import java.util.HashMap;

import java.util.HashSet;

import java.util.Set;

public class twoSum {

public static void main(String args[]){

int[] nums = {1,3, 2, 7, 11, 15};

int target = 9;

int[] res = twoSum3(nums, target);

for(int a : res)

System.out.print(a + " ");

}

/\*Brute Force

\* Time complexity : O(n^2) Because there are two for loop which traversal the arrary twice, so the time complexity is O(n^2).

\* Space complexity : O(1) There is no other space needed.

\* \*/

public static int[] twoSum1(int[] nums, int target) {

int [] res = new int[2];

if(nums==null || nums.length < 2)

return res;

//Traverse the array, use two pointers, one points to the current number

// another one point to the next, then judge whether their sum equals to target

// if true, return these two indices

for(int i = 0; i < nums.length - 1; i++){

for(int j = i + 1; j < nums.length; j++){

if(nums[i] + nums[j] == target){

res[0] = i;

res[1] = j;

}

}

}

return res;

}

/\*use HashMap

\* Time complexity : O(n) Only traverse the array once.

\* Space complexity : O(n) Used a hashmap to store value, so space complexity is O(n)

\* \*/

public static int[] twoSum2(int[] nums, int target){

int[] res = new int[2];

if(nums==null || nums.length < 2)

return res;

HashMap<Integer,Integer> hashMap = new HashMap<>(); // use hashMap to store the value which have been visited

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

// every time visit a number, then find whether (target-number) is in the hashMap

// if true, return these two indices

// if false, put this number into hashMap then next loop

if(!hashMap.containsKey(target - nums[i])){

hashMap.put(nums[i], i);

}else{

res[0] = hashMap.get(target - nums[i]);

res[1] = i;

break;

}

}

return res;

}

}