] + nums[j] == target) {
return new int[]{i, j};
}
}
}
return new int[]{}; // guaranteed one solution, so not reached
}
}
Saved
Upgrade to Cloud Saving
Ln 15, Col 1
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Case 3
12°C
Partly cloudy
Search
ENG
IN
19:41
24-01-2026


geeksforgeeks.org/problems/minimum-number-of-jumps-1587115620/1
Search...
Get 90% Refund
Courses
Tutorials
Practice
Jobs
Problem
Editorial
Submissions
Comments
Minimum Jumps
Difficulty: Medium
Accuracy: 11.91%
Submissions: 1.1M
Points: 4
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
Java (21)
Start Timer
class Solution {
public int minJumps(int[] arr) {
// code here
int n = arr.length;
if (n <= 1) return 0;
if (arr[0] == 0) return -1;
int jumps = 1;
int maxReach = arr[0];
int steps = arr[0];
for (int i = 1; i < n; i++) {
// Reached the end
if (i == n - 1) {
return jumps;
}
maxReach = Math.max(maxReach, i + arr[i]);
steps--;
if (steps == 0) {
jumps++;
if (i >= maxReach) {
return -1;
}
steps = maxReach - i;
}
Custom Input
Compile & Run
Submit
12°C
Partly cloudy
Search
ENG
IN
20:08
24-01-2026