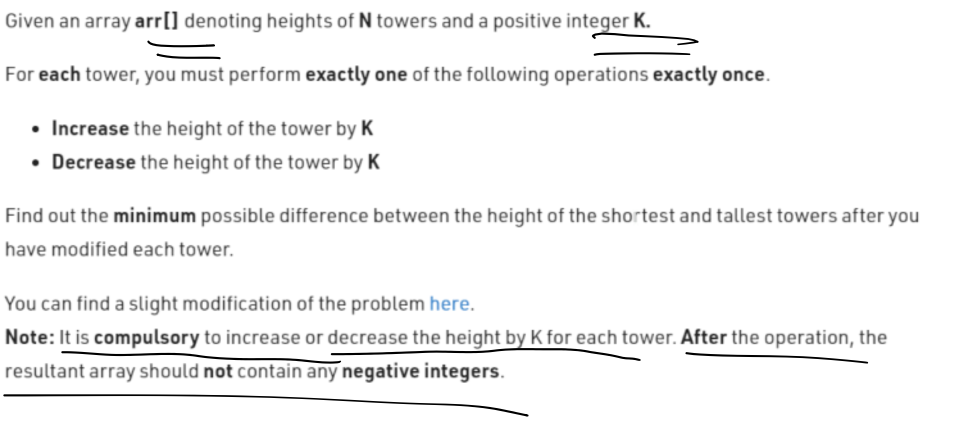
Day 6

12 November 2022 15:04



Given an array **arr[]** denoting heights of **N** towers and a positive integer **K.**

For **each** tower, you must perform **exactly one** of the following operations **exactly once**.

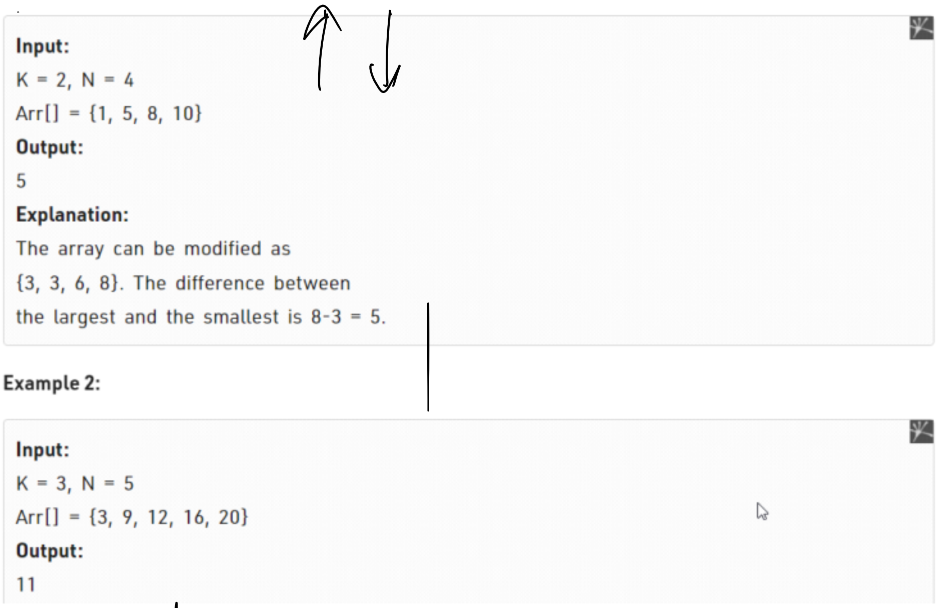
- **Increase** the height of the tower by **K**
- **Decrease** the height of the tower by **K**

Find out the **minimum** possible difference between the height of the shortest and tallest towers after you have modified each tower.

You can find a slight modification of the problem here.

Note: It is **compulsory** to increase or decrease the height by K for each tower. **After** the operation, the resultant array should **not** contain any **negative integers**.

