Code

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int t = scn.nextInt();
    HashMap<String, ArrayList<String>> map = new HashMap<>();
    for (int i = 0; i < t; i++) {
        String operation = scn.next();
        if (operation.equals("add")) {
            String empId = scn.next();
            String name = scn.next();
            String design = scn.next();
            String department = scn.next();

            ArrayList<String> arr = new ArrayList<>();
            arr.add(name);
            arr.add(design);
            arr.add(department);

            map.put(empId, arr);
        } else if (operation.equals("update")) {

            String empId = scn.next();
            String design = scn.next();

            ArrayList<String> arr = map.get(empId);
            arr.set(1, design);

            map.put(empId, arr);

        } else if (operation.equals("delete")) {

            String empId = scn.next();
            map.remove(empId);

        } else if (operation.equals("show")) {

            String empId = scn.next();

            if ( map.containsKey(empId) ) {
                ArrayList<String> arr = map.get(empId);
                for (String s : arr) {
                    System.out.print(s + " ");
                }
                System.out.println();
            } else {
                System.out.println("-1");
            }

        }
    }
}
```

⇒ Variation of hashmap (HashSet)

HashSet :- This follows all the properties of hashmap except it can contain only key

↳ repeated elements are not allowed in hashset

i/p

"abc"
"efg"
"Abc"
"efg"

set

String
abc
~~efg~~ efg
Abc

Note:-
↳ hashset is the best D.S to identify the duplicacy

Syntex

```
HashSet<DataType> set = new HashSet<>();
```

Inbuilt f^n

```
set.add( key );        // add element in set
set.remove ( key );    // remove  "  from  "
set.contains (key);    // check if present or not
set.size() / set.isEmpty()   } evergreen
```

Unique Number of Occurrences

$$arr = \begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 3, & 5, & 5, & 7, & 3, & 3, & 3 \\ \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow \end{array}$$

true

hash map

hash set

$$\left.\begin{array}{l} 3 \longrightarrow 4 \\ 5 \longrightarrow 2 \\ 7 \longrightarrow 1 \end{array}\right\} 3 \quad == \quad \left.\begin{array}{l} 4 \\ 2 \\ 1 \end{array}\right\} 3$$

code

```java
public static boolean uniqueNumberOfOccurences(int[] arr, int n) {
    HashMap<Integer, Integer> map = new HashMap<>();
    for (int i = 0; i < n; i++) {
        int curr = arr[i];
        if ( map.containsKey(curr) ) {
            int oldFreq = map.get(curr);
            map.put(curr, oldFreq + 1);
        } else {
            map.put(curr, 1);
        }
    }

    HashSet<Integer> set = new HashSet<>();
    for (Map.Entry<Integer, Integer> e : map.entrySet()) {
        int key = e.getKey();
        int val = e.getValue();

        set.add(val);
    }

    if (map.size() == set.size()) {
        return true;
    } else {
        return false;
    }
}
```

T.C = $O(n)$

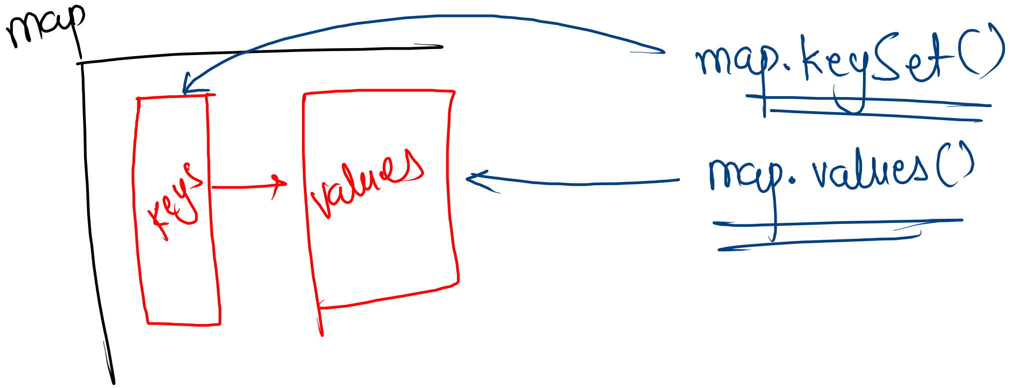
S.C = $O(n)$

n = size of array

code

```java
public static boolean uniqueNumberOfOccurences(int[] arr, int n) {
    HashMap<Integer, Integer> map = new HashMap<>();
    for (int i = 0; i < n; i++)
        map.put( arr[i], map.getOrDefault( arr[i], 0 ) + 1 );

    HashSet<Integer> set = new HashSet<>(map.values());
    return map.size() == set.size();
}
```



Two Sum 14

$arr = [\overset{0}{2}, \overset{1}{7}, \overset{2}{11}, \overset{3}{15}]$, ans = 0, 1

target = 9

$\underline{arr[i] + arr[j] == target}$

A1 ⟶ n^2

A2 ⟶ 2 pointers :- $\underline{n\log(n)}$

A3 ⟶ T.C = O(n) , S.C = O(n)

$$arr = [\overset{0}{2}, \overset{1}{7}, \overset{2}{11}, \overset{3}{15}]$$
$$\uparrow \quad \uparrow \quad \uparrow \quad \uparrow$$

$\boxed{t = 9}$

smartness

(num1) (num2)

arr[i] + arr[j] == tar

map

$$\begin{tikzcd}
2 \arrow[r] & 0 \\
7 \arrow[r] & 1 \\
11 \arrow[r] & 2 \\
15 \arrow[r] & 3
\end{tikzcd}$$

i = 0, arr[i] = 2 , $\boxed{arr[j] = 7}$ ✓

if (map.containsKey (tar - arr[i]))

i = 1, arr[i] = 7 , $\boxed{arr[j] = 2}$

i = 2, arr[i] = 11, $\boxed{arr[j] = -2}$

i = 3, arr[i] = 15, $\boxed{arr[j] = -6}$

psudo code

```
1) create a hashmap
2) traverse in array
       store each element along with
       its index
3) traverse in array
       curr = arr[i]
       check if arr[j] is present in map
       (where arr[j] = tar - arr[i])
           check if both have diff index
               print
```

Code

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int target = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }

    twoSum(arr, n, target);
}
public static void twoSum(int[] arr, int n, int target) {
    HashMap<Integer, Integer> map = new HashMap<>();
    for (int i = 0; i < n; i++) {
        map.put( arr[i], i );   <- smart move
    }

    for (int i = 0; i < n; i++) {
        int num1 = arr[i];
        int num2 = target - num1;
        if ( map.containsKey(num2) ) {
            if ( i != map.get(num2) ) {
                System.out.println( i + " " + map.get(num2) );
                return;
            }
        }
    }
}
```

T.C = $O(n)$

S.C = $O(n)$

Valid Anagram

str1 = "geekster"

str2 = "eeekgstr"

g → 1	
e → 3	
k → 1	
s → 1	
t → 1	
r → 1	