1. Two Sum

Easy

9720314FavoriteShare

Given an array of integers, return indices of the two numbers such that they add up to a specific target.

You may assume that each input would have *exactly* one solution, and you may not use the *same* element twice.

Example:

Given nums = [2, 7, 11, 15], target = 9,  
  
Because nums[0] + nums[1] = 2 + 7 = 9,  
return [0, 1].

class Solution {  
public:  
 vector<int> twoSum(vector<int>& nums, int target) {  
 vector<int> results;  
 std::map<int,int> mapper;  
 for(int i=0;i<nums.size();i++){  
 if(mapper.find(nums[i])!=mapper.end()){  
 results.push\_back(i);  
 results.push\_back(mapper[nums[i]]);  
 }else{  
 mapper[target-nums[i]]=i;  
 }  
 }  
 return results;  
 }  
};

Success  
Details   
Runtime: 16 ms, faster than 57.98% of C++ online submissions for Two Sum.  
Memory Usage: 10.8 MB, less than 0.99% of C++ online submissions for Two Sum.

class Solution {

public int[] twoSum(int[] nums, int target) {

HashMap<Integer, Integer> hm = new HashMap<Integer,Integer>();

int []ret=new int[2];

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

if(hm.containsKey(nums[i])){

ret[0]=i;

ret[1]=hm.get(nums[i]);

}else hm.put(target-nums[i],i);

}

return ret;

}

}

Runtime: 3 ms, faster than 99.58% of Java online submissions for Two Sum.

Memory Usage: 39.3 MB, less than 21.27% of Java online submissions forTwo Sum.