Minimize

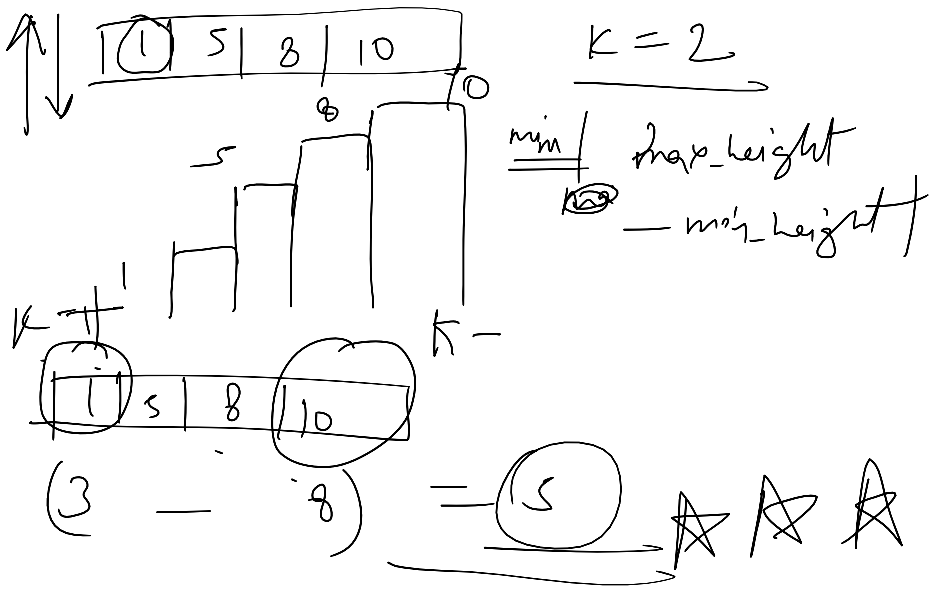


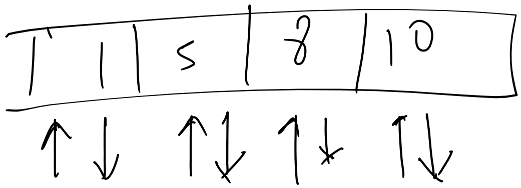
Input:
K = 2, N = 4
Arr[] = {1, 5, 8, 10}
Output:
5
Explanation:
The array can be modified as
{3, 3, 6, 8}. The difference between
the largest and the smallest is 8-3 = 5.

Example 2:

