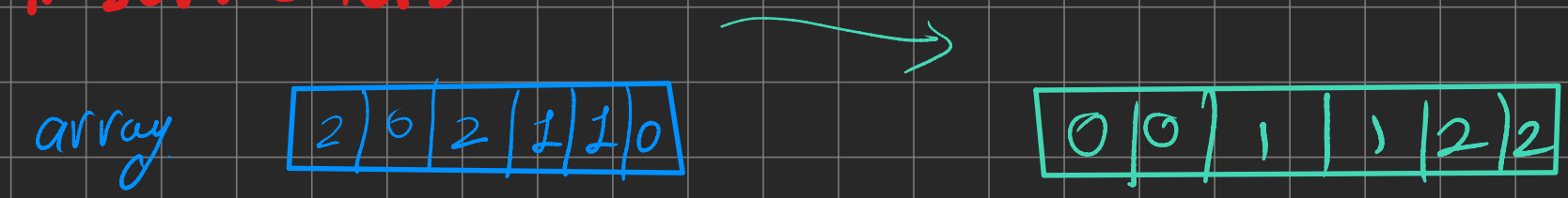
sort colors



Method ① → sorting.

```
vector<int> vec = { 1, 5, 8, 9, 6, 7, 3, 4, 2, 0 };

// Defining the range as whole vector
  auto first = vec.begin();
  auto last = vec.end();
```

$O(n \log n)$

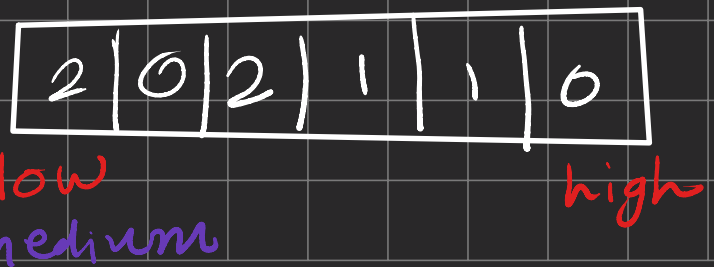
$O(n)$

Mthod ② counting

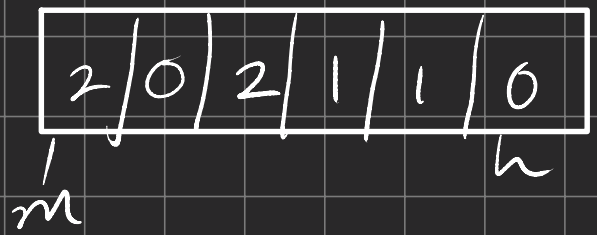
total zero = 2, total one = 2, total two = 2

then add 0,1,2 according to counting into array

Mthod ③ : Inplace solution (three pointer approach)



→ low और zeros को left में लाएंगे।
→ high और twos को right में लाएंगे।
→ medium और one को medium में लाएंगे।

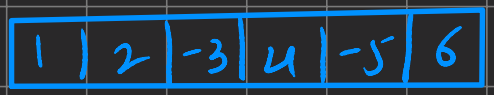


m = 0 → swap(low, mid)
l++, m++

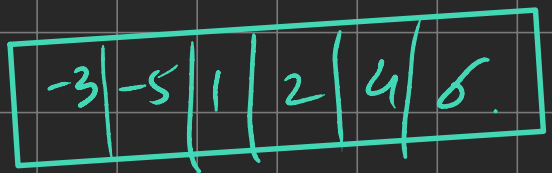
m = 1 → m++

m = 2 → swap(mid, high)
h--

Move all Negative nums on left side of array.

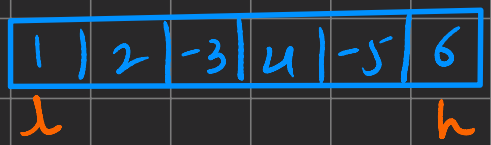


assumption order not matters.



① sort array (increasing order) STL

② Dutch national Flag Algo (two pointer approach)



maintain negative values

maintain positive values.

if low = -ve → low ++

elseif high = +ve → high --

else swap(low, high)

means left has +ve
high has -ve num.

