Day- 4

odd → input length

Except one element, All elements appears twice. Find that element

2(n) + 1 ⇒ odd

① → O(n²) + O(1) → O(nlogn) + O(n) ②

Input: ar[] = {7, 3, 5, 4, 5, 3, 4}

Output: 7

n

7 → 1
3 → 2
5 → 2
4 → 2

3 3 4 4 5 5 7
i j

O(n) + O(n) ③

O(n) + O(1) *

n/2 + 1

key	v
7	1
3	1 2
5	1 2
4	1 2

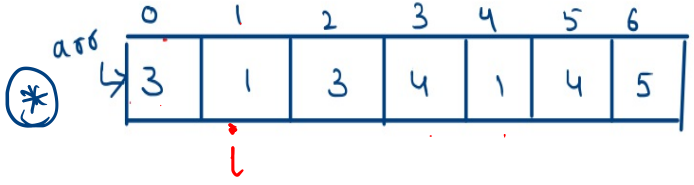
PS

```
temp = arr[0]
for(i=1; i<n; i++)
    temp = temp ^ arr[i]
```

O(n) + O(1) ⇒

temp = arr[0]

Print(temp)



12th

→ Ex-OR operator [⊕] ^

A, B are Boolean Variables ↳ 0/1

A	B	A+B	A·B	A⊕B
0	0	0	0	0 ✓
0	1	1	0	1 ✓
1	0	1	0	1
1	1	1	1	0 ✓

temp = 3 ^ 1 ^ 3 ^ 4 ^ 1 ^ 4 ^ 5

2 0 ^ 5 = 5 ✓

3 → 1 1

1 → 0 1

1 0 ⇒ 2

2 → 10

3 → 11

01 ⇒ 1

3 ^ 1 ^ 3 = 1

0 ^ 1 = 1

$0 \oplus X = X$

$X \oplus X = 0$ ✓

$0 \oplus 1 = 1$

$0 \oplus 3 = 3$

EA

(*) Check given number is Even/Odd with out using Arithemetic Operators ✓

n%2 == 0 → even
else → odd
*

↳ +, *, -, %, /

$n_1 = 10 \Rightarrow (1\ 0\ 1\ \boxed{0})_2$

$n_2 = 11 \Rightarrow (1\ 0\ 1\ \boxed{1})_2$ *

$n_3 = 7 \Rightarrow (0\ 1\ 1\ \boxed{1})_2$ *

$n_3 = 8 \Rightarrow (1\ 0\ 0\ \boxed{0})_2$

odd → last bit : 1

even → last bit : 0

$n_1 = 10$

$n_1 \& 1$

```
  1010
& 0001
------
  0000   ⇒ 0
```

$n_2 = 11$; $n_2 \& 1$

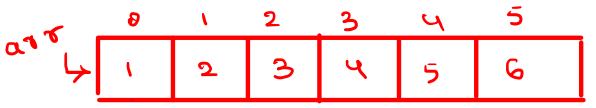
```
  1011
& 0001
------
  0001   ⇒ 1
```

n & 1
↳ 0 even
↳ 1 odd

✓ 2) Rotate array by d elements [left], anti-clock wise direction

input:-

arr[6]= { 1,2,3,4,5,6 } and d=2



n = 6

d = 2

output :-

d=1 ==> 2,3,4,5,6,1 ✓

d=2 ==> 3,4,5,6,1,2 [final output]

d=1 2 3 4 5 6 1

d=2 3 4 5 6 1 2 (*) o/p

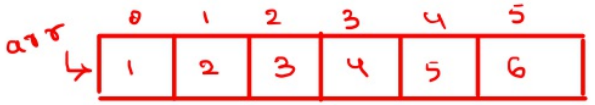
n = 6, d = 7

*

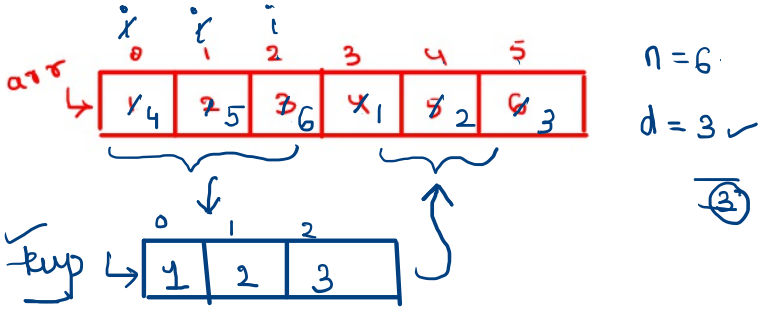
6

1

d = d % n


arr
0
1
2
3
4
5
1
2
3
4
5
6

4 5 6 1 2 3



n = 6

d = 3 ✓

(2)

