Problem- Solving - 4

Question : Majority Element.

Ex: 3 3 4 2 4 4 2 4 4

$freq(n) > \frac{n}{2}$ $N = 9$

Ans: 4 $\left[\frac{n}{2}+1\right]$

(n)

Atmax 1 majority Element. $n - \left(\frac{n}{2}+1\right) = \left(\frac{n}{2}-1\right)$

Ex 3 3 4 2 4 4 2 4 $N = 8$

$> \frac{8}{2} > 4$

No Majority Ele

Brute Force : 3 3 4 2 4 4 2 4 4

consider every element

T.C :

$T.C: O(n^2)$
$S.C: O(1)$

```
for(i=0 to n) {
    for( j=0; j<n; j++) {
```

4 1

Frequency of any element

Approach 2 :

N=9

1) Hash map

3 3 4 2 4 4 2 4 4

hashmap = { 3 : 2, 4 : 5, 2 : 2 }

T.C : O(n) + O(n)

S.C : O(n)

Approach 3 :

A = 3 3 4 2 4 4 2 4 4

sort(A) = 2 2 3 3 4 4 4 4 4

0 1 2 3 4 5 6 7 8

$(\frac{n}{2}+1)$ is the frequency

A = 3 3 1 2 1 1 2 1 1

majority = 1

sort(A) = 1 1 1 1 1 2 2 3 3

0 1 2 3 4 5 6 7 8

A = 3 3 2 1 2 2 1 2 2

sort = 1 1 2 2 2 2 2 3 3

possible candidate.

majority = 2

Majority : Sort & take Middle element

A = 5 7 8 9 2

$A' = \quad 2 \quad 5 \quad 7 \quad 8 \quad 1$

1) Sort the Array →
2) count frequency of Middle Element

T.C: $n\log n + n = \boxed{O(n\log n)}$

S.C: $\boxed{O(1)}$ 5

freq++

→ 1 1 2 2 3 4 5 5 5

Moore's Algorithm:

T.C: $O(n)$
S.C: $O(1)$

1) Find a possible candidate for majority element. $O(n)$

2) Verify if this is majority element.

[count(ele) > $\frac{n}{2}$]

count++;
count--;

count = 0
maj = 7

count > 0

count = 0
majority = 7
count > 0

Ex 1 7 2 7 7 2 7 7 3

majority = 7

5 − (2+1) 0

count(7) = 5
count(2) = 2
count(3) = 1

$\boxed{\text{count (majority)} - \text{count(remaining ele)} > 0}$

$\left(\frac{n}{2}+1\right) - \left(\frac{n}{2}-1\right)$

	0	1	2	3	4	5	6	7
A:	7	2	7	7	2	7	7	3

count > 0

majority: ~~2~~ 7

count : ~~1~~ ~~0~~ ~~1~~ ~~2~~ ~~1~~ ~~2~~ ~~3~~ 2

possible candidate = 7

→ check if 7 is the majority element.

majority = 5
count = 1

Ex

d	d	d	d	d
1	2	3	4	5

majority = ~~1~~ ~~2~~ ~~3~~ ~~4~~ 5

count = ~~1~~ ~~0~~ ~~1~~ ~~0~~ ~~1~~ ~~0~~ ~~1~~ ~~0~~ 1

majority = 5
count = 1

d	d	d	d	d	d	d	d	d
0	1	2	3	4	5	6	7	8
7	2	7	4	7	5	7	1	7

major = 7
count = 1

N = 9

majority = ~~7~~ ~~2~~ ~~7~~ ~~4~~ ~~7~~ ~~5~~ ~~7~~ ~~1~~ 7

count = ~~1~~ ~~0~~ ~~1~~ ~~0~~ ~~1~~ ~~0~~ ~~1~~ ~~0~~ ~~1~~ ~~1~~ ~~1~~ ~~1~~ 1

