Find common elements in three sorted arrays

Given three arrays sorted in non-decreasing order, print all common elements in these arrays.

Examples:

ar1[] = {1, 5, 10, 20, 40, 80}

ar2[] = {6, 7, 20, 80, 100}

ar3[] = {3, 4, 15, 20, 30, 70, 80, 120}

Output: 20, 80

ar1[] = {1, 5, 5}

ar2[] = {3, 4, 5, 5, 10}

ar3[] = {5, 5, 10, 20}

Output: 5, 5

A simple solution is to first find [intersection of two arrays](https://www.geeksforgeeks.org/union-and-intersection-of-two-sorted-arrays-2/)and store the intersection in a temporary array, then find the intersection of third array and temporary array. Time complexity of this solution is O(n1 + n2 + n3) where n1, n2 and n3 are sizes of ar1[], ar2[] and ar3[] respectively.

The above solution requires extra space and two loops, we can find the common elements using a single loop and without extra space. The idea is similar to [intersection of two arrays](https://www.geeksforgeeks.org/union-and-intersection-of-two-sorted-arrays-2/). Like two arrays loop, we run a loop and traverse three arrays.  
Let the current element traversed in ar1[] be x, in ar2[] be y and in ar3[] be z. We can have following cases inside the loop.  
1) If x, y and z are same, we can simply print any of them as common element and move ahead in all three arrays.  
2) Else If x < y, we can move ahead in ar1[] as x cannot be a common element 3) Else If y < z, we can move ahead in ar2[] as y cannot be a common element 4) Else (We reach here when x > y and y > z), we can simply move ahead in ar3[] as z cannot be a common element.

// C++ program to print common elements in three arrays

#include <iostream>

using namespace std;

// This function prints common elements in ar1

void findCommon(int ar1[], int ar2[], int ar3[], int n1, int n2, int n3)

{

    // Initialize starting indexes for ar1[], ar2[] and ar3[]

    int i = 0, j = 0, k = 0;

    // Iterate through three arrays while all arrays have elements

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

    {

         // If x = y and y = z, print any of them and move ahead

         // in all arrays

         if (ar1[i] == ar2[j] && ar2[j] == ar3[k])

         {   cout << ar1[i] << " ";   i++; j++; k++; }

         // x < y

         else if (ar1[i] < ar2[j])

             i++;

         // y < z

         else if (ar2[j] < ar3[k])

             j++;

         // We reach here when x > y and z < y, i.e., z is smallest

         else

             k++;

    }

}

// Driver program to test above function

int main()

{

    int ar1[] = {1, 5, 10, 20, 40, 80};

    int ar2[] = {6, 7, 20, 80, 100};

    int ar3[] = {3, 4, 15, 20, 30, 70, 80, 120};

    int n1 = sizeof(ar1)/sizeof(ar1[0]);

    int n2 = sizeof(ar2)/sizeof(ar2[0]);

    int n3 = sizeof(ar3)/sizeof(ar3[0]);

    cout << "Common Elements are ";

    findCommon(ar1, ar2, ar3, n1, n2, n3);

    return 0;

}

Find Union and Intersection of two unsorted arrays

Given two unsorted arrays that represent two sets (elements in every array are distinct), find union and intersection of two arrays.

For example, if the input arrays are:  
arr1[] = {7, 1, 5, 2, 3, 6}  
arr2[] = {3, 8, 6, 20, 7}  
Then your program should print Union as {1, 2, 3, 5, 6, 7, 8, 20} and Intersection as {3, 6}. Note that the elements of union and intersection can be printed in any order.

[**Recommended: Please solve it on “*PRACTICE* ” first, before moving on to the solution.**](https://practice.geeksforgeeks.org/problems/intersection-of-two-arrays/0)

**Method 1 (Naive)**  
***Union:***  
1) Initialize union U as empty.  
2) Copy all elements of first array to U.  
3) Do following for every element x of second array:  
…..a) If x is not present in first array, then copy x to U.  
4) Return U.

***Intersection:***  
1) Initialize intersection I as empty.  
2) Do following for every element x of first array  
…..a) If x is present in second array, then copy x to I.  
4) Return I.

Time complexity of this method is O(mn) for both operations. Here m and n are number of elements in arr1[] and arr2[] respectively.

