Homework Questions of 12th February 2023

Q1 Print column wise sum in 2D array

i/p →

1	2	3
4	5	6
7	8	9

o/p → 12 15 18

Initially take sum = 0 and we have to do the column wise traversal & add that value to the sum. As the inner loop is finished, this means that we finished traversing the 1st column.

Code

```
void columnWiseSumPrint(int arr[][3],
                        int rows, int cols){

for(int i=0; i<rows; i++){

    int sum = 0;
    for(int j=0; j<cols; j++){
                          → column wise
        sum = sum + arr[j][i]; traversal

    }
    cout << "Sum of column" << (i+1)
         << " is " << sum << endl;

}
```

Q2 Sort 0s, 1s and 2s in the array

i/p →	2	0	2	1	1	0
o/p →	0	0	1	1	2	2

Algorithm - 1

One approach to this problem can be like making 3 variables with names as

zeroCount, oneCount, twoCount which will store the count of 0, 1 and 2 respectively. Then we will run 3 while loop until these count gets 0 & adding desired element like 1st 0s, then 1s & then 2s.

Code

```
void sortColors (vector <int> & nums) {

    int zeroCount = 0;
    int oneCount = 0;
    int twoCount = 0;
    for (int i=0; i< nums.size(); i++) {

        if(nums[i] == 0)
            zeroCount++; //Store no of 0s
        else if (nums[i] == 1)
            oneCount++; //Store no of 1s
        else
            twoCount++; //Store no of 2s

    }
    int i=0;

    while (zeroCount--) {
        arr[i++] = 0; //Adding 0s
    }
    while (oneCount--) {
        arr[i++] = 1; //Adding 1s
    }
```

```
while (twoCount--) { //Adding 2s
        arr[i++] = 2;
  }
}
```

The problem with the above approach is that it is not a one pass solution.

Algorithm-2

We will be solving the question via the 3 pointer approach.

2	0	2	1	1	0
l, m					h

```
if (nums[m] == 0) {
        swap(nums[l], nums[m]);
        l++;
        m++;
}
else if (nums[m] == 1)
        m++;
else {
        swap(nums[m], nums[h]);
        h--;
}
```

1) nums[m] == 2
swap h and m & do h--

0	0	2	1	1	2
l, m				h	

2) nums[m] == 0
swap l and m & do l++, m++

0	0	2	1	1	2
	l, m			h	

3) nums[m] == 0
swap l and m & do l++, m++

0	0	2	1	1	2
		l, m		h	

4) nums[m] == 2
swap h and m and do h--

0	0	1	1	2	2
		l, m	h		

5) nums[m] == 1
Simply do m++

0	0	1	1	2	2
		l	m, h		

6) nums[m] == 1
Simply do m++

0	0	1	1	2	2
		l	h	m	

now m > h & hence exit the loop.

l → place 0s correctly
m → place 1s correctly
h → place 2s correctly

Code

```
void sortColors(vector<int> nums) {

    int l = 0;
    int m = 0;
    int h = nums.size() - 1;

    while (m <= h) {

        if (nums[m] == 0) {
            swap(nums[l], nums[m]);
            l++;
            m++;
        }
        else if (nums[m] == 1) {
            m++;
        }
        else {
            swap(nums[h], nums[m]);
            h--;
        }
    }
}
```

Continued on Pg 251

Homework Questions of 12th February 2023

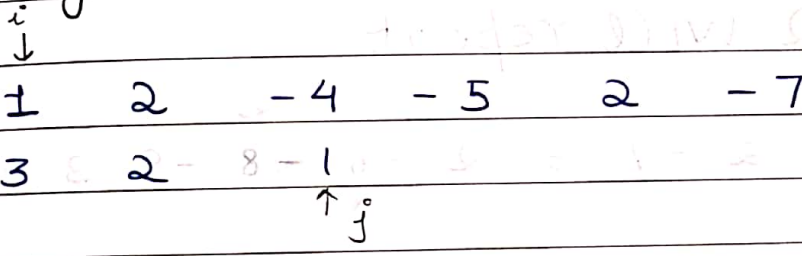
Q3 Move all negatives to one side of the array.

i/p → 1, 2, 3, -4, 5, -6

o/p → -6 -4 3 2 5 1 → o/p may be something else

Our motive is to move all the negatives to the left side of the array.

