Q1 Given a matrix of size N×M.

Find max submatrix sum s.t. the submatrix starts at row 0 & ends at row N-1 →

Eg M[3][5]

	0	1	2	3	4
0	-3	2	3	4	-6
1	5	5	-5	2	2
2	-4	-3	1	1	1
	-2	4	-1	7	-3

0,0 to 2,0

0,1 to 2,1

Kadane's Algo

0 (1 2 3) 4

→ ⇒ -2, (4, -1, 7,) -3 ⇒ Solved

r_1, c_1, r_2, c_2

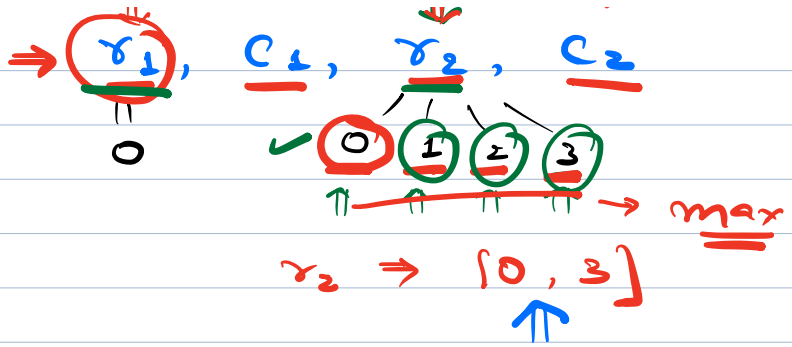
r_1 = 0 (i)

r_2 = N-1 (j)

Q2 Given a matrix of size N×M

Find max submatrix sum s.t. the submatrix starts at row 0 but can end at any row.

	0	1	2	3	4	5
0	2	-4	1	5	-1	2
1	1	3	2	-7	3	3
2	0	-1	1	3	4	-7
3	1	-1	-6	4	-4	6



Case I $r_1 = 0$, $r_2 = 0$

Kadane

2, -4, (1, 5, -1, 2) = 7

→ 0 1 2 3 4 5

Case II $r_1 = 0$, $r_2 = 1$

0 1 2 3 4 5

→ 0 2, -4, 1, 5, -1, 2

→ 1 1, 3, 2, -7, 3, 3

→ (3, -1, 3, -2, 2, 5) → 10

Case III $r_1 = 0$, $r_2 = 2$

0, 1, 2

→ 3, -1, 3, -2, 2, 5

row2 → 0, -1, 1, 3, 4, -7

→ (3 (0), -2 (1), 4 (2), 1 (3), 6 (4)) -2 (5) → 12

c_1 c_2

Ans

Case IV $r_1 = 0$, $r_2 = 3$

→ [3, -2, 4, 1, 6, -2]
1, -1, -6, 9, -9, 6

4, -3, -2, (5, 2, 4) ⇒ 11

Q3 Given a matrix NXM

Cal max submatrix sum s.t submatrix can start at any row & end at any row.

r1	r2 ⇓	
0	[0 - 3] →	Range
1	[1 - 3]	
2	[2 -3]	
3	[3 -3]	

Code

// gr1 gr2

ans = INT_MIN;

for (r1 = 0; r1 < N; r1++){ →

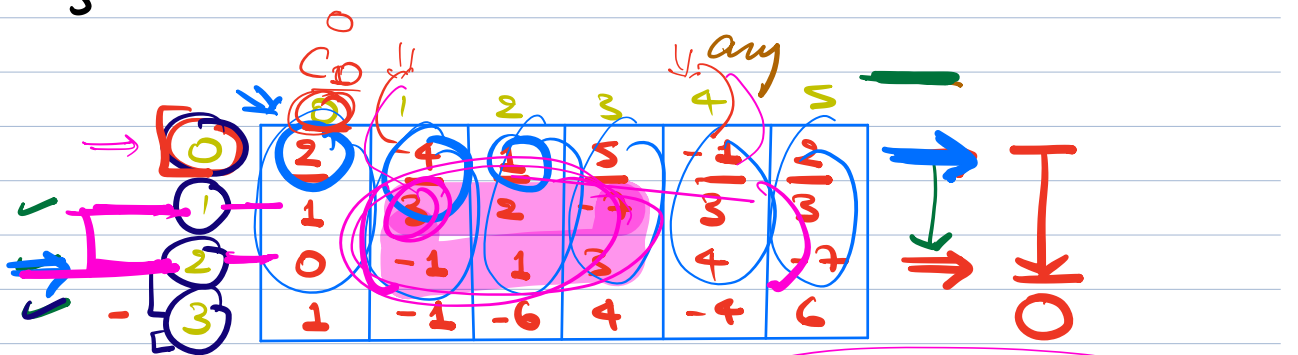
O(N^2) → int sum[m] = {0};