Find duplicate Number

constraints
* num.length = n+1
* 1 <= num[i] <= n

| 1 | 3 | 4 | 2 | 2 | → 2

| 3 | 1 | 3 | 4 | 2 | → 3

n+1 = 5
n = 4

N+1 = 5
N = 4

Method ① Sort.

| 1 | 2 | 2 | 3 | 4 |

→ i = i+1 } during loop on sorted array.

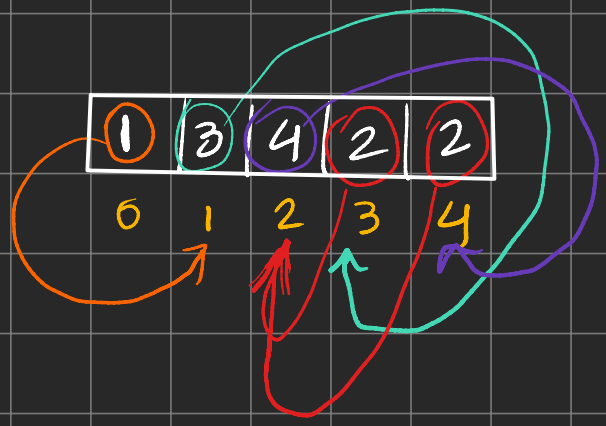
Mtd ② Negative making method

| 1 | 3 | 4 | 2 | 2 |

* => n+1 = num.length
n+1 = 5
n = 4

* num[i] ∈ [1, 4]

Here elements of array behaves like an index.



Here while visiting the index we are making them (-ve)

when we visit same index two times with any element that time that element is repeated we can say that.

IF (num[num[i]]) ⟶ position visited → make it (-ve)

absolute value

IF it is already visited that means that elemet is duplicate

num[i] ⟶ 3 ⇝ -3 1 -3 4 2 2

num[-3] abs⇒ num[3] ⟶ 2 ⇝ -2 1 -3 4 -2 2

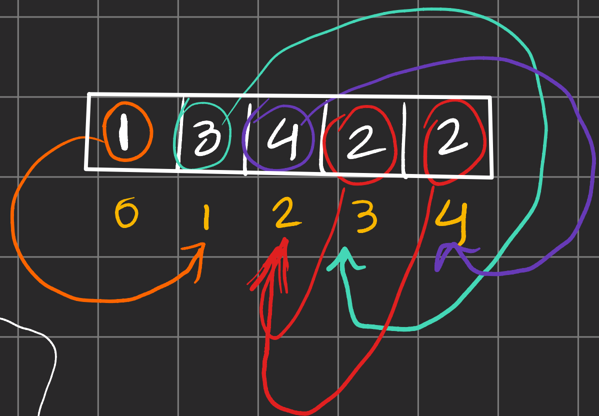
num[4] → 2 ⤳ -2   1 -3 4 -2 -2

num[-2] —abs→ num[2] → 4 ⤳ -4   1 -3 -4 -2 -2

num[-2] —abs→ num[2] → -4

It means element is duplicate

Mtd: ③ Position Method.



इस element को उसके Index पर रख दो।

change position of element according to the index. in last num[0] is repeated element

1	3	4	2	2
3	1	4	2	2
2	1	4	3	2
4	1	2	3	2
2	1	2	3	4

```
while( num[0] != num[num[0]])
    swap(num[0], num[num[0]])

return num[0]
```

