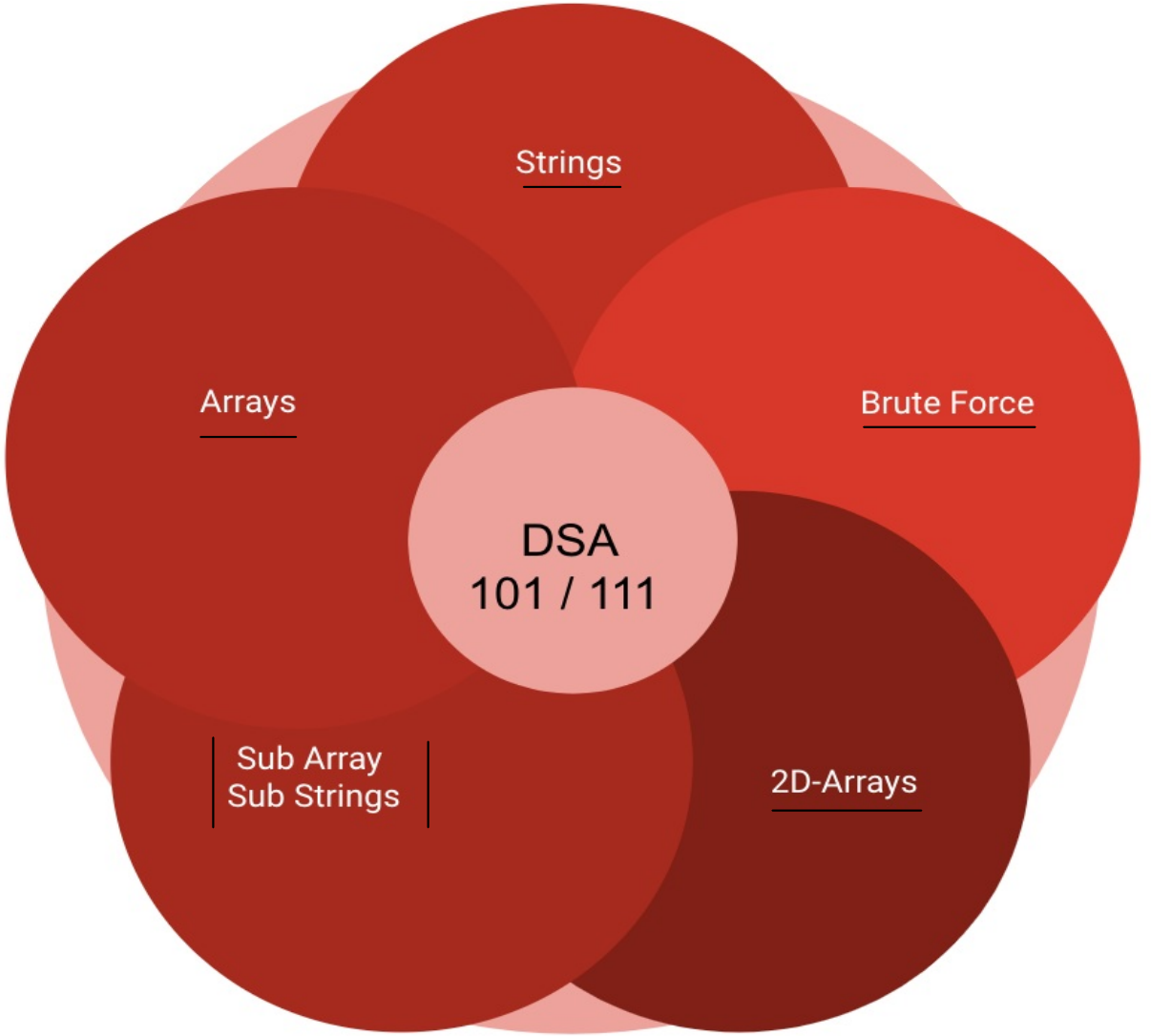
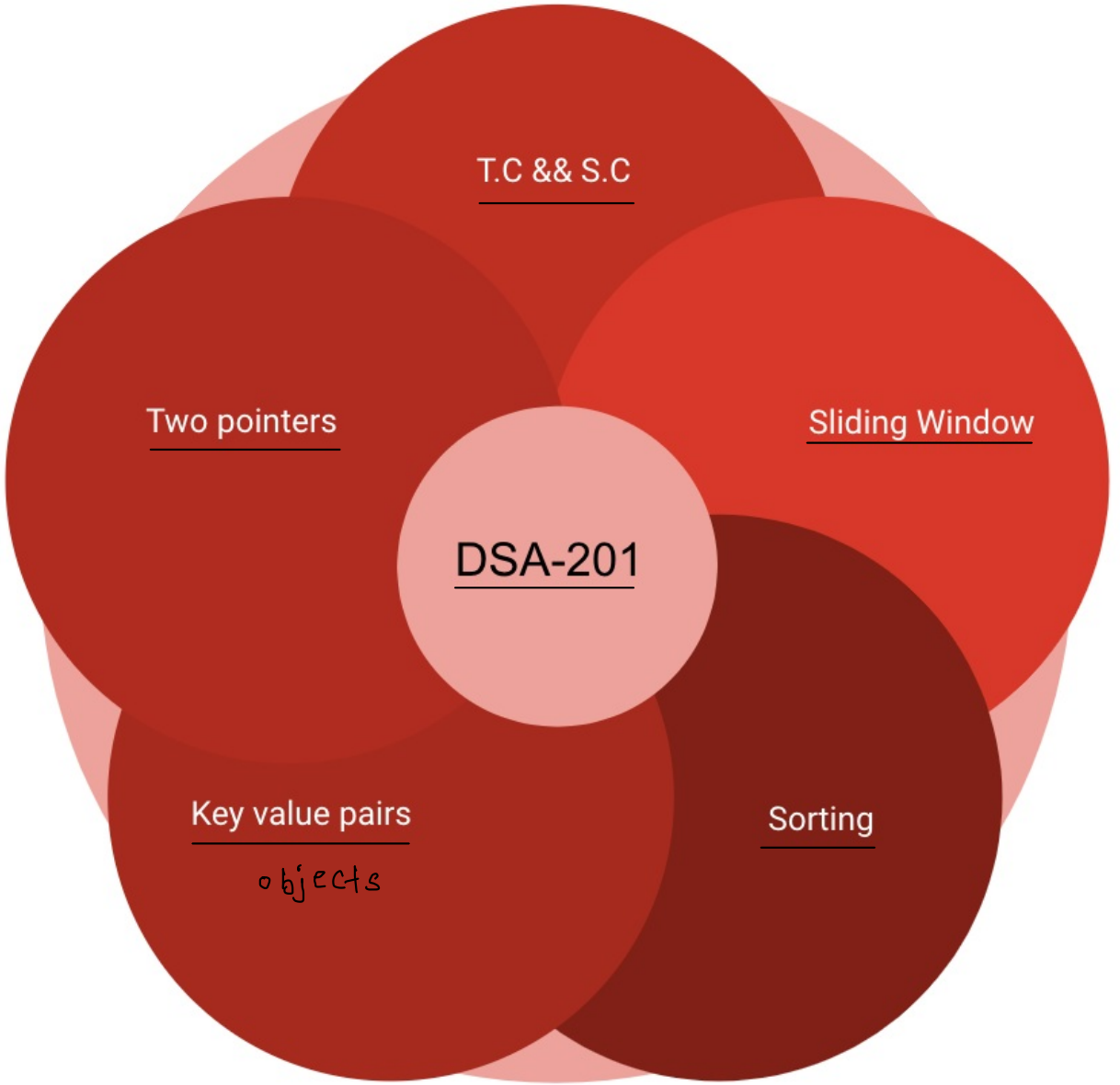
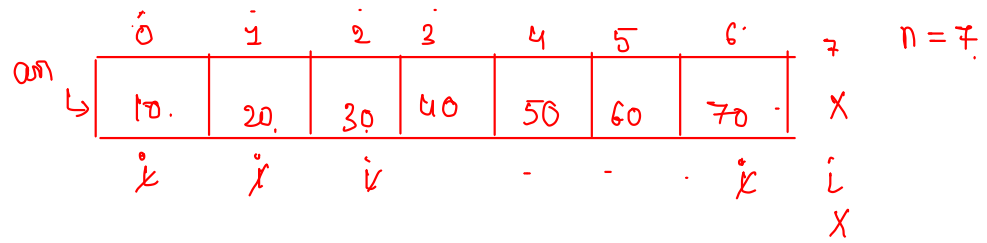
DSA - 201