$\left(\frac{n}{2}+1\right)$

```
int findCandidate( int a[], int N) {
    int majority = a[0];
        count = 1
    for(i = 1 ; i < n; i++) {
        if (a[i] == majority) {
            count++;
```

}

else {

count - - ;

}

if(count = = 0) {
majority = a[i];
count = 1;
}

return majority;

}

N = 10

(6)

(N) $\left(\frac{N}{2}\right)$

| 5 | | 5 | . | 5 | . | 5 | . | 5 |

$\left(\frac{N}{2}+1\right)$ $\left(\frac{N}{2}+1\right)$ Similar

Question: Count Special Prime elements

$\boxed{(P_1 < P_2 < P_3)}$

Given 3 prime numbers P_1, P_2, P_3 that Generate first $\boxed{N}$ numbers that are multiply of P_1, P_2, P_3

$14 = 2 \times 7$

$12 = 2^2 \times 3$

$1 = 2^0 \cdot 3^0 \cdot 5^0$

14.

$x = P_1 P_2 P_3 P_4 \ldots$

2, 3, 5

$P_1^0 \cdot P_2^0 \cdot P_3^0$

$7 = 7 \times 1$

$P_1 = 2, \quad P_2 = 3, \quad P_3 = 5$

Sorted

1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, ...

$\boxed{P_1^1 P_2^5 P_3^k}$

N = 11 = (15) (14) N = 11 (15)

N = 100

$4 = 2 \times 2$

$4 = 2^2 \cdot 3^0 \cdot 5^0$

$5 = 5 \times 1$

$5 = 5^1$

1, 2, 3, 4 15

$6 = 2^1 \times 3^1$

$10 = (2, 5)$

Approach 1:

2, 3, 5 $\quad 2 \times 2 = 4$

8

for(i = 1 to INF) {

if(i%P1 == 0 && i%P2 == 0 && i%P3 == 0)