Created by Notein

Missing elements from array with duplicates

1	3	5	3	4

→ N: size of array → $a[i] \in [1, N]$

→ 2 missing

1, 2, 3, 4, 5

① Apply negative marking

1	3	5	3	4

index → 1 2 3 4 5

→

-1	3	-5	-3	-4

1 2 3 4 5

positive → so that index(2) ala element missing 😊

② sorting + swapping method

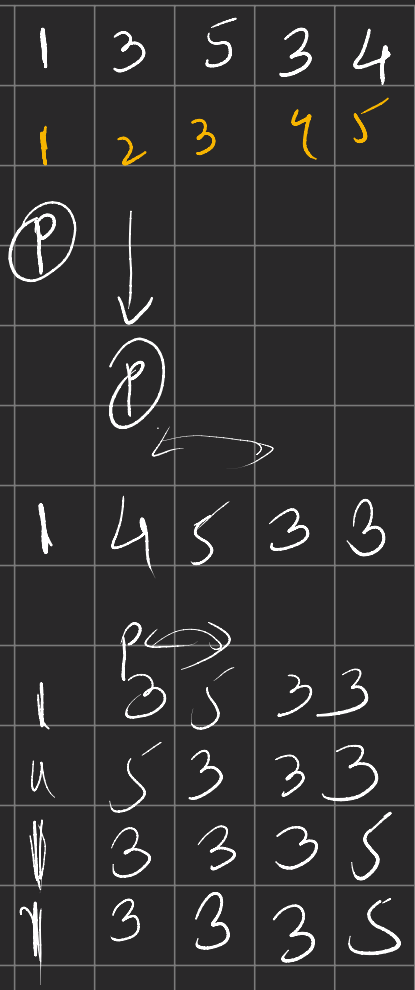
1	3	5	3	4

sort →

1	3	3	4	5

1 2 3 4 5

→ mismatching → 2 is missing



a[1] = 1 == a[p] → p++

a[3] = 5 → a[5] = 4

a[p] ↔ a[5]

3 ↔ 4

```java
// sorting + swapping method
int i=0;
while(i<size){

    int index = b[i]-1;

    if(b[i] != b[index]){
        swap(b[i], b[index]);
    }
    else{
        i++;
    }

}
```

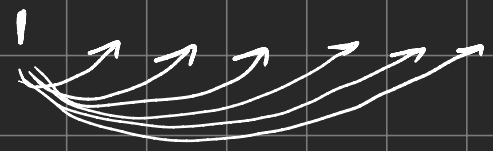
sorting

Find First Repeating element

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

→ 5 is First repeating

①



Tc. $O(n^2)$

②

1	5	3	4	3	5	6

* Hashing

1 → 1
5 → 2
3 → 2
4 → 1
6 → 1

} Traversing and store

$O(n)$

* Iterate and can check one by one elemat

Hashing with unordered map.

```cpp
#include <iostream>
#include <unordered_map>
using namespace std;

int main(){

    int arr[]= {1, 5, 3, 4, 3, 5, 6};
    int size = sizeof(arr)/sizeof(arr[0]);

    unordered_map<int, int> hash; // [int, int]

    // Count frequency of each element
    for(int i=0; i<size; i++){
        hash[arr[i]]++;
    }

    // Print the frequency of each element
    for(auto& pair : hash){
        cout << "Element " << pair.first << " appears " << pair.second << " times."
    }

    return 0;
}
```

key	value
1	1
5	2
3	2
4	1
6	1

bash — Copy code

```
Element 1 appears 1 times.
Element 5 appears 2 times.
Element 3 appears 2 times.
Element 4 appears 1 times.
Element 6 appears 1 times.
```

if (hash[arr[i]] > 1

↳ means duplicate there

✡ Another Way of hashing

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

make new array and then store the value according to index

0 → appears → 0 times

1 → " → 1 times . . .

common elements in 3 sorted Array

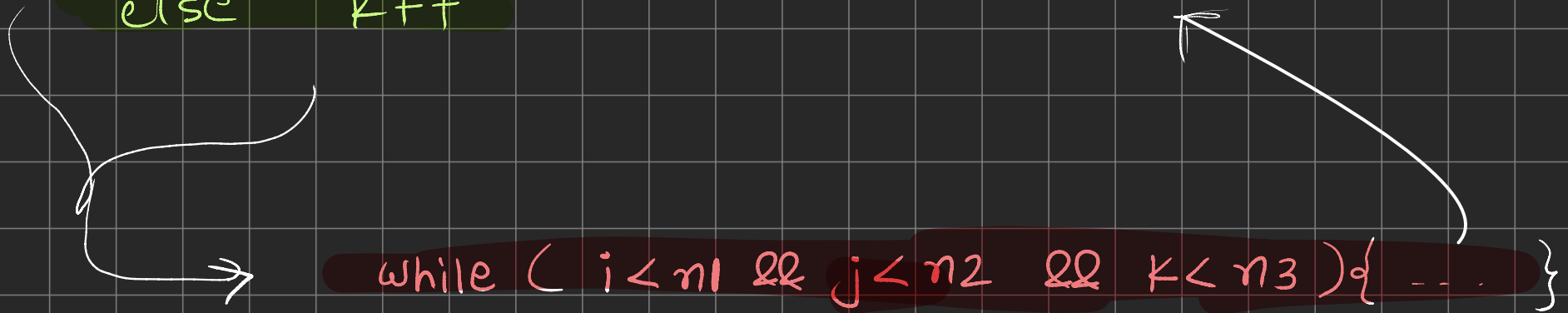
Input: arr1 = [1, 5, 10, 20, 40, 80] , arr2 = [6, 7, 20, 80, 100] , arr3 = [3, 4, 15, 20, 30, 70, 80, 120]
Output: [20, 80]
Explanation: 20 and 80 are the only common elements in arr, brr and crr.

