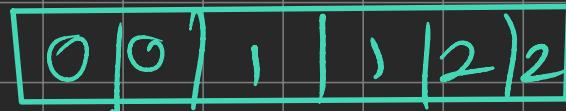
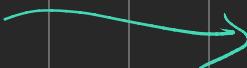


sort colors

array.



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();
```

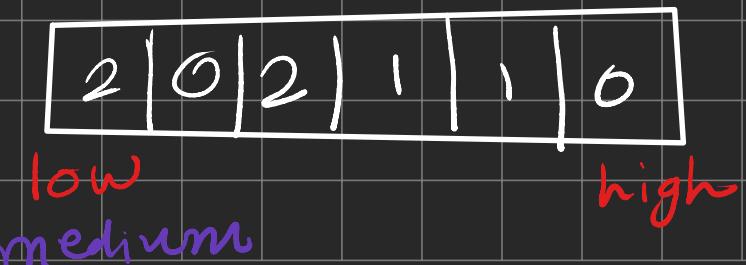
Method ②

Counting

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

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

Method ③ : Inplace solution (three pointer approach)



$O(n \log n)$

$O(n)$

→ low 2112 zeros → left → 0112011

→ high 0112 two's → right → 0112011

→ medium 2112 one → medium → 0112011

2	0	2	1	1	0
m					n

$m=0 \rightarrow \text{swap}(\text{low}, \text{med})$

$l++, m++$

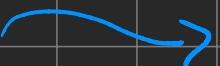
$m=1 \rightarrow m++$

$m=2 \rightarrow \text{swap}(\text{med}, \text{high})$

$h--$

Move all Negative numbers on left side of array.

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



assumption
order not
matters.

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

① sort array (increasing order) STL

② Dutch national Flag Algo (two pointer approach)

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

l
h
means
maintain
negative
values

→ maintain
positive
values.

if $low = -ve \rightarrow low++$

else $high = +ve \rightarrow high--$

use $swap(low, high)$

means l est has +ve
high has -ve num.

Find duplicate Number

constraints

* $\text{num.length} = n+1$

* $1 \leq \text{num}[i] \leq n$

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

 → 2

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

 → 3

$$N+1 = 5$$

$$N=4$$

$$\begin{aligned}n+1 &= 5 \\n &= 4\end{aligned}$$

Method ①. Sort

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

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

Method ②. Negative making method

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

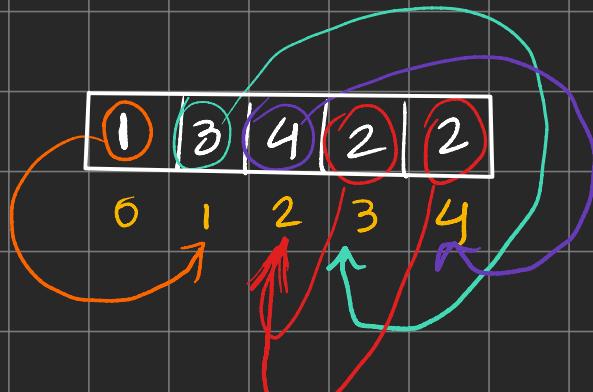
$$\cancel{\Rightarrow} n+1 = \text{num.length}$$

$$n+1 = 5$$

$$n = 4$$

$$\cancel{\Rightarrow} \text{num}[i] \in [1, 4]$$

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



Here elements of array behaves like an index.

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

IF $(num[num[r]])$ → position visited → make it (-ve)

absolute value

IF it is already visited
that means that element is duplicate

$num[i]$ → 3 ↗ -3

1 -3 4 2 2

$num[-3]$ ↗ ab → $num[3]$ → 2 ↗ -2

1 -3 4 -2 2

$\text{num}[4] \rightarrow 2 \rightsquigarrow -2$ 1 -3 4 -2 -2

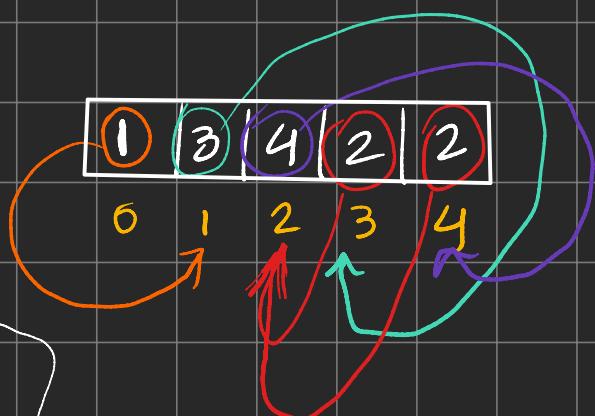
$\text{num}[-2] \xrightarrow{\text{abs}} \text{num}[2] \rightarrow 4 \rightsquigarrow -4$ 1 -3 -4 -2 -2

$\text{num}[-2] \xrightarrow{\text{abs}} \text{num}[2] \rightarrow \underline{-4}$



It means element is duplicate

Mtd ③ Position Method.



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

* If element gets index 4 & its already there.
* change position of element according to the index in last $\text{num}[0]$ is repeated element

while ($\text{num}[0] \neq \text{num}[\text{num}[0]]$)

swap ($\text{num}[0], \text{num}[\text{num}[0]]$)

return $\text{num}[0]$

Missing elements from array with duplicates

1	3	5	3	4
---	---	---	---	---

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

2 missing

1, 2, 3, 4, 5

① Apply negative marking

1	3	5	3	4
---	---	---	---	---

\rightarrow

-1	3	-5	-3	-4
----	---	----	----	----

index \rightarrow 1 2 3 4 5

1 2 3 4 5

↓ positive \rightarrow so that index(2) a[2] element missing is

② Sorting + swapping method

1	3	5	3	4
---	---	---	---	---

sort \rightarrow

1	3	3	4	5
---	---	---	---	---

1 2 3 4 5

mismatching \rightarrow 2 is missing

1 3 5 3 4
1 2 3 7 5

$$a[i] = i \xrightarrow{i=p} p++$$

(P)



(P)



1 4 5 3 3

P



1 3 5 3 3
1 5 3 3 3
1 3 3 3 5
1 3 3 3 5

$$a[3] = 5$$

$$\curvearrowright a[5] = 4$$

$$a[p] \leftrightarrow a[5]$$

$$3 \leftrightarrow 4$$

// 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



* Iterate and can check one by one element