Algorithm



$i = 0$, $j = arr.size() - 1$

1) $arr[i] < 0 \Rightarrow$ This means that it is the correct position of negative element & simply do $i++$.

2) $arr[j] < 0$ && $arr[i] > 0$ → swap & then do $i++$ and $j--$

3) arr[j] > 0, then this is the correct position of +ve element & hence do j--.

This is a 3 step algorithm.
Step-by-step process

1) 1 2 -4 -5 2 -7 3 2 -6 -8 -9 3 2 1
↑i ↑j

arr[j] > 0 → j-- till we reach -9.

2) 1 2 -4 -5 2 -7 3 2 -6 -8 -9 3 2 1
↑i ↑j

arr[i] > 0 && arr[j] < 0, this means swap & do i++, j--

3) -9 2 -4 -5 2 -7 3 2 -6 -8 3 2 1
↑ ↑
i j

Again step-2 will repeat

4) -9 -8 -4 -5 2 -7 3 2 -6 3 2 1
↑ ↑
i j

arr[i] < 0, simply do i++. This is done till we reach 2.

5) -9 -8 -4 -5 2 -7 3 2 -6 2 1 3 2 1
↑ ↑
i j

arr[i] > 0 && arr[j] < 0, this means

we need to do swap & then i++ & j--.

6) -9 -8 -4 -5 -6 -7 3 2 2 2 1 3 2 1
↑ ↑
i j

Now arr[i] < 0, simply do i++.

7) -9 -8 -4 -5 -6 -7 3 2 2 2 1 3 2 1
↑ ↑
i j

arr[j] > 0, j-- till we reach -7 but here i > j & hence exit from the loop.

Code

```
void moveNegatives (vector<int>&arr) {

    int i=0;
    int j = arr.size() - 1;
    while (i <= j) {
        if (arr [i] < 0)
            i++;
        else if (arr [j] < 0) {
            swap (arr [i], arr [j]);
            i++;
            j--;
        }
        else {
            j--;
        }
    }
}
```

Q4 Missing element from an array with duplicates

i/p → 1 3 5 3 4
o/p → 2

Algorithm

1	3	5	3	4
0	1	2	3	4

1) $i = 0$
index = 1
arr [index - 1] = arr [0] = 1 > 0
Make it -1 so that we treat it as visited.

-1	3	5	3	4
0	1	2	3	4

2) $i = 1$
index = 3
arr [index - 1] = arr [2] = 5 > 0
Make it -5 so that we treat it as visited.

-1	3	-5	3	4
0	1	2	3	4

3) $i = 2$
index = 5
arr [index - 1] = 4 > 0
Make it -4 so we treat it as visited.

-1	+3	-5	3	-4
0	1	2	3	4

4) $i = 3$
index = 3
arr[index-1] = arr[2] = -5 > 0 (False)
so do nothing.

5) $i = 4$
index = 4
arr[index-1] = arr[3] = 3 > 0
Make it -3 so that we treat it as negative.

-1	3	-5	-3	-4

Now traverse the new array & find the index of all positive element & print ans as 1 + index for each +ve element

Here ans = 1+1 = 2

Code

```
void printMissing (vector<int>&arr) {

  for (int i=0; i< arr.size(); i++) {
        int index = abs(arr[i]);
        if (arr[index-1] > 0) {
            arr[index-1] = -arr[index-1];
        }
  }
  for (int i=0; i< arr.size(); i++) {
```

```
        if (arr[i] > 0) {
            cout << (i+1) << endl;
        }
    }
}
```

QS Find first repeating element in the array.

i/p → 1 5 3 4 3 5 6
o/p → 5 , 1 =< size <= 10^6

Algorithm-1
For each element, we check for its occurence in the further array & if found, then simply return the element but the problem with this solution is that it has time complexity = $O(n^2)$ as 2 nested for loops are used.