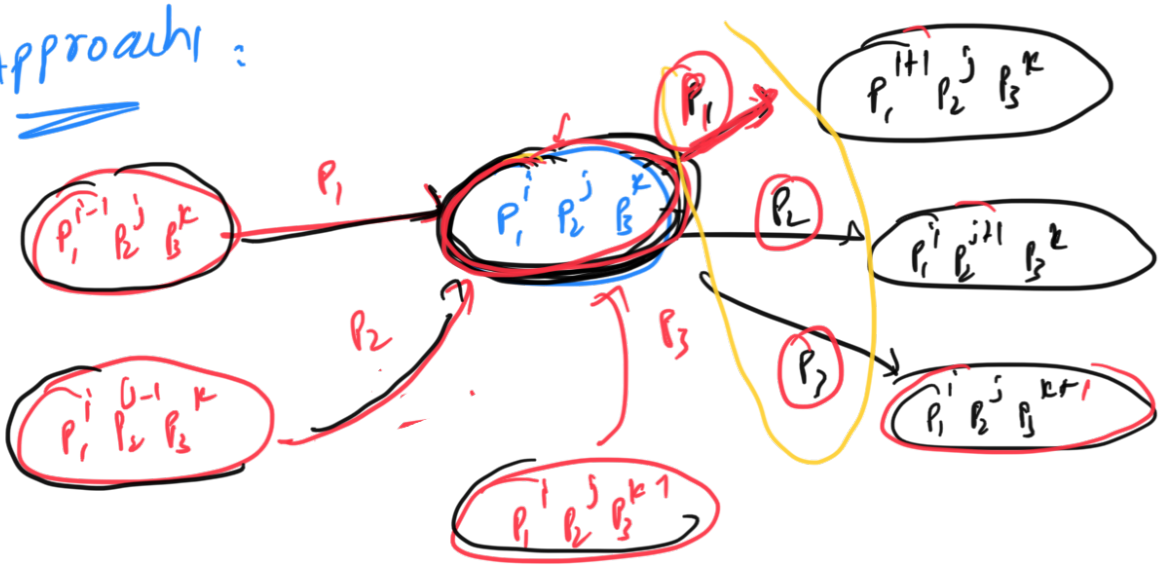
'i' belongs to series

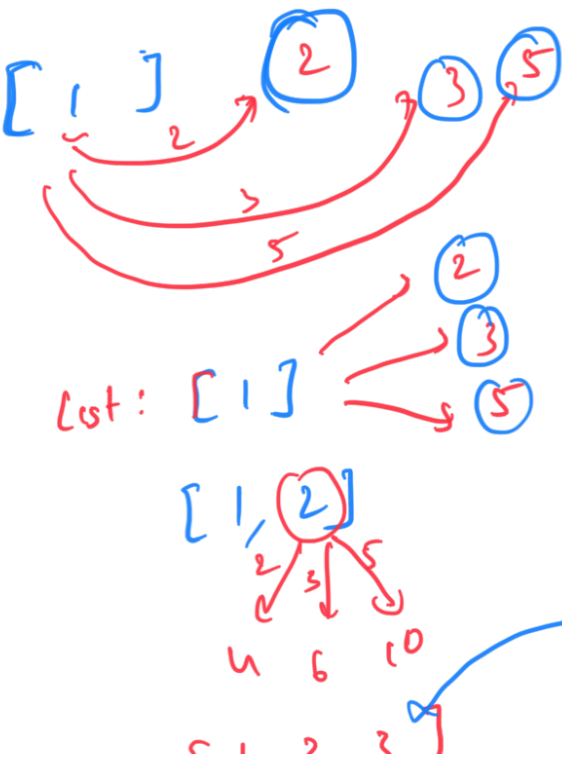
2, 3, 5

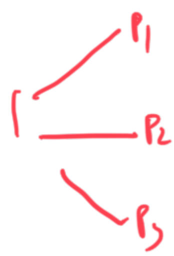
(14)

(8) $P_1^i P_2^j P_3^k$ P_1, P_2, P_3

Approach:







$P_1 = 11$
$P_2 = 17$
$P_3 = 31$

[3, 5]

[3, 4, 5, 6, 10]

[4, 5, 6, 10, 6, 9, 15]

[1, 2, 3]

6 9 15

Simulate in Elegant Way

$P_1 = 2$
$P_2 = 3$
$P_3 = 5$

[1]

P_1 P_2 P_3

$1 \times P_1 = 2$
$1 \times P_2 = 3$
$1 \times P_3 = 5$
} min

$P_1 \times 1$

[1, 2]

P_2 P_3 P_1

$P_2 \times 1 = 3$
$P_3 \times 1 = 5$
$P_1 \times 2 = 4$
} min

[1, 2, 3]

P_3 P_2 P_1

$P_3 \times 1 = 5$
$P_2 \times 2 = 6$
$P_1 \times 2 = 4$
} min

$\rightarrow P_1 \times 2$

[1, 2, 3, 4]

P_3 P_2 P_1

$P_1 \times 1 = 2$
$P_2 \times 1 = 3$
$P_3 \times 1 = 5$
} Min

$P_1 = 2$
$P_2 = 3$
$P_3 = 5$

S = [1]

P_1 P_2 P_3

(n)

Every iter: 3 operation
↓
(n)

$P_2 \times 1$

[1, 2]

$P_2 \times 1 = 3$
$P_3 \times 1 = 5$
} min

$[3n] = O($

$P_1 \times 2 = 4$

$[\ 1,\ 2,\ 3\ ]$

$P_3 \times 1 = 5$
$P_2 \times 2 = 6$
$P_1 \times 2 = 4$ } min

$P_1 \times 2$

$[\ 1,\ 2,\ 3,\ 4\ ]$

$P_3 \times 1 = 5$
$P_2 \times 2 = 6$
$P_1 \times 3 = 6$

$P_3 \times 1$

$[\ 1,\ 2,\ 3,\ 4,\ 5\ ]$

$P_3 \times 2 = 10$
$P_2 \times 2 = 6$
$P_1 \times 3 = 6$ } min

Increment 2 (P_1) P_2++

$[\ 1,\ 2,\ 3,\ 4,\ 5,\ 6\ ]$

T.C:

