Day- 4

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

even ?

2 k + 1 ⇒ odd

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

Output: 7

hm

K	V
7	1 → 1
3	2
5	2
4	2

n/2

① BF → O(n²) T.C

O(1) S.C

② Sorting → O(nlogn) T.C

O(1) S.C

③ K + V → O(n) T.C

O(n) S.C

2 + 3 - 5 - 3 + 5 - 2 = 0

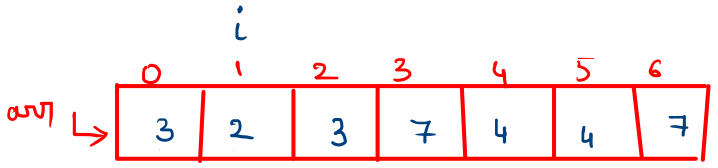
Note:-

1. $x \oplus x = 0$
2. $0 \oplus x = x$
3. $x \oplus y = y \oplus x$

$x, y \in W$ ↳ $\{0, 1, 2, \ldots\}$

$\Rightarrow O(n)$ T.C

$O(1)$ S.C



$\Rightarrow$ O/p (2)

```
res = arr[0]
for(i=1; i<n; i++)
{
    res = res ^ arr[i]
}
return res
```

$res = 3 \wedge 2 \wedge 3 \wedge 7 \wedge 4 \wedge 4 \wedge 7$

$= 0 \wedge 2$

$= (2)$

② even / odd [without using +, −, %, /]

MSB ↓ LSB

0 ⇒ 0 0 0 0

⇒ 1 → 0 0 0 **1**

2 → 0 0 1 0 ✓

⇒ 3 → 0 0 1 **1**

4 → 0 1 0 0 ✓

⇒ 5 → 0 1 0 **1**

6 → 0 1 1 0 ✓

⇒ 7 → 0 1 1 **1**

even 10 8 1

10 → 1 0 1 0

& 0 0 0 1

0 0 0 0

odd →

9 → 1 0 0 1

& 0 0 0 1

0 0 0 1

12 → 1 1 0 0

& 0 0 0 1

0 0 0 0

13 → 1 1 0 1

0 0 0 1

0 0 0 1

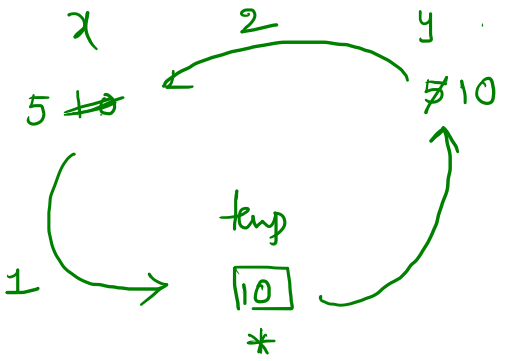
```
if( n & 1 == 1)
    print(" odd")
else
    print(" even")
```

③ Swap 2 numbers, without using 3rd variable.

x = 10
y = 5

code ⇒

x = 5
y = 10



+, −, *, %

^, +, •

Alg-1

a = a + b

b = a − b

a = a − b

a	b
~~5~~	~~10~~
~~15~~	5
10	

Alg-2 [^]

1) a = a ^ b

2) b = a ^ b

3) a = a ^ b

a	b
~~5~~	~~10~~
5 ^ 10	5 ^ 10 ^ 10
~~5~~ ^ 10 ^ ~~5~~	5
10	

a	b
10	5

11:45

D - B

12 → ( )$_2$

2 | 12
2 | 6 — 0
2 | 3 — 0
1 — 1

$(1100)_2$ → Binary rep

37 ⇒

2 | 37
2 | 18 — 1
2 | 9 — 0
2 | 4 — 1
2 | 2 — 0
1 — 0

$(100101)_2$ →

cross-check:-

2^5	2^4	2^3	2^2	2^1	2^0
1	0	0	1	0	1

⇒ $(1*32)+0+0+(1\times4)+0+(1\times1)$

$= 37$

39 →

32 + 7

↳ 4 + 2 + 1

32	16	8	4	2	1
1	0	0	1	1	1

$\Rightarrow (100111)_2$

97 →

↓

64 + 33

↳ 32 + 1

64	32	16	8	4	2	1
1	1	0	0	0	0	1

$\Rightarrow (1100001)_2$

*

If A, B are boolean variables

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

+ : Bitwise OR (|)

· : " AND (&)

⊕ : " ex-OR (^)

↑ programatically represent

13 ⊕ 14 = ?

