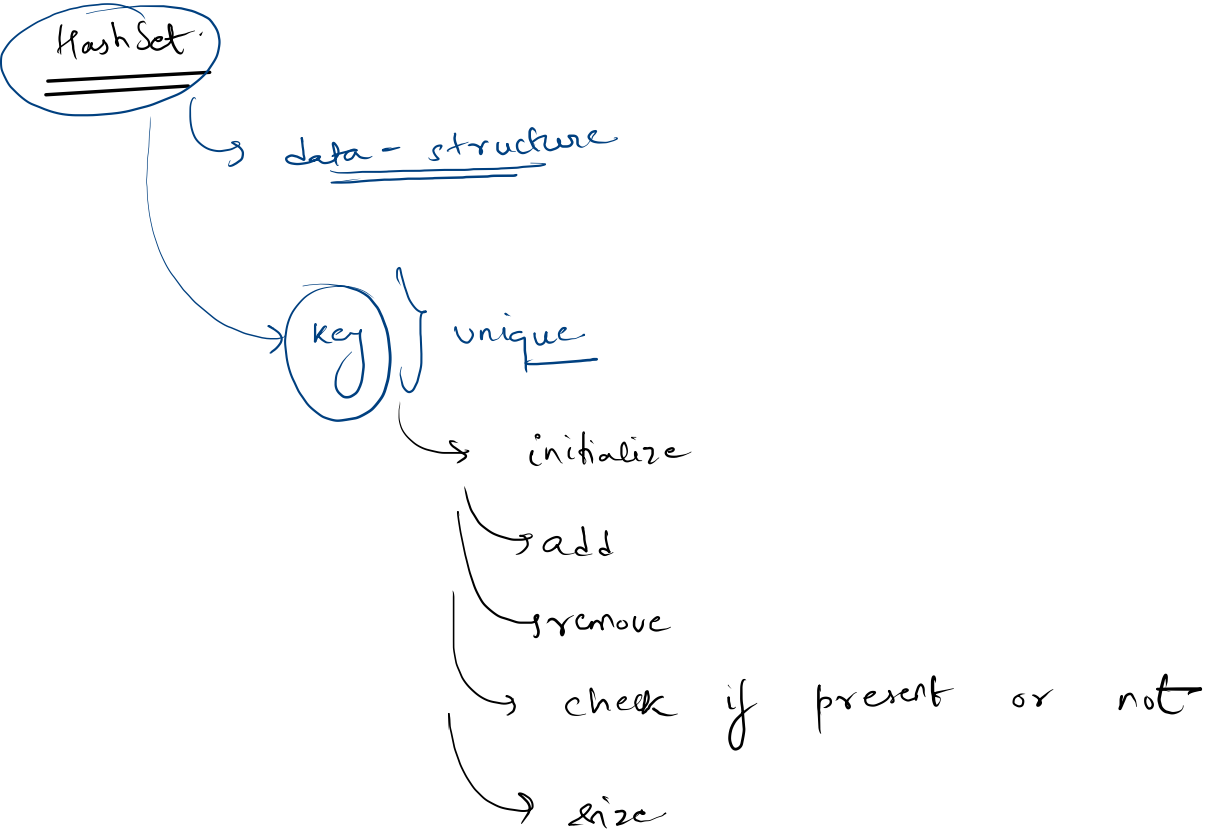

HashSet
data - structure
key
unique
initialize
add
remove
check if present or not
size

Main.java

```java
import java.util.*;
import java.util.HashSet;
public class Main
{
    public static void main(String[] args) {
        //ArrayList<Integer> arr = new ArrayList<>();
        HashSet<Integer> hs = new HashSet<>();
        hs.add(10);
        hs.add(20);
        hs.add(30);
        hs.add(10);

        hs.remove(10);
        hs.add(90);


        System.out.println(hs.size());
        System.out.println(hs);
        System.out.println(hs.contains(80));
        System.out.println(hs.contains(20));
    }
}