$6 = P_2 \times 2$
$6 = P_1 \times 3$

(Continue)
Reach N numbers

Question:
Given 3 sorted array, choose 3 values one from each array s.t
a, b, c

(a) minimise $\max(a,b,c) - \min(a,b,c)$

A[] =	5	10	17	20	21	24	
					18	23	35
			9	16			
B[] =	-1	1			15	19	25
C[] =	-8	-2	6	11			

(b)

a	b	c	max(a,b,c)	min(a,b,c)	ans
		-8	10	-8	18
10	-1	11	17	11	6
17	16	6	6	1	5
5	1	25	25	23	$\boxed{2}$
24	23	19	20	18	$\boxed{2}$
20	18	11	10	9	$\boxed{2}$
10	9	25	35	24	$\boxed{11}$
24	35				

(2, 8, 11)
11 - 2 = $\boxed{9}$

A = 1 2 3
B = 7 8 9
C = 10 11 12

~~3, 7, 10~~
3, 7, 10
$\boxed{7}$

Brute Force:

consider all possible triplets

for(i = 0 to n)
  for(j = 0 to m)
    for(k = 0 to p)

T.C : $O(n^3)$
S.C : $O(1)$

(5, -1, -2) = 7
(10, -1, 6) = 11
(10, -1, 6) = 6
(5, -1, 6) =

(5, -1, 8) =
(5, 9, 6) = 4
(5, -1, -2) = 7

3-Pointer

A = 5 10 17 20 21 24
9 16 11 23 35

P_1 P_2

B = -1 1 6 11 15 19 23

C = ~~8~~ -2 6 11

ans = 4

P_1++
P_2++
P_3++

a	b	c	max	min	ans
				-8	13
5	-1	-8	5		
5	-1	-2	5	-2	7
5	-1	6	6	-1	7
5	1	6	6	1	5
5	9	6	9	5	4
10	9	6	10	6	4
10	9	11	11	9	2
10	16	11	16	10	6

we are moving pointers not to get a lesser value rather

we are moving the pointers because we cannot get a better with the current min element

$P_1 = 0 \rightarrow n$
$P_2 = 0 \rightarrow n$
$P_3 = 0 \rightarrow n$
(3n)
$\boxed{O(n)}$

A = 1 2
B = 3 4
C = 5 6

a	b	c	max	min	ans
			5	1	4
1	3	5			3
2	3	5	5	2	

. h l one pointer

continue process until the array reach end of

1) (2,3) (4,5) a, b, c
2) (1,1) (6 7)

<u>Question</u>: Submatrix sum Queries

	0	1	2	3
0	5	4	2	9
1	1	-3	6	10
2	3	14	5	17
3	7	17	3	11
4	9	13	9	16

5 x 4

(r_1, c_1)
(r_2, c_2)

TL : (1,1)
BR : (4,2)

Ans: 64

1 Query:
O(Q.n.m)

Q queries

TL (0,0)
BR (4,3)

(n*m)

<u>Brute Force</u>:

(r_1, c_1) — TL
(r_2, c_2) — BR

```
for( i = r1; i <= r2; i++)
   for(j = c1; j <= c2; j++))
      sum += mat[i][j];
```

