

# GeeksforGeeks

A computer science portal for geeks

## GeeksQuiz

- [Home](#)
- [Algorithms](#)
- [DS](#)
- [GATE](#)
- [Interview Corner](#)
- [Q&A