for (r2 = r1; r2 < N; r2++){ →

```
for(i = 0; i < M; i++){
    sum[i] += M[r2][i];
}
```

O(M) maxSum{ // Apply Kadane's on sum array. ⇒

```
ans = max(ans, maxSum);
}
}
```



⇒ 1, 2, 3, -4, 7, -4

O(N) Prefix

r_1 0	2	-4	1	5
r_1 1	3	-1	3	-2
r_2 2	0	-2	4	1
3	1	-3	-2	3

1D array

r_1, r_2 ⇒ 1D sum array

Kadane's

O(N²)

T.C. = $O(N^2 \times (M + M))$

$= O(N^2 \times 2M)$

$= O(N^2 \times M)$

Q2 Given an unsorted array of integer of size N.

Find the first missing natural no.

Eg $A[5] = \{3, -2, 1, 2, 7\}$

1, 2, 3, ~~4~~ ⇒ Ans

$A \Rightarrow [2, 4, -1, -6, 3, 7, 8, 4, -3]$

$\Rightarrow 1$

Soln

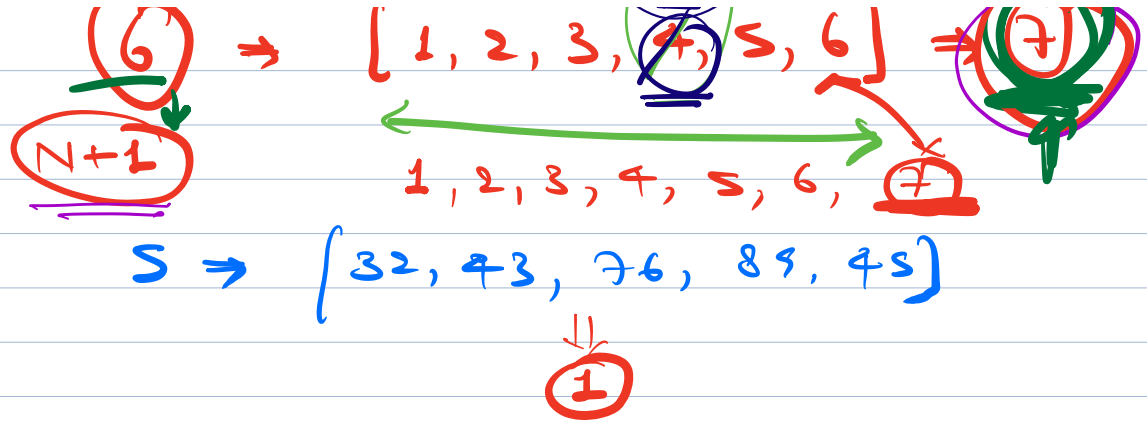
1) Brute force

Iterate over natural no. from 1 & check if it is present or not.

ans =

$A \rightarrow N \quad [1, 2, 3, 4 \ldots\ldots N] \Rightarrow N+1$

10^5



1) Brute force

```
for ( i = 1 ; i <= N + 1 ; i++) {
    // check if i is present in A.
```

1) Iterate ⇒ O(N)

2) HashSet ⇒ O(1)
   S.C. = O(N)

}

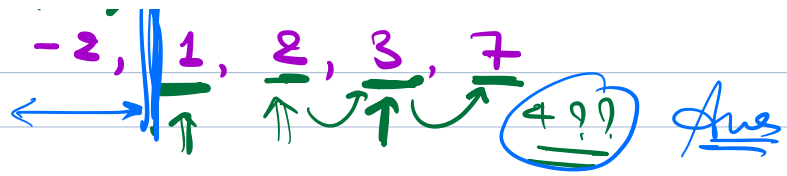
T.C.	$O(N^2)$	$O(N)$
S.C.	$O(1)$	$O(N)$

⇒ No extra space allowed.