T.C: (n*m).Q = $\boxed{O(n.m.Q)}$

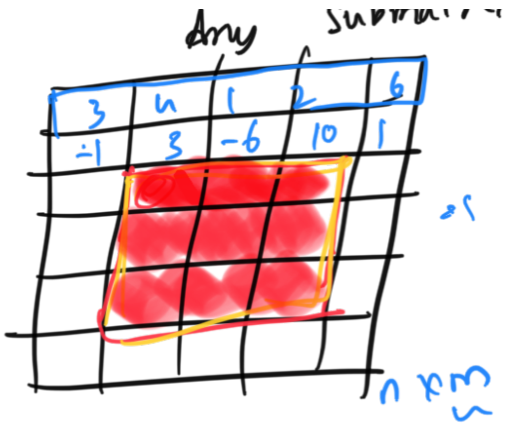
$n = 10^3$
$m = 10^3$
$Q = 10^6$

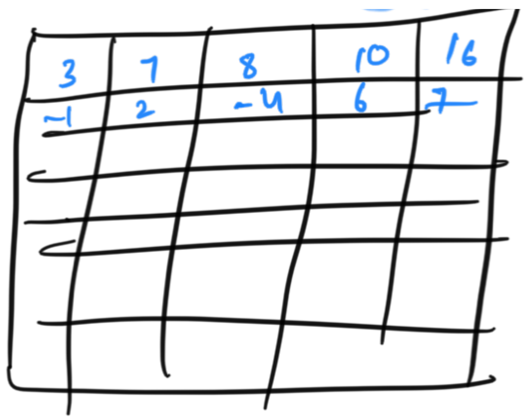
<u>Approach 2</u>:

Prefix Sum.

Grandchildren

submatrix is a collection of row





T.C: $O(m) * n$

Prefix sum: $O(n \cdot m)$

$O(1) * n$

1 Query: $O(n)$

q query: $O(n \cdot q)$

T.T.C: $O(n \cdot m + q \cdot n)$ 10^9

$n = 10^3$

$m = 10^3$

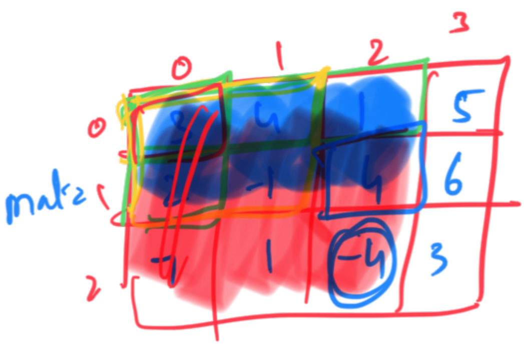
$q = 10^6$

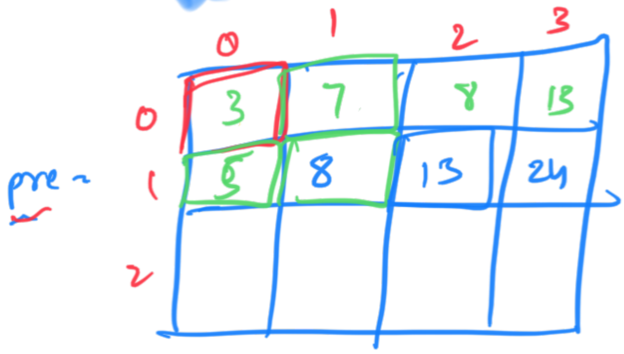
TLE'd

$10^6 \times 10^3 = 10^9$

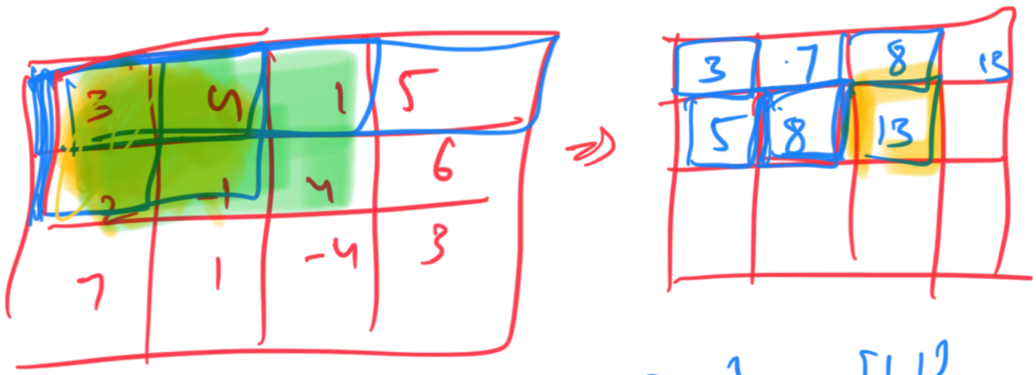
2D Prefix Sum:

