Lecture 48 | Hash Map

- `Hash-Map` : A data structure that stores <key, value> pairs.
- eg : `"Milind" : 4356, "Raj" : 1234, "Ravi" : 3456` (key-value pairs)
- Example problem : To find index of the key in array, arr = [1,2,3,4,5,6,7,8,9,10], key = 5 => index = 4 (index starts from 0)
- Lets say we need to find m elements in the array arr = [1,2,3,4,5,5,5,5,5,5] and key = 5, then we need to find the m elements in the array. => [5,5,5,5,5,5]
- Hashmap is an optimsation over normal traversal of array.
- Traversal of array is O(n) and hashmap/hashset is O(1)
- Searching in hashmap is O(1) (constant time) and insertion is O(1) (constant time) faster than array traversal.

Hashmap Operations

Operation	Time Complexity
Searching (value get...)	O(1) or O(Lambda)
Insertion (value put...)	O(1)
Check if present or not	O(1)

- Lambda : Hashing Constant

Hashmap / Hashtable (A deep dive into the topic)

- Hashmap is a data structure that stores key-value pairs. Lets take an example of a hashmap.

Country (key)	Population (value)
"India"	428
"China"	603
"USA"	400

- Hashmap combinations : <String, Integer>; <String, String>, <Integer, String>, <Integer, Integer>, <String, Double> ...
- Key is supposed to be unique.

Hashmap Operations in Java

`hm.put(key, value)` : Inserts the key-value pair into the hashmap. In O(1) time

> hm.put("India", 428); hm.put("China", 603); hm.put("USA", 400);

key	value

key	value
India	428
China	603
USA	400

> hm.put("India", 200); hm.put("USA", 28); hm.put("Dubai", 530);

key	value
India	200 (gets updated)
China	603
USA	28 (gets updated)
Dubai	530 (gets added)

- Important Points
  - When same key gets inserted, value gets updated.

`hm.get(key)` : Returns the value of the key. In O(1) time

> hm.get("India"); // Returns 200 hm.get("China"); // Returns 603 hm.get("USA"); // Returns 28 hm.get("Dubai"); // Returns 530 hm.get("Canada"); // Returns null

- Important Points
  - If key is not present, returns null.

`hm.containsKey(key)` : Returns true if key is present in the hashmap. In O(1) time

> hm.containsKey("India"); // Returns true hm.containsKey("China"); // Returns true hm.containsKey("USA"); // Returns true hm.containsKey("Dubai"); // Returns true hm.containsKey("Canada"); // Returns false

`hm.keySet()` : Returns the set of keys in the hashmap. In O(1) time

> hm.keySet(); // Returns Set {India, China, USA, Dubai}

Array vs Hashmap

- factorial storage

key	value
0	1
1	1

key	value
2	2
3	6
4	24
5	120

- to get the value of nth factorial, we need to store all the values in an array.

a[] = [1, 1, 2, 6, 24, 120]

- to get value of fact(5) = 120 can get O(1) time by using array and O(lambda) ~ O(1) time by using hashmap

Practical Example

```
import java.util.*;
// import java.util.HashMap;

public class Main {
    public static void main(String[] args) {
        // declare and initialise hashmap
        // HashMap<keyDataType, valueDataType> hashMapName = new
HashMap<keyDataType,
        // valueDataType>();
        HashMap<String, Integer> hm = new HashMap<String, Integer>();

        // add elements to hashmap
        hm.put("India", 628);
        hm.put("China", 837);
        hm.put("Dubai", 120);

        // print hashmap
        System.out.println(hm); // might not get in order due to hashmap
    }
}
```

> Output :

```
> java Main.java
{China=837, Dubai=120, India=628}
```

> On Updation :

```
import java.util.*;
// import java.util.HashMap;
```

```
public class Main {
    public static void main(String[] args) {
        // declare and initialise hashmap
        // HashMap<keyDataType, valueDataType> hashMapName = new
HashMap<keyDataType,
        // valueDataType>();
        HashMap<String, Integer> hm = new HashMap<String, Integer>();

        // add elements to hashmap
        hm.put("India", 628);
        hm.put("China", 837);
        hm.put("Dubai", 120);
        hm.put("India", 200);
        hm.put("Pak", 837);
        hm.put("USA", 443);

        // print hashmap
        System.out.println(hm); // might not get in order due to hashmap
    }
}
```

> Output :

```
> java Main.java
{USA=443, China=837, Pak=837, Dubai=120, India=200}
```

`hm.get(key)` : Returns the value of the key

```
// get value from hashmap
System.out.println(hm.get("India")); // Returns 200
```

`hm.containsKey(key)` : Returns true if key is present in the hashmap

```
// containsKey() method returns true if key is present in hashmap
boolean isChinaPresent = hm.containsKey("China"); // returns true or false
depending on key's presence
boolean isBangladeshPresent = hm.containsKey("Bangladesh"); // returns
false
System.out.println(isChinaPresent); // true
System.out.println(isBangladeshPresent); // false
```

`hm.keySet()` : Returns the set of keys in the hashmap

```
// keySet() method returns set of keys
for (String key : hm.keySet()) {
    System.out.println(key);
}
```

`hm.size()` : Returns the number of key-value pairs in the hashmap

```
// size
System.out.println(hm.size()); // 5 : number of elements/entry in hashmap
```

[PROBLEM 1] Highest Frequency Character

`Easy`

> 1. You are given a string str.
> 2. You are required to find the character with maximum frequency.

