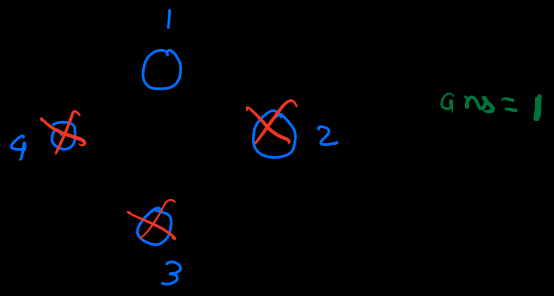
1. Josephus Problem
2. K^{th} symbol - Hard
3. Valid Sudoku
4. Count Subarray with sum ZERO

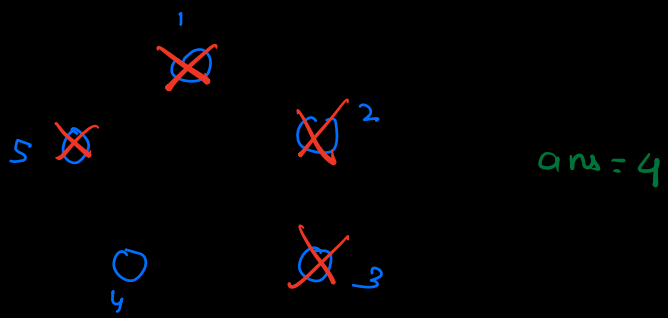
Q1 Josephus Problem.

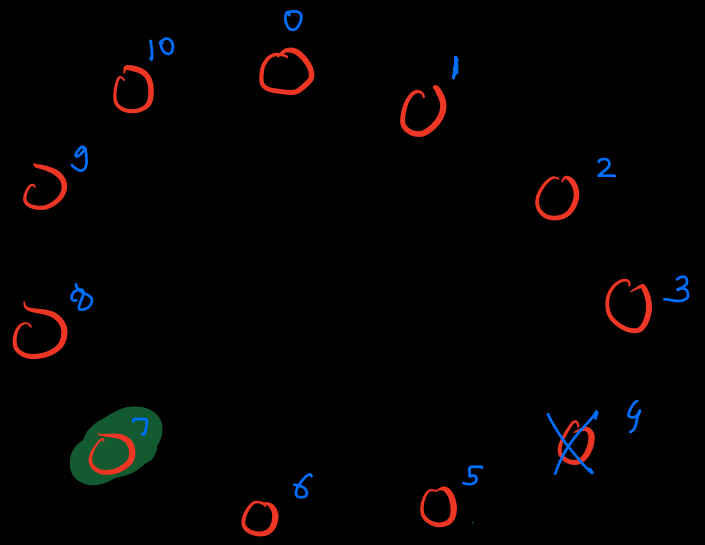
N person, kill every K^{th} person. Last man standing

