DAY 123

Hashmap and Heap Started

Date → 27 Jan
Day → Thursday

Date
Page No.

HashMap and Heaps

Hashmap

1. → HashMap<Key, Value> is a part of Java Collection
2. This class is found in Java.util package.
3. A hashmap is a data structure which has an amazing property that most of the operations which we perform on it are done in O(1) time complexity.
   The data is stored in a hash map in the form of key value pairs.

eg1)
for eg → Population Map

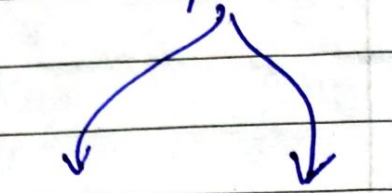
→ Population of various countries

Country	Population (in millions)
India	1391
China	1398
USA	329
Indonesia	268

Key → Country
Value → Population

Datatype of Key → String
Datatype of Value → Integer

अगर हम data Hashmap में store करवा रहे हैं तो data will be stored in the form of Key-value pair मतलब हमारा data pairs की form में store होगा।



1 Key और 1 value को मिलाके एक Pair बनता है, और उसे हम बोलते हैं Key-Value Pair.

4. Hashmap is a Map. Map is a key-value mapping, which means every key is mapped to exactly 1 value and we can use the key to retrieve the corresponding value from a map.

5. The advantage of a HashMap is that time complexity to insert and retrieve a value is O(1) on average.

eg.2) College Canteen Menu

Day	Item
Mon	Samosa, Parantha
Tue	Pizza, pasta
Wed	Veg Roll, Pizza
Thu	Pasta, Chilli Potato
Fri	Chowmein, Samosa.
Sat	Pasta, Pizza.
Sun	Parantha, Samosa.

Day → Key
Menu → Value

Datatype of Key → String.
Datatype of value → array of Strings.

Eg 3.

Item	Price (₹)
Samosa	10
Paratha	25.
Chowmein	40
Veg Roll	45.
Pizza	45.
Pasta	45
Chilli Potato	50.

Key → Item
Value → Price.

Datatype of Key → String
Datatype of value → Intiger.

1. Create a HashMap

It is compulsory to not use int here. हमेशा पहले capital I वाला Integer ही लिख सकते हैं

Date / /
Page No.

```
HashMap<String, Integer> hm = new HashMap<>();
```

- Key → datatype है String
- Value → datatype है Integer.
- name of the HashMap
- like any other data structure, Hashmap is also created using new keyword

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

{ }

↳ This is what an empty hashmap looks like. So, let's now try to insert some values into it and then display it.

2. Put into Hashmap

→ put(Key, Value) function can be used to put some values into the Hashmap.

Date / Page No.

The time complexity of Put function is → O(1).

There are 2 possible cases while we are using the put(Key, value) function in a hashmap.

a) When the Key is not present

↓

If the Key is not present in the HashMap, it will get inserted.

b) The Key is already present

→ If the key is already present, we cannot insert the same Key again
(यानी हमारे पास duplicate key होगी)
(दो duplicate keys नहीं होनी चाहिए)

→ Only the value of the existing key will get updated in the Hashmap.

put → add. (अगर key पहले से present नहीं है तो key add करेगा)

put → update (अगर key पहले से present है तो उसकी value को update करेगा)

तो put function हमारे लिए key add का भी काम करसकता है और key update का भी।

→ hashmap

```
hm.put("India", 1391);
hm.put("China", 1398);
hm.put("USA", 329);
hm.put("Indonesia", 268);

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

{USA: 329, China: 1398, India: 1391, Indonesia: 268}

We have again tried to insert the same key "Indonesia" which is already present

```
hm.put("Indonesia", 270);

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

{USA: 329, China: 1398, India: 1391, Indonesia: 270}

Value updated

* We must observe that the Hashmap has changed the order in which the elements occur.
So, this is something that we can not control in HashMap.
We can not control the order of occurence of elements in a Hashmap.