Constraints

> 0 < str.length() <= 100 There will be a single character with highest frequency

Format

Input

> A string str

Output

> The character with highest frequency

Example

Sample Input

> zmszeqxllzvheqwrofgcuntypejcxovtaqbnqyqlmrwitc

Sample Output

> q

Solution

```
import java.util.*;

public class HighFreqChar {
    public static void main(String[] args) {
        // to get maximum frequency character using hashmap
        Scanner sc = new Scanner(System.in);
        String s = sc.nextLine();
        HashMap<Character, Integer> hm = new HashMap<Character, Integer>
();
        for (int i = 0; i < s.length(); i++) {
            char ch = s.charAt(i);
            if (hm.containsKey(ch)) {
                int val = hm.get(ch);
                hm.put(ch, val + 1); // repeat character
            } else {
                hm.put(ch, 1); // first time seeing this character
            }
            // hm.put(ch, hm.getOrDefault(ch, 0) + 1);
        }

        char maxFreqChar = s.charAt(0);
        for (Character key : hm.keySet()) {
            if (hm.get(key) > hm.get(maxFreqChar)) {
                maxFreqChar = key; // update maxFreqChar if current key is
greater than maxFreqChar
            }
        }
        System.out.println(maxFreqChar);
        sc.close();
    }
}
```

> Output :

```
> java HighFreqChar.java
aaaabbb
a
```

Dry Run

Character	Integer
a	4
b	3

> So, we have 4 a's and 3 b's. Output is a because it has highest frequency.

- A neat way to handle if else condition

  > hm.put(ch, hm.getOrDefault(ch, 0) + 1);

[PROBLEM 2] Get Common Elements - 1

`Easy`

> 1. You are given a number n1, representing the size of array a1.
> 2. You are given n1 numbers, representing elements of array a1.
> 3. You are given a number n2, representing the size of array a2.
> 4. You are given n2 numbers, representing elements of array a2.
> 5. You are required to print all elements of a2 which are also present in a1 (in order of their occurence in a2). Make sure to not print duplicates (a2 may have same value present many times).

Constraints

> 1 <= n1, n2 <= 100 0 <= a1[i], a2[i] < 10

`Time complexity should be O(n)`

Format

Input

> A number n1 n1 number of elements line separated A number n2 n2 number of elements line separated

Output

> All relevant elements of a2 in separate lines (no duplicacy)

Example

> Sample Input

```
9
5
5
9
8
5
5
8
0
3
18
9
7
1
0
3
6
5
9
1
```

```
1
8
0
2
4
2
9
1
5
```

Sample Output

```
9
0
3
5
8
```

Solution

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

public class GetCommonElements {

    public static void main(String[] args) throws Exception {
        Scanner sc = new Scanner(System.in);
        int n1 = sc.nextInt();
        int a1[] = new int[n1];
        for (int i = 0; i < n1; i++) {
            a1[i] = sc.nextInt();
        }
        int n2 = sc.nextInt();
        int a2[] = new int[n2];
        for (int i = 0; i < n2; i++) {
            a2[i] = sc.nextInt();
        }
        // frequency map of a1[]
        HashMap<Integer, Integer> hm = new HashMap<Integer, Integer>();
        for (int ele : a1) {
            hm.put(ele, hm.getOrDefault(ele, 0) + 1); // it will add the
element if it is not present in the map
        }
        // frequency map of a2[]
        for (int ele : a2) {
            if (hm.containsKey(ele)) {
                System.out.println(ele);
                // remove
                hm.remove(ele); // it will remove the element if it is
present in the map
            }
        }
        sc.close();
    }
}
```

[PROBLEM 3] Get Common Elements - 2

`Easy`

> 1. You are given a number n1, representing the size of array a1.
> 2. You are given n1 numbers, representing elements of array a1.
> 3. You are given a number n2, representing the size of array a2.
> 4. You are given n2 numbers, representing elements of array a2.
> 5. You are required to find the intersection of a1 and a2. To get an idea check the example below:

- if a1 -> 1 1 2 2 2 3 5
- and a2 -> 1 1 1 2 2 4 5
- intersection is -> 1 1 2 2 5

`Note -> Don't assume the arrays to be sorted. Check out the question video.`

Constraints

- 1 <= n1, n2 <= 100
- 0 <= a1[i], a2[i] < 10 `Time complexity should be O(n)`

Format

Input

- A number n1
- n1 number of elements line separated
- A number n2
- n2 number of elements line separated

Output

- All relevant elements of intersection in separate lines
- The elements of intersection should be printed in order of their occurence in a2.

Example

Sample Input

```
7
1
1
```

```
2
2
2
3
5
7
1
1
1
2
2
4
5
```

Sample Output

```
1
1
2
2
5
```

Solution

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

public class GetCommonElement2 {

    public static void main(String[] args) throws Exception {
        Scanner scn = new Scanner(System.in);
        int n1 = scn.nextInt();
        int[] a = new int[n1];
        for (int i = 0; i < n1; i++)
            a[i] = scn.nextInt();

        int n2 = scn.nextInt();
        int[] b = new int[n2];
        for (int i = 0; i < n2; i++)
            b[i] = scn.nextInt();

        // FreqMap of A array
        HashMap<Integer, Integer> hm = new HashMap<>();
        for (int ele : a)
            hm.put(ele, hm.getOrDefault(ele, 0) + 1);

        for (int ele : b) {
            if (hm.containsKey(ele) && hm.get(ele) > 0) {
                System.out.println(ele);
                int oldFreq = hm.get(ele);
                int newFreq = oldFreq - 1;
                hm.put(ele, newFreq);
            }
        }

    }

}
```