13 : 1 1 0 1

14 : 1 1 1 0 ⊕

0 0 1 1 ⇒ $(0011)_2$ → 3

Ex1: 13 ⊕ 0

1 1 0 1

⊕ 0 0 0 0

1 1 0 1 → 13

Ex2 13 ⊕ 13 = 0

1 1 0 1

⊕ 1 1 0 1

0 0 0 0 → 0

Note: ① $x \oplus x = 0$ ② $x \oplus 0 = x$

$x \in N$

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

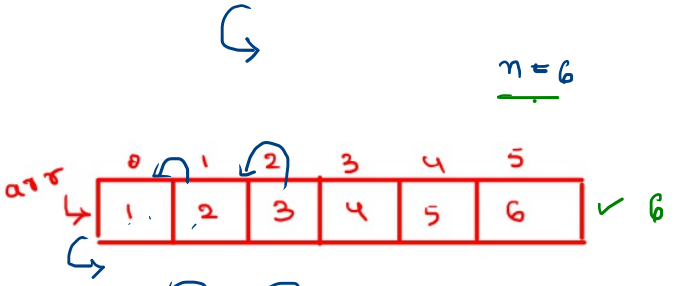
input:-

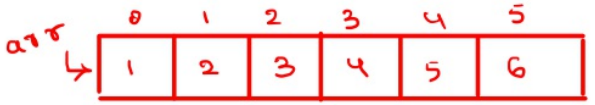
arr[6]= { 1,2,3,4,5,6 } and d=2

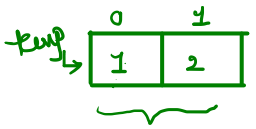
output :-

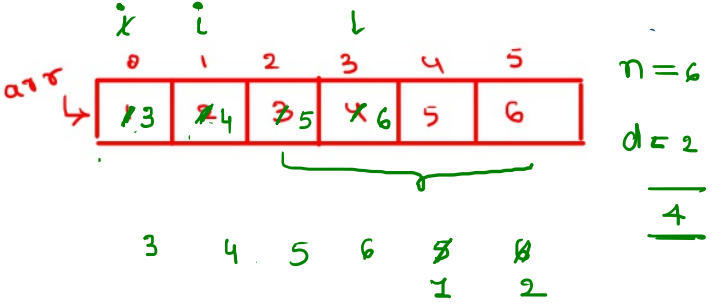
d=1 ==> 2,3,4,5,6,1

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




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





Brute-Force

* 1) take temp array of size d
* 2) copy first d elements of given array into temp array → loop (d)
* 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)

