**Two Sum in O(n):**

A more efficient approach is to use a hash table (unordered\_map in C++). We can iterate through the array once, and for each element, check if the target minus the current element exists in the hash table. If it does, we have found a valid pair of numbers. If not, we add the current element to the hash table.

*#include <unordered\_map>*

*class Solution {*

*public:*

*vector<int> twoSum(vector<int>& nums, int target) {*

*vector<int> comb;*

*unordered\_map<int,int> umap;*

*for (int i = 0; i < nums.size(); i++){*

*int cur = nums[i];*

*if (umap.find(target - cur) == umap.end()){*

*umap[cur] = i;*

*}*

*else{*

*comb.push\_back(umap[target - cur]); // Pushes the index of Target-Cur into vector*

*comb.push\_back(i);*

*return comb;*

*}*

*}*

*return comb;*

*}*

*};*