Sprint-1 [Day-1]


Strings
Arrays
Brute Force
DSA
101 / 111
Sub Array
Sub Strings
2D-Arrays

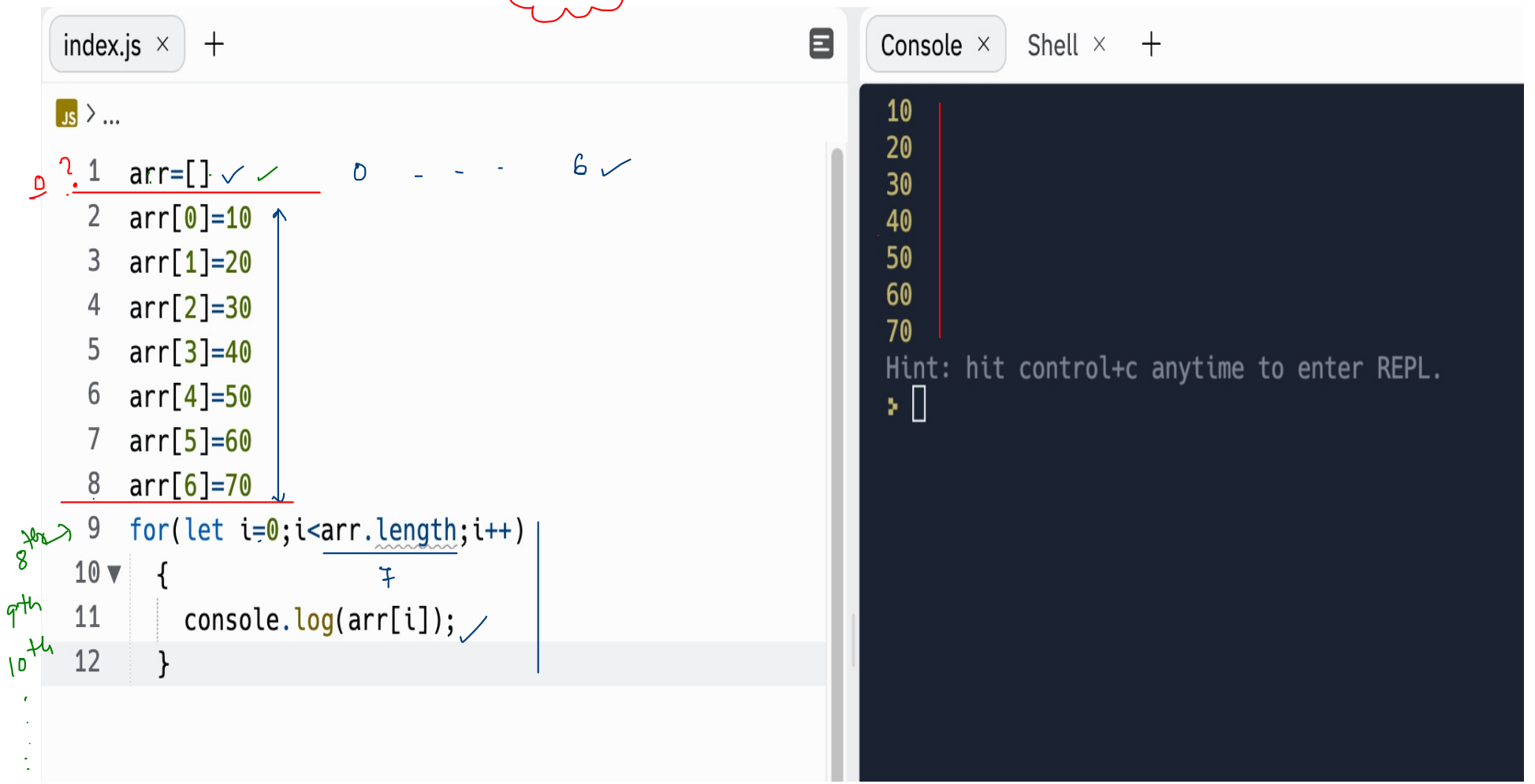

T.C && S.C
Two pointers
Sliding Window
DSA-201
Key value pairs
objects
Sorting

1-D Array




```
for(i=0;i<n;i++)
{
        console.log(arr[i])
}
```

arr = [ ]


index.js
Console
Shell
1 arr=[]
2 arr[0]=10
3 arr[1]=20
4 arr[2]=30
5 arr[3]=40
6 arr[4]=50
7 arr[5]=60
8 arr[6]=70
9 for(let i=0;i<arr.length;i++)
10 {
11 console.log(arr[i]);
12 }
10
20
30
40
50
60