Input:
K = 3, N = 5
Arr[] = {3, 9, 12, 16, 20}
Output:
11

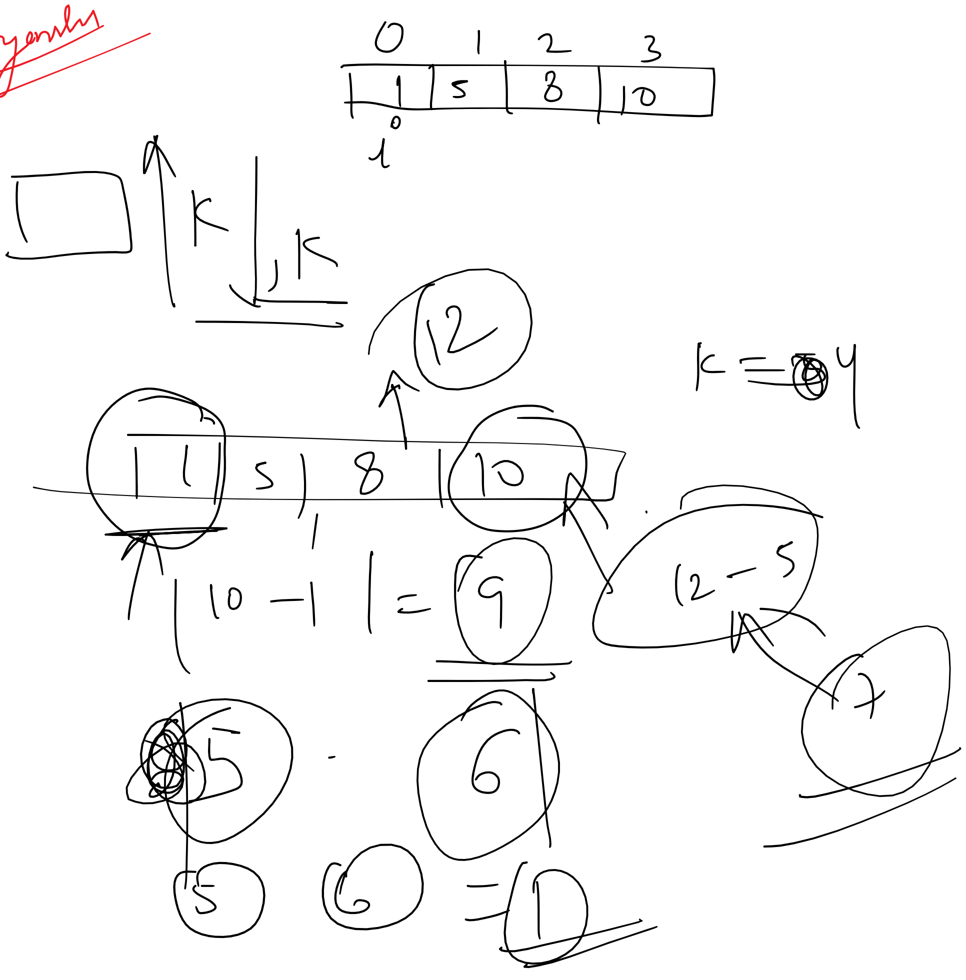
the heights 2



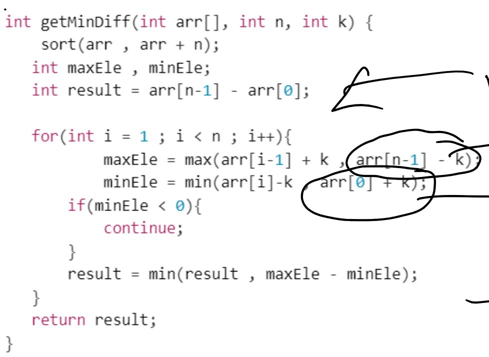


min
max] difference (minimize)

Satyendra



(Baseline)



```
int getMinDiff(int arr[], int n, int k) {
    sort(arr , arr + n);
   int maxEle , minEle;
   int result = arr[n-1] - arr[0];

   for(int i = 1 ; i < n ; i++){
          maxEle = max(arr[i-1] + k ,arr[n-1] - k);
          minEle = min(arr[i]-k , arr[0] + k);
       if(minEle < 0){
          continue;
       }
       result = min(result , maxEle - minEle);
   }
   return result;
}
```

← without any operations

→ largest

→ smallest

Algorithm ① sort the heights

k = 2

Algorithm ①

| 1 | 5 | 8 | 10 |

k = 2

| 1 | 5 | 8 | 10 |

baseline = (10 - 1) = 9 → ans

k = 2

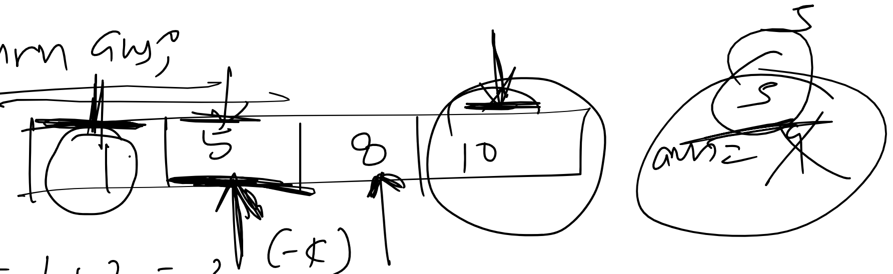
longest = 10 - k = 8

smallest = 1 + k = 3

```
for(int i=1; i<n; i++){
    mini = min(smallest, arr[i] - k);
    maxi = max(longest, arr[i-1] + k);
    ans = min(ans, maxi - mini);
}
return ans;
```

Ex|

k = 2



smaller = 1 + 2 = 3

longest = 10 - 2 = 8

5 - 2 = 3

8 - k = 6

5 + 2 = 7

```
for(i = 1 → to n-1)
    s = min(5,3)       // s = 3
    l = max(8, 1+k)    // l = 8
```

~~// ans = 5~~

10-2 (8, 3) // 3 ⑦

max (8+2, 8) // 10

max (8+2, 8) //10

Ex-2

3	9	12	16	20
	i			

k = 3

ans = 17

smallest = 3 + K = 6 — 0th index ignore

largest = 20 - k = 17 n-1th index ignore

i = 1 to n-1

i = 1 :

- s = min(s, 9-3) // s = 6
- l = max(l, arr[0]+k) // l = 17

$3+3$

(17, 6)

17 - 6 → 11 ✓

$\boxed{ans = 11}$

i = 2 :

s = 6   l = 17   ans = 11   k = 3

s = min(s, arr[i]-k) // s = 6

//12 - 3 = 9

l = max(l, arr[i-1]+k) // l = 17

max(17, 12)

11

i = 3 :

s = 6   l = 17   ans = 11   k = 3

s = min(s, 16-3) // (13, 6) // 6

$l = \max(l, 12+3)$ // $(17, 15)$ // 17

$\boxed{11}$

$i=4$ $s=6$ $l=17$ $ans=11$ $k=3$

$s = \min(s, 20-3)$ // 6

$l = \max(l, 16+3)$ // $(17, 19)$ 19

$l-8$ $\boxed{13}$

$ans = \min(13, 11)$

$\boxed{ans=11}$

$\boxed{i≥5}$

$\boxed{ans=11}$ ✓

Ex-3

1	2	5	8	10	13
0	1	2	3	4	5

$k=3$

$ans = 13-1 \Rightarrow \boxed{12}$

smallest $= \boxed{4}$ // 0th index ignore

largest $= 13 - k = \boxed{10}$ // n-1th index ignore

$i=1$: $s=4$ $l=10$ $ans=12$ $k=1$

$s = \min(s, 2-3)$

$m =$

~~20~~ ~~8~~ $\boxed{10}$

skip

$i=2$: $s=4$ $l=10$ $ans=12$ $k=3$

$s = \min(s, 5-3)$ // $(4,2)$ // $s=2$

$l = \max(l, 2+3)$ // $(10,5)$ // $l=10$

$ans = \min(\underset{12}{\underline{ans}}, \underset{8}{\underline{l-s}})$ // $\underline{ans=8}$