```

Unique Number of Occurrences

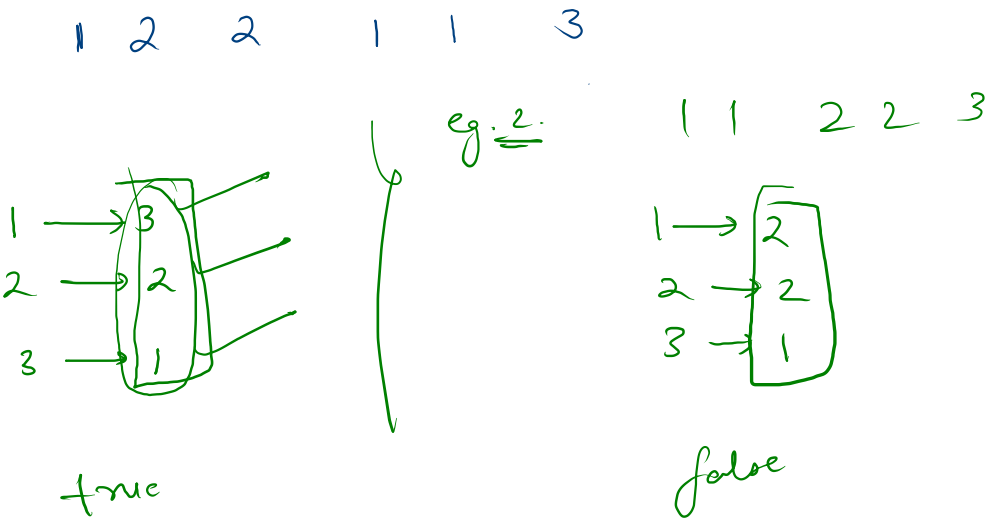
Take an array of integers **arr** as input from user and print **"true"** if the number of **occurrences** of each value in the array is unique, else print **"false"**.

Sample Input 0

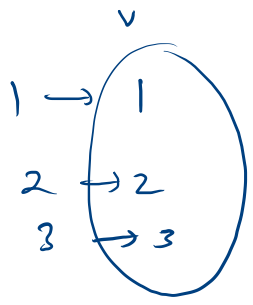
```
6
1 2 2 1 1 3
```

Sample Output 0

```
true
```



1 2 2 3 3 3

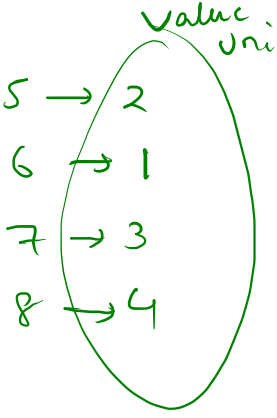


true

5 5 6 7 7 7 8 8 8 8

value unique

true.



eg.

5 7 7 7 8 8 8 8 5 6

Step 1. → freq. map. (freq)

5 → 2
7 → 3
8 → 4
6 → 1

freq.size() = 4

Step 2. → put all values in hashset.

hs.

hs.size() = 4

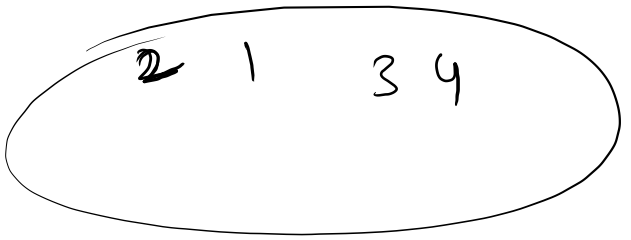
5 5 6 7 7 7 8 8 8 8 9 9

step 1 freq

5 → 2
6 → 1
7 → 3
8 → 4
9 → 2

} freq.size() → 5

step 2 put all values in hashset.



} hs.size() = 4

```java
    public static void main(String[] args) {
        //ArrayList<Integer> arr = new ArrayList<>();
        HashMap<Integer, Integer> hm = new HashMap<>();
        hm.put(10, 500);
        hm.put(20, 324);
        hm.put(30, 784);
        hm.put(40, 473);
        hm.put(50, 112);
        hm.put(70, 112);

        System.out.println(hm.keySet());

        HashSet<Integer> allVal = new HashSet<>(hm.values());


//      for(int key : hm.keySet()){
//          //System.out.println(key + " -- >  "  + hm.get(key));

//          allVal.add( hm.get(key));
//      }

        System.out.print(allVal);

