Lab session on

Maths & Two pointers

Content

01. Pascal Triangle → Combinations property

02. Count Divisor → Prime no. logic

03. Check pair sum

04. Pair sum II

05. Pair Difference

} Two pointer

01. Given N, Generate pascal triangle for the no. N

N = 5

	0	1	2	3	4
0	1				
1	1	1			
2	1	2	1		
3	1	3	3	1	
4	1	4	6	4	1

	0	1	2	3	4
0	$^0C_0$				
1	$^1C_0$	$^1C_1$			
2	$^2C_0$	$^2C_1$	$^2C_2$		
3	$^3C_0$	$^3C_1$	$^3C_2$	$^3C_3$	
4	$^4C_0$	$^4C_1$	$^4C_2$	$^4C_3$	$^4C_4$

Brute force → For each & every position, calculate ^iC_j & store it.

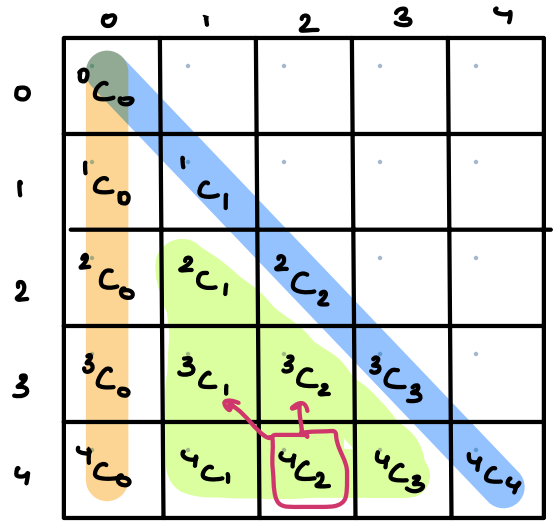
$$^iC_j = \frac{i!}{j!\,(i-j)!}$$

$5! = 120$

$10! = 362880$

$20! = 2.4 * 10^{18}$

} Factorials grows rapidly.



Combination property

01. $^{N}C_{0} = 1$

02. $^{N}C_{N} = 1$

03. $^{N}C_{r} = {}^{N-1}C_{r} + {}^{N-1}C_{r-1}$

$^{4}C_{2} = {}^{3}C_{2} + {}^{3}C_{1}$

```
int [][] ans = new int[A][A]

ans[0][0] = 1;
for (i=1; i<A; i++)
    for(j=0; j<=i; j++)
        if (j==0 || j==i) ans[i][j] = 1;
        else ans[i][j] = ans[i-1][j] + ans[i-1][j-1];
return ans;
```

TC : O(n²)

SC : O(1)

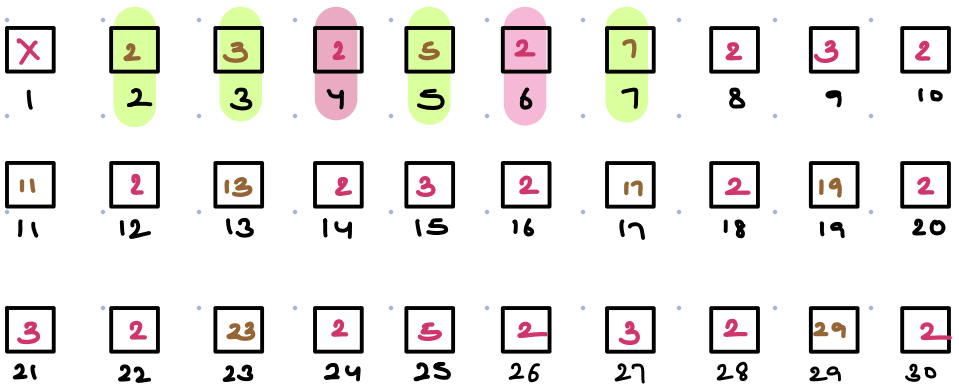
Smallest prime factor

Given N, return the smallest prime factor for all the values from 2 to N

N = 10 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10

Ans = 2 3 2 5 2 7 2 3 2

Assume → Every no. is spf of itself



if (A[i] == i) → prime no.

↓

go on multiples of i & update every multiple with the value of i if i was not updated before.

```
int [] getspf (int N){
    int [] spf = new int [N+1]

    // at element i, spf[i] = i;

    for (i=2; i<=n; i++){
        spf[i] = i;
    }

    for (i=2; i*i<=n; i++){
        if ( spf[i] == i ){      // i is a prime
            for ( j=i*i; j<=n; j+=i ){
                if (spf[j]==j) spf[j] = i;
            }
        }
    }

    return spf;
}
```

Count Divisors

Given an A[N], get count of divisor for each & every element

A[] =

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

1	1	1	1	1	1
	2	3	2	5	2
			3		4
			6		

Ans[] = 1 2 2 4 2 3

Brute force → Iterate on array, get the factors of A[i] using countfactor fn & update answer array

7:55 → 8:05 AM

```java
public class Solution {
    public int[] solve(int[] A) {
        int[]ans=new int[A.length];
        for(int i=0;i<A.length;i++){
            ans[i]=countfactors(A[i]);
        }
        return ans;
    }
    public int countfactors(int n){
        int cnt=0;
        for(int i=1;i*i<=n;i++){
            if(n%i==0){
                if(i==n/i)cnt++;
                else cnt+=2;
            }
        }
        return cnt;
    }
}
```

Reset

✔ Correct Answer.