The time complexity of get() function is O(1).

Date / / Page No.

3. Get a value in HashMap

The get(key) function in a hashmap is used to get the value corresponding to a particular key in the hashmap.

There can be 2 possible cases for this :-

a) If the key exists

If the key exists, then we will get the value of that key by using the get() function.

b) If the key doesn't exist.

If the key does not exist, then the get(key) function returns null.

* `System.out.println(hm.get("India"));` → 1391.

"India" → key

India key already exist in the hm hashmap with value = 1391.

तो हमने India की value को get करलिया

* `System.out.println(hm.get("Utopia"));` → null

The key "Utopia" does not exist. So null value will be returned.

We should get the value of a key only in the Integer

↳ Capital I Integer datatype can store null as

```
Integer x = hm.get("Utopia");
```

We can store null here

Returns null.

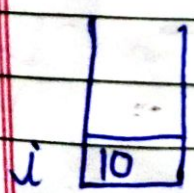
```
int x = hm.get("Utopia");
```

int datatype cannot store null. so we will get a null pointer exception.

return null

We cannot store null in variable of int datatype because int is a primitive datatype

int i=10;



Primitive Datatype Stack पर ही store होते हैं.

There are total 8 primitive datatype

- byte
- short
- int
- long
- float
- double
- char
- boolean

Primitive datatype की Wrapper Class भी होती है

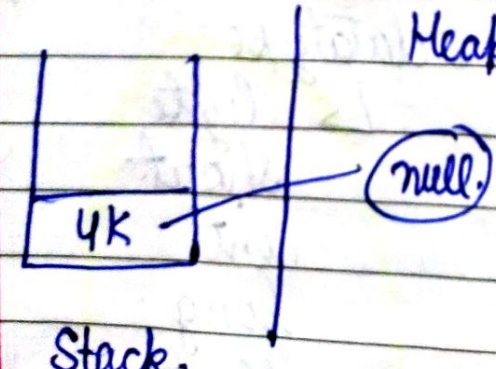
Primitive type	Wrapper Class
byte	Byte
int	Integer
short	Short
long	Long
float	Float
double	Double
boolean	Boolean
char	Character

Primitive datatype के variable stack में store होते हैं

Wrapper Class के objects Heap में store होते हैं।

int i = null ✗ } Primitive datatype में हम null store नहीं करा सकते क्योंकि stack पर जितना भी variable और stack में हम null नहीं डाल सकते।

Integer i = null } Wrapper Class के objects में हम null डाल सकते हैं। Heap में हम null store कर सकते हैं।



get → If key is not present → null
If key is present → value.

Date / /
Page No.

7. Contains Key in Hashmap

↳ The time complexity of ContainsKey() function is O(1).

The ContainsKey() is a Boolean function.

We pass a particular key value as a parameter to this function. If the hashmap contains key, it returns true else returns false.

```
System.out.println(hm.containsKey("China"));
```

true. } We get true for the key "China" as it exists in the hashmap.

```
System.out.println(hm.containsKey("Pakistan");
```

false. } We get false for the key "Pakistan" as the Hashmap does not consist the key.

5. KeySet in HashMap.

We know that we have key-value pairs in Hashmap. We can get only the keys in a hashmap by using the keySet() function.

The keySet() function returns the set of all the keys in a Hashmap.
We can store it in a Set container

To get the Set of keys, we can store it in a Set Container.

```
Set<String> keys = hm.keySet();
```

We have stored all the keys in this Set.

```
System.out.println(keys);
[USA, China, India, Indonesia]
```

Keys (order हमेशा unpredictable और changed होती है)
keys की

Set एक ऐसा collection होता है जिसमे हमेशा unique values होगी। जैसे Hashmap में keys हमेशा unique होती है।

Implementation of KeySet

Date / /
Page No.

```
import java.util.*;
public class Main
{
  public static void main (String [] args)
  {
    HashMap<String, Integer> hm = new HashMap<>();
    hm.put("India", 1391);
    hm.put("China", 1398);
    hm.put("USA", 329);
    hm.put("Indonesia", 268);

    Set<String> key = hm.keySet();

    for(String key : keys)
    {
      Integer val = hm.get(key);
      System.out.print (key + ":" + val);
    }
  }
}
```

Output → USA : 329 China : 1398 India : 1391
Indonesia : 268

∘∘ We have seen the following functions.