2) Sorting

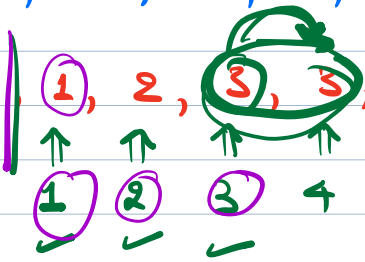
A → 3, -2, 1, 2, 7

Sort ⇒ -2, 1, 2, 3, 7



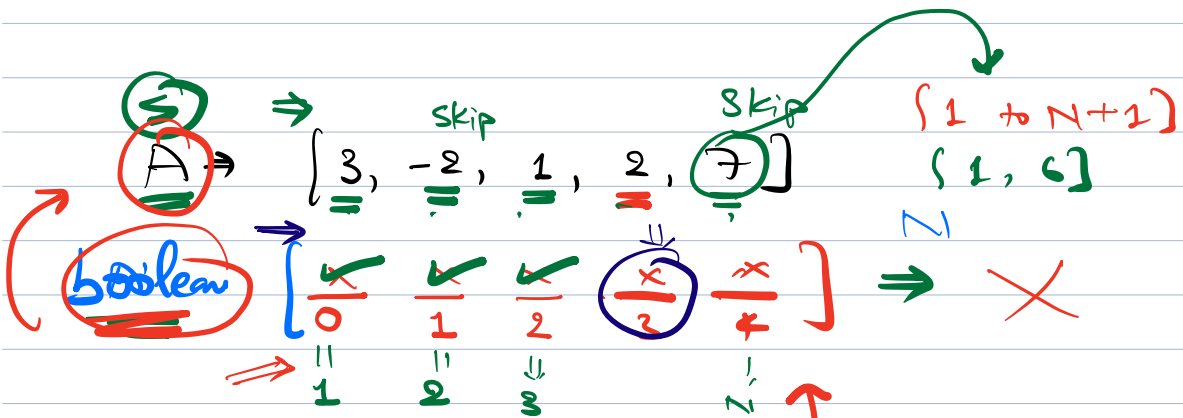
A ⇒ 3, -2, 1, 2, 7, 3, 4

S ⇒ -3, -2

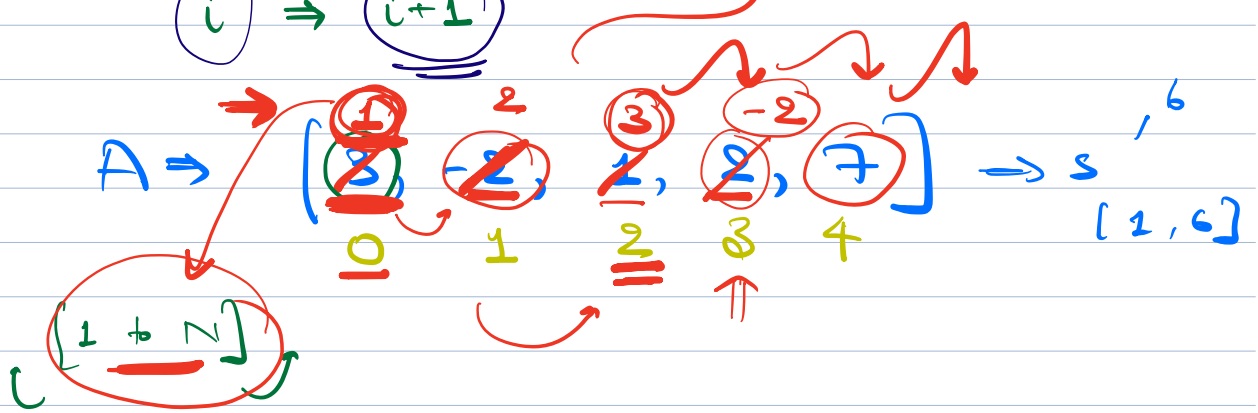


O(N×N) ✗

T.C. = O(N log N + N) ✗

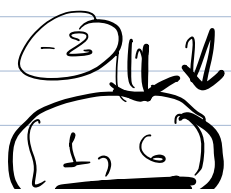


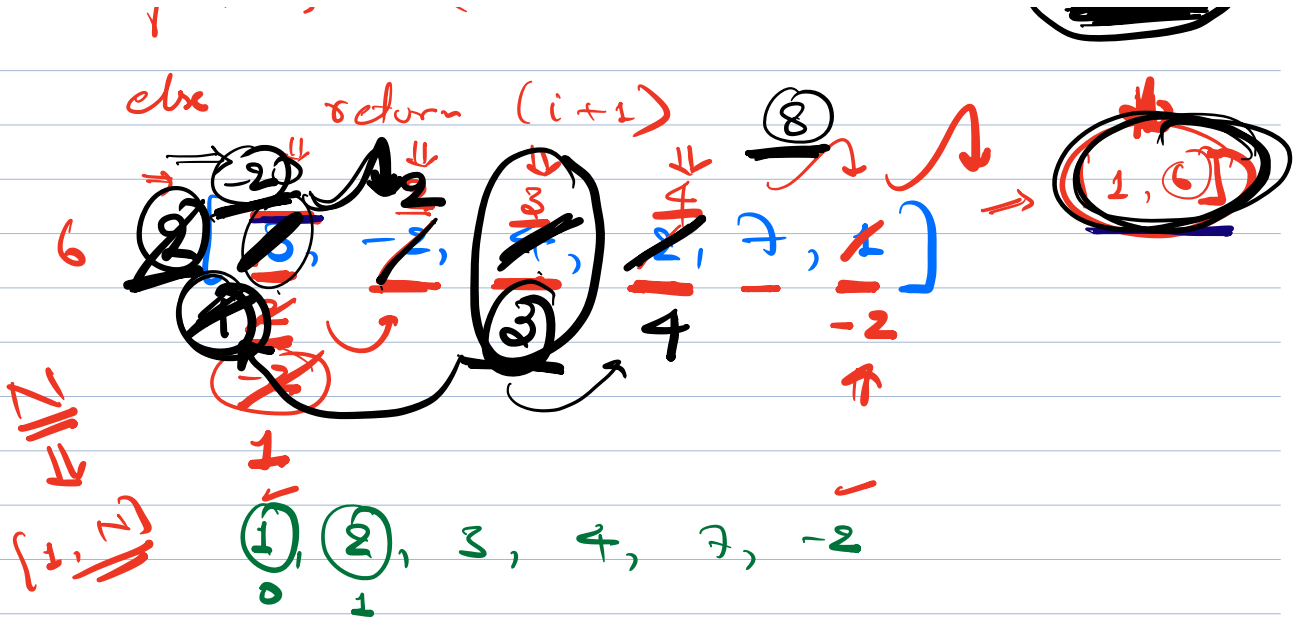
i ⇒ i+1



A ⇒ [1, 2, 3, -2, 7]
0 1 2 3

∴ A[i] = = (i+1) move ahead.



