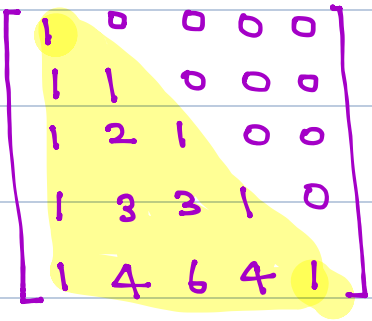
Lab Session on Prime Numbers & two pointers

We will Start at 7:05 AM

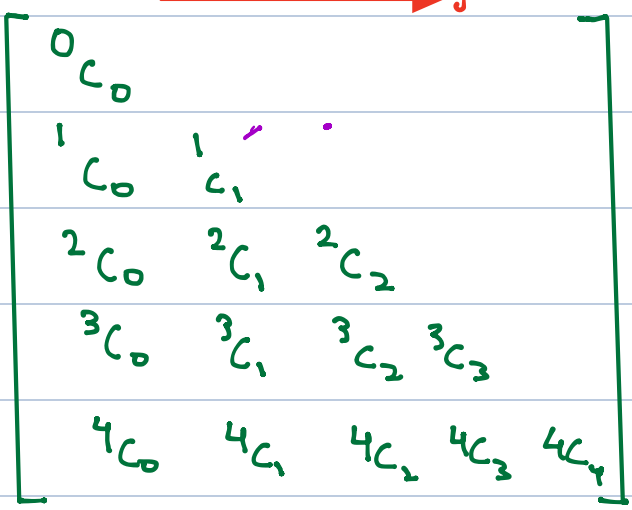
① Pascal's triangle

Write a program to print the pascal triangle up to N rows.

N = 5



→ Considering 0 based indexing -



$i \rightarrow {}^{i}C_{0}\ \ {}^{i}C_{1}\ \ {}^{i}C_{2} \ldots\ {}^{i}C_{i}$

Pascal Triangle

$O(n) =$

$$^{n}C_{r} = \frac{n!}{r!\,(n-r)!}$$

$$\frac{n*(n-1)*(n-2)*\cdots*1}{r!*(r-1)!\cdots *}$$

Approach 1 → Keep calculating $^{i}C_{j}$ for every position.

T.C → $O(n^3)$

→ n^2 items × $O(n)$ T.C for calculating answer for each position

Approach 2:

$$^{n}C_{r} = {}^{n-1}C_{r-1} + {}^{n-1}C_{r}$$

n ↓ r ↓

$i = 3, \quad j = 2$

Find → 3rd row, 2nd col.

→ $^{3}C_{2} = {}^{2}C_{1} + {}^{2}C_{2}$

$$^{n}C_{r} = {}^{n-1}C_{r-1} + {}^{n-1}C_{r}$$

```
arr[i][j] = arr[i-1][j-1] + arr[i-1][j]
```

Pseudo Code-

```
mat[N][N]
for( i → 0 to N-1) {
    for( j → 0 to i) {
        if(j==0 || j==i) {
            mat[i][j] = 1
        } else {
            mat[i][j] = mat[i-1][j-1] + mat[i-1][j]
        }
    }
}
```

T.C → $O(N^2)$

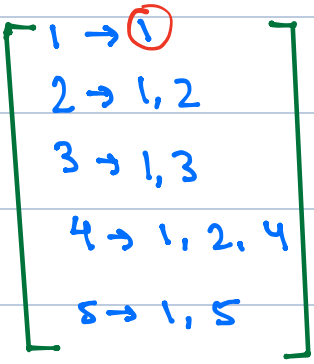
S.C → $O(N^2)$

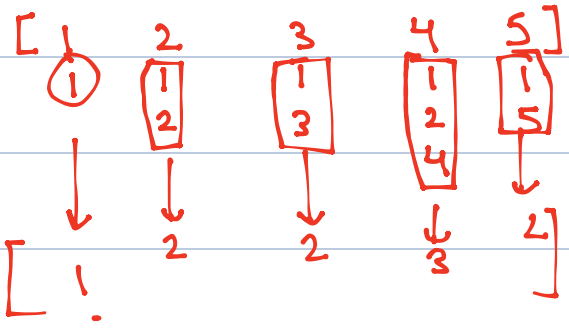
Count of Divisors -

Given N, return the count of divisors of all numbers till N.

N = 5

Ans: [1, 2, 2, 3, 2]





Approach 1: Find factors of each number.

↳ $O(\sqrt{N})$ → For each number.