```

#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;
}

```

Hashing with unordered map.

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

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

A	1	5	10	20	40	80
	i					

B	6	7	20	80	100
	j				

C	3	4	15	20	30	70	80	120
	k							

All are
sorted
arrways

if ($A[i] == B[j] == C[k]$) $\rightarrow i++, j++, k++$

else if ($A[i] < B[j]$) $\rightarrow i++$

else if ($B[j] < C[k]$) $\rightarrow j++$

else $k++$

\Rightarrow while ($i < n1 \text{ and } j < n2 \text{ and } k < n3$) { ... }

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

SET does not allow DUPLICATES

cpp

Copy code

```
#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;

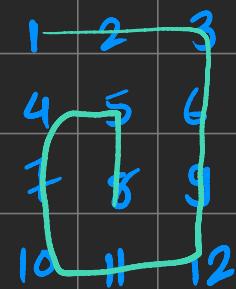
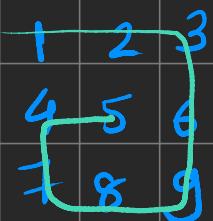
    return 0;
}
```

Y

~~another sol~~

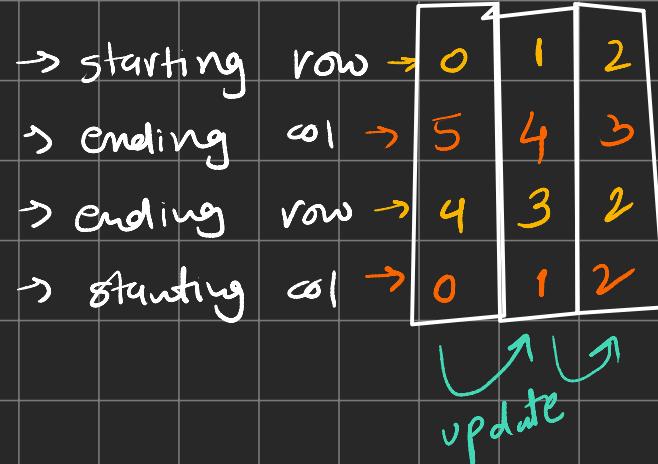
* we can remove all duplicates from all arrays and then store them in vector array so they don't have duplicates.

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



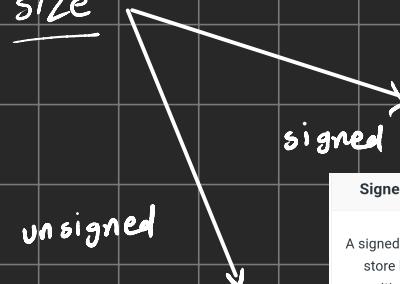
$m \times n$

$5 \times 6 = 30$ total elements

\downarrow
 }
 $v.size()$
 $v[0].size()$

Factorial of Large number soln

→ 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

A → $\begin{array}{cccc} 9_0 & 5_1 & 4_2 & 9_3 \end{array}$

T
B 2 | 1 | 4

B → $\begin{array}{cccc} 2_0 & 1_1 & 4_2 \\ \hline 9_0 & 7_1 & 6_2 & 3_3 \end{array}$

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

carry = 0

int x = a[i] + b[i] + carry

int digit = x % 10

carry = x / 10

$$\begin{aligned} ① \quad x &= 9+4+0 = 13 & ② \quad x &= 4+1+1 = 6 \\ \text{digit} &= 13 \% 10 = 3 & \text{digit} &= 6 \% 10 = 6 \\ \text{carry} &= 13 / 10 = 1 & \text{carry} &= 6 / 10 = 0 \end{aligned}$$

$$\rightarrow 7! = 1 \times 2 \times 3 \times 4 \times 5 \times 6 \times 7 = 5040 \rightsquigarrow \boxed{5040}$$

i

$$\text{carry} = 0$$

$$\text{int } \alpha = \text{ans}[j] * i + \text{carry}$$