→ hm.put("India", 240);
→ hm.get("India");
→ hm.containsKey("India");
→ hm.keySet();

All these functions are taking O(1) time complexity

तो हमने कुछ basic & most commonly used functions in HashMap अभी देखे

इन सभी functions की time complexity है O(1) तो इसीलिए HashMap हमारे लिए बहुत important है because it can help us to reduce the Time Complexity and improve the performance of our code.

1. Highest Frequency Character

Date / /
Page No.

हमें एक string given होगी। हमें उस string में से maximum occurring character को print कराना है।

↳ जो character सबसे ज्यादा occur करता है string में, हमें वो character print कराना होगा।

हम 1 frequency map बनाएँगे।

↳ हर character की frequency store होगी

↳ Key = character

↳ value = frequency.

String → a b b c b a b c d d a b b d a b b b.

a	4
b	9
c	2
d	3

→ क्योंकि b की frequency सबसे ज्यादा है तो b is the Highest Frequency character

Here, the frequency of 'a' is maximum

eg-1) abracadabra.

हमें 1 String दी गयी है।

और हमे वो character print करना है जिसकी frequency सबसे ज्यादा है।

The frequency of 'a' is maximum.
Hence 'a' is printed.

eg-2) abcccdbabccccd

हम String के हर character पर iterate करेंगे और उस character की frequency बढ़ाते चलेंगे

We will make a HashMap table (Key-Pair) for this string.

String: Character	Integer: Frequency
a.	✗2
b.	✗✗3
c	✗✗✗✗✗6
d.	✗2

Now, we can clearly see the frequency of 'c' is maximum so, 'c' is the Highest Frequency character.

∴ We iterate through every character of the String and then increase the count against that character.

Approach

1. We create a new HashMap of Character vs Integer named hm.
   - Key का datatype है "Character"
   - value का datatype है "Integer"
   - name of the Hashmap.

2. We iterate through all the character of the String and check if the hashmap contains the given character.
   If it does, then, we get the frequency of that character and store in the old variable.

   हम String के सारे characters पर iterate करेंगे और check करेंगे, क्या hashmap में ये character पहले से exist करते हैं?
   अगर hashmap में ये character पहले से exist करते हैं तो हम old variable में इस character की frequency store करेंगे
   - old variable का नाम है जिसमें character की existing frequency हम store करेंगे

   Then, we increase it by 1 and store in the new. The 'new' gets inserted in hashmap against character
   - new variable में old+1 store करेंगे

   And then new gets inserted in the hashmap against the character.
   - new ↓ value of new variable

3. Else if the hashmap does not contain that character. then we add it to hashmap with its frequency 1.

4. We initialize a variable "max" with the first character of the given string where "max" is the character with highest frequency.

```
char max = str.charAt(0);
```

max में वो character store होगा जिसकी frequency सबसे ज्यादा है।

Initialized with first character of given string.

5. We apply the for loop for every character of the Hashmap.

If the frequency of current character is greater than the frequency of "max", then that character gets stored in max.