$i=3$: $s=2$ $l=10$ $ans=8$ $k=3$

$s = \min(s, 8-3)$ // $\min(2,5)$ // $\boxed{2}$

$l = \max(l, 5+3)$ // $\max(10,8)$ // $\underline{10}$

$ans = \underline{\boxed{10-2}}$

$\underline{i=4:}$ $s=2$ $l=10$ $ans=8$ $k=3$

$s = \min(s, 10-3)$ // 2

$l = \max(l, 8+3)$ // $\underline{11}$

$ans = \boxed{11-2}$ $\underline{\boxed{9}}$

$i=5$: $s=2$ $l=11$ $ans=8$ $k=3$

$s = \boxed{13-3}$ // 2

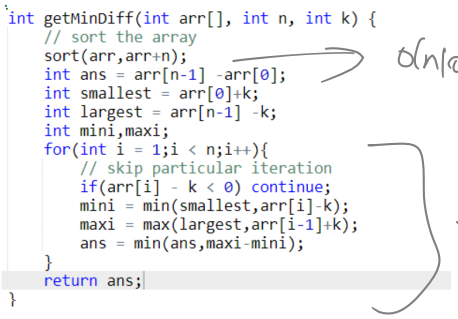
$l = \boxed{10+3}$ // 13

$\boxed{13-2}$ $\underline{\boxed{11}}$

$\boxed{ans = 8}$ ★★

$\boxed{ans = 8}$ ☆☆

T.C - O(nlogn + n)



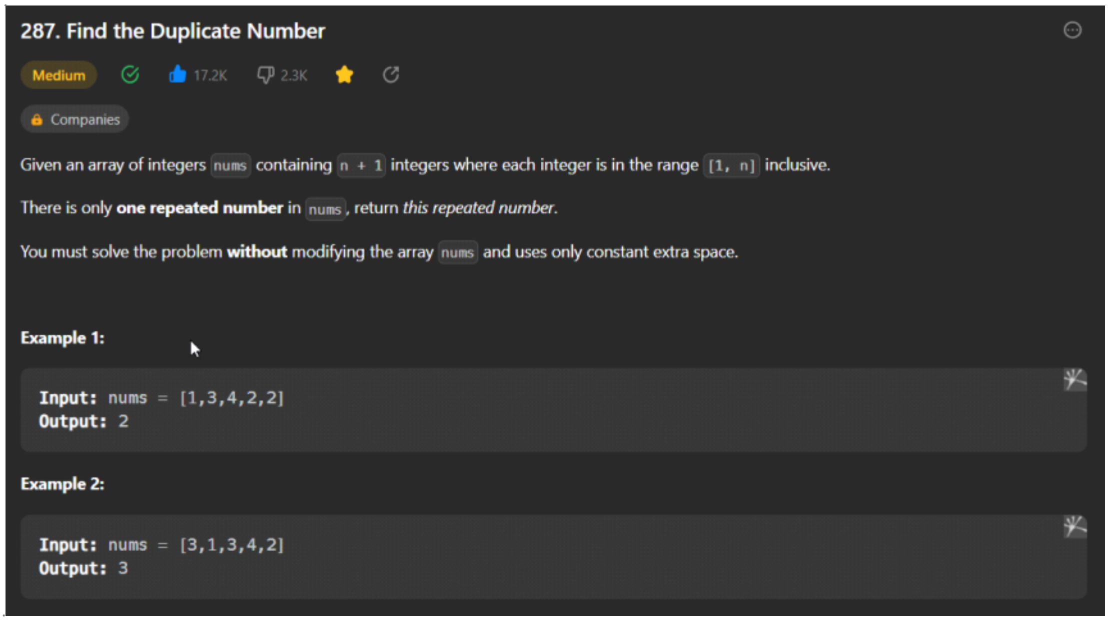
```
int getMinDiff(int arr[], int n, int k) {
    // sort the array
    sort(arr,arr+n);
    int ans = arr[n-1] -arr[0];
    int smallest = arr[0]+k;
    int largest = arr[n-1] -k;
    int mini,maxi;
    for(int i = 1;i < n;i++){
        // skip particular iteration
        if(arr[i] - k < 0) continue;
        mini = min(smallest,arr[i]-k);
        maxi = max(largest,arr[i-1]+k);
        ans = min(ans,maxi-mini);
    }
    return ans;
}
```

→ O(nlogn)

n

↓

O(nlogn)



287. Find the Duplicate Number

Medium 17.2K 2.3K

Companies

Given an array of integers `nums` containing `n + 1` integers where each integer is in the range `[1, n]` inclusive.