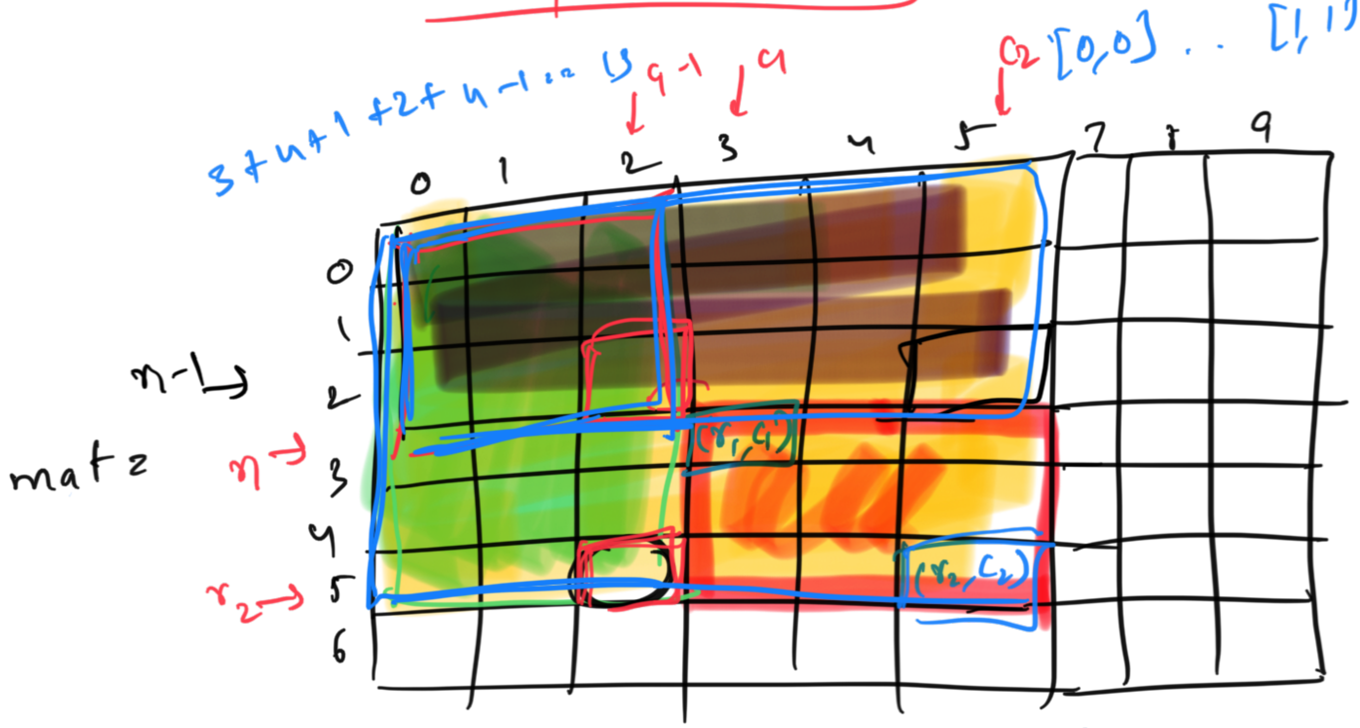
pre[i][j] = contain sum of submatrix from (0, 0) to (i, j)





pre[i][j]: sum([0, 0] .... [i, j])