इस hashmap की सभी keys पर loop लगाते हैं। i.e (हर character पर loop लगेगा)
और अगर किसी character की frequency ज्यादा होगी तो वो character होगा
Max

6. After comparing the frequency of all the characters, we print the 'max'


```
import java.util.*;
public class Main
{
  public static void main (String[] args)
  {
    Scanner scn = new Scanner(System.in);
    String str = scn.next();
    HashMap<Character, Integer> hm = new HashMap<>();
    for(int i=0; i<str.length(); i++)
    {
      char ch = str.charAt(i);
      if (hm.containsKey(ch))
      {
        int old = hm.get(ch);
        int new = old+1;
        hm.put(ch, new);
      }
      else{
        hm.put(ch, 1);
      }
    }
    char max = str.charAt(0);
    for(Character key : hm.keySet())
    { if(hm.get(key) > hm.get(max))
        { max=key;}
    }
    System.out.println(max);
}
```

हमने एक नया hashmap बनाया hm नाम का

String के सभी characters पर loop लगाया

अगर string में वो character पहले से है तो उसकी value update करो (old में ++ किया)

अगर वो character hm में exist नहीं करता तो एक new put करेगा उसको character के लिए, with value 1 और उसे hm hashmap में डालदो

keySet() से सभी keys को receive करेगा

max मतलब maximum freq वाला character

हमने पूरे keys के set पर loop लगाया।

अगर key की value max की value से ज्यादा है तो max में वो key डालदो।

last में maximum freq वाले character को print करदो।

Time Complexity of Highest Frequency Character function is O(n)

↳ क्योंकि 2 for loop लगे हैं → O(n)+O(n)

→ O(n)

2. Get Common Elements - 1

Date / /
Page No.

हमें 2 array a1, a2 given होंगी

a1	1	1	2	2	2	3	5
a2	1	1	1	2	2	4	5

We are required to print all the elements of a2 which are also present in a1 (in order of their occurence in a2).

(हमें एक element 1 बार ही print करवाना है).
Make sure to not print duplicates.

So the output for a_1 & a_2 → 1 2 5.

तो पहले हम Array 1 के सभी elements के लिए एक HashMap (frequency map) बनाएँगे

a1	1 ✓	1 ✓	2 ✓	2 ✓	2 ✓	3 ✓	5 ✓
a2	1	1	1	2	2	4	5

Key	value
1	~~1~~ 2
2	~~1~~ ~~2~~ 3
3	1
5	1

→ अब हम array 2 पर traverse करेंगे, हर element को check करेंगे और अगर वो element 1st frequency map में present होगा तो हम उस element को print करेंगे

↳ Because it is present in freq map of array 1 & array 2.

So basically it is present in both the arrays.
So it is a common element.

→ Now, we will remove that element from the frequency map of array 1. if its present in map & array 2

a2 1 1 1 2 2 4 5.

हम array 2 पर travel करेंगे हमें 1 मिला first element 1 is also present in the hashmap.

Integer	Frequency
~~1~~	~~1~~ 2
2	~~1~~ ~~2~~ 3
3	1
5	1

Output : 1

a2 1 1 1 2 2 4 5.

Integer	frequency
~~2~~	~~1~~ ~~2~~ 3
3	1
5	1

Output : 1 2.

Page No.

a3 1 1 1 2 2 4 5.

Integer	frequency
3	1
5	1

output : 1 2 5.

```
import java.util.*;
public class Main{

public static void main (String [] args)
  {
  Scanner scn = new Scanner (System.in);
  int n1 = scn.nextInt();
  int [] arr1 = new int [n1];
  for (int i = 0; i < arr1.length ; i++)
  {
  arr1[i] = scn.nextInt();
  }
  int n2 = scn.nextInt();
  int [] arr2 = new int [n2];
  for (int i = 0; i < arr2.length ; i++)
  {
  arr2 [i] = scn.nextInt();
  }
HashMap < Integer , Integer > fmap = new HashMap <>();
for (int val : arr1)
{
  if (fmap.containsKey (val) == (false))
    {
    fmap.put (val, 1);
    }
```

Hash map बनाया

array 1 के लिए frequency map बनाया

```
else {
    int oldfreq = fmap.get(val);
    int newfreq = oldfreq + 1;
    fmap.put(val, nf);
  }
}
for(int val : arr2)
  { if(fmap.containsKey(val)==true)
    {
      System.out.println(val);
      fmap.remove(val);
    }
  }
}
}
```

अगर val पहले से fmap में है तो old frequency +1 करदो value और new frequency होगी freq map में update.

