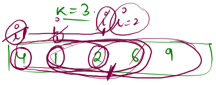
Previous. 2 question

TC. & SC.

3 Question

1. Sliding window.



K=3

for (i=0 → n-k]

max=~~2~~ ~~6~~ 9

max=4

max=~~1~~ ~~2~~ 6.

4 6 9 → o/p.

4 6 9

find Me - 6.

A → 2 ($i=0$) 1 -2 -4 3

B → for (-1 1 2 4 6)

Math.abs (A[0]) == B[2] ✓ A[i].

↳ Syso (A[i])

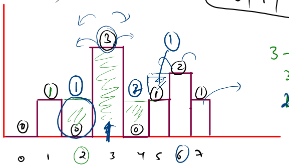
Store Maximum. (Trapping. (C U R) O(n²)

0	1	0	3	0	1	2	1

w=1

ans=~~1~~ ~~3~~ 4

r	l
3	1
3	3
3	2
3	2



3 - h[3] = 0

3 - 3 = 0

2 - h[4] = 2

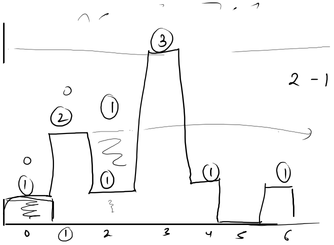
2 - h[5]

2 - 1 = 1

idx → 3

l → 3 , 3

3 - h[3] = 0.



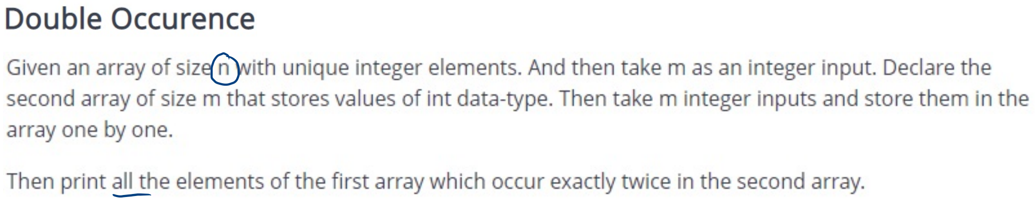
2 - 1 = 1

l	r
1	3

1 - 1 = 0

2 3

2 3



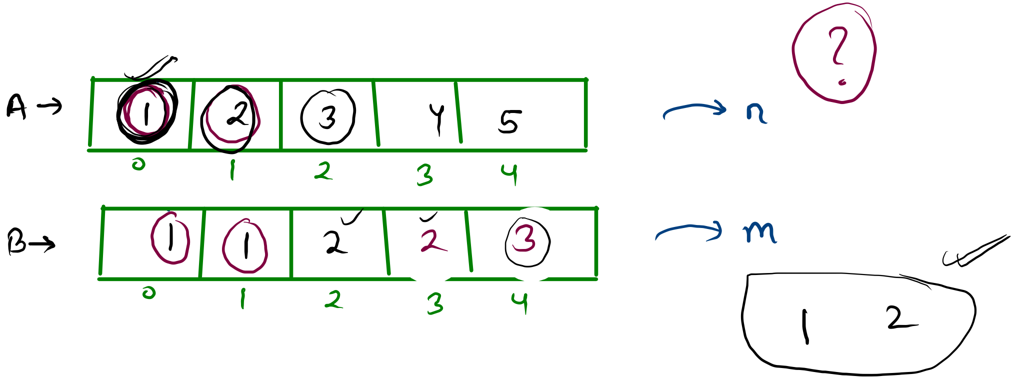
Double Occurence

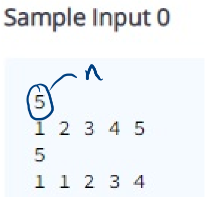
Given an array of size n with unique integer elements. And then take m as an integer input. Declare the second array of size m that stores values of int data-type. Then take m integer inputs and store them in the array one by one.

Then print all the elements of the first array which occur exactly twice in the second array.

A → 1 2.

o/p
1.



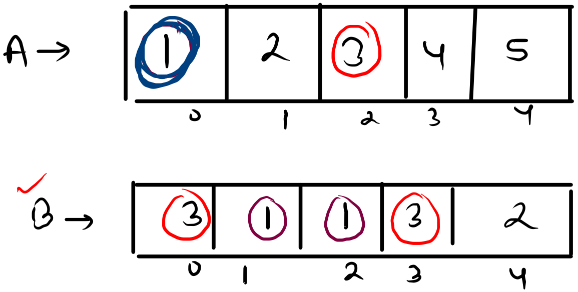


Sample Input 0

n

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

eg. logic.



Count = 2 → Syso(A[0])

A[0]

Mirror Image 4

abs.