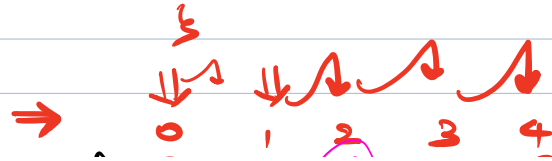


Code

```
for (i = 0; i < N; i++) {
    while ( A[i] > 0 && A[i] < N) {
        if ( A[i] == (i+1) {
            break;
        }
        else {  if (A[i] == A[A[i]-1]) { break; }
            swap ( i , A[i]-1) ;
        }
    }
}
```

```
swap (x, y) {
    temp = A[x];
    A[x] = A[y]
    A[y] = temp;
}
```

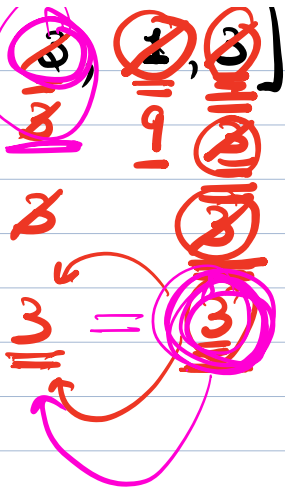
x → x-1



N = 5

[1, 5]

A: [9, 6, 3, 2, 3]



T.C. = O(N)