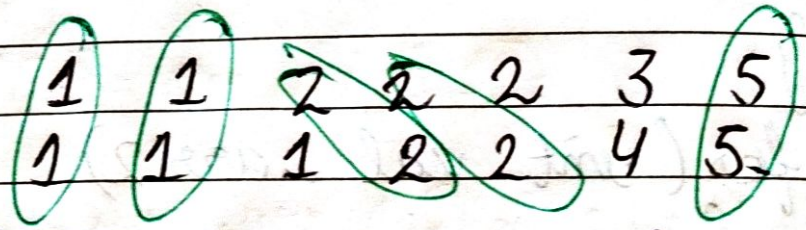
अगर array2 की element(val) इसी freq map में मिलगया तो उस value (element) को print करदो और val को fmap से remove करदो

(3). GET COMMON ELEMENTS -2.

↳ इसमे correct integration हमे print करनी होगी।

a1	1	1	2	2	2	3	5
a2.	1	1	1	2	2	4	5

output → 1 1 2 2 5.



→ 1) पहले array 1 का frequency map बनाओ

character	frequency
1	✗ 2
2	✗ ✗ 3
3	1
5.	1.

हमने Hashmap बनाया array 1 के लिए
→ ये array 1. का frequency map है।

→ 2) अब हमे array 2 पर traverse करना होगा।

और अगर array 2 का कोई element freqMap में मिले और उस key की value > 0 है तो उसे print करदो और उस key की value मे से 1 minus करदो।

Date / /
Page No. 1

array 2 ✓ ✓ ✓ ✓ ✓ ✓ ✓
1 1 1 2 2 4 5

character	frequency
1	~~2~~ 1 0
2	~~3~~ 2 1
3	1
5	~~1~~ 0

output ✓.
1 1 2 2 5.

* Approach

1. We make a frequency table (a Hashmap) for elements of array 1.

char	freq.
1	~~2~~ 1
2	~~3~~ 2 1
3	1
5	~~1~~ 0

2. Then we iterate through every element of array 2 and if that element is present in our Hashmap, then it is printed & the value corresponding to that element (key) is reduced by 1 in Hashmap. (उस key की value में से 1 minus हो जाता है).
If the element is not present in the Hashmap then we do nothing.

char	freq.
1	~~2~~ 1 0
2	~~3~~ ~~2~~ 1
3	1
5	~~1~~ 0

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

public class Main
{
    public static void main(String args)
    {
        Scanner scn = new Scanner(System.in);

        int n1 = scn.nextInt();
        int arr1[] = new int[n1];
        for(int i=0; i<arr1.length; i++)
        {
            arr1[i] = scn.nextInt();
        }

        int n2 = scn.nextInt();
        int arr2[] = new int[n2];
        for(int i=0; i<arr2.length; i++)
        {
            arr2[i] = scn.nextInt();
        }
```