$d = n \% d$

$d < n$

$d + n - d + d = n + d$

$\Rightarrow O(n)$ T.C

$\Rightarrow O(d)$ S.C

Multiple-sub-arrays-reverse

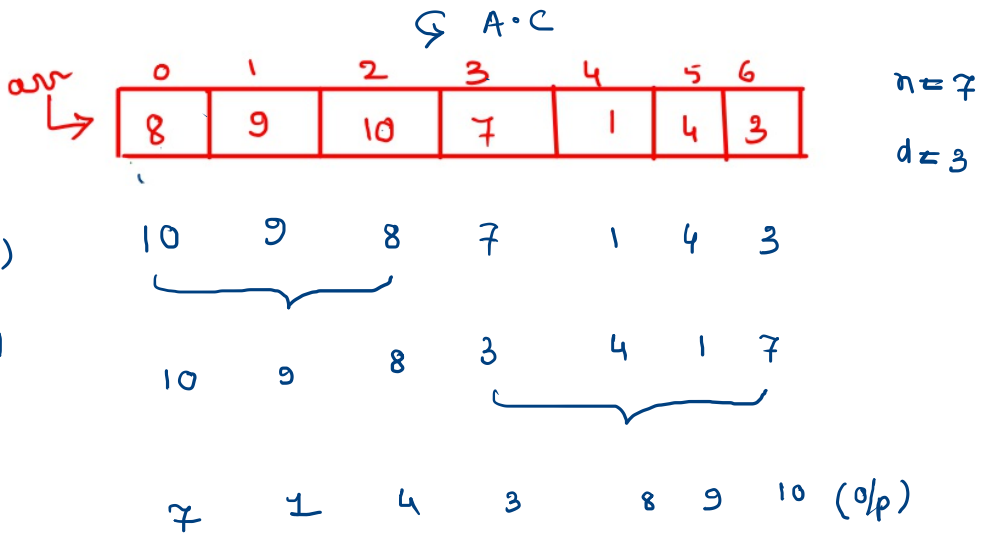
⇒ 1) reverse(arr,0,d-1) : 0 to 2 ⇒ O(d)

⇒ 2) reverse(arr,d,n-1) : 3 to 6 ⇒ n-d

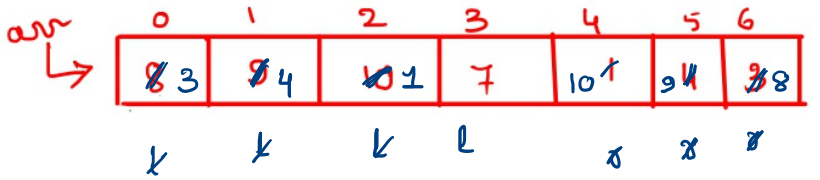
⇒ 3) reverse(arr,0,n-1) : 0 to 6 ⇒ n

O(n) T.C

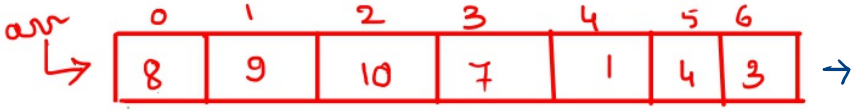
O(1) S.C



* Reverse the array



3 4 1 7 10 9 8



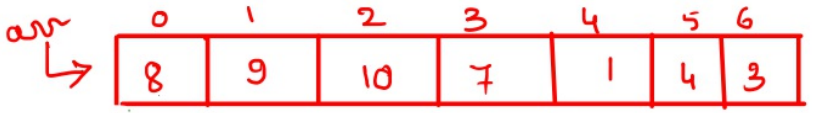
AP1

```
rev[] =
i ← n-1 to 0
rev.push(arr[i])
```

AP2

2-ptr

O(n)


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

Home work : please do for same problem, clock wise

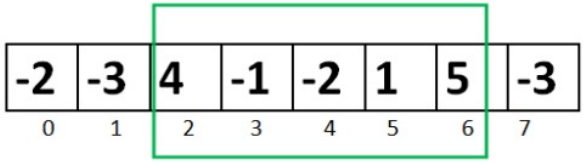
arr[6] = { 1,2,3,4,5,6} and d=2

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.

Largest Subarray Sum Problem



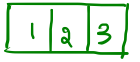
4 + (-1) + (-2) + 1 + 5 = 7

Maximum Contiguous Array Sum is 7

BF (n)

↳ generate all sub-arrays

n = 3



len(1) = 3 [1] [2] [3]

len(2) = 2 [1 2] [2 3]

len(3) = 1 [1 2 3] [1 3] X

1 + 2 + 3 = 6

n = 4

1 + 2 + 3 + 4

General:

1 + 2 + 3 + ... + n

$= \frac{n(n+1)}{2}$ ✔

Brute-Force

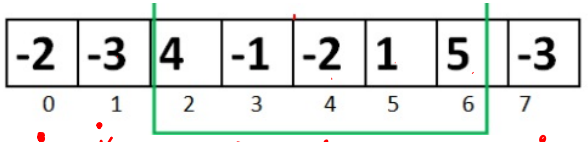
↳ Generate All Sub-Arrays $\left[\frac{n(n+1)}{2}\right] \simeq O(n^2)$

