

WORKSHEET 2

Student Name: Avreet Singh

UID: 22BCS16488

Branch: CSE

Section/Group: NTPP-602-A

Semester: 6

Date of Performance: 20/01/2025

Subject Name: AP LAB II

Subject Code: 22CSH-359

1. Aim: Arrays, Stacks, Queues

- **Problem 1.2.1: Two Sum**

Problem Statement: Given an array of integers nums and an integer target, return the indices of the two numbers such that they add up to target. Each input has exactly one solution, and you cannot use the same element twice.

- **Problem 1.2.2: Jump Game II**

Problem Statement: You are given a 0-indexed array nums of length n. You are initially positioned at nums[0]. Each element nums[i] represents the maximum length of a forward jump from index i. Return the minimum number of jumps to reach nums[n - 1].

2. Algorithm:

Problem 1.2.1

- Iterate through all pairs of indices (a, b) in the array, where $b > a$.
- Check if the sum of the numbers at indices a and b equals the target.
- If a match is found, return the indices [a, b].
- If no match is found after checking all pairs, return an empty list.

Problem 1.2.2

- Initialize jumps to count the number of jumps, current_end for the farthest point reachable in the current jump, and farthest for the farthest point overall.
- Iterate through the array (excluding the last index).
- For each index, update farthest as the maximum of its current value and the sum of the current index and its jump value.
- If the current index reaches current_end:
- Increment the jumps count.
- Update current_end to farthest.
- If current_end is greater than or equal to the last index, break the loop.
- Return the total jumps.

3. Source code/Implementation:

Problem 1.2.1

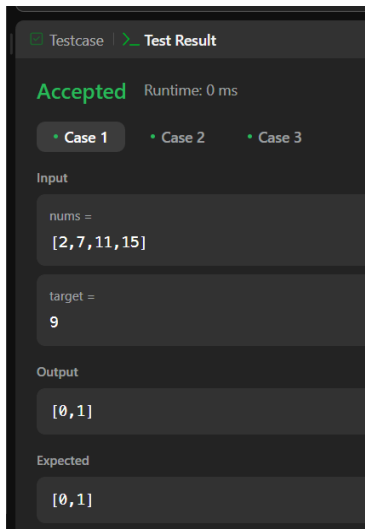
```
class Solution:
    def twoSum(self, nums, target):
        for a in range(len(nums)):
            for b in range(a + 1, len(nums)):
                if nums[a] + nums[b] == target:
                    return [a, b]
        return []
```

Problem 1.2.2

```
class Solution:
    def jump(self, nums):
        n=len(nums)
        jumps=0
        current_end=0
        farthest=0
        for a in range(n - 1):
            farthest = max(farthest, a + nums[a])
            if a == current_end:
                jumps += 1
                current_end = farthest
            if current_end >= n - 1:
                break
        return jumps
```

4. Screenshots of outputs:

• Problem 1.2.1



Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

nums =
[2,7,11,15]

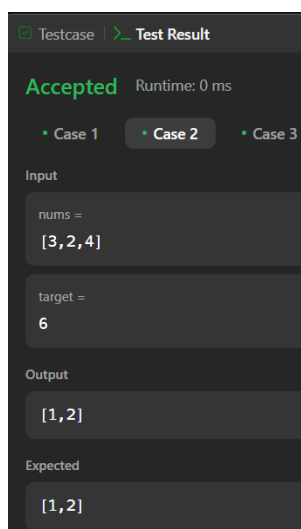
target =
9

Output

[0,1]

Expected

[0,1]



Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

nums =
[3,2,4]

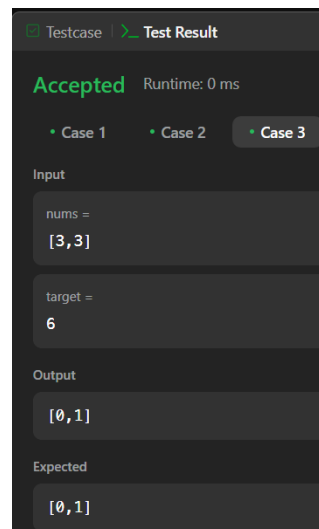
target =
6

Output

[1,2]

Expected

[1,2]



Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

nums =
[3,3]

target =
6

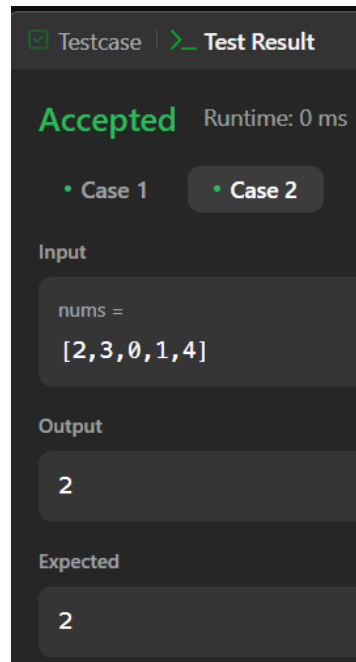
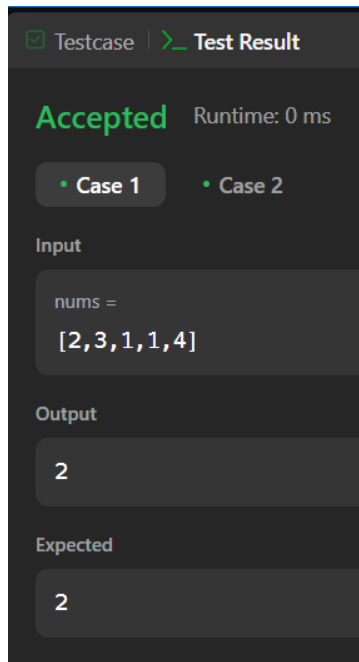
Output

[0,1]

Expected

[0,1]

Problem 1.2.2



5. Learning Outcomes:

- Learn to find two numbers in an array that sum to a target and return their indices using both brute-force and optimized methods.
- Understand how to minimize the number of jumps required to reach the end of an array using a greedy approach.
- Gain experience in handling array manipulation problems with multiple solutions.
- Understand how to optimize time complexity when solving problems with a fixed input size.
- Develop problem-solving skills for scenarios involving dynamic decision-making based on array values.