```
HashMap<Integer, Integer> fmap = new HashMap<>();
for(int val : arr1)
{
  if(fmap.containsKey(val)==false)
  {
    fmap.put(val, 1);
  }
  else
  {
    int oldfreq = fmap.get(val);
    int newfreq = oldfreq + 1;
    fmap.put(val, newfreq);
  }
}
for(int val : arr2)
{
  if(fmap.containsKey(val)==true
     && fmap.get(val) > 0
  )
  {
    System.out.println(val);
    int oldfreq = fmap.get(val);
    int newfreq = oldfreq - 1;
    fmap.put(val, newfreq);
  }
}
```

Array 2 के सभी elements पर iterate करेंगे

अगर array2 का element freq map में है और उस key की value 0 से ज्यादा है तो उस element को print करेंगे और map में उसकी value -1 करेंगे

(4) Longest consecutive Subsequence

We are given an array

eg → | 10 | 5 | 9 | 1 | 11 | 8 | 6 | 15 | 3 | 12 | 2 |

↳ The following consecutive sequences can be formed from the given array.

1 2 3

5 6

8 9 10 11 12

15.

The longest sequence here is of 5 elements.

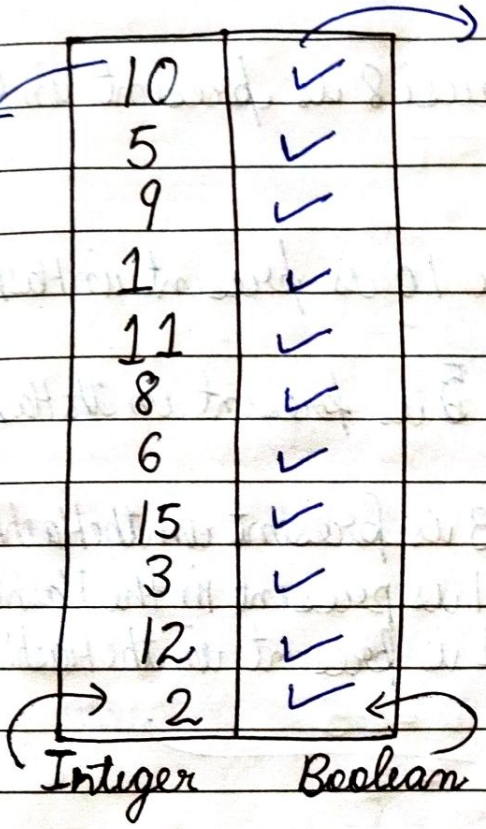
8 - 9 - 10 - 11 - 12.

Approach

Date / / Page No.

Step 1) हम एक hashmap बनाते हैं (Integer v/s Boolean) for all the elements of the array and store "true" against them all initially

10 5 9 1 11 8 6 15 3 12 2



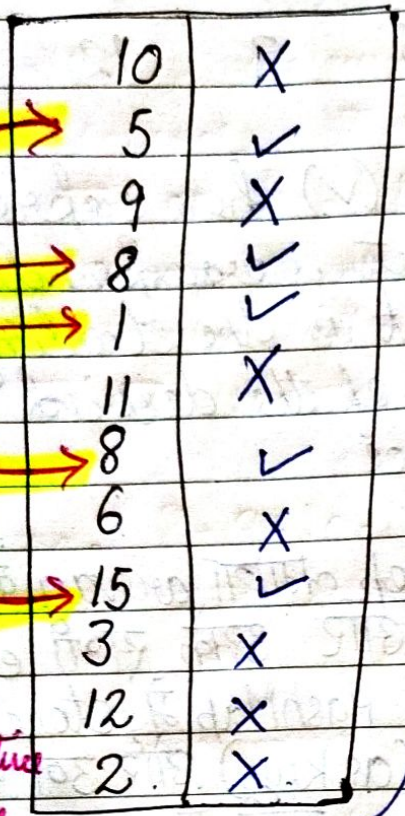
The true (✓) here represents that its corresponding element is the starting element of the desired sequence.

एक loop लगाया array के ऊपर और उसके सभी elements को इस HashMap में store कराया (as keys) और उन सब की value डाली True.

Step 2) अब हम पूरे array पर iterate करेंगे loop लगाके & we will check whether a number 1 less than that element is present in the HashMap or not. If the element is not present, then we do nothing

Eg → अगर हम एक element पर हैं → let say → 10. फिर हम check करते हैं कि क्या 10 − 1 i.e 9 is present in the HashMap or not. अगर हमें previous element (i.e 9) अगर मिलगया HashMap में, तो इसका मतलब 10 can not be the starting element of the consecutive sequence. (9 अगर Hashmap में है तो 9 या 9 से कम कोई element ही इस sequence का starting element हो सकता है)

अगर 9 HashMap में हमे नही मिलता, तो 10 ही होता Sequence का starting element, then the current element (10) would have been left as true.



10	X
5	✓
9	X
8	✓
1	✓
11	X
8	✓
6	X
15	✓
3	X
12	X
2	X

Now, these elements which are true are the starting index of desired consecutive sequence.

Because 9 is present in HashMap

Because 8 is present in HashMap

Because 10 is present in HashMap

Because 5 is present in the HashMap

Because 3 is present in the HashMap.
Because 11 is present in the HashMap
Because 1 is present in the HashMap.

After iterating on all the elements of array

Now, the values of keys have been updated in our HashMap.

Step 3) for every true element, we find the consecutive elements in the hashmap.

We will have 2 variables → msp and mlen

msp → max starting point

mlen → max length

for the starting point of sequence of maximum length

for the length of sequence of maximum length

5 → is the same key as its value in the Hashmap.

↳ for 5, we find all the consecutive elements in the hashmap.

5-6 } The consecutive sequence for 5 becomes 5-6.

↳ The size of this sequence is 2.

mlen = 2. } maximum length ⇒ 2

msp = 5. } maximum starting point → 5

Now, the next sequence with '1' as the starting element is 1-2-3.

↓

length of this sequence is greater than mlen ↳ 2.

So, we will update mlen & msp.

↓ 3 ↓ 1.

So the maximum length (mlen) = 3.
maximum starting point = 1.

This goes on until we find we find the maximum length out of all sequences.

(जब तक हमे सभी sequences में से सबसे maximum length वाला sequence न मिल जाए, तब तक ये loop चलेगा और maximum length (mlen) और maximum starting point (msp) को हम update करते जाएँगे)

So, 8-9-10-11-12

this is our maximum sequence.

So the value of mlen = 5
msp = 8

Step 4) Now, we are having our value of

mlen (maximum length)
msp (maximum starting point).

→ तो msp(8) से 5 consecutive element हमें print कराने होंगे i, mlen

8, 9, 10, 11, 12.