Yellow : $pre[r_2][c_2]$

Green : $pre[r_2][c_1-1]$

Violet : $pre[r_1-1][c_2]$

Blue : $pre[r_1-1][c_1-1]$

Yellow − Green − Violet

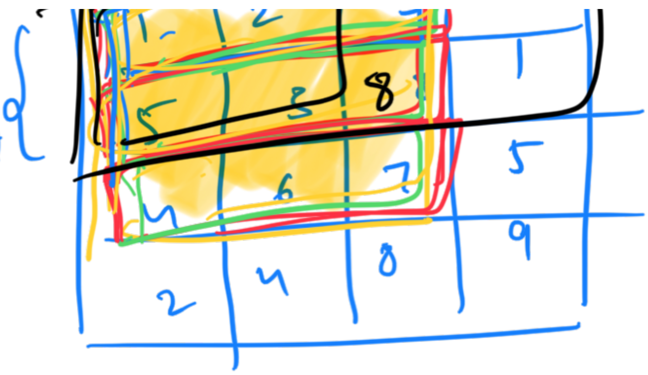
Red = Yellow − Green − Violet + Blue

$$= pre[r_2][c_2] - pre[r_2][c_1-1] - pre[r_1-1][c_2] + pre[r_1-1][c_1-1] \rightarrow O(1)$$

1+2+5+3 = $\boxed{11}$

prefix_row

mat



prefix_colm

1	3	6	10
6	11	22	27
10	21	39	49
12	27	53	72

T.C: O(n.m)

1) Prefix sum of Rows
2) Prefix of column.

T.C: O(n×m)
+O(n×m)

$\boxed{\text{T.C: } O(n \times m)}$

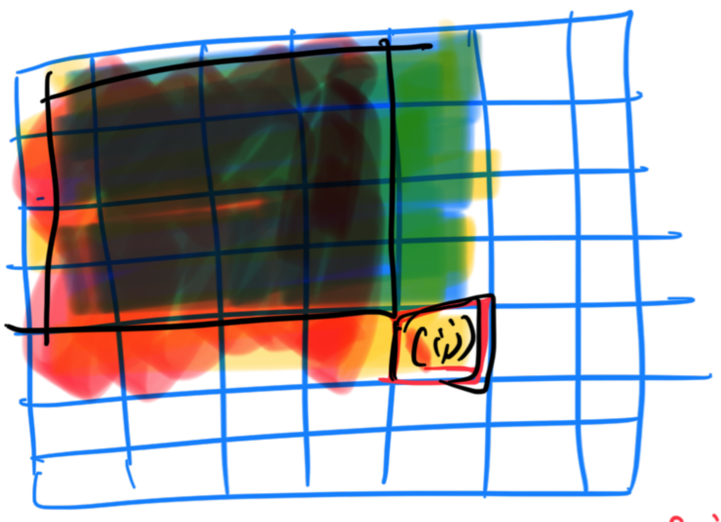
T.T.C: O(n.m) + O(Q.1)

$\boxed{\text{T.C: } O(n.m + Q)}$

S.C: O(n.m)

pre[i][j]

pre[]=



RED = pre[i][j-1]
BLUE = pre[i-1][j]
pre[i-1][j-1]

Black = p..

pre[i][j] = pre[i][j-1] + pre[i-1][j] - pre[i-1][j-1] + mat[i][j]

$\boxed{\text{One Iteration}}$

3	1	2	6
1	4	3	7
2	6	1	5
4 3		10	4

=>

3	4	6	12	
4				
2				
0				

```
for(i=1; i<n; i++) {
    for(j=1; j<m; j++) {
        pre[i][j] =
```

$P_1 = 2$

$P_2 = 3$

$P_3 = 5$

$8 = 2^3$

d

1	2	3	4	5	6	8	9	10	

set = { 2, 3, 5, 4, 6 } -> O(1)