```

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

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();

        HashMap<Integer, Integer> hm = new HashMap<>();     //freq map
        //Step 1: make freq map
        for(int i = 0; i < n; i++){
            int x = scn.nextInt();
            if(hm.containsKey(x)){
                int oldVal = hm.get(x);
                hm.put(x, oldVal + 1);
            }else{
                hm.put(x, 1);
            }
        }
        //Step 2: add all values to hs
        HashSet<Integer> hs = new HashSet<>(hm.values());

        //Step 3: compare result
        System.out.println(hm.size() == hs.size());

    }
}
```

Valid Anagram 5

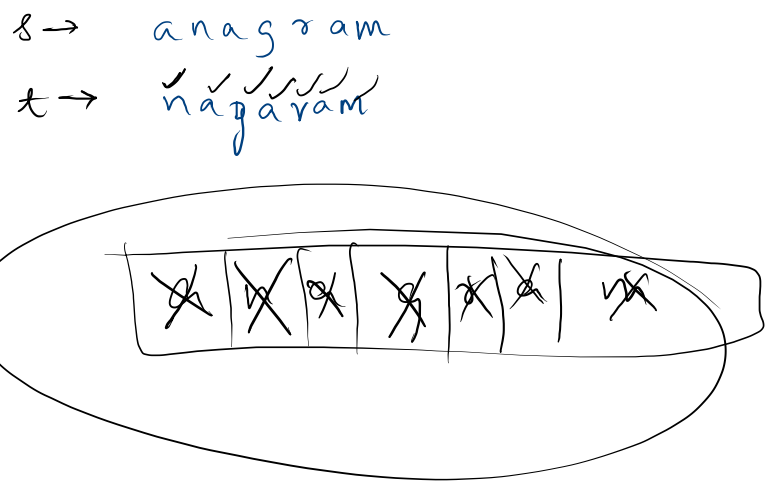
Given two strings s and t, return true if t is an anagram of s, and false otherwise. An Anagram is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.

Sample Input 0

```
anagram
nagaram
```

Sample Output 0

```
true
```



s → anagram

t → nagaram

f1.equals(f2)

true/false

f1.(s).

a → 3

n → 1

g → 1

r → 1

m → 1

f2(t)

n → 1

a → 3

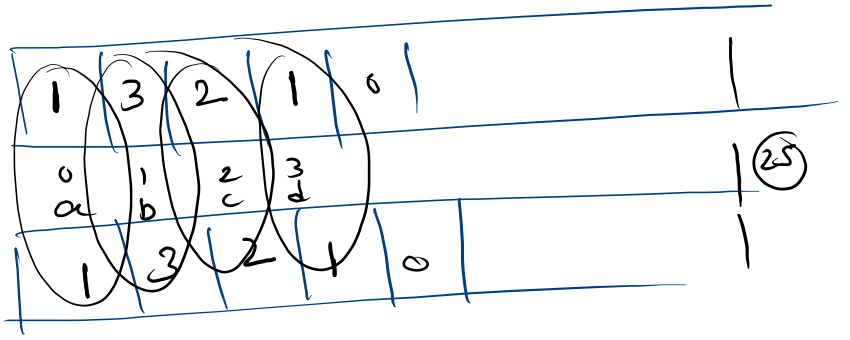
g → 1

m → 1

r → 1

a b b b c c d

c d c b a b b



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

public class Solution {
    public static HashMap<Character, Integer> getFreqMap(String x){
        HashMap<Character, Integer> hm = new HashMap<>();
        for(int i = 0; i < x.length(); i++){
            char ch = x.charAt(i);
            if(hm.containsKey(ch)){
                hm.put(ch, hm.get(ch) + 1);
            }else{
                hm.put(ch,  1);
            }
        }
        return hm;
    }
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        String s = scn.next();
        String t = scn.next();

        if(s.length() != t.length()){
            System.out.println(false);
        }else{
            HashMap<Character, Integer> f1 = getFreqMap(s);
            HashMap<Character, Integer> f2 = getFreqMap(t);
            System.out.println(f1.equals(f2));
        }
    }
}
```

s → anagram

t → nagaram