$$\text{ans}[j] = \alpha \cdot 10$$

$$\text{carry} = \alpha / 10$$

$$\rightarrow i = 2$$



$$\alpha = 1 * 2 + 0 = 2$$

$$\text{ans}[j] = 2 \cdot 10 = 2$$

$$\text{carry} = 2 / 10 = 0$$



$$\rightarrow i = 3$$



$$\alpha = 2 * 3 + 0 = 6$$

$$\text{ans}[j] = 6 \cdot 10 = 6$$

$$\text{carry} = 6 / 10 = 0$$

$\rightarrow i=4$



$$x = 6 * 4 + 0 = 24$$

$$\text{ans}[j] = 24 \cdot 10^{10} = \frac{24}{10} = 4$$

$$\text{carry} = 24 \cdot 10 = 2$$

if (carry)

ans.push(carry)

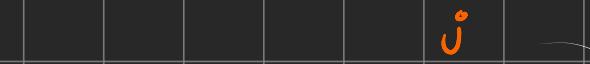
$\rightarrow i=5$



$$x = 4 * 5 + 0 = 20$$

$$\text{ans}[j] = 20 \cdot 10^{10} = \frac{20}{10} = 0$$

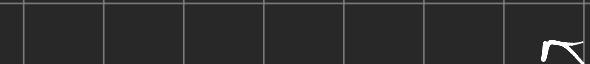
$$\text{carry} = 20 \cdot 10 = 2$$



$$x = 2 * 5 + 2 = 12$$

$$\text{ans}[j] = 12 \cdot 10^{10} = 2$$

$$\text{carry} = 1$$



→ factorial of large num

$$7! \rightarrow 1 \times 2 \times 3 \times 4 \times 5 \times 7$$

carry = 0

$$\text{int total} = \text{factNum} * i + \text{carry}$$

$$\text{ans}[i] = \text{total} \% 10$$

$$\text{carry} = \text{total} / 10$$

1	2	
0	1	2
0	1	2

$$2! = 2 \times 1 = 2$$

$$3! = 3 \times 2 = 6$$

$$4! = 4 \times 6 = 24 \rightarrow$$

$$\text{digit} = \frac{2}{10} = 24 \% 10 = 4$$

$$\text{carry} = 2$$

$$5! = 5 \times 4 = 20 \rightarrow$$

$$\text{digit} = 0$$

$$\text{carry} = 2$$

$$4! = 4 \times 6 + 0 \\ = 24$$

$$\begin{array}{r} 2 \\ 10 \sqrt{24} \\ \underline{20} \\ 4 \end{array} \rightarrow \text{digit} = 4 \quad \text{carry} = 2 \rightarrow \text{ans}[i] = \text{digitpush}$$

4 2

$$5! = 5 \times 4 + 0 \\ = 20$$

$$\rightarrow -5 \times 2 + 2$$

$$= -10 + 2 \\ = 12$$

$$\begin{array}{r} 2 \\ 2 \sqrt{20} \\ \underline{20} \\ 0 \end{array} \rightarrow \text{digit} = 0 \rightarrow \text{push}$$

$$\begin{array}{r} 4 \\ 10 \sqrt{12} \\ \underline{10} \\ 2 \end{array} \rightarrow \text{digit} = 2 \rightarrow \text{push}$$

20 0 2 1

```
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;  
}
```

1/1

