Hướng dẫn giải bài toán ”Two Sum”

1. Mô tả bài toán

Cho một danh sách các số nguyên $nums$ và một số nguyên $target$, hãy tìm hai chỉ số i, j $(i \neq j)$ sao cho:

$$nums[i] + nums[j] = target$$

Yêu cầu:

- Chỉ có duy nhất một cặp thỏa mãn điều kiện.
- Không sử dụng lại một phần tử hai lần.

Đầu vào:

- Một danh sách số nguyên $nums$ với $2 \leq |nums| \leq 10^4$.
- Giá trị của các phần tử: $-10^9 \leq nums[i] \leq 10^9$.
- Một số nguyên $target$ với $-10^9 \leq target \leq 10^9$.

Đầu ra: Một danh sách chứa hai chỉ số $[i, j]$.

2. Hướng dẫn giải bài toán từng bước

Bước 1: Hiểu vấn đề

- Cần tìm hai số trong danh sách có tổng bằng $target$. - Phải xác định chỉ số thay vì giá trị của các số.

Bước 2: Chiến lược giải quyết

Sử dụng **Hash Map** để lưu trữ các giá trị đã gặp và chỉ số của chúng. Quá trình được thực hiện như sau:

1) Duyệt qua danh sách $nums$.
2) Tính phần chênh lệch $diff = target - nums[i]$.
3) Kiểm tra nếu $diff$ đã xuất hiện trong hash map:
    - Nếu có: Trả về chỉ số của $diff$ và chỉ số hiện tại i.
    - Nếu không: Lưu $nums[i]$ và chỉ số i vào hash map.

Bước 3: Pseudocode

```
1. Create an empty hash map.
2. Loop through each element in nums:
   a. Compute diff = target - nums[i].
   b. If diff exists in the hash map:
      - Return [hashmap[diff], i].
   c. If not:
      - Store nums[i] in the hash map with value i.
3. End the loop.
```

Listing 1: Pseudocode for Two Sum

Bước 4: Cài đặt Python

```python
def twoSum(nums, target):
    hashmap = {}
    for i, num in enumerate(nums):
        diff = target - num
        if diff in hashmap:
            return [hashmap[diff], i]
        hashmap[num] = i
```

Listing 2: Python Implementation of Two Sum

3. Ưu và nhược điểm

Ưu điểm

- Độ phức tạp thời gian: $O(n)$ do chỉ duyệt qua danh sách một lần.

- Dễ hiểu và cài đặt.
- Hiệu quả với dữ liệu đầu vào lớn (đến 10,000 phần tử).

Nhược điểm

- Sử dụng thêm bộ nhớ $O(n)$ để lưu hash map.
- Không kiểm tra đầu vào không hợp lệ (ví dụ, không tồn tại cặp số thỏa mãn).

4. Hướng cải thiện

- Bổ sung kiểm tra đầu vào để đảm bảo tính an toàn, ví dụ:
  - Trả về giá trị mặc định nếu không tìm thấy cặp số (dành cho ứng dụng mở rộng).
- Cải thiện bộ kiểm thử:
  - Thử nghiệm với các danh sách lớn.
  - Kiểm tra hiệu năng khi dữ liệu chứa nhiều số trùng lặp.
- Tối ưu bộ nhớ bằng cách giải quyết trực tiếp mà không cần hash map (áp dụng nếu danh sách đã sắp xếp).

5. Kết luận

Hàm `twoSum` là một giải pháp hiệu quả và phù hợp cho bài toán "Two Sum" với độ phức tạp thời gian $O(n)$. Tuy nhiên, cần mở rộng khả năng xử lý lỗi và kiểm thử thêm nhiều trường hợp đặc biệt để tăng tính ứng dụng.