70
Hint: hit control+c anytime to enter REPL.

```
// arr[6]=70
// for(let i=0;i<arr.length;i++)
//   {
//     console.log(arr[i]);
//   }

let n=7

let arr=new Array(n)
arr[0]=10
arr[1]=20
arr[2]=30
arr[3]=40
arr[4]=50
arr[5]=60
arr[6]=70
for(let i=0;i<arr.length;i++)
  {
    console.log(arr[i]);
  }
```

used to create object

Size

name

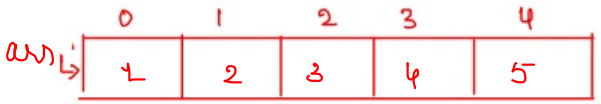
Array type

N. = 1000.

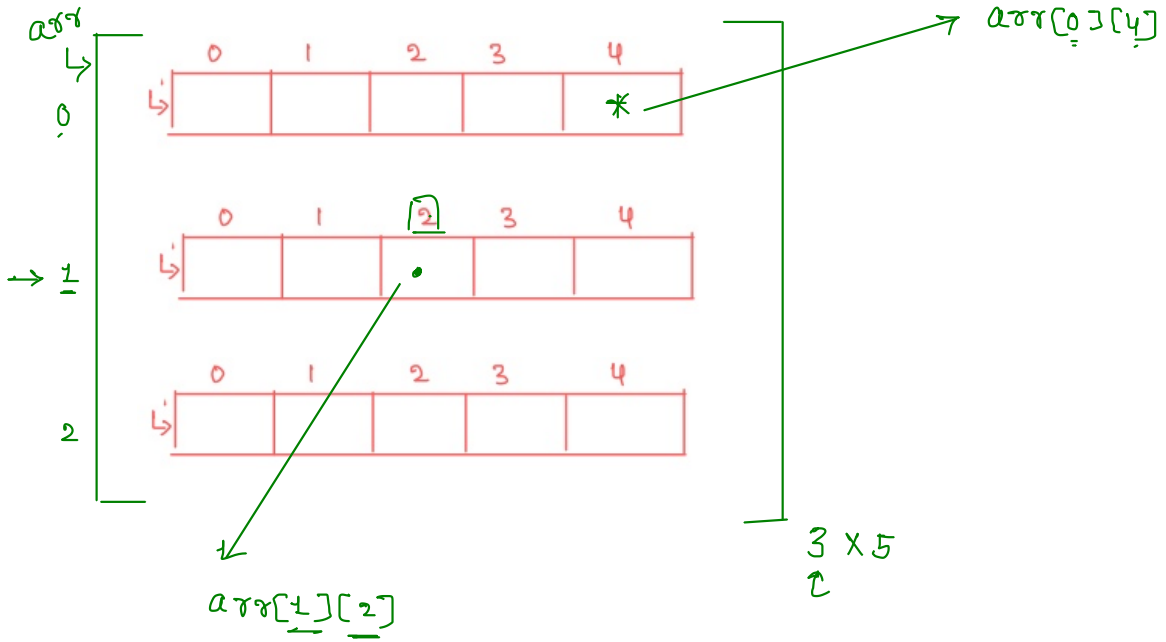
```
10
20
30
40
50
60
70
Hint: hit control+c
```