Ex N = 4
K = 2



N = 5
K = 3





11 person

K = 5

Josephus (10,5) = 2

N - 1 person

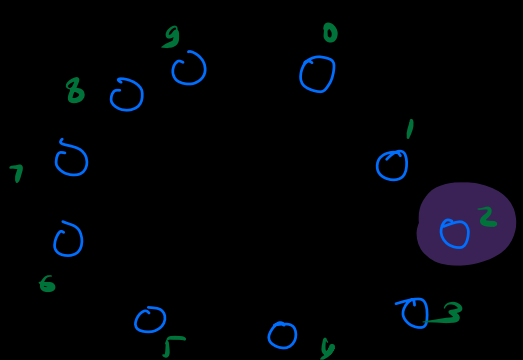
K^{th}

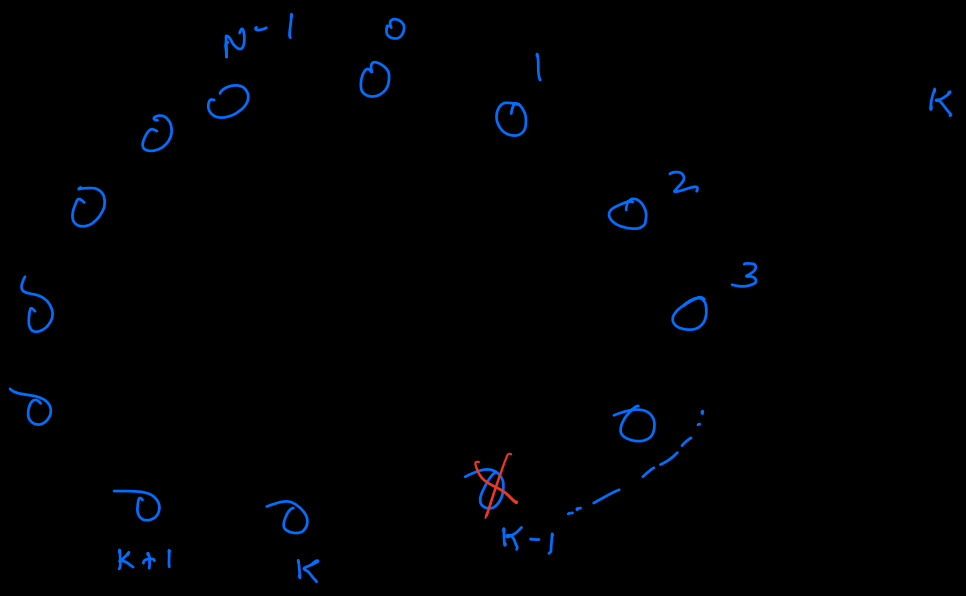
Start point 5,

↳ which index will left out if I start from 0. and kill every K^{th} person

```
int Josephus ( int N, int K)
{
```

