Two Sum Using Hashmap

n = 4
t = 9
arr = [2, 7, 11, 15]

```
Scanner sc = new Scanner(System.in);
int n = sc.nextInt();
int t = sc.nextInt();
int arr[] = new int[n];
for(int i=0; i<n; i++){
  arr[i] = sc.nextInt();
}
```

[2, 7, 11, 15], target = 9

diff = 9 - 2 = 7

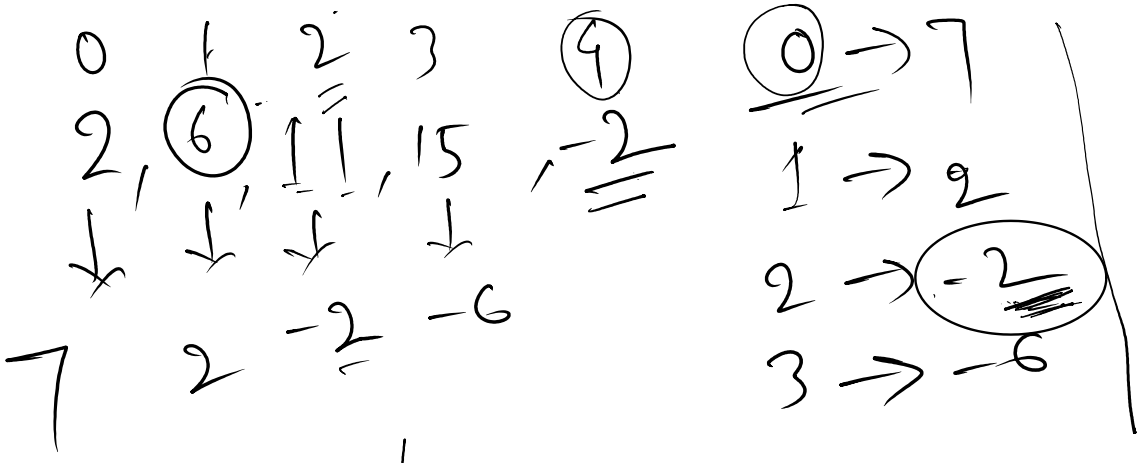
```
hm.put(7, 0);
for(int i=0; i<arr.length; i++)
  valuereq = target - arr[i];
```

// 9 - 2 = 7

7, 0 _ :)

T.O.

```
// 9-2
hm.put(valuereq, i);
if (hm.containsKey(arr[i]))
int index = hm.get(arr[i]);
S.O.Pln(i + " " + index);
 break;
 }
}
```



2, 4

11 + (-2) = 9.

```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        /* Enter your code here. Read input from STDIN. P
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int t = sc.nextInt();
        int arr[] = new int[n];
        for(int i=0;i<n;i++){
            arr[i] = sc.nextInt();
        }
        HashMap<Integer, Integer> hm = new HashMap<>();
        for(int i=0;i<n;i++){
            int valuereq = t-arr[i];

            if(hm.containsKey(arr[i])){
                int index = hm.get(arr[i]);
                System.out.println(index+" "+i);
                break;
            }
             hm.put(valuereq,i);
        }
    }
}
```

Valid Anagram

s = "anagram"
t = "nagaram"

s → hm1 → a → 3
n → 1
g → 1
r → 1
m → 1

t → hm2 → n → 1
a → 3
g → 1
r → 1
m → 1

1. It should be of same length
2. Create hashmap for first string and store its frequency.
3. Create hashmap for second string and store its frequency.
4. Compare if both hashmap are same return true, otherwise return false.

```java
public class Solution {

    public static void main(String[] args) {
        /* Enter your code here. Read input from STDIN. Print output t
        Scanner sc = new Scanner(System.in);
        String s = sc.next();
        String t = sc.next();
        HashMap<Character,Integer> shm = new HashMap<>();
        HashMap<Character,Integer> thm = new HashMap<>();
        if(s.length()!=t.length()){
            System.out.println("false");
            return;
        }
        for(int i=0;i<s.length();i++){
            if(shm.get(s.charAt(i))==null){
                shm.put(s.charAt(i),1);
            }else{
                shm.put(s.charAt(i),shm.get(s.charAt(i))+1);
            }
        }
        for(int i=0;i<t.length();i++){
            if(thm.get(t.charAt(i))==null){
                thm.put(t.charAt(i),1);
            }else{
                thm.put(t.charAt(i),thm.get(t.charAt(i))+1);
            }
        }
        if(shm.equals(thm)){
            System.out.println("true");
        }else{
            System.out.println("false");
        }

    }
}
```

Longest Substring without Repeating Characters

String s = "abcabcbb"

0 1 2 3 4 5

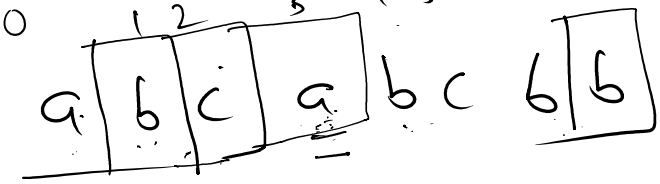
1. abc
2. bca
3. cab
4. abc

0 1 2 3 4 5 6 7

"abcabcbb"

We will solve this by using sliding window

abcabcbb



maxlength = 3

start = 0

abcb ✗

bcb ✗

cb ✓

① a → 3, b → 1, c → 2 } Hashmap

bca

```
if (hm.containsKey(s.charAt(i))) {
    start = hm.get(s.charAt(i)) + 1;
```