T.C → $O(N\sqrt{N})$

Code:

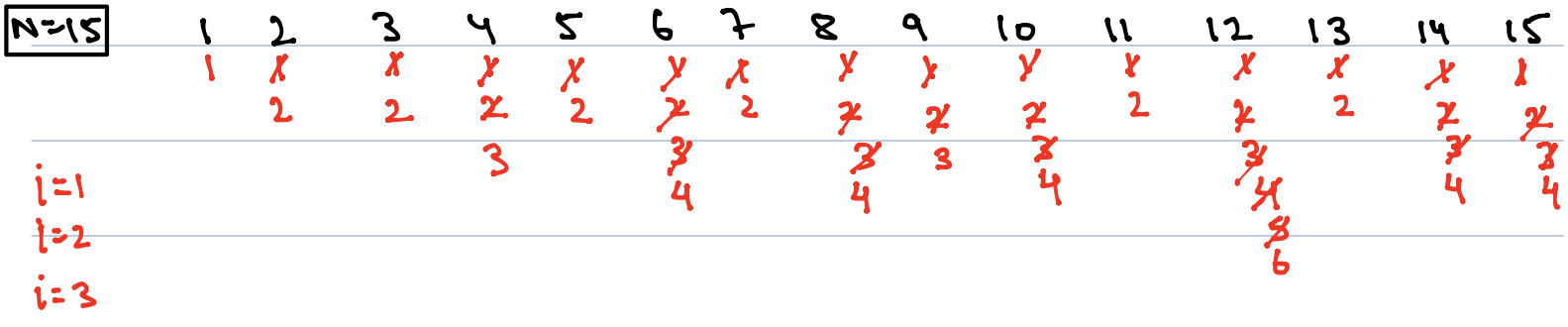
```
int countDivisors(int n) {
    count = 0
    for (i = 1; i*i <= n; i++) {
```

```
        if ( n%i ==0) {
            if (i = n/i)  count++;
            else count+=2;
        }
    }
    return count;
}

for ( i → 1 to N) {
    print ( countDivisors (i));
}
```

Count of factors

Approach 2:



Code:

```
ans[N+1];                                   → N x
for ( i →1 to N) {                          → log N
    for ( j=i ; j<=N; j+=i){                → Updating the count of
```

ans[j] ++;

}

}

return ans[]

factors

- Time Complexity = ?

1 → 1, 2, 3, 4, 5, 6 ... 15 = N

2 → 2, 4, 6, 8, 10, 12, 14 = N/2

3 → 3, 6, 9, 12, 15 = N/3

4 → 4, 8, 12 = N/4

Count the iterations. →

$$\left[\overset{1}{N} + \overset{1}{\frac{N}{2}} + \overset{1}{\frac{N}{3}} + \overset{1}{\frac{N}{4}} + \overset{1}{\frac{N}{5}} \cdots \overset{1}{\frac{N}{N}} \right]$$

$$= N\left[\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} \cdots \frac{1}{N} \right]$$

$$= N \int_{1}^{N} \frac{1}{x} dx \qquad = N \log N$$

T.C → O(N log N)

Check Pair Sum:

Given an Array of integers B, and a target sum A.

Check if there exists a pair (i,j) such that Bi + Bj = A and i!=j.

A = 8 :: B = [3, 5, 1, 2, 1, 2]

True

A = 21 :: B = [9, 10, 7, 10, 9, 1, 5, 1, 5]

False

Approach 1: Check all combinations of pairs

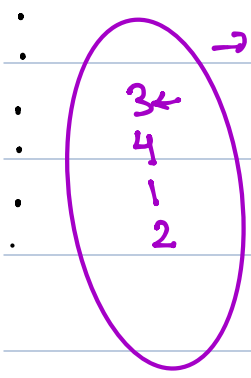
↳ Two nested loops iterating through all numbers.

↳ $O(N^2)$

```
for(i → 0 to n-1){
    for( j = i+1 → n-1){
        if ( A[i] + A[j] == sum) return true;
    }
}
return false;
```

Approach 2: Use Set.

A = 8 :: B = [3, 4, 1, 2, 5, 2]



→ Contains no. we have already seen.

T.C → O(N)

S.C → O(N)

Look → A - arr[i] in Set → return True.

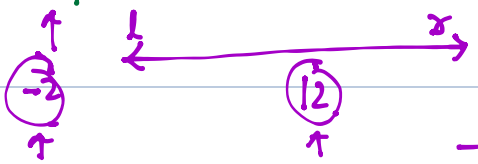
* Let's say that array is Sorted.