Jos(10,5) = 2





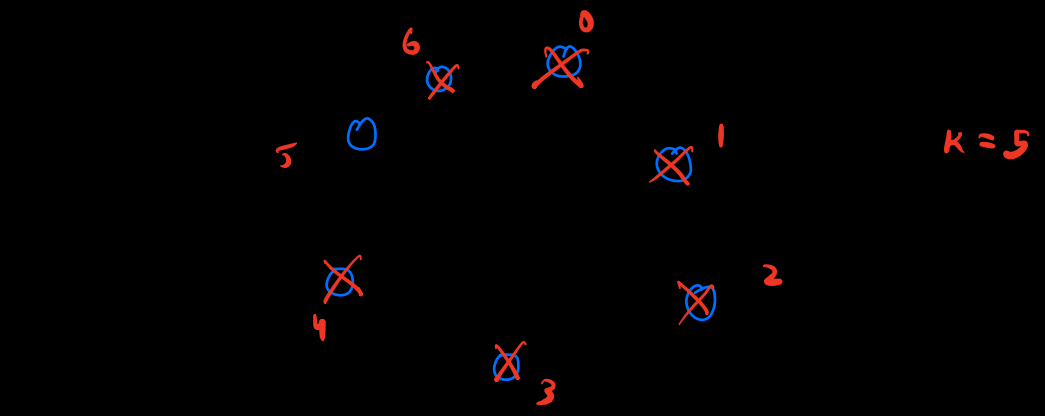
Josephus (N-1, K) = x

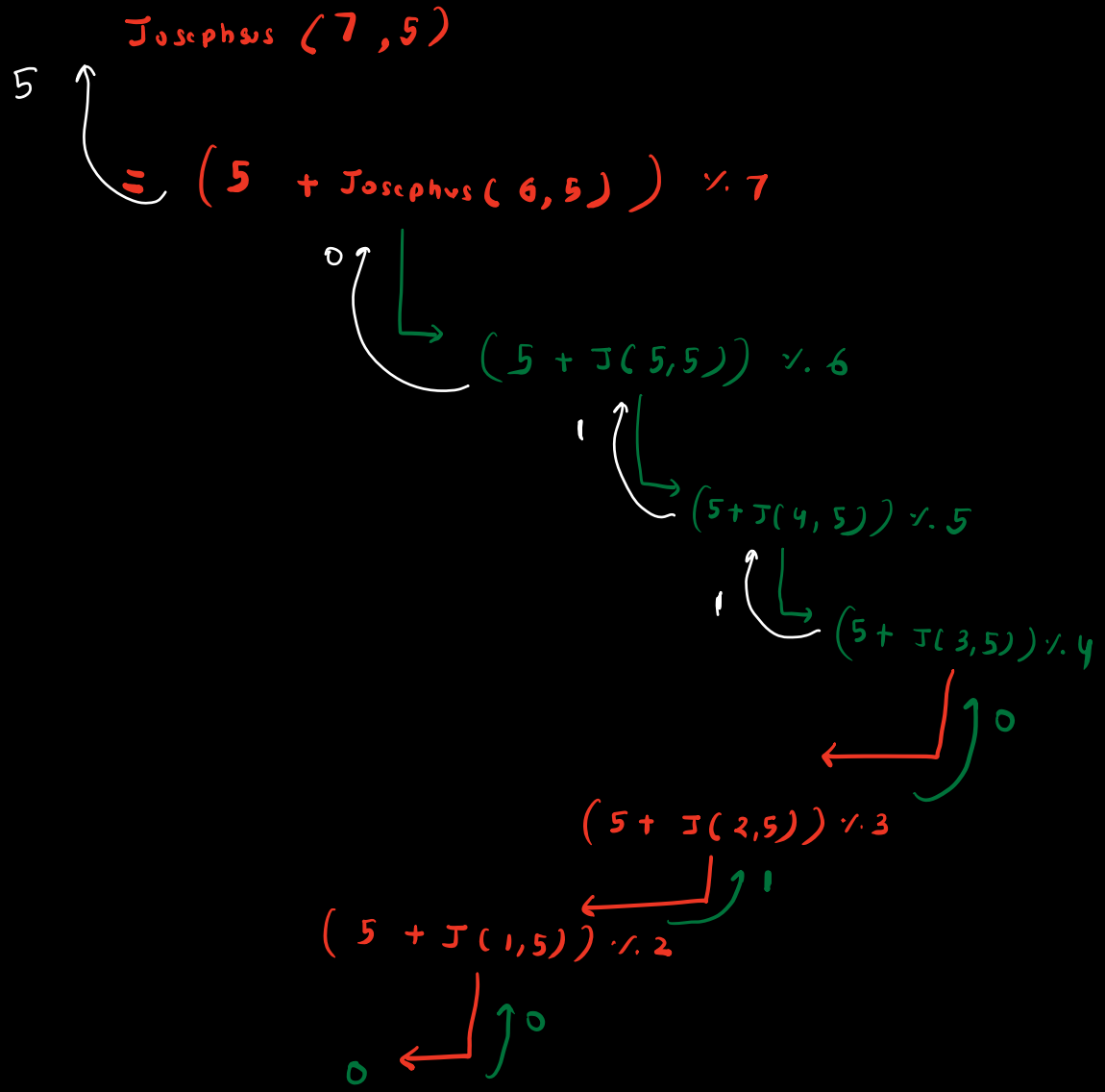
```
int Josephus(int N, int K)
{
    if (N==1) return 0

    int x = Josephus(N-1, K)

    return (x+K) %N
}
```

TC: O(N)

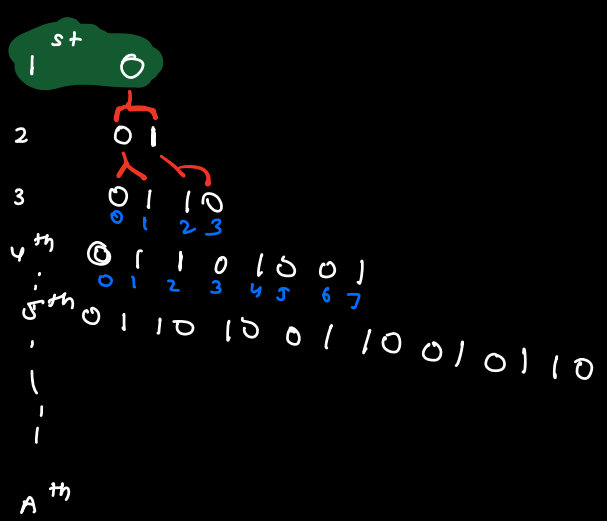

6
0
5
0
1
K = 5
4
2
3


Josephus (7, 5)
5
= (5 + Josephus(6,5)) %. 7
0
(5 + J(5,5)) %. 6
1
(5 + J(4,5)) %. 5
1
(5 + J(3,5)) %. 4
0
(5 + J(2,5)) %. 3
1
(5 + J(1,5)) %. 2
0
0

```
int Josephus(n, k)
{
    =
    =
```

```
main(n, k)
{
    ans = Josephus(n, k)
    return ans+1
```

Q.