Brute-Force :-

→ d = d % n

(1) take temp array of size d

(2) copy first d elements of given array into temp array → d times

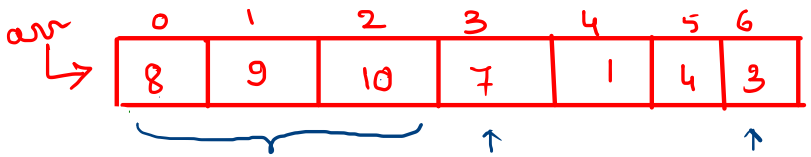
(3) shift n-d elements to left, using a for loop [arr[i] = arr[i+d]] → n - d

(4) put all elements of temp array to last d positions in given array ✓ → d

O(n) + O(d)

d == n

Multiple-sub-arrays-reverse



d = 3 (left rotate)
n = 7

O(d) → (1) reverse(arr,0,d-1) 0 to 2

O(n-d) → (2) reverse(arr,d,n-1) 3 to 6

O(n) → (3) reverse(arr,0,n-1) 0 to 6

↳ arr ⇒ o/p

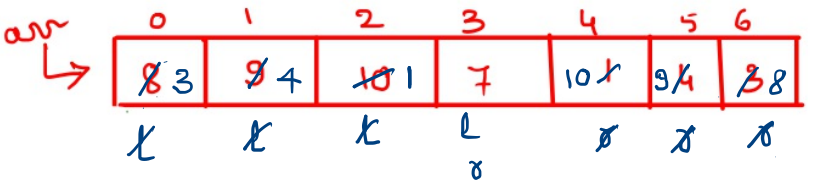
10 9 8 7 1 4 3 ←

10 9 8 3 4 1 7 ←

7 1 4 3 8 9 10 ⇒ o/p (?)

→ O(n) + O(1)

(2-ptr + swap)



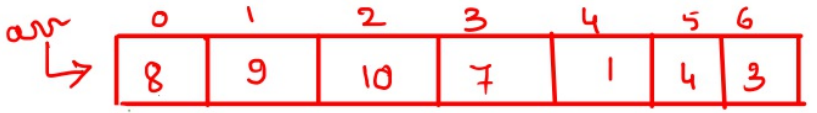

```
while(l<r)
{
  .....
}
```

reverse arr

3 4 1 7 10 9 8

O(n) ⇒ reverse(arr, l, r)

```
{
  .....
}
```


arr
0
1
2
3
4
5
6
8
9
10
7
1
4
3

* Home work : please do for same problem, clock wise

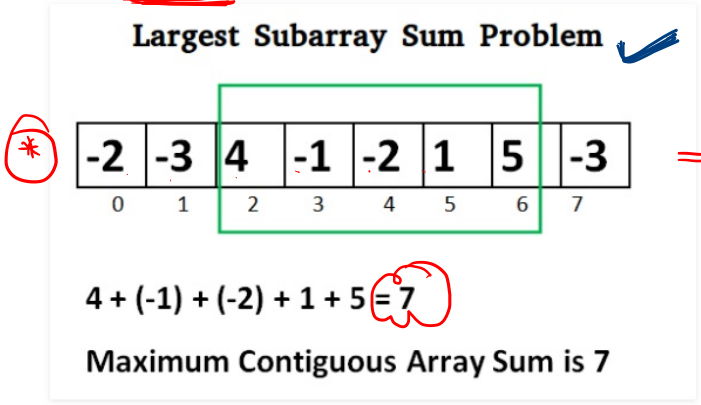
→ Notes

arr[6] = { 1,2,3,4,5,6} and d=2

d=1 ==> 6,1,2,3,4,5

* d=2 ==> 5,6,1,2,3,4 [final output]

Write an efficient program to find the sum of contiguous subarray within a one-dimensional array of numbers that has the largest sum.



max_sum = 0 → basic

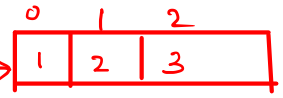
Local_sum = 0

{ unit - 2 }

1) B.F

n → # of sub-Array's

Sum — arr



$\frac{n(n+1)}{2} \Rightarrow O(n^2)$

↳ $O(n^2) \times O(n)$

$O(n^3)$

⇒ 1 2 3 → length (1)

1 2 2 3 → length (2)

