

### 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.

#### Code:

```
#include <vector>
#include <unordered_map>
using namespace std;
class Solution {
public:
    vector<int> twoSum(vector<int>& nums, int target) {
        unordered_map<int, int> numMap;
        for (int i = 0; i < nums.size(); i++) {
            int complement = target - nums[i];
            if (numMap.find(complement) != numMap.end()) {
                return {numMap[complement], i};
            }
            numMap[nums[i]] = i;
        }
        return {};
    }
};
```

### 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]`.

#### Code:

```
class Solution {
public:
    int jump(vector<int>& nums) {
        int n = nums.size();
        if (n <= 1) return 0;
        int maxReach = nums[0];
        int currentMax = nums[0];
        int jumps = 1;
        for (int i = 1; i < n - 1; i++) {
            maxReach = max(maxReach, i + nums[i]);
            if (i == currentMax) {
                jumps++;
                currentMax = maxReach;
            }
        }
        return jumps;
    }
};
```