$\boxed{i=2}$

i = 3

i = 4

i = 5

i = 6

i = 8

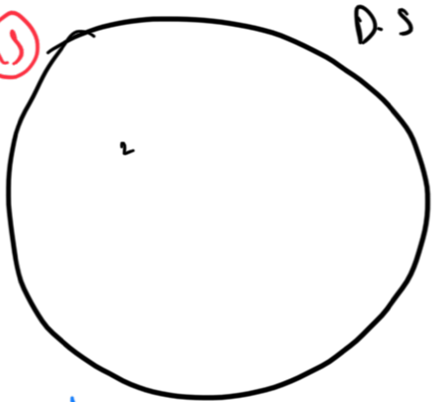
```
for ( i = 2 ; i < INF ; i++ ) {

}
```

T.C : $\boxed{O(n)}$

$8 = 2^3$

8/2 = 4

2x 3x 5x



D.S

~~10~~ $N = 10^6$ num

ans = 10^{12}

$O(10^{12})$ is TLE

$N = 10^6$

$O(n) = (10^6) = 1s$

$N = 10^6$

10^6th number in the series is X

T.C: $O(X)$

$P_1 = 37, P_2 = 37, P_3 = 101$

1000th number = 10^{10}

10^{12}

for(i=1 to 10^{12})

set = {1, 2, 3, 5, 4, 6, 10

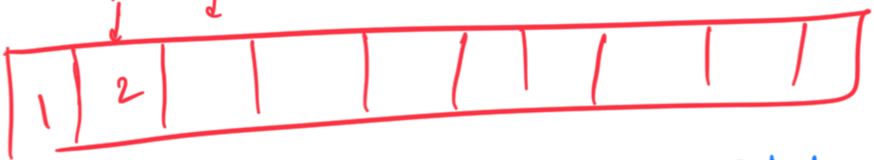
$P_1 = 2$

$P_2 = 3$

$P_3 = 5$.

i=2

10^6



for (i = 1 to INF) {

N^{th} = number in series.

$N = 10^6$

Special Prime Elements

```
specialPrimeElements(int p1, int p2, int p3, int N){
      int a = 0, b = 0, c = 0;
      series[N+1];
      series[0] = 1;
      for(int i = 1; i <= N; i++){
           min_ele = min(p1*series[a],  p2*series[b], p3*series[c] );
           series[i] = min_ele;

           if(min_ele == p1*series[a]) a++;
           if(min_ele == p2*series[b]) b++;
           if(min_ele == p3*series[c]) c++;
      }
      return series;
}
```

Majority Element : Moore's Algo

```
int findCandidate(int a[], int size)
{
    int majority = a[0], count = 1;
    for (int i = 1; i < size; i++) {
        if (majority == a[i])
            count++;
        else
            count--;
        if (count == 0) {
            majority = a[i];
            count = 1;
        }
    }
    return majority;
}
```

2D Prefix Sum matrix generation

```
pre[0][0] = mat[0][0];
// Initialize 1st row
for(int i = 1; i < m; i++)
      pre[0][i] = pre[0][i-1] + mat[0][i];

// Initialize 1st column
for(int i = 1; i < n; i++)
      pre[i][0] = pre[i-1][0] + mat[i][0];

for(int i = 1; i < n; i++)
      for(int j = 1; j < m; j++)
            pre[i][j] = pre[i-1][j] + pre[i][j-1] - pre[i-1][j-1] +
a[i][j]
```

Triplet Minimum

```
int tripletMin(int A[], int B[], int C[], ){
      int p1 = 0, p2 = 0, p3 = 0;
      ans = INT_MAX;
      while(p1<n && p2<n && p3<n){
            val = max(a[p1],a[p2],a[p3]) - min(a[p1],a[p2],a[p3]));
            ans = min(ans, val);

            int minm =  min(a[p1],a[p2], a[p3] );
            if(minm == a[p1]) p1++
            else if(minm == a[p2]) p2++;
            else p3++;
      }
      return ans;
}
```