2D-Array

↳ collection
of 1D Array's



⇒ Size(arr) = 5



3D Array

↳ Collection of 2D-Array's

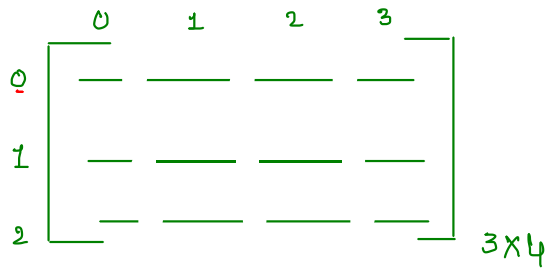
1D → [ ]

2D → [ ][ ]

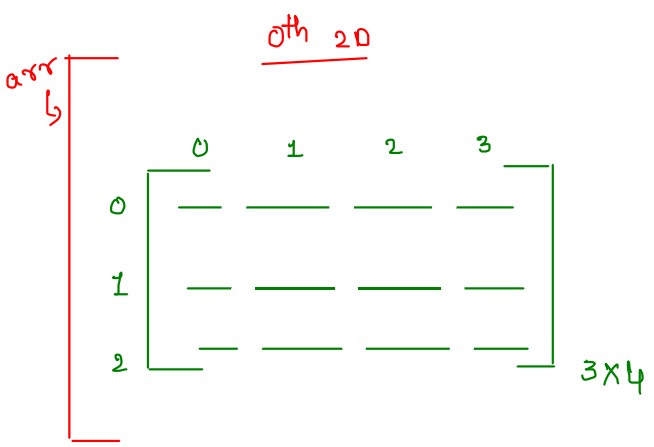
3D → [ ][ ][ ]

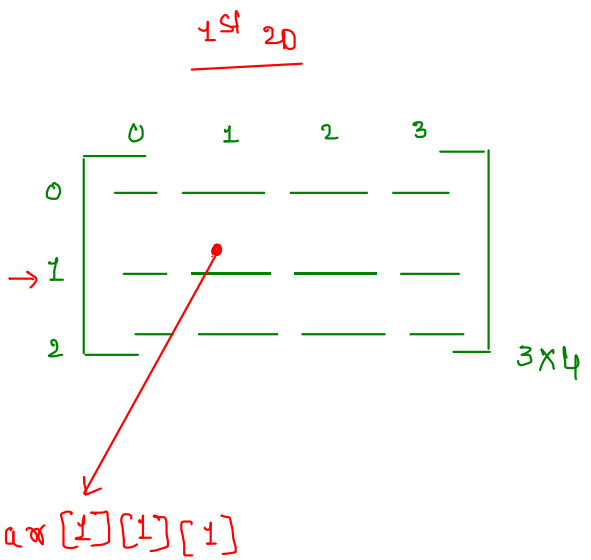
4D → [ ][ ][ ][ ]