**Method 2 (Use Sorting)**  
1) Sort arr1[] and arr2[]. This step takes O(mLogm + nLogn) time.  
2) Use [O(m + n) algorithms to find union and intersection of two sorted arrays](https://www.geeksforgeeks.org/union-and-intersection-of-two-sorted-arrays-2/).

Overall time complexity of this method is O(mLogm + nLogn).

**Method 3 (Use Sorting and Searching)**  
***Union:***  
1) Initialize union U as empty.  
2) Find smaller of m and n and sort the smaller array.  
3) Copy the smaller array to U.  
4) For every element x of larger array, do following  
…….b) Binary Search x in smaller array. If x is not present, then copy it to U.  
5) Return U.

***Intersection:***  
1) Initialize intersection I as empty.  
2) Find smaller of m and n and sort the smaller array.  
3) For every element x of larger array, do following  
…….b) Binary Search x in smaller array. If x is present, then copy it to I.  
4) Return I.

Time complexity of this method is min(mLogm + nLogm, mLogn + nLogn) which can also be written as O((m+n)Logm, (m+n)Logn). This approach works much better than the previous approach when difference between sizes of two arrays is significant.

Thanks to [use\_the\_force](https://disqus.com/by/use_the_force/) for suggesting this method in a comment [here](https://www.geeksforgeeks.org/union-and-intersection-of-two-sorted-arrays-2/).

Below is the implementation of this method.

// A C++ program to print union and intersection

/// of two unsorted arrays

#include <iostream>

#include <algorithm>

using namespace std;

int binarySearch(int arr[], int l, int r, int x);

// Prints union of arr1[0..m-1] and arr2[0..n-1]

void printUnion(int arr1[], int arr2[], int m, int n)

{

    // Before finding union, make sure arr1[0..m-1]

    // is smaller

    if (m > n)

    {

        int \*tempp = arr1;

        arr1 = arr2;

        arr2 = tempp;

        int temp = m;

        m = n;

        n = temp;

    }

    // Now arr1[] is smaller

    // Sort the first array and print its elements (these two

    // steps can be swapped as order in output is not important)

    sort(arr1, arr1 + m);

    for (int i=0; i<m; i++)

        cout << arr1[i] << " ";

    // Search every element of bigger array in smaller array

    // and print the element if not found

    for (int i=0; i<n; i++)

        if (binarySearch(arr1, 0, m-1, arr2[i]) == -1)

            cout << arr2[i] << " ";

}

// Prints intersection of arr1[0..m-1] and arr2[0..n-1]

void printIntersection(int arr1[], int arr2[], int m, int n)

{

    // Before finding intersection, make sure arr1[0..m-1]

    // is smaller

    if (m > n)

    {

        int \*tempp = arr1;

        arr1 = arr2;

        arr2 = tempp;

        int temp = m;

        m = n;

        n = temp;

    }

    // Now arr1[] is smaller

    // Sort smaller array arr1[0..m-1]

    sort(arr1, arr1 + m);

    // Search every element of bigger array in smaller

    // array and print the element if found

    for (int i=0; i<n; i++)

        if (binarySearch(arr1, 0, m-1, arr2[i]) != -1)

            cout << arr2[i] << " ";

}

// A recursive binary search function. It returns

// location of x in given array arr[l..r] is present,

// otherwise -1

int binarySearch(int arr[], int l, int r, int x)

{

    if (r >= l)

    {

        int mid = l + (r - l)/2;

        // If the element is present at the middle itself

        if (arr[mid] == x)  return mid;

        // If element is smaller than mid, then it can only

        // be presen in left subarray

        if (arr[mid] > x)

          return binarySearch(arr, l, mid-1, x);

        // Else the element can only be present in right subarray

        return binarySearch(arr, mid+1, r, x);

    }

    // We reach here when element is not present in array

    return -1;

}

/\* Driver program to test above function \*/

int main()

{

    int arr1[] = {7, 1, 5, 2, 3, 6};

    int arr2[] = {3, 8, 6, 20, 7};

    int m = sizeof(arr1)/sizeof(arr1[0]);

    int n = sizeof(arr2)/sizeof(arr2[0]);

    cout << "Union of two arrays is n";

    printUnion(arr1, arr2, m, n);

    cout << "nIntersection of two arrays is n";

    printIntersection(arr1, arr2, m, n);

    return 0;

}