There is only **one repeated number** in `nums`, return *this repeated number*.

You must solve the problem **without** modifying the array `nums` and uses only constant extra space.

Example 1:

```
Input: nums = [1,3,4,2,2]
Output: 2
```

Example 2:

```
Input: nums = [3,1,3,4,2]
Output: 3
```

| 1 | 3 | 4 | 2 | 2 | (1→4)

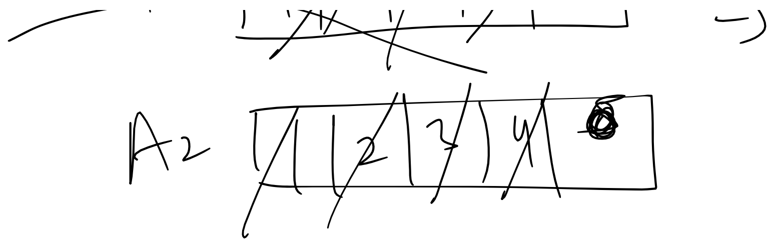
1 ⑤ n+1 integer

↘ ④

n=3 → [1,n]

| 1 | 2 | 2 | 3 |

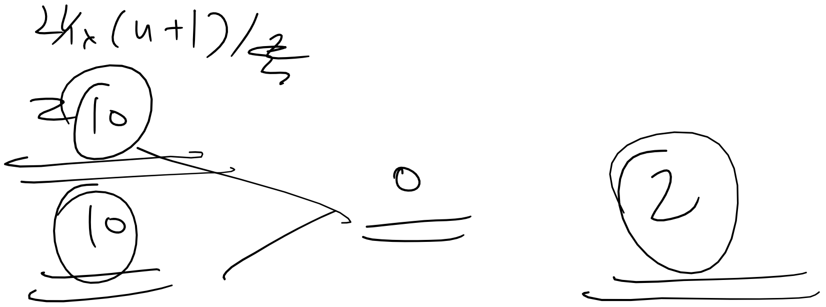
A1 ~~| 1 | 3 | 4 | 2 | 2 |~~ =>



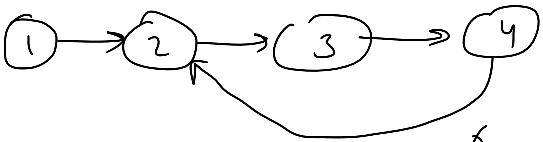
[2,2,2,2,2]

From <https://leetcode.com/problems/find-the-duplicate-number/>

2	2	2	2	2
0	1	2	3	4



Cycle Detection in linked list

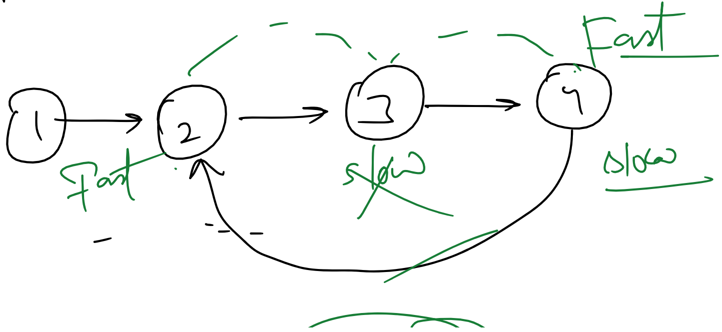


cycle present है या नहीं ||

Slow and fast pointer

1 step at a time

two step at a time



n+1 elements   [1, n]

(0 → n)