0 → 01
1 → 10

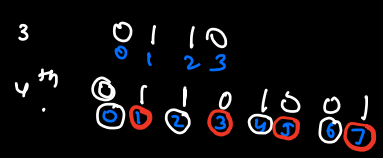
A=4
B=0 ans=0

A=4
B=2 ans=1

A=4
B=5 ans=0

A=3
B=1 ans=1

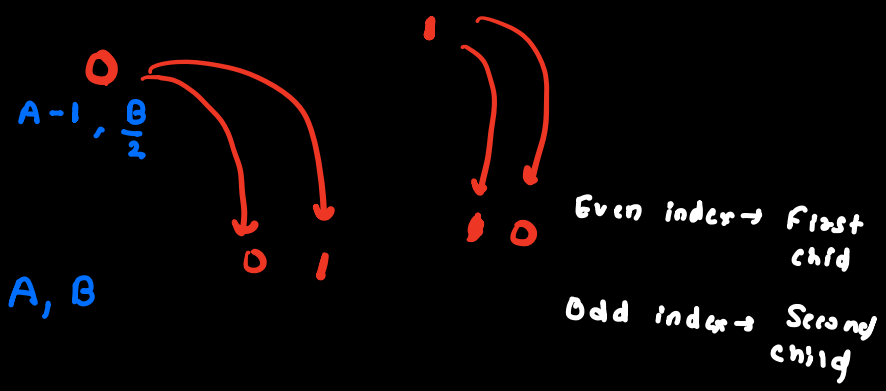
A, B B^{th} index in A^{th} row



child	Parent
0	0
1	0
2	1
3	1

par[i] = $\frac{i}{2}$

4	2
5	2
6	3
7	3



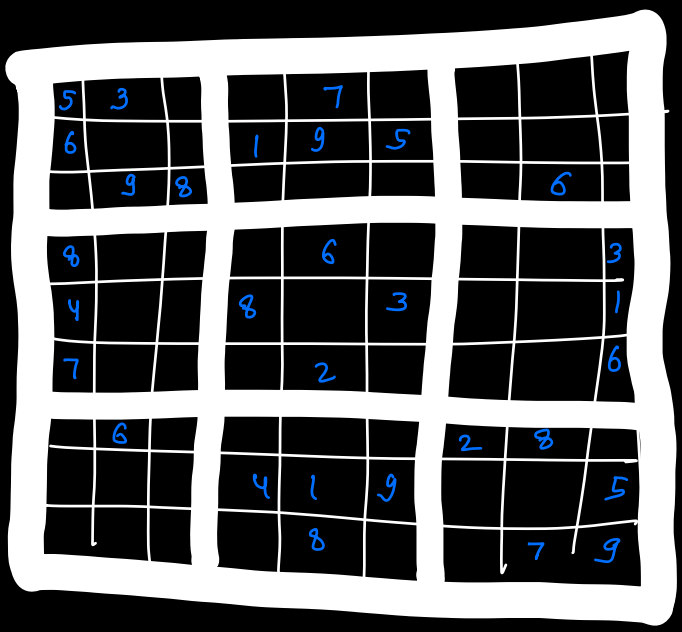
A^{th} row B^{th} index

```
int Kth index ( A , B )
{
    if ( A == 1 )  return 0

    int parvalue = Kth index ( A-1, B/2 )

    if ( B % 2 == 0 )        if ( B & 1 == 0 )
    {
        return parvalue
    }
    else
    {
        return 1 - parvalue
    }
}
```

Q.