Algorithm-2
Make a count Array & initialize it with size = 10^6 and initialize all the elements with 0. Now store each element count in that array. Then traverse the input array & check if count of that particular element is greater than or equal to 2, then return that element. This solution has time complexity of O(n) but also space complexity of O(n).

Code

```
int firstRepeated (int arr[], int n){
    int countArray [1000000] = {0};
    for (int i=0 ; i<n ; i++) {
        countArray [arr[i]]++;
    }
    for (int i=0 ; i<n ; i++) {
        if (countArray [arr[i]] >= 2) {
            int ans = arr[i];
            return ans;
        }
    }
    return -1;
}
```

Q6 Find the common elements in 3 arrays which are sorted.

i/p → 1, 5, 10, 20, 40, 80
6, 7, 20, 80, 100
3, 4, 15, 20, 30, 70, 80, 120

o/p → 20 80

Algorithm

$i=0, j=0, k=0$

1) $A[i] == B[j] == C[k]$ → Print the value of $A[i]$ & do $i++$, $j++$ and $k++$

2) A[i] <= B[j] , i++;

3) B[j] <= C[k], j++;

4) k++;

Also we can use set data structure to take care of duplicate elements & if we don't want to use set, then just remove the duplicates from all 3 arrays and then apply the above logic.

Code

```
void commonElements (vector<int> & A,
vector<int> & B, vector<int> & C) {

 int n1 = A.size();
 int n2 = B.size();
 int n3 = C.size();
 set<int> st;
 int i=0, j=0, k=0;

 while (i<n1 && j<n2 && k<n3) {

 if(A[i]==B[j] && B[j]==C[k]){
        st.insert(A[i]);
        i++;
        j++;
        k++;
 }
 else if (A[i] < B[j]) {
        i++;
 }
```

```
else if (B[j] < C[k]){
        j++;
}
else {
        k++;
}
}
for (auto i : st) {
        cout << i << " ";
}
}
```

Q] Factorial of large numbers.

5! = 120 → Can be easily stored in int but 100! is a very large number which can not be stored in the int & hence we use vector to store the digits of the factorial & hence we can store the very large number.

Before doing this question, we can do sum of 2 numbers stored in the vector.

A →	9	4	5	9 ← i
B →		2	1	4 ← j

1) Initially carry = 0

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

x = 9 + 4 + 0

x = 13

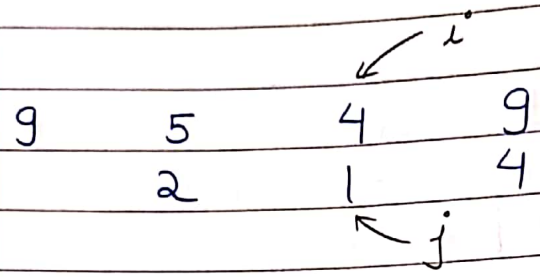
int digit = x % 10;

carry = x / 10;

Store the digit in the output array.