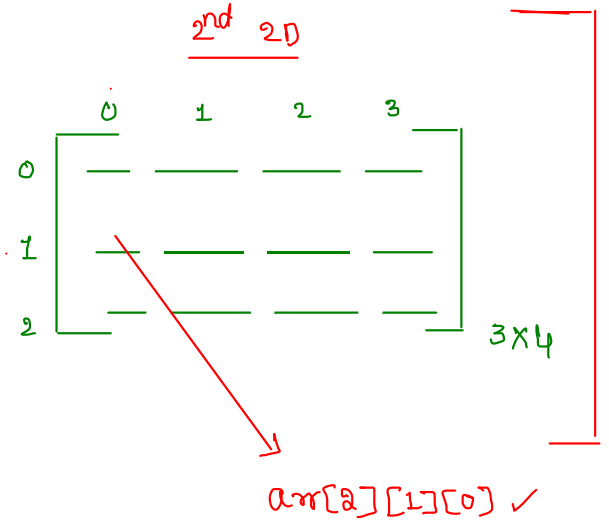
↳ collection of 3D arrs ✓



R / S







↳ collection of Rows & Columns

2-D Arrays [Matrix]

1. # of rows == # of columns

Square Matrix

let n = 5 ⇒ 5 x 5

give, ONLY one-dim

1. # of rows != # of columns

Rectangular Matrix

Ex

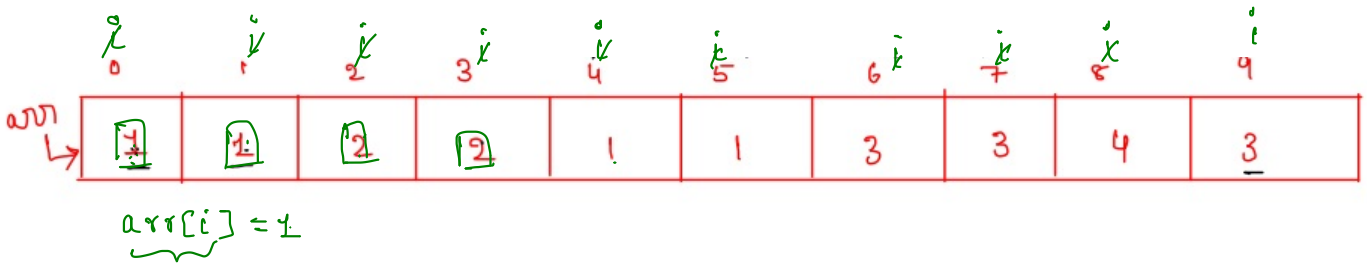
n = 3

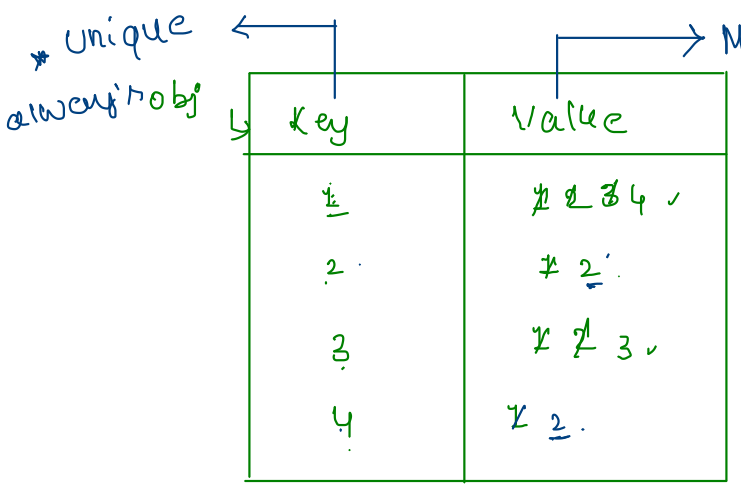
m = 4

⇒ 3 x 4

3D-Array

Objects [Key-Value Pair]





