**Two Sum**

Question: Given an array of integers nums and an integer target, return *indices of the two numbers such that they add up to target*.

**Example**:

**Input:** nums = [2,7,11,15], target = 9

**Output:** [0,1]

**Output:** Because nums[0] + nums[1] == 9, we return [0, 1].

**Similar questions:**

[3Sum](https://leetcode.com/problems/3sum/)

Medium

[4Sum](https://leetcode.com/problems/4sum/)

Medium

[Two Sum II - Input array is sorted](https://leetcode.com/problems/two-sum-ii-input-array-is-sorted/)

Easy

[Two Sum III - Data structure design](https://leetcode.com/problems/two-sum-iii-data-structure-design/)

Easy

[Subarray Sum Equals K](https://leetcode.com/problems/subarray-sum-equals-k/)

Medium

[Two Sum IV - Input is a BST](https://leetcode.com/problems/two-sum-iv-input-is-a-bst/)

Easy

[Two Sum Less Than K](https://leetcode.com/problems/two-sum-less-than-k/)

**Solution**:

class Solution(object):

def twoSum(self, nums, target):

"""

:type nums: List[int]

:type target: int

:rtype: List[int]

"""

# Method 1: This is a brute force way which searches all possible pairs of numbers.

This method takes about 28 ms to finish.

# index = []

# for i in range(len(nums)):

# for j in range(len(nums) - i - 1):

# if nums[j+i+1] == target - nums[i]:

# index = [i, j+i+1]

# return index

# Method 2: This method utilizes the default function of list, i.e., in and index.

We check if the remaining value is in the list. If so, obtain the index of that number in the list.

Using this method, we have to search the entire list for the remaining value.

This method takes about 20 ms.

# index = []

# for i, num in enumerate(nums):

# remain = target - num

# if remain in nums:

# index = [i, nums.index(remain)]

# break

# return index

# Method 3: This method uses a hash map (i.e., a dictionary) to speed up the search.

At first, we create a dictionary in which the keys are the numbers and the values are the indexes.

We then check if the remaining value is in the dictionary. If so, we obtain the corresponding value (i.e., the index in the original list).

Using this method, we have to search the entire dictionary for the remaining value.

This method takes about 20 ms.

Ndic = {}

for i, num in enumerate(nums):

Ndic[num] = i

for v, k in Ndic.items():

remain = target - v

if remain in Ndic:

return [k, Ndic[remain]]

# Method 4: Similar to method 3, we also create a dictionary with this method.

We check if the remaining value is the dictionary. If not, we save this value and its index into the dictionary.

Using this method, we don’t need to search the entire list/dictionary. We only need to search those numbers we have seen.

This method takes about 12 ms.

# h = {}

# for i, num in enumerate(nums):

# n = target - num

# if n not in h:

# h[num] = i

# else:

# return [h[n], i]