O/p → | | | | 3 |

2)



x = 4 + 1 + 1 = 6

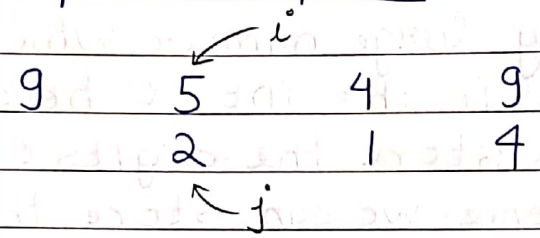
↳ carry

digit = 6 % 10 = 6

carry = 6 / 10 = 0

O/p → | | | 6 | 3 |

3)



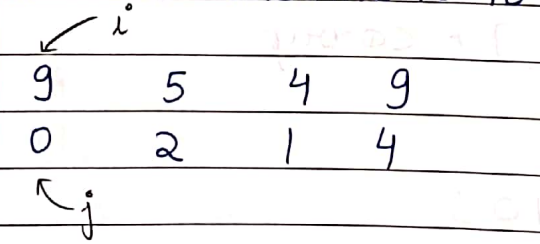
x = 5 + 2 + 0 = 7

digit = 7 % 10 = 7

carry = 7 / 10 = 0

O/p → | | 7 | 6 | 3 |

4) Now number in B array is over so we can assume it ~~to~~ be 0.



$x = 9+0+0 = 9$

digit = $x \% 10 = 9$

carry = $x / 10 = 0$

o/p → | 9 | 7 | 6 | 3 |

If at end carry is left, then we need to add that also in our output array.

Code

```
void findSum (vector <int> &a , vector <int> &b){

    vector <int> ans;
    int carry = 0;
    int i = a.size() - 1;
    int j = b.size() - 1;
    while (i >= 0 && j >= 0) {
        int x = a[i] + b[j] + carry;
        int digit = x % 10;
        ans.push_back (digit);
        carry = x / 10;
        i--;
        j--;
    }
    while (i >= 0) { //If first number has more digits
        int x = a[i] + 0 + carry;
        int digit = x % 10;
        ans.push_back (digit);
        carry = x / 10;
        i--;
    }
```

```
while (j >= 0) { // If 2nd number has more digits.
    int x = b[j] + carry;
    int digit = x % 10;
    ans.push_back(digit);
    carry = x / 10;
    j--;
}
if (carry) {
    ans.push_back(carry);
}
reverse(ans.begin(), ans.end());

for(int i=0; i<ans.size(); {
    cout << ans[i] << " ";
}
}
```

Factorial algorithm

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

Now we have to store each digit of the factorial in the vector.

ans → | | | | | | 1 | ← initially put 1

1) $i = 2, j = 0$

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

ans.size() = 1

↱ carry

$x = 1 \times 2 + 0 = 2 + 0 = 2$

$ans[j] = x \% 10 = 2 \% 10 = 2$

$carry = 2/10 = 0$

	2

2) $i = 3, j = 0$

$x = 2 \times 3 + 0 = 6 + 0 = 6$

$ans[j] = 6 \% 10 = 6$

$carry = 6/10 = 0$

	6

3) $i = 4, j = 0$

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

$ans[j] = 24 \% 10 = 4$

$carry = 24/10 = 2$

↗ push carry if carry is there

	2	4

4) $i = 5, j = 0$

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

$ans[j] = 20 \% 10 = 0$

$carry = 20/10 = 2$

$j = 1$

$x = 2 \times 5 + 2 = 12$

$ans[j] = 12 \% 10 = 2$

$carry = 12/10 = 1$

Pushed carry ↗

	1	2	0

5) $i = 6$, $j = 0 - 2$
Again applying formulae we get 720

	7	2	0

6) $i = 7$, $j = 0 - 2$

$720 \times 7 = 5040$ } Again by applying the formulae

5	0	4	0

↑ carry pushed

Code

```
vector <int> factorial (int n) {

    vector <int> ans ;
    ans.push_back (1);
    int carry = 0 ;

    for (int i = 2 ; i <= n ; i++) {
        for (int j = 0 ; j < ans.size() ; j++) {

            int x = ans[j] * i + carry ;
            ans[j] = x % 10 ;
            carry = x / 10 ;
    }
    while (carry) {
            ans.push_back (carry % 10) ;
            carry = carry / 10 ;
    }
```

```
    carry = 0;
}
reverse(ans.begin(), ans.end());
return ans;
}
```