↳ find out each & every sub-Array sum $\Rightarrow O(n)$

$n^2 * n \Rightarrow O(n^3)$ T.C

$O(1)$ S.C

*Kadane's Algo



-2 > -∞ ✓

-3 > -2 ✗

Sum = ~~-2~~ 4 7

Cum-sum = 0 ~~-3~~ 0 4 ~~3~~ 1 2 7 4

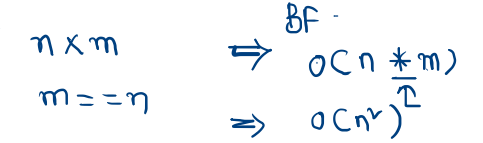
Algo

```
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 ✓
}
```

T.C : O(n)

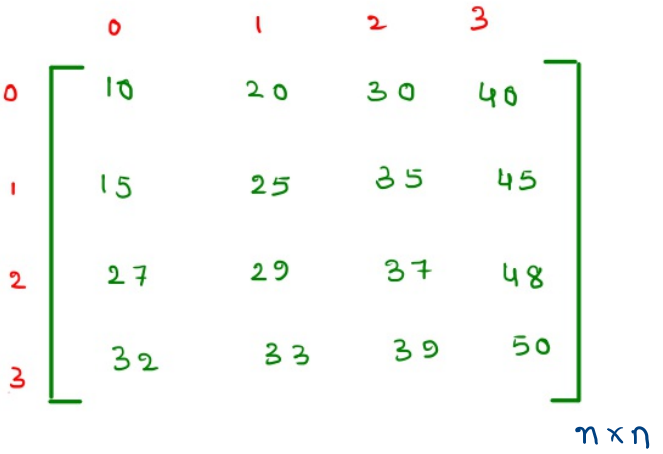
S.C : O(1)

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

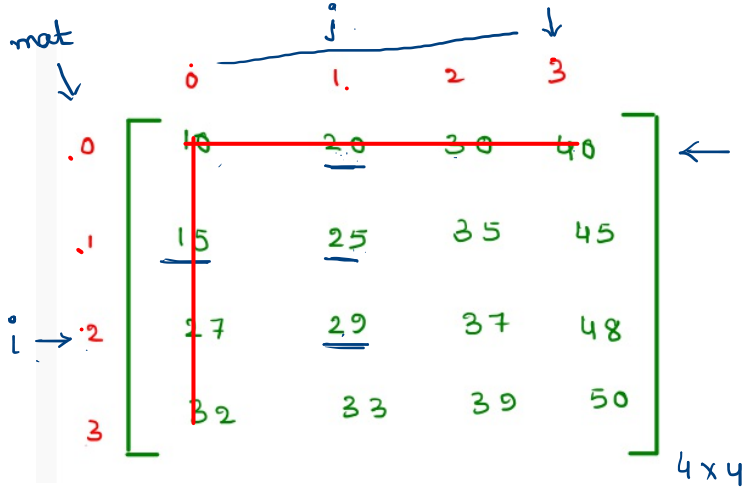



```
Input: mat[4][4] = { {10, 20, 30, 40},
                     {15, 25, 35, 45},
                     {27, 29, 37, 48},
                     {32, 33, 39, 50}};
               x = 29
Output: Found at (2, 1)
Explanation: Element at (2,1) is 29

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
```



Let $n \times m$



Key = 29

i = 0, j = 3

mat[i][j] == key C_1

print(i, j)
return;

mat[i][j] > key C_2

j--

mat[i][j] < key C_3

i++

let Mat → square

AP_1 - n rows, n col } $O(n^2)$ T.C

AP_2 $2n \Rightarrow O(n)$ T.C

$O(n+m)$ [T.C]

$O(1)$ [S.C]

square_matrix

```
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;
}
```

x → key

C1 C2 C3