Approach 1:

↳ Binary Search

[-1, -2, [3, 7, 12, 20]]

target = 10



→ True

↳ To Search a no. in a Sorted array → log N

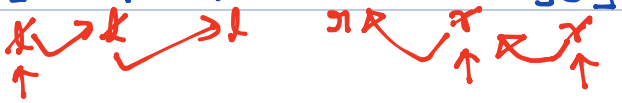
T.C → O(n log n)

* Approach 2: Two pointers.

Sorted →

arr → [2 4 12 15 21 30]

Sum = 27



arr[l] + arr[r] = 32

↳ = 23

↳ = 25

↳ = 33

↳ = 27 =

l = 0, r = n - 1

```
while ( l < r) {
    if ( A[l] + A[r] == target) return True;
    if (A[l] + A[r] < target) {
        l++;
    } else {
        r--;
    }
}
return False;
```

Time Complexity → O(N)

S.C → O(1)

Pairs with given Sum - II

Given a sorted array of integers (not necessarily distinct) A and an integer B, find and return how many pair of integers (A[i], A[j]) such that i != j have sum equal to B.

A = [1, 1, 1]

B = 2

Ans: 3

Examples: Target = 10

[1, 4, 5, 6, 7, 10] ans = 1

→ [1, 4, 4, 5, 6, 7, 10] ans = 2

→ [1, 4, 4, 5, 6, 6, 6, 7, 10] ans = 6

→ [1, 4, 4, 5, 5, 5, 5, 6, 6, 6, 7, 10] ans = 12

Map → Number → frequency.

Target = 10

Count = 6 + 6

1 → 1

4 → 2

5 → 4

6 → 3

7 → 1

10 → 1

= 6

+6

4 x 4 ✗

$^4C_2$ = 6

Code:

0 1 2 3 4 5 6 7 8 9 10 11
[1, 4, 4, 5, 5, 5, 5, 6, 6, 6, 7, 10]

i=0, j=N-1, count=0

```
while (i < j) {
    if ( arr[i] + arr[j] == target) {
        if (arr[i] == arr[j]) {
            x = j - i + 1;
            count += (x x (x-1))/2;
            break;
        }
        leftVal = arr[i] ; lcount =0;
        while ( arr[i]== leftVal) { lcount++   i++ }
        rightVal = arr[j] ;  rcount =0;
        while ( arr[j] == rightVal) { rcount++   j-- }
        count += lcount x rcount; ] x
    } else if ( arr[i] + arr[j] < target) {
        i++;
    } else {
        j--;
    }
}
return count;
```

$^{n}C_2$

↳ $\frac{n \times (n-1) \times (n-2)!}{2 \times (n-2)!}$

3

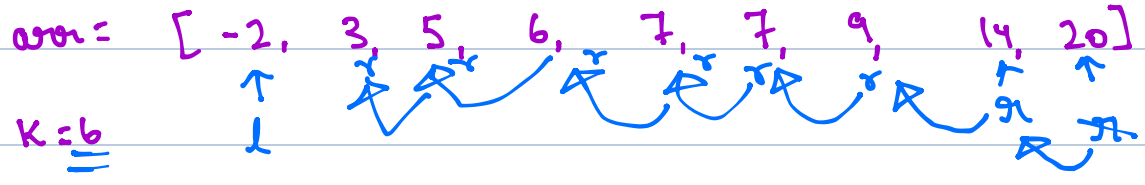
Time Complexity → O(n)

Space Complexity → O(n) ↑ Hashmap O(1) ↑ Without Hashmap

Pair Difference K -

Given sorted arr[] of N integers.

Check if there a pair with difference = K.



arr[j] − arr[i] → 22 = reduce

Ambiguity.

→ Answer Does not → 16 → 11

l → 0

r → end ✗

Exist

↳ 9
↳ 9

↳ 7
↳ 5

k = 6

arr = [-2, 3, 5, 6, 7, 7, 9, 14, 20]

l r r r r r r

arr[r] - arr[l] = 0
↳ 5
↳ 7
↳ 2
↳ 3
↳ 4
↳ 4
↳ 6

True : A pair exists
→ With both pointers at start.

Pseudo Code

```
i=0  j=1
while (j<N) {
    diff = arr[j] - arr[i]
    if (diff == abs(k)) {
```

```
            return True;
        } else if ( diff < k) {
            j++;
        } else {
            i++;
        }
    }
```

T.C → O(N)

S.C → O(1)

① → Brute force

② → Set (Same as above)

③ → B.S

④ → Two Pointers