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

1 2 3 → length (3)

n = 3

$\frac{3+2+1}{4+3+2+1} = (6)$

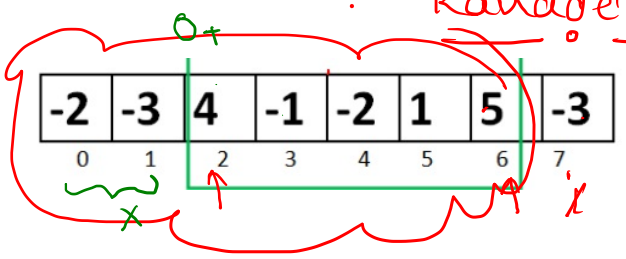
$= (10)$

n = 4

Brute-Force

Kanade's Algo:

Int. Min Value



- SUM = -2 4 7 ✓ ⇒ SUM
- CURR_SUM = 0 -2 0

-3 0 4 3 1 2 7 4

```
function fun(arr[],n)
{
   sum=-infinity
   curr_sum=0
   for(i=0;i<n;i++)
   {
      curr_sum=curr_sum+arr[i]
      if(curr_sum>sum)
         sum=curr_sum
      if(curr_sum<0)
        curr_sum=0
   }
   return sum
}
```

(7)

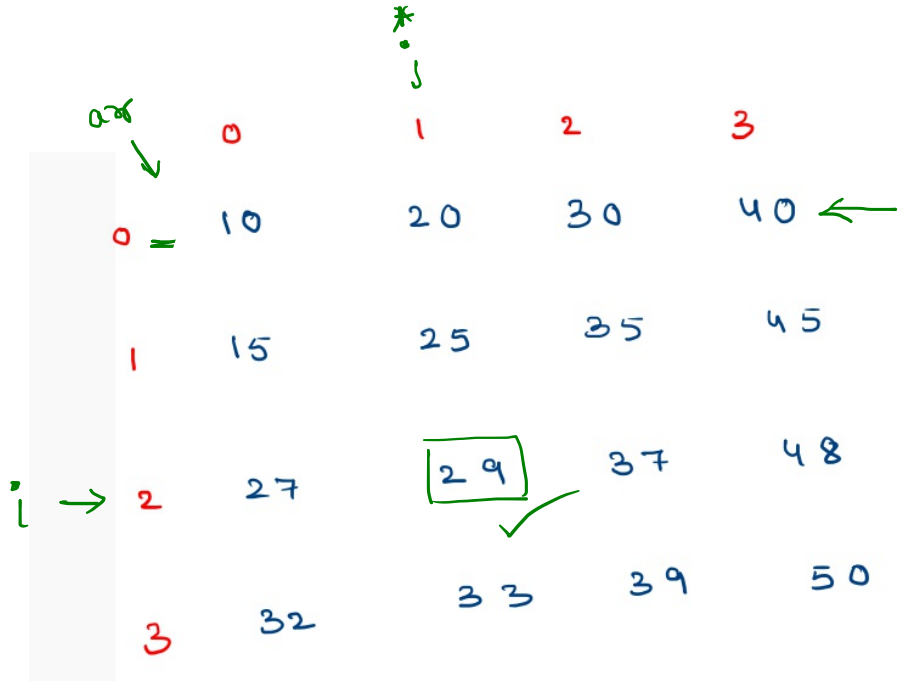
CS = 0 -2 → -3 → -5

Search an element in row-wise, column-wise sorted

Input: mat[4][4] = { {10, 20, 30, 40},
{15, 25, 35, 45},
{27, 29, 37, 48},
{32, 33, 39, 50}};
x = 29
Output: Found at (2, 1)
Explanation: Element at (2,1) is 29

Input : mat[4][4] = { {10, 20, 30, 40},
{15, 25, 35, 45},
{27, 29, 37, 48},
{32, 33, 39, 50}};
x = 100
Output : Element not found
Explanation: Element 100 is not found

	0	1	2	3
0	10	20	30	40
1	15	25	35	45
2	27	29	37	48
3	32	33	39	50



row
col
} sorted

key = 29

→ =/>/<

arr[i][j] V/s key

key < arr[i][j]

j--

key > arr[i][j]

i++

key == arr[i][j]

return

O(n*m)

↳

O(m+n)

```
search(int[][] mat,int  n, int  x)
{
     int  i = 0, j = n - 1;
    while  (i < n && j >= 0)
    {
        if  (mat[i][j] == x)
        {
            print("element found");
            return;
        }
        if  (mat[i][j] > x)
            j--;
        else
            i++;
    }
    print("element is not found");
    return;
}
```

⇒ n x n