$TC: O(N * \sqrt{max(A[i])})$

$SC: O(1)$

* Prime factorisation

→ process to break a number into multiples of prime factor.

N = 48

2	48
2	24
2	12
2	6
3	3
	1

$N = 48 \longrightarrow 2^4 * 3^1$

No. of divisors $= (4+1) * (1+1)$

$= 5 * 2 = 10$

1, 2, 3, 4, 6, 8, 12, 16, 24, 48

N = 300

2	300
2	150
3	75
5	25
5	5
	1

$300 = 2^2 * 3^1 * 5^2$

$ans = 1 * (2+1) * (1+1) * (2+1)$

$= 3 * 2 * 3$

$= 18$

Steps

01. Create spf array of size equal to largest ele

sz = (max + 1)

02. Iterate on array & get A[i] = val

→ while (val > 1)

→ get spf for val & divide it as long as possible, maintain a count along it

→ ans = ans * (cnt + 1);

3

```java
public class Solution {
    public int[] solve(int[] A) {
        int n=A.length;
        int maxi=Integer.MIN_VALUE;
        int[]res=new int[n];
        for(int i=0;i<n;i++)maxi=Math.max(maxi,A[i]);
        int[]spf=getspf(maxi+1);

        for(int i=0;i<n;i++){
            int val=A[i];
            int ans=1;

            while(val>1){
                int cnt=0;
                int s=spf[val];

                while(val%s==0){
                    val=val/s;
                    cnt++;
                }
                ans=ans*(cnt+1);
            }
            res[i]=ans;
        }
        return res;

    }
    public int[]getspf (int sz){
        int[]spf=new int[sz];
        for(int i=1;i<sz;i++)spf[i]=i;

        for(int i=2;i*i<sz;i++){
            if(spf[i]==i){
                for(int j=i*i;j<sz;j+=i){
                    if(spf[j]==j) spf[j]=i;
                }
            }
        }
        return spf;
    }
}
```

O(n)

maxi

N

log (maxi)

TC: O(N log(maxi))

SC: O(maxi)

O(maxi)

Problem 3 Pairs with given sum

Problem Statement

Given an integer sorted array `A` *and an integer* `k`*, find any pair (i, j) such that* `A[i] + A[j] = k`*,* `i != j`*.*

A[ ] = { 1 3 5 10 20 23 30 }

k = 23 → True

k = 40 → True

Check if there exists a pair with sum k

`A [ ] = { -3, 0, 1, 3, 6, 8, 11, 14, 18, 25 }`

`k = 12` ⟶ True

Brute force → check for all pairs of A & see if

A[i] + A[j] == k

TC : $O(n^2)$

SC : O(1)

→ Use hashset

TC : O(N)

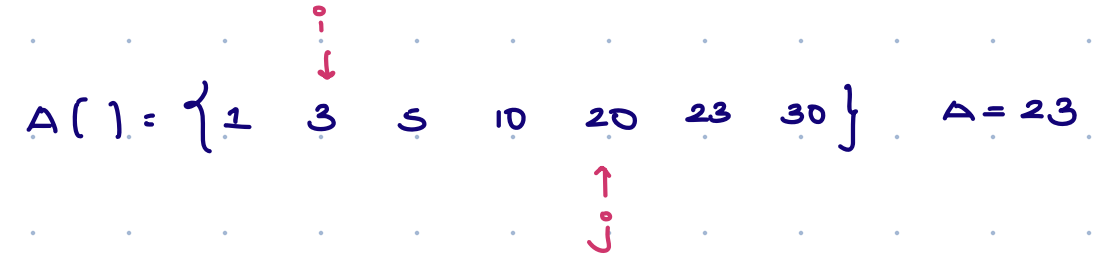
SC : O(N)

9:02 → 9:08 pm

Idea 3 → Two pointer technique

↳ where to start

↳ how ptrs will be updated.



A[i] + A[j]	K	
1 + 30 = 31	> 23	⟶ decrease sum, j--;
1 + 23 = 24	> 23	⟶ decrease sum, j--;
1 + 20 = 21	< 23	⟶ increase sum, i++
3 + 20 = 23	== 23	⟶ return true

```
Arrays.sort(A)
int i=0, j=n-1;
while (i<j)
{
    int sum= A[i] + A[j]
    if (sum == k) return true;
    else if (sum > k) j--;
    else i++;
}
return false;
```

TC : O(nlogn)

SC : O(N)