```
import java.util.*;
import java.io.*;
public class Main
{
    public static void main(String[] args)
    {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int arr[] = new int[n];
        for(int i=0; i<arr.length; i++)
        {
            arr[i]=scn.nextInt();
        }

        HashMap<Integer, Boolean> map= new HashMap<>();
        for(int val:arr)
        {
            map.put(val, true);
        }

        for(int val:arr)
        {
            if(map.containsKey(val-1)==true)
            {
                map.put(val, false);
            }
        }

        int msp=0;
        int mlen=0;
```

→ इसी array का size input किया

इसने n size का array बनाया

इसने array के elements input किए

इसने एक Hashmap बनाया जिसमें Key का datatype होगा Integer और values का datatype होगा Boolean

इसने array के सभी elements को Hashmap में डाला

सभी Keys की values initially true रखी है

उपर के element से पहले वाली digit (element) Hashmap में है तो वो starting point नहीं हो सकता sequence का

initially maximum starting point 0 रखी है (max sequence का starting point) → सबसे बड़ी sequence का starting digit

maximum length भी 0 होगी initially → सबसे बड़ी sequence की length

```
for (int val : arr)
{
    if (map.get(val) == true)
    {
        int tsp = val;
        int tlen = 1;

        while (map.containsKey(tsp + tlen) == true)
        {
            tlen++;
        }

        if (tlen > mlen)
        {
            msp = tsp;
            mlen = tlen;
        }
    }
}
for (int i=0; i < mlen; i++)
{
    System.out.println(msp + i);
}
}.
```

for (int val : arr) → array के सभी element पर Loop लगाया और वो element select किए जिनकी values Hashmap में true है

int tsp = val; → temporary starting point

int tlen = 1; → temporary length

while (...) → हम हर element का ... करेगे और temporary length को increase करते चलेगे

if (tlen > mlen) → अगर templeng की value max length से ज्यादा है तो max starting point (msp) और mlen को update करदो.

for (int i=0; i < mlen; i++) → फिर msp और msp से next (mlen) element print करादो.