duplicates are allowed

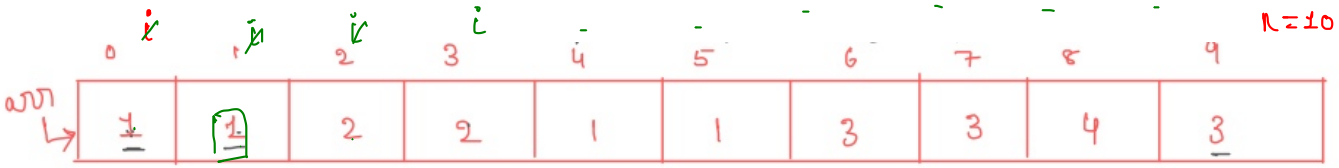
arr[i] → ✓ (present) → (arr[i], value+1)

arr[i] → X (not present) → (arr[i], 1)

```
function fun(arr,n)
{
    let obj={}
    for(let i=0;i<n;i++)
    {
        if(obj[arr[i]]==undefined)
        {
            obj[arr[i]]=1
        }
        else
        {
            obj[arr[i]]=obj[arr[i]]+1
        }
    }
}
```

size of arr

obj[arr[i]]++



obj

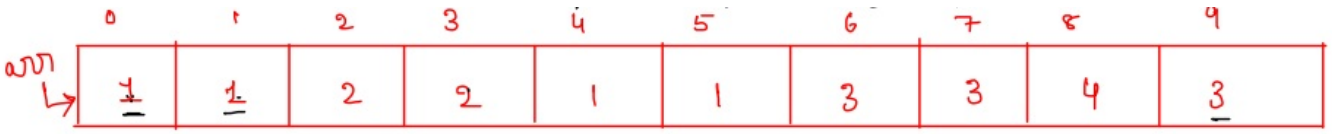
Key	Value
1	2
2	2

2

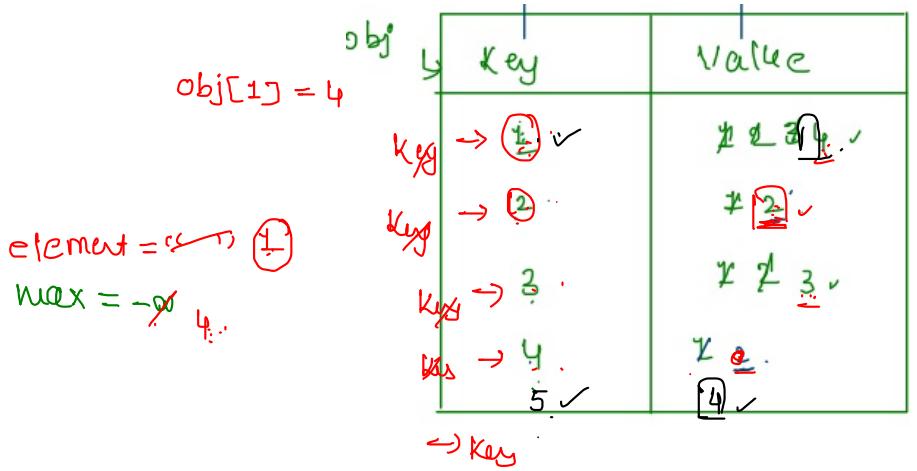
obj[arr[i]]
1

```
function fun(arr,n)
{
        let obj={}
        for(let i=0;i<n;i++)
        {
                if(obj[arr[i]]==undefined)
                {
                        obj[arr[i]]=1
                }
                else
                {
                        obj[arr[i]]=obj[arr[i]]+1
                }
        }
        let max=-infinity
        let element=" "
        for(let key in obj)
        {
                if(obj[key] > max)
                {
                        max=obj[key])
                        element=key
                }
        }
        return element
}
```

↳ Console.log(element)



Find an ele, which is having, highest frequency.



op: -1

max ele
↑
start with
-∞

4 > -∞

Min ele
:
start with
+∞