DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
Discover. Learn. Empower.

Experiment 1.2

Student Name: Jayant Sharma
UID: 22BCS16668
Branch: BE-CSE
Section/Group: NTPP-602-A
Semester: 6th
Date of Performance:20-01-25
Subject Name: AP LAB-II
Subject Code: 22CSP-351

1.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.

2.Algorithm:

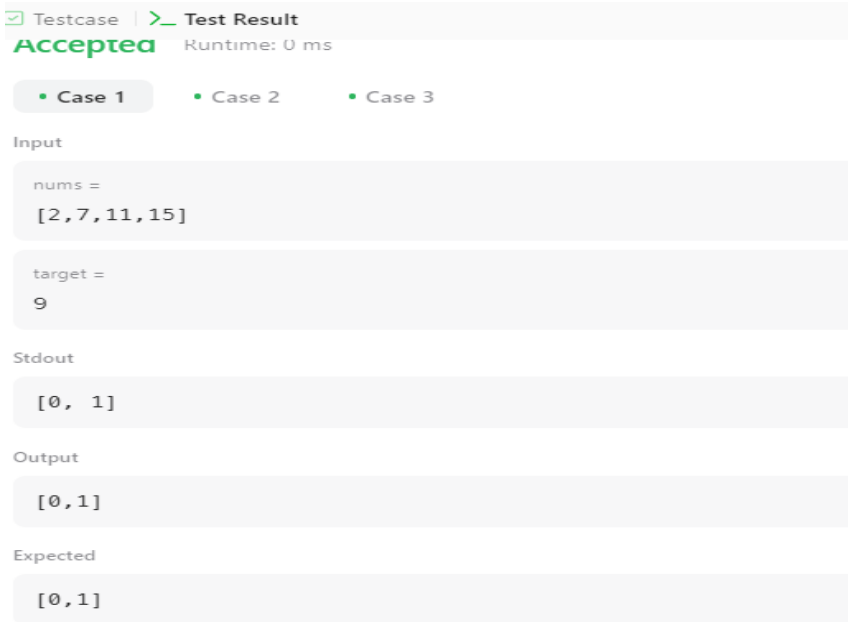
1. Initialize an empty hash map (dict).
2. Iterate through the nums array:
3. For each element num, calculate the complement: complement = target - num.
4. Check if the complement exists in the hash map:
5. If it does, return the indices of the complement and the current number.
6. If it doesn't, add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

Code: 1.2.1

```
class Solution(object):
    def twoSum(self, nums, target):

        s1 = []
        s2 = []


        for i in range(len(nums)):
            s1.append((nums[i], i))


        while s1:
            num1, idx1 = s1.pop()

            s2.append((num1, idx1))


            for num2, idx2 in s2:
                if num1 + num2 == target and idx1 != idx2:
                    return [idx1, idx2]

        return []


sol = Solution()
nums = [2, 7, 11, 15]
target = 9
print(sol.twoSum(nums, target))
```

\

Output:



CODE: 1.2.2

</> Code

Python Auto

```python
class Solution(object):
    def jump(self, nums):
        """
        :type nums: List[int]
        :rtype: int
        """
        n = len(nums)
        if n <= 1:
            return 0

        jumps = 0
        farthest = 0
        current_end = 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

```

OUTPUT:

Testcase Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

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

Output

2

Expected

2

CODE: 1.2.3

Code

Python Auto

```python
class Solution(object):
    def simplifyPath(self, path):

        stack = []
        parts = path.split("/")

        for part in parts:
            if part == "..":
                if stack:
                    stack.pop()  # Go up one directory level
            elif part and part != ".":
                stack.append(part)  # Add valid directory/file name
```

OUTPUT:

Testcase | Test Result

Accepted Runtime: 0 ms

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

Input

path =
"/home/"

Output

"/home"

Expected

"/home"

5.Learning Outcomes:-

- **Array Manipulation** – Solve problems using arrays and index-based operations.
- **Efficient Algorithms** – Apply hashing and greedy methods for optimization.
- **Stack Usage** – Use stacks for directory path simplification.
- **Edge Case Handling** – Manage constraints like duplicates and empty inputs.
- **Problem-Solving Skills** – Break down problems and debug efficiently.