[Click on the logo to find **Problem Statement**]

Intuition

Imagine you have a list of numbers and a target value. The problem asks you to find any two numbers in the list that add up to the target. One way to do this is by considering all possible pairs of numbers in the list and checking if any of these pairs adds up to the target.

The outer loop of the algorithm allows you to pick a number from the list and fix it as the first element of a potential pair. The inner loop then iterates through the remaining numbers in the list, trying to find a second number that, when added to the first number, equals the target.

By systematically checking all pairs of numbers, the algorithm exhaustively explores all possible combinations. If it finds a pair where the elements add up to the target, it immediately returns the indices of these elements as the solution.

While this approach is straightforward and easy to implement, it is not the most efficient. The reason is that it checks all possible pairs, leading to a quadratic time complexity. However, it serves as a basic and intuitive way to solve the problem by brute-forcing through all the options.

Approach

Nested Loop Iteration: Use two nested loops to iterate through all pairs of elements in the input list nums. The outer loop variable, say i, represents the first index of the pair, ranging from 0 to len(nums) - 1. The inner loop variable, say j, represents the second index of the pair, ranging from i + 1 to len(nums) - 1.

Pair Comparison: For each pair of indices (i, j) where i is less than j, compare the elements at these indices in the input list. Check if nums[i] + nums[j] equals the target value.

Solution Found: If a pair is found where the elements add up to the target, return the indices [i, j] as the solution. The solution is found in constant time during this step.

No Solution Found: If no such pair is found after checking all possible pairs, return an empty list to indicate that no solution exists.

Complexity

- Time complexity:$O(N^2)$
- Space complexity: O(1)

Code

```python
class Solution:
    def twoSum(self, nums: List[int], target: int) -> List[int]:
        for i in range(len(nums)):
            for j in  range(i+1, len(nums)):
                if nums[i] + nums[j] == target:
                    return [i,j]
        return[]
```