Input: arr1 = [1, 2, 3, 4, 5] , arr2 = [6, 7] , arr3 = [8,9,10]
Output: [-1]
Explanation: There are no common elements in arr, brr and crr.

Input: arr1 = [1, 1, 1, 2, 2, 2], B = [1, 1, 2, 2, 2], arr3 = [1, 1, 1, 1, 2, 2, 2, 2]
Output: [1, 2]
Explanation: We do not need to consider duplicates

①

```
if ( A[i] == B[j] == C[k] )  ⟶  i++ , j++ , k++

else if ( A[i] < B[j] )  ⟶  i++

else if ( B[j] < C[k] )  ⟶  j++

else  k++
```



→ Here we are storing values in array and sometimes because of duplication there are [3,3] like this so how to avoid this → using SET data structure in STL

SET doesnot allow DUPLICATES

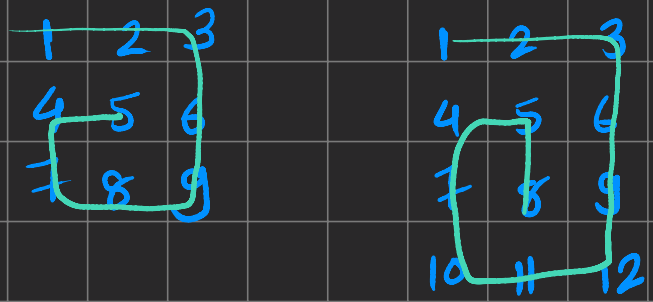
cpp Copy code

```cpp
#include <iostream>
#include <set>
using namespace std;

int main() {
    // Declare a set of integers
    set<int> ans;

    // Sample array
    int A[] = {3, 1, 4, 1, 5, 9, 2, 6, 5};

    // Insert elements of the array into the set
    for (int i = 0; i < 9; ++i) {
        ans.insert(A[i]);
    }

    // Output the elements of the set
    cout << "Elements in the set (unique and sorted): ";
    for (auto& x : ans) {
        cout << x << ' ';
    }
    cout << endl;

    return 0;
}
```

another sol

* we can remove all duplicates from all arrays and then store them in vector array so they donot have duplicates.

Spiral Print Matrix



→

	0	1	2	3	4	5
0	1	2	3	4	5	6
1	7	8	9	10	11	12
2	13	14	15	16	17	18
3	19	20	21	22	23	24
4	25	26	27	28	29	30

$m \times n$

$5 \times 6 = 30$ total elements

V.size()

V[0].size()

→ starting row → 0 1 2

→ ending col → 5 4 3

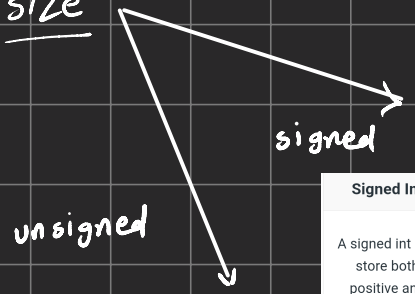
→ ending row → 4 3 2

→ starting col → 0 1 2

update

Factorial of Large number

→ int size



Signed Int	Unsigned Int
A signed int can store both positive and negative values.	Unsigned integer values can only store non-negative values.
A signed integer can hold values from $-2^{32}/2 - 1$ (-2147483648) to $2^{32}/2 - 1$ (2147483647)	A 32-bit unsigned integer can store only positive values from 0 to 2^{32} -1 (4294967295)

Data Types	Sizes in byte	Sizes in bits	Range formula 2^n-1	Ranges
int	4 bytes	32bits	2^{32}-1	-2,147,483,648 to 2,147,483,647
unsigned int	4 bytes	32 bits	2^{32}-1	0 to 4294967295
float	4 bytes	32 bits	2^{32}-1(5 points)	3.4×10^{-38} to $3.4 \times 10^{+38}$
double	8 bytes	64 bits	2^{64}-1(15 points)	1.7×10^{-308} to $1.7 \times 10^{+308}$
long double	10 bytes	80 bits	2^{80}-1(19 points)	1.7×10^{-4932} to $1.7 \times 10^{+4932}$
char	1 byte	8 bits	2^8-1	0 to 255

→ Add two numbers represented by two arrays.

