Experiment 1.2

Student Name: Anand Kumar **UID: 22BCS50097**
Branch: CSE **Section: IOT_627-A**
Semester: 6th **DOP: 22/01/25**
Subject: AP Lab-2 **Subject Code:22CSP-351**

Aim:

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].

Problem 1.2.3: Simplify Path

- **Problem Statement**: Given a string path, which is an absolute path to a file or directory in a Unix-style file system, convert it to the simplified canonical path.

Objective:

Arrays, stacks, and queues are all data structures understanding and implementation on leet code

Implementation

1. Initialize an empty hash map (dict).
2. Iterate through the nums array:
    - For each element num, calculate the complement: complement = target - num.
    - Check if the complement exists in the hash map:
        - If it does, return the indices of the complement and the current number.
        - If it doesn't, add the current number and its index to the hash map.
3. Return the indices of the two numbers that add up to the target.

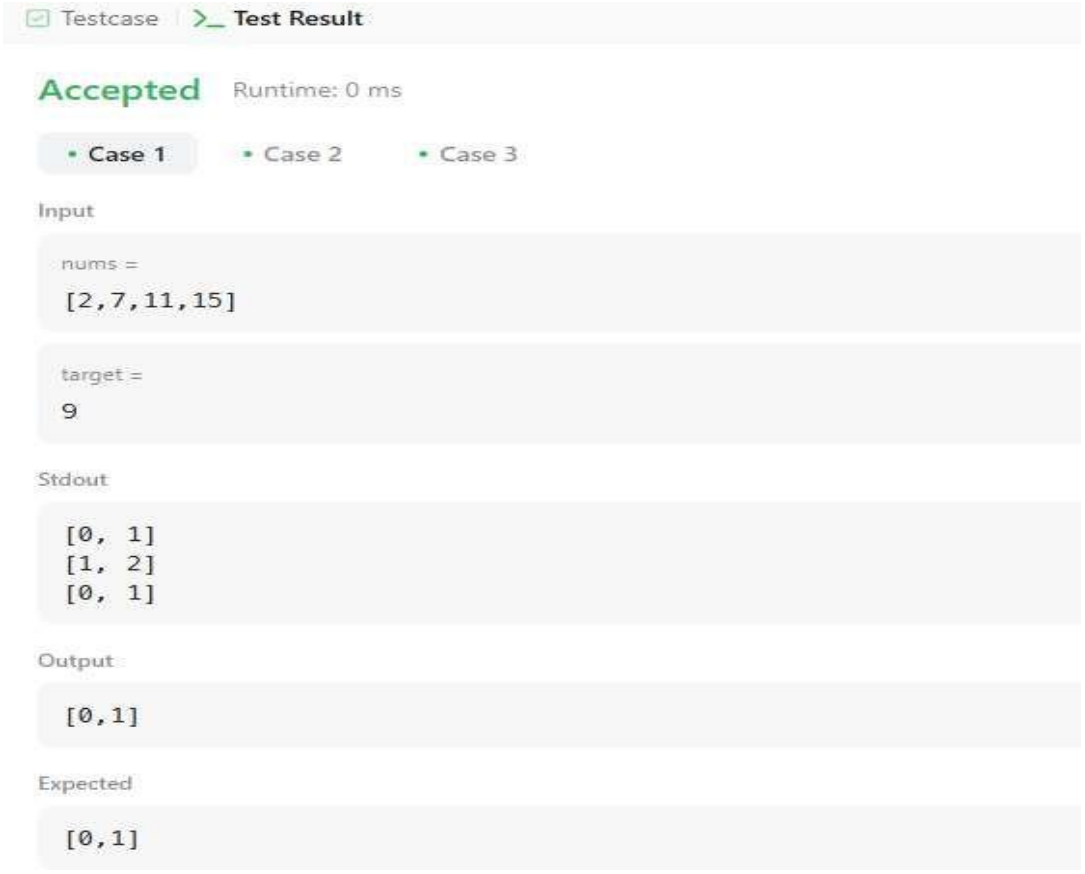
Code: 1.2.1

Code

Python3 Auto

```python
class Solution:
    def twoSum(self, nums, target):
        seen = {}
        for i, num in enumerate(nums):
            complement = target - num
            if complement in seen:
                return [seen[complement], i]
            seen[num] = i

# ROSH
solution = Solution()

nums1 = [2, 7, 11, 15]
target1 = 9
print(solution.twoSum(nums1, target1))

nums2 = [3, 2, 4]
target2 = 6
print(solution.twoSum(nums2, target2))

nums3 = [3, 3]
target3 = 6
print(solution.twoSum(nums3, target3))

```

Output:


Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Case 3
Input
nums =
[2,7,11,15]
target =
9
Stdout
[0, 1]
[1, 2]
[0, 1]
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]

CODE: 1.2.2

Code

Python3 Auto

```python
class Solution:
    def jump(self, nums):
        n = len(nums)
        jumps = 0
        current_end = 0
        farthest = 0

        for i in range(n - 1):
            farthest = max(farthest, i + nums[i])
            if i == current_end:
                jumps += 1
                current_end = farthest
                if current_end >= n - 1:
                    break

        return jumps

# ROSH
solution = Solution()

nums1 = [2, 3, 1, 1, 4]
print(solution.jump(nums1))

nums2 = [2, 3, 0, 1, 4]
print(solution.jump(nums2))

```

Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

nums =
[2,3,1,1,4]

Stdout

2
2

Output

2

Expected

2

Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

nums =
[2,3,0,1,4]

Output

2

Expected

2

CODE: 1.2.3

</> Code

Python3 Auto

```
    def simplifyPath(self, path):
        stack = []
        parts = path.split('/')
        for part in parts:
            if part == '..':
                if stack:
                    stack.pop()
            elif part and part != '.':
                stack.append(part)
        return '/' + '/'.join(stack)

# ROSH
solution = Solution()

path1 = "/home/"
print(solution.simplifyPath(path1))

path2 = "/home//foo/"
print(solution.simplifyPath(path2))

path3 = "/home/user/Documents/../Pictures"
print(solution.simplifyPath(path3))

path4 = "/../"
print(solution.simplifyPath(path4))

path5 = "/.../a/../b/c/../d/./"
print(solution.simplifyPath(path5))
```

Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3 • Case 4 • Case 5

Input

path =
"/home/"

Stdout

```
/home
/home/foo
/home/user/Pictures
/
/.../b/d
```

Output

"/home"

Expected

"/home"

Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3 • Case 4 • Case 5

Input

path =
"/home//foo/"

Output

"/home/foo"

Expected

"/home/foo"

Testcase | >_ Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • **Case 3** • Case 4 • Case 5

Input

path =
"/home/user/Documents/../Pictures"

Output

"/home/user/Pictures"

Expected

"/home/user/Pictures"

Testcase | >_ Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3 • **Case 4** • Case 5

Input

path =
"/../"

Output

"/"

Expected

"/"

Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3 • Case 4 • Case 5

Input

path =
"/.../a/../b/c/../d/./"

Output

"/.../b/d"

Expected

"/.../b/d"