Ex1

0	1	2	3	4
1	3	4	2	2

slow = arr[0] // 1
fast = arr[0] // 1

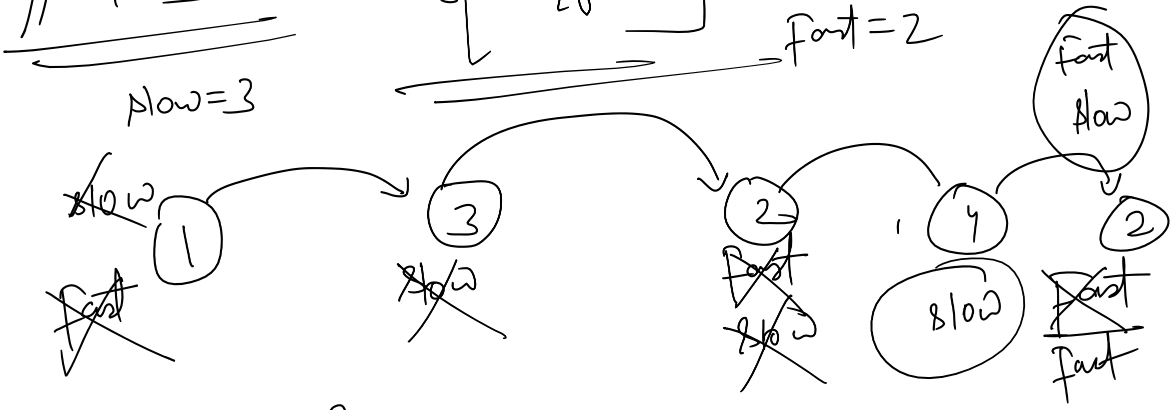
arr [2]
arr[4]

slow 1 step
//arr[slow]

fast 2 steps
arr[arr[fast]]

fast=2

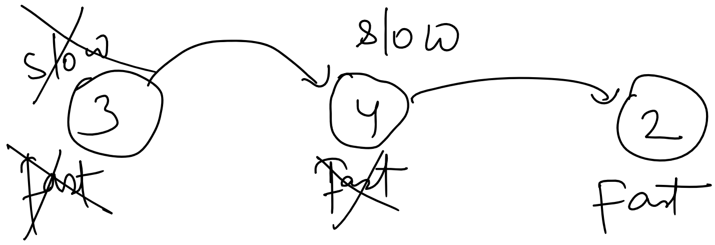
slow=3



~~slow~~ arr[2]

0	1	2	3	4
3	1	3	4	2

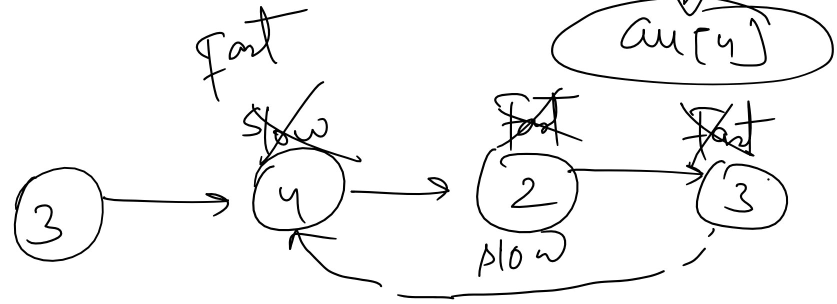
slow = arr[0] , fast = arr[0]
slow = 3   fast = 3



★ slow = 3 ⇒ 1 jump   arr[3]

★ fast = 3 ⇒ 1 jump ⇒ arr[3] ⇓ (4)

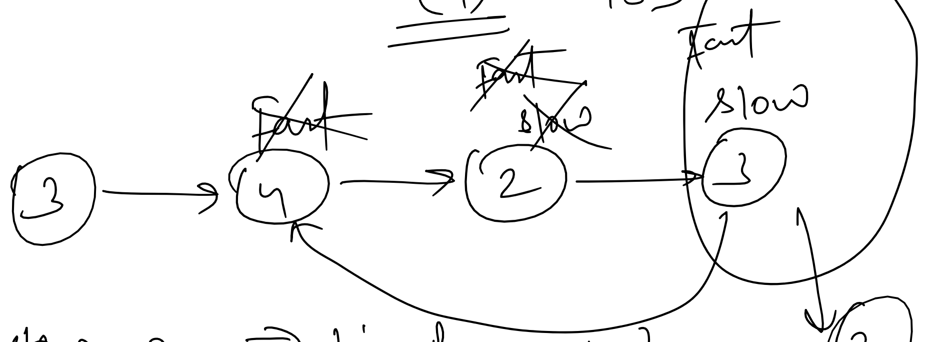
arr[4]



★ slow = 4 ⇒ 1 jump = arr[4]

★ fast = 2 ⇒ 1 jump = arr[2] (3)

(4) arr[3]



★ slow = 2 ⇒ 1 jump = arr[2]