Given an array of size n with unique integer elements. And Then take m as an integer input. Declare the second array of size m that stores values of int data-type. Then take m integer inputs and store them in the array one by one.

Then print all the unique elements of the first array whose additive inverses are present in the second array.

Sample Input 0

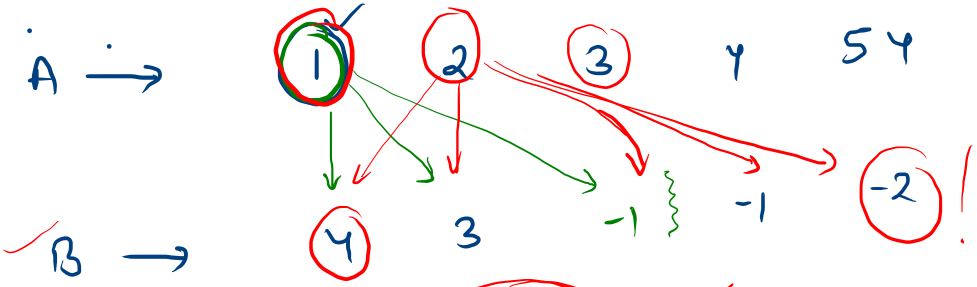
n

m

```
5
1  2  3  4  54
5
4  3  12  -1  -2
```

Sample Output 0

```
1  2
```

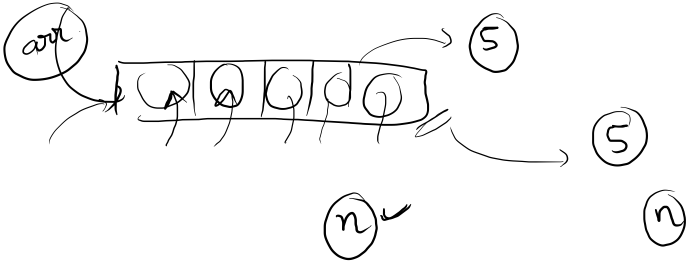


1 2

$A[i] + B[j] == 0)$

Time Complexity & Space Complexity

→ no. of input ∝ no. of opertion.



TC → relation.

no. of operation and i/p provided

1 → 1 O(1)

n → n → O(n).

O(1). ?

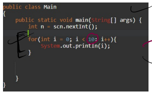
1 opertion ---- for any inp.

O(1).

declaring a variable.

doing operation take 1 oper.

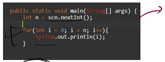
① n = 10   n = 1000



```
public class Main
{
    public static void main(String[] args) {
        int n = scn.nextInt();

        for(int i = 0; i < 10; i++){
            System.out.println(i);
        }


    }
}
```

$O(1) = O(\text{const})$

$O(\text{const}) = O(1) = O(\text{const} . 1) = \text{const } \boxed{O(1)}$



```
public static void main(String[] args) {
    int n = scn.nextInt();

    for(int i = 0; i < n; i++){
        System.out.println(i);
    }
```

n = 100

100 times

oper ∝ input → $\boxed{O(n)}$

(∝ op)

O → Oh → Worst → $O(n)$

ω → omega → Best → $\omega(n)$

Θ → theta → Average. → $\Theta(n)$

linear

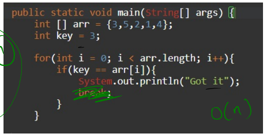
$\omega(1)$ →

$O(n)$

TC relation b/w → input & operation.

$O(1)$ → a+b; int val = 10; print( ) → 1; if( ) else( ) → $O(1)$.

$O(n)$.



```
public static void main(String[] args) {
    int [] arr = {3,5,2,1,4};
    int key = 3;

    for(int i = 0; i < arr.length; i++){
        if(key == arr[i]){
            System.out.println("Got it");
            break;
        }
    }
```

O, ω, Θ → (n)

$O(n)$

5 4 2 1 3

① 1 operation

linear Search.

best case TC → $\omega(1)$

worst case TC → $O(n)$

Avg TC → $\Theta(n)$.

$1 + 2 + 3 + 4 + 5.$

n

$\frac{\cancel{n}(n+1)}{2\cancel{n}}$

$1 + 2 + 3 + 4 ... + 5.$

oper ∝ $\frac{(n+1)}{2}$

$\frac{n(n+1)}{2}$

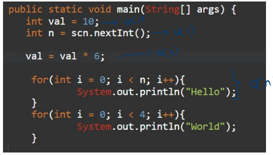
oper ∝ $\frac{n}{2} + \frac{1}{2}$ → const.

