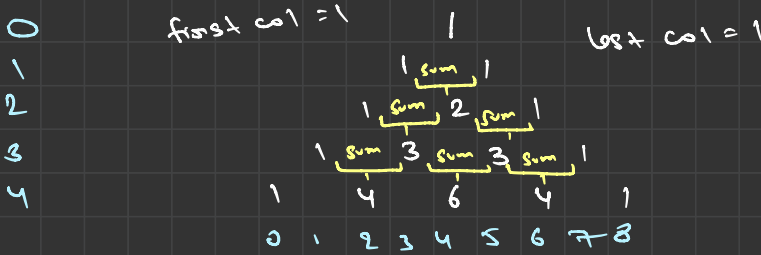



Row

Pascal Δ



There are three questions

① Give R and C tell the element at that place

$$\therefore R=5 \quad C=3$$

$$\text{ans} = 6$$

② Print any n^{th} row of Pascal Δ
 $\therefore n=4 \Rightarrow 1 \ 3 \ 3 \ 1$

③ Draw complete Pascal Δ

① formula $\Rightarrow R-1$
 for $R=5$ $C=3$ $C-1$

$5-1$ \Rightarrow 4 C
 $3-1$ \Rightarrow 2

$n C r \Rightarrow \frac{n!}{r!(n-r)!} \Rightarrow \frac{4!}{2!(4-2)!} \Rightarrow \frac{4!}{2! \times 2!}$

$\Rightarrow \frac{4 \times 3 \times 2 \times 1}{2 \times 1 \times 2 \times 1} = 2 \times 3 \Rightarrow 6$

using optimizing method

fun nCr (n-1, c-1)

① use loop for $n!$, $r!$, $(n-r)!$

② use optimizing method $\Rightarrow TC \Rightarrow O(r) + O(n) + O(n-r)$



optimizing \Rightarrow don't use for loop / brute force

$7 C 2 \Rightarrow \frac{7!}{2!(7-2)!} \Rightarrow \frac{7!}{2! \times 5!} \Rightarrow \frac{7 \times 6 \times 5!}{2 \times 1 \times 5!}$

$10 C 3 \Rightarrow \frac{10!}{3! \times 7!} \Rightarrow \frac{10 \times 9 \times 8 \times 7!}{3 \times 2 \times 1 \times 7!}$

$n(5) \Rightarrow 10(3) \Rightarrow \frac{10 \times 8 \times 7}{3 \times 2 \times 1}$
 fun HCR (n, n)
 {
 res = 1;
 for (i = 0; i < n; i++)
 {
 res = res * (n - i);
 res = res / (i + 1);
 }
 return res;
 }

because we know
 loop will run
 n times

or we can
 say that

$\frac{10 \times 8 \times 7}{1 \times 2 \times 3}$
 \downarrow
 $\frac{10}{1} \times \frac{8}{2} \times \frac{7}{3}$

10 is basically $n \rightarrow n - 0 \rightarrow n - i$
 9 $\rightarrow n - 1 \rightarrow n - i$
 8 $\rightarrow n - 2 \rightarrow n - i$

becoz for $i = 0$ we take 1
 $i = 1 \rightarrow 2$
 $i = 2 \rightarrow 3$

Time complexity $\rightarrow O(n)$
 Space complexity $\rightarrow O(1)$

② N row $\rightarrow n$ elements

we know
that

$arr[c-1]$

```
for (c = 1; c <= n; c++)  
    print (funNcr (n-1, c-1));  
}
```

Time complexity = $O(n \times n)$

```

num = 11;
num = num - n;
for (row = 0; row < n; row++)
{
    for (col = 0; col < n - row; col++)
    {
        cout << num++;
    }
    num--;
    num = num - 2 * (n - 1 - row);
    cout << endl;
}

```

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

$n = 4$

count = 10

$$10 - 5 = 5$$