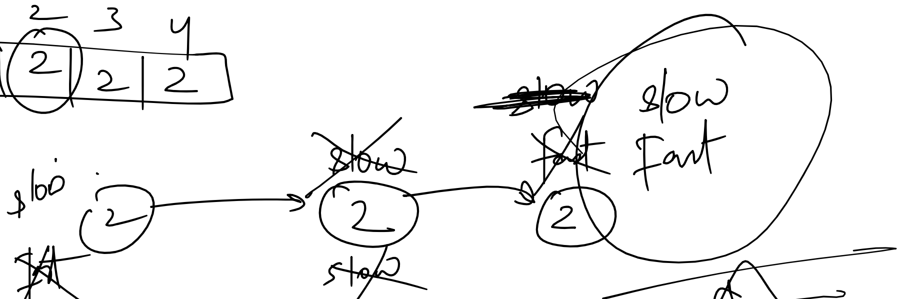
Ans 3

★

0	1	2	3	4
2	2	2	2	2

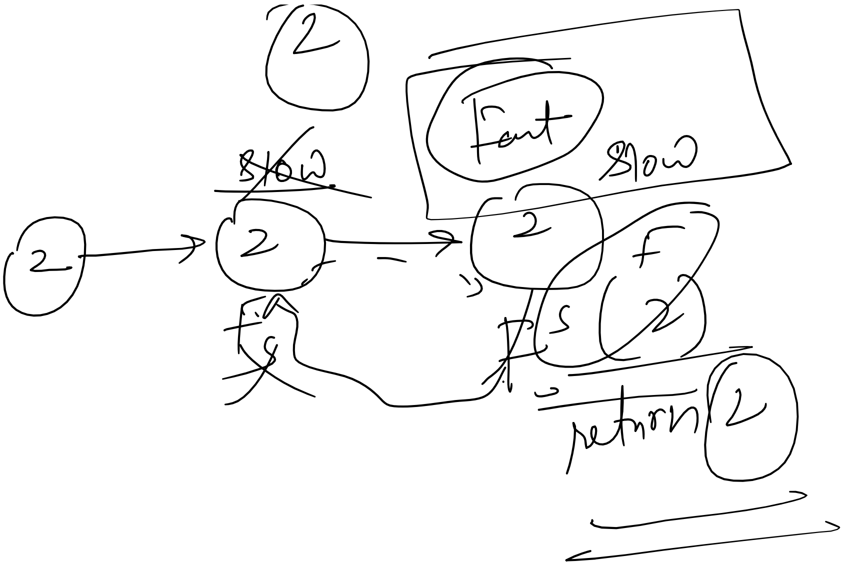
slow = 2

Fast = 2



slow ⇒ 1 jump ⇒ arr[2) ⇒ 2

fast ⇒ 2 jump ⇒ arr[2] ⇒ 2 ⇓ 2

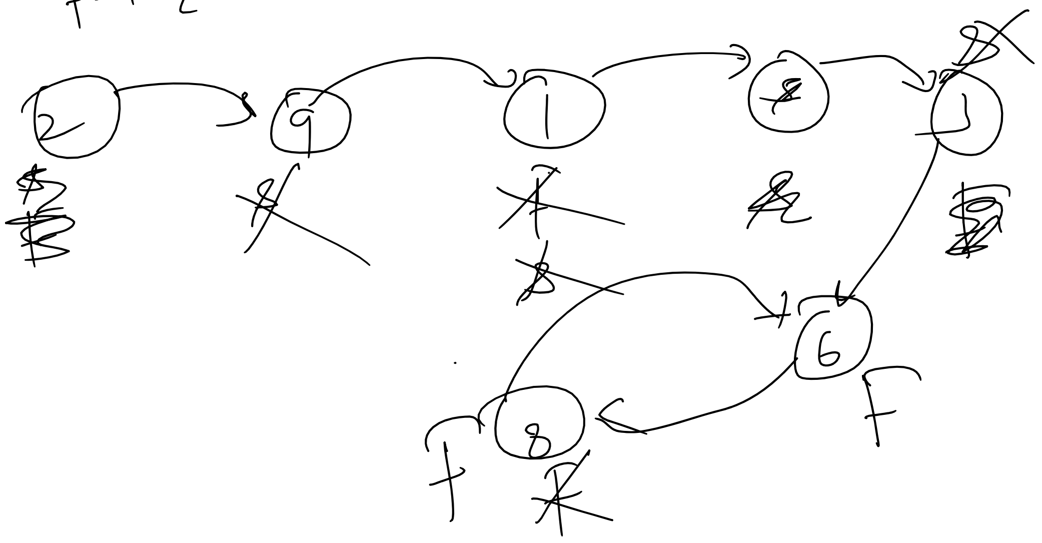


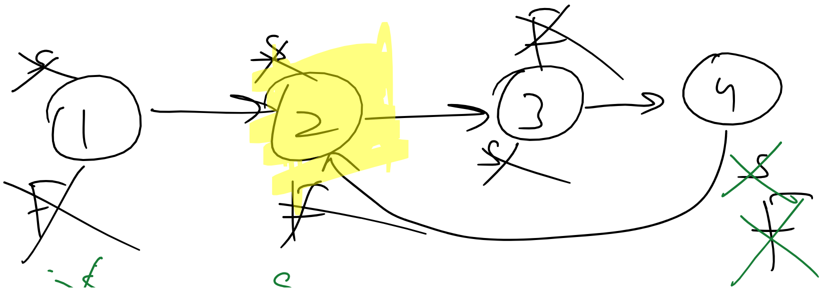
[2,5,9,6,9,3,8,9,7,1]

From <https://leetcode.com/problems/find-the-duplicate-number/>