$\frac{f(n+3)}{}$ → 1

oper ∝ $\frac{1}{2} . n$

oper ∝ n

$\Theta(n)$

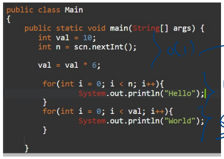


```java
public static void main(String[] args) {
    int val = 10;
    int n = scn.nextInt();

    val = val * 6;

     for(int i = 0; i < n; i++){
             System.out.println("Hello");
     }
     for(int i = 0; i < 4; i++){
             System.out.println("World");
     }
```

Const → o(1)

o(n)

⇒ O(n).



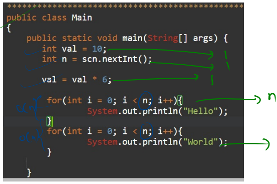
```java
public class Main
{
    public static void main(String[] args) {
        int val = 10;
        int n = scn.nextInt();

        val = val * 6;

         for(int i = 0; i < n; i++){
                 System.out.println("Hello");
         }
         for(int i = 0; i < val; i++){
                 System.out.println("World");
         }

    }
```

o(1) → n = 2 → 2000

O(n) → O(n)

60. → o(1)

fixe



```java
*******************************************************
public class Main
{
    public static void main(String[] args) {
        int val = 10;
        int n = scn.nextInt();

        val = val * 6;

         for(int i = 0; i < n; i++){
                 System.out.println("Hello");
         }
         for(int i = 0; i < n; i++){
                 System.out.println("World");
         }

    }
```

1

1

1

→ n

→ n

$O(1) + O(n) + O(n)$

$= O(2n)$

$= O(n)$

$O(2n + 3)$ → $O(n)$.

i=0 → (n operation)

i=1 → (n times)

i=2 → (n times)

|

:

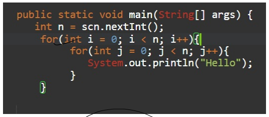
i=n (n times).

$\underbrace{n + n + n ---- + n}_{n \text{ times}}$ = n.n

= O(n²)

O(n²)

O(n²).



```java
public static void main(String[] args) {
    int n = scn.nextInt();
    for(int i = 0; i < n; i++){
        for(int j = 0; j < n; j++){
            System.out.println("Hello");
        }
    }
```

n=5

i=0 {0, 1, 2, 3, 4} 5 times ⇒ n

i=1 {0, 1, 2, 3, 4} → n

i=2 {0, 1, 2, 3, 4} → n

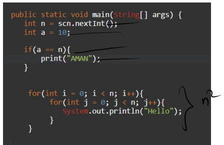
:

i=n-1 {0, 1, 2 --- } → n

1 → n

n → n×n.

O(n²)

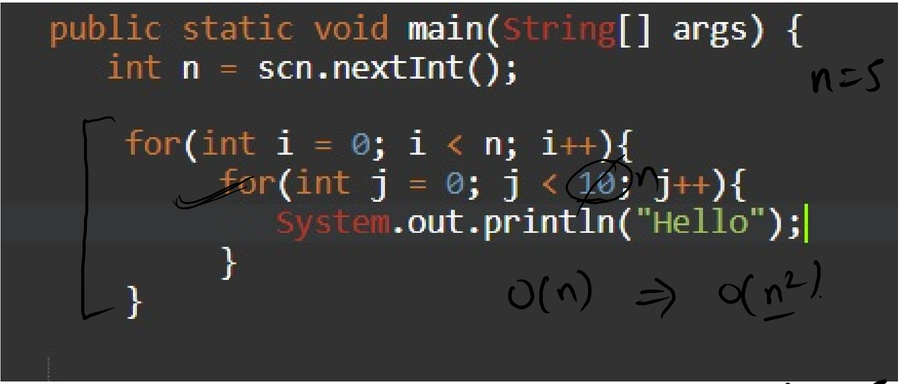


```java
public static void main(String[] args) {
    int n = scn.nextInt();
    int a = 10;

    if(a == n){
        print("AMAN");
    }


     for(int i = 0; i < n; i++){
         for(int j = 0; j < n; j++){
             System.out.println("Hello");
         }
     }
```

n²

n² + const

= O(n²)



```
public static void main(String[] args) {
   int n = scn.nextInt();

     for(int i = 0; i < n; i++){
        for(int j = 0; j < 10; j++){
           System.out.println("Hello");
        }
     }
```

~~10~~ n

n=5

$O(n) \Rightarrow O(\underline{n^2})$

$i=0 \rightarrow \boxed{10}$

$i=1 \rightarrow 10$

$i=2 \rightarrow 10$

const

$n = \underline{500}$.

$i=0$

$i=1$

$1 \rightarrow$ const.

$n \rightarrow n.$ const.

$\Rightarrow \underline{O(n)}$.

$\underline{\text{const raint}}$ $\boxed{n \le 10}$