Longest Consecutive sequence

[100, 4, 200, 3, 1, 2] → 2^n → 26

Consecutive Value → 1, 2, 3, 4, 5, 6, 7, 8, 9. . . —

100, 101, 102, 103 ✓

100, 200, 205, 206 ✗

1, 2, 3, 4.

1, 2

1, 2, 3

1, 2, 3, 4. → longest consecutive sequence

100, 4, 200, 3, 1, 2

Using HashMap, we can do it in O(n)

1. traverse from 0 to n-1.

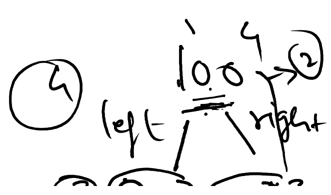
index
0 → 100.

99, 100, 101

2. Check if left of 100 which 99, is present in hashmap

99, 100 → 2

100 101 → 2



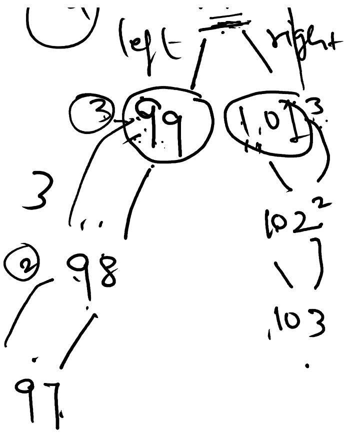
100 which 99, 3
present in hashmap
or not

```
if(hm.containsKey(99)){
  left = hm.get(99);
}
left=0
```

100, 101 → 2

99, 100, 101 → 3

100 → 1.



Here, we are using hashmap to store value along with its length of possible sequence

```
if (hm.containsKey(101)){
  right = hm.get(101);
}else{
  right=0;
}
```

total length = left+right+1;

100, 4, 200, 3, 1, 2, 4

100

leftlength = hm.get(99) = null
= 0

100 - 1

rightlength=0

rightlength - -

length = 0+1+0 ; maxlen = 1

hm.put(arr[i], length);

4
---

left = hm.get(3) = null = 0

right = hm.get(5) = null = 0

length = 0+1+0 = 1 maxlen = 1

hm.put(4, 1)

100 - 1
4 - 1

200
---

left = hm.get(199) = 0

right = hm.get(201) = 0

length = 0+1+0 = 1 → maxlen = 1

hm.put(200, 1);

100 - 1
4 - 1
200 - 1

3
---

left = hm.get(2) = 0

right = hm.get(4) = 1

length = 0+1+1 = 2 maxlen = 2

hm.put(3, length);

hm.put(3-left, length);

hm.put(3+right, length);

(4, 2)

100 - 1
4 - 2
200 - 1
3 - 2

→ 1+1 = 2

1+2 = 3

3, 4, 5

(3, 4)

(4, 2)

3, 4, 5 $\boxed{3, 4}$

100 - 1
4 - 2
200 - 1
3 - 2
1 - 1

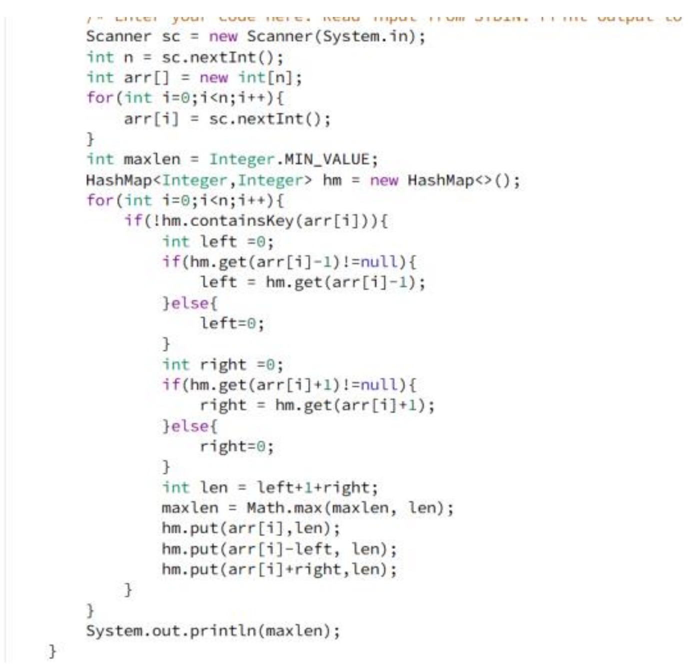
1

left = hm.get(0) = 0
right = hm.get(2) = 0
length = 0 + 1 + 0 = 1
maxlen = 2
hm.put(1, length);
hm.put(1 - left, length);
hm.put(1 + right, length);

2

100 - 1
4 - 4
200 - 1
3 - 2
1 - 4
2 - 4

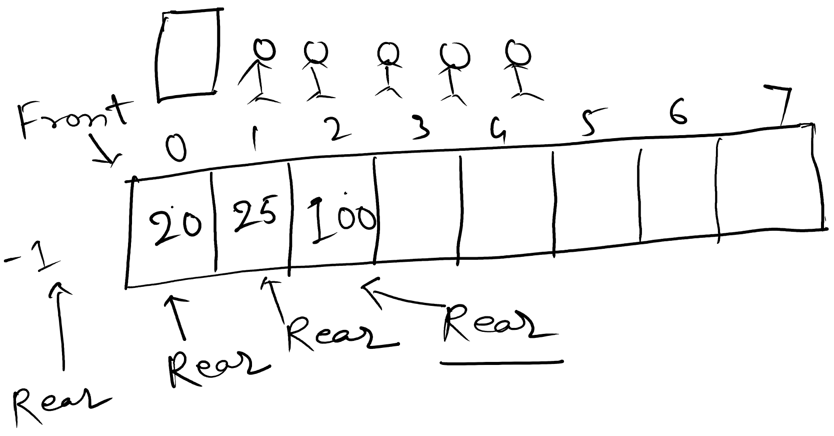
left = hm.get(1) = 1
right = hm.get(3) = 2
length = 1 + 1 + 2 = 4
maxlen = 4
hm.put(2, 4);
hm.put(2 - left, 4);
hm.put(2 + right, 4)



```
    /* Enter your code here. Read input from STDIN. Print output to
    Scanner sc = new Scanner(System.in);
    int n = sc.nextInt();
    int arr[] = new int[n];
    for(int i=0;i<n;i++){
        arr[i] = sc.nextInt();
    }
    int maxlen = Integer.MIN_VALUE;
    HashMap<Integer,Integer> hm = new HashMap<>();
    for(int i=0;i<n;i++){
        if(!hm.containsKey(arr[i])){
            int left =0;
            if(hm.get(arr[i]-1)!=null){
                left = hm.get(arr[i]-1);
            }else{
                left=0;
            }
            int right =0;
            if(hm.get(arr[i]+1)!=null){
                right = hm.get(arr[i]+1);
            }else{
                right=0;
            }
            int len = left+1+right;
            maxlen = Math.max(maxlen, len);
            hm.put(arr[i],len);
            hm.put(arr[i]-left, len);
            hm.put(arr[i]+right,len);
        }
    }
    System.out.println(maxlen);
}
```

Queue :- It is a linear data structure

FIFO → First In First Out



Queue is an interface.
It can be implemented by LinkedList

Syntax

Queue<Type> queue_name = new LinkedList<>();

Queue<Integer> my_queue = new LinkedList<>();

(i.) add(value) → It will insert value
(ii.) remove() → It will remove the value
(iii.) size() → To get the size of queue
(iv.) peek() → It will give you first value which is at front