Question: <https://leetcode.com/problems/two-sum/>

Constraints: Assume there is exactly one solution.

Thus we can conclude that nums[x]+nums[y]=target where x, y are unique positions.

Brute Force Approach would be to check all possible pairs and check which one matches with the target. But that would result in a time complexity of O(n^2).

Now we can optimizing it by memorizing the value which we have already traversed so that we need not to re-iterate the array.  
So lets consider we are at nums[y], then target-nums[y] must be equal to nums[x] thus if we have already came across nums[x] and memorized it then we will instantly find out the both the elements, or if we haven’t yet came across nums[x] then when we will come across nums[x] then we can use same logic to find out nums[y].

For memorizing we have used HashMap, and since we need to return the position of 2 elements. So we will store nums[x] as key and x as value.  
  
Solution:

class Solution {

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

int cur\_sum=0,b[]=new int[2];

Map<Integer, Integer> map=new HashMap<>();

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

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

b[0]=map.get(target-nums[i]);

b[1]=i;

}

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

}

return b;

}

}