0	1	2	3	4	5	6	7	8	9
2	5	9	6	9	3	8	9	7	1

slow = 2
Fast = 2





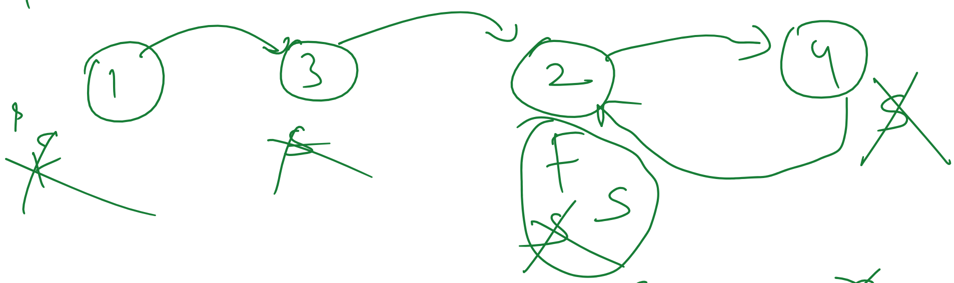
slow → 1 step

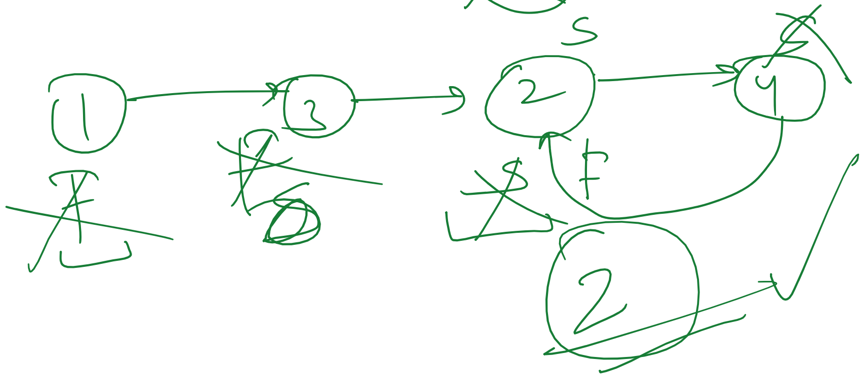
Fast → 2

1	3	4	2	2
0	1	2	3	4

slow = ~~0~~ 1

fast = ~~2~~ 4



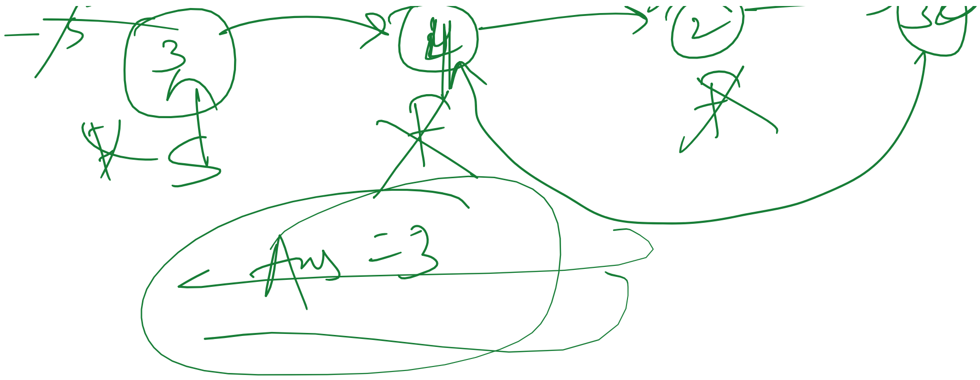


3	1	3	4	2
0	1	2	3	4

slow = 3

fast = 3

☆☆ বাপিরে নদী গাড়ী।



Ans = 3