July 06, 2025

DOUBT SESSION

#1. Power of Two : Whether a number is a power of 2 or not.

Sol 1.

	Binary
1	1
2	10
4	100
8	1000
16	10000

0	0
1	1
3	11
7	111
15	1111

16 && 15 = 0

If a number n follows the following relation:

n && n-1 == 0

Then, we can say that n is a power of 2. otherwise no.

⇒ O(1)

Or,

Sol 2.

```
while (n%2==0) n=n/2
if (n==1) return 1
return 0
```

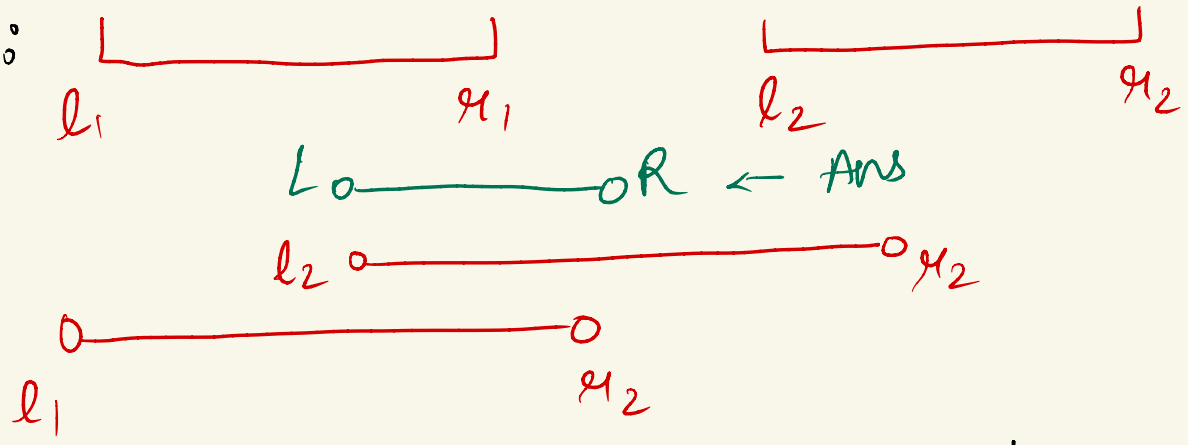
⇒ O(log n)

Or,

Sol 3. If there only 1 one in the binary representation of n then it can be said as a power of 2.

#2 Two Intervals : finding intersection

Given:



Soln approach

Max (l_1, l_2) ⇒ L Starting point

Min (R_1, R_2) ⇒ R Ending point

if (Starting point <= ending point)

intersection exist

L & R are valid.

else

L & R are invalid

#3 GCD

Use in-built STL fxn to find gcd of a & b.

⇒ __gcd(a, b)

Or,

Use Euclidean GCD Algorithm

#4 Lucky Numbers : a no consisting only of 4 & 7

```
int lucky(int x)
{
    while (x)
    {
        int digit = x%10
        if (digit != 4 && digit != 7) return 0;
    }
    return 1
}
```

check every digit of the number using modulo 10 and stop as soon as you encounter anything other than 4 & 7.

#5 Sum of odd numbers b/w X & Y : O(1) soln

If x is even , start from x+1

If x is odd , start from x+2

If y is even , ending should be y-1

If y is odd , ending will be y-2.

→ S<=ending no

$S_n = \frac{n}{2}[a+l]$ AP

1/2

```
int x,y;
cin>>x>>y;

int starting=min(x,y),ending=max(x,y);
if(x%2)starting+=2;
else starting+=1;
if(y%2)ending-=2;
else ending-=1;

if(starting<=ending)
{
    int n=((ending-starting)/2)+1;
    int ans= (n*(starting+ending))/2;
```

starting
ending

2/2

```
    if(starting<=ending)
    {
        int n=((ending-starting)/2)+1;
        int ans= (n*(starting+ending))/2;
        cout<<ans<<endl;
    }
    else
    {
        cout<<0<<endl;
    }
}
signed main()
```

AlgoZenith

#6 Three Numbers : O(K) solution

$x + y + z = S$

$0 \le x, y, z \le K$

$x + y = S - z$

$T = S - z$

$x + y = T$

Range of T : $S - K \le T \le S$

$\therefore 0 \le T \le 2K$

Ex:

- K = 8, $0 \le x, y \le 8$

$x + y = 7$

$0 + 7 = 7$
$1 + 6$
$2 + 5$
$3 + 4$
$4 + 3$
$5 + 2$
$6 + 1$
$7 + 0$

8 solutions

- $x + y = 3$

$0 + 3$
$1 + 2$
$2 + 1$
$3 + 0$

4 soln

- $x + y = 15$

$8 + 7$
$7 + 8$

2 soln

- $x + y = 10$

$10 - 8 = 2$

$8 + 2$
$7 + 3$
$6 + 4$
$5 + 5$
$4 + 6$
$3 + 7$
$2 + 8$

going till 2

going till 8

7 soln

The sum is T and Range of T is $0 \le T \le k$

when sum $\le k$, $T \le k$

no. of solution = T+1

when sum > k

diff ↑ sum

Starting range: $X \le k$, $Y = T - k$.

no. of soln = k - diff + 1 (use prev examples to understand this)

```
void solve()
{
    int k, s;
    cin >> k >> s;

    /*
    x+y+z=s;
    0<=s<=3k
    x+y=s-z;
    x+y=T
    max(0,s-k)<=s-z<=s
    */

    int ans = 0;
    for (int i = max((int)0, s - k); i <= s; i++)
    {
        if (i <= k)
        {
            ans += i + 1;
        }
        else if(i<=2*k)
        {
            int d = i - k;
            ans += (k - d + 1);
        }
    }
    cout << ans << endl;
}
signed main()
{
```

if $T \le k$

↳ T+1

else

diff = T - k

ans = k - diff + 1

#7 Another Pyramid Problem

N=4

	1	2	3	4	5	6	7
1	–	–	–	1			
2	–	–	1	2	1		
3	–	1	2	3	2	1	
4	1	2	3	4	3	2	1

row = n = 4

col = (2*n) - 1

= 7

```
void solve()
{
    int n;
    cin >> n;
    int row = n, col = (2 * n) - 1;
    for (int i = 1; i <= row; i++)
    {
        for (int blank = 0; blank < n - i; blank++)
        {
            cout << "* "; // spaces
        }
        for (int num = 1; num <= i; num++)cout << num << " ";
        for (int num = i - 1; num >= 1; num--)cout << num << " ";
        for (int blank = 0; blank < n - i; blank++)
        {
            cout << "* "; // after space
        }
        cout << endl;
    }
}
```

Spaces to print for every row : n - i

row = 4

1 → 3 → n - 1

4 - 1

⇒ 3

2 → 2 → n - 2

4 - 2

⇒ 2