Same no should not be → 1. same row → hashset
2. same col → hashset
3. 3x3 box

If 5 present in 4th row

hmrow[4].contains(5)

hashset <int> hsrow[9]

hashset <int> hscol[9]

hashset <int> hs3x3[9]

TODO

how you will check this

```
for( i=0 ; i<9 ; i++)
{
    for(j=0 ; j<9 ; j++)
    {
        if( ith row have arr[i][j] OR
            jth col have arr[i][j] OR
            corresponding 3x3 box have arr[i][j])
            return false
    }
}
```

return True

TC : O(N×m)

TC : O(9×9)

TC : O(81)

TC : O(1)

Q. Count subarrays with zero sum.

	0	1	2	3
A = [	1	-1	-2	2]

[0,1]

[2,3]

[0,3]

ans = 3

[i, j] = pre[j] - pre[i-1] = 0

pre[j] = pre[i-1]

	0	1	2	3	4	5	6
Prefixsum [	3	4	3	2	5	3	4]
	×	×	1	×	×	2	1

1 + 2 + 1 = 4.

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

1 + 2 + 2 + 1 = 6

frequency

hashmap < int, int > hm

mp[0] = 1

sum = 0

for(i = 0; i < N; i++)

{

sum = sum + arr[i]

ans = ans + hm.get(sum)

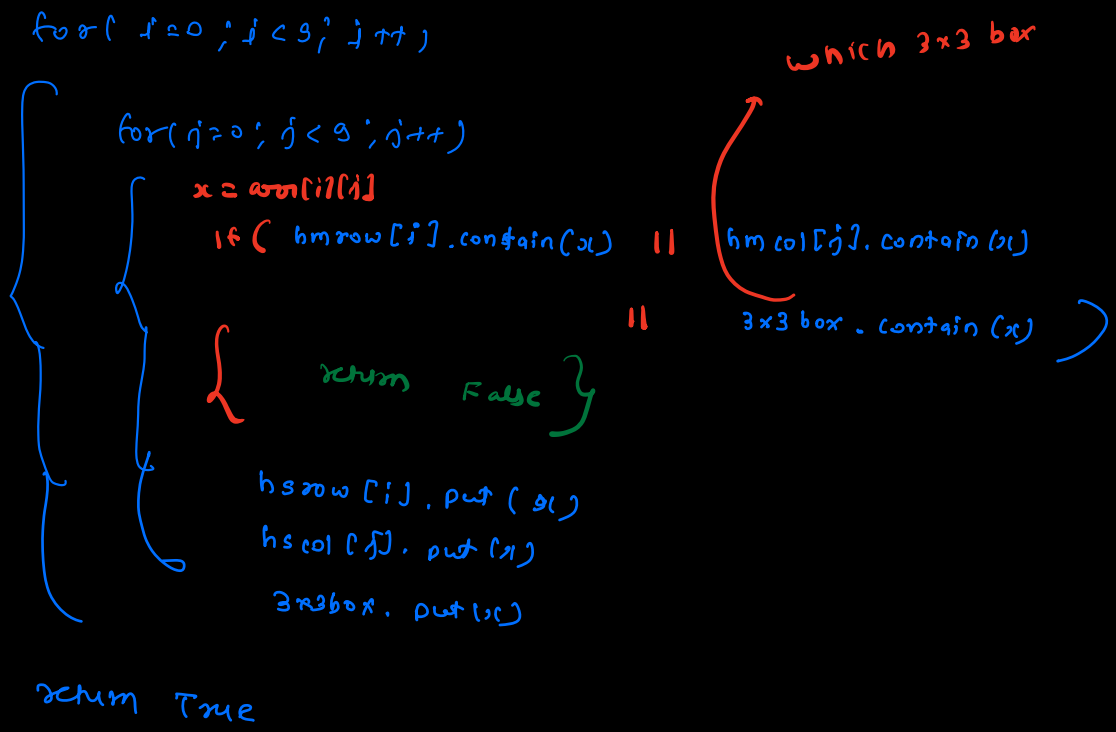
update sum freq. by 1

}

return ans.

TC: O(N)

hm.put(sum, hm.get(sum) + 1)


for(i=0 ; i<9 ; i++)
for(j=0 ; j<9 ; j++)
x = arr[i][j]
if (hmrow[i].contain(x) || hmcol[j].contain(x)
|| 3x3 box . contain (x)
which 3x3 box
return False
hsrow[i].put(x)
hscol[j].put(x)
3x3box.put(x)
return True