A	9	5	4	9
+				
B		2	1	4

→ if inend, carry ≠0 then put it before answer.

```
carry = 0
int x = a[i] + b[i] + carry
int digit = x % 10
carry = x / 10
```

① x = 9+4+0 = 13
digit = 13 % 10 = 3
carry = 13/10 = 1

② x = 4+1+1 = 6
digit = 6 % 10 = 6
carry = 6/10 = 0

→ $7! = 1 \times 2 \times 3 \times 4 \times 5 \times 6 \times 7 = 5040$ ⇝

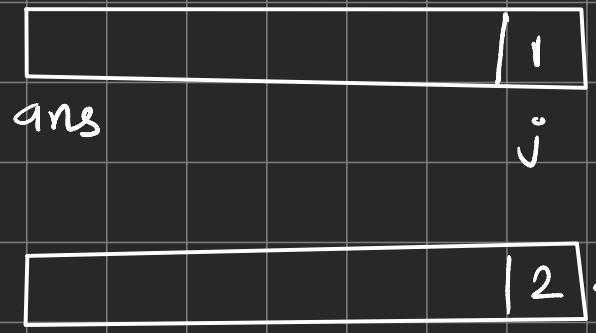
5	0	4	0

i

carry = 0

```
int x = ans[j] * i + carry
ans[j] = x % 10
carry = x / 10
```

→ i = 2



x = 1 * 2 + 0 = 2

ans[j] = 2 % 10 = 2

carry = 2 / 10 = 0

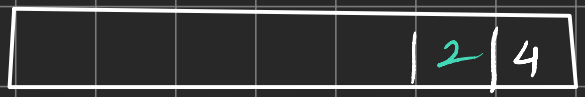
→ i = 3

x = 2 * 3 + 0 = 6

ans[j] = 6 % 10 = 6

carry = 6 / 10 = 0

→ i = 4



x = 6 * 4 + 0 = 24

ans[j] = 24 % 10 = 4

$10\overline{)24}^{2}$, $\frac{20}{4}$

carry = 24 / 10 = 2

if (carry)
ans.push(carry)

→ i = 5

j

x = 4 * 5 + 0 = 20

ans[j] = 20 % 10 = 0

$10\overline{)20}^{2}$, $\frac{20}{0}$

j

carry = 20 / 10 = 2

x = 2 * 5 + 2 = 12

ans[j] = 12 % 10 = 2

carry = 1

→ factorial of large num

7! → 1 × 2 × 3 × 4 × 5 × 7

carry = 0

int total = factNum * i + carry

ans[i] = total % 10

carry = total / 10

0 4 2

2! = 2 × 1 = 2

3! = 3 × 2 = 6

4! = 4 × 6 = 24 → digit = 24 % 10 = 4

carry = 2

5! = 5 × 4 = 20 → digit = 0, carry = 2

4! = 4 * 6 + 0

= 24

digit → ans[i] = digit push

carry

4 2

5! = 5 × 4 + 0

= 20

→ 5 × 2 + 2

= 10 + 2

= 12

0 2 1

```cpp
vector<int> factorial(int n) {
    // code here

    vector<int> ans;
    ans.push_back(1);

    for( int factNum = 2; factNum <= n; factNum++ ){

        //cout << "factNum is" << factNum << endl;
        int carry = 0;

        for( int i=0; i<ans.size(); i++ ){

            int total = factNum * ans[i] + carry;
            int digit = total % 10;
            ans[i] = digit;
            carry = total/10;

        }

        while( carry != 0 ){
            int digitFromCarry = carry%10;
            ans.push_back( digitFromCarry );
            carry = carry/10;
        }

        // for(auto i:ans){
        // cout << ans[i] << " ";
        // }

    }



    reverse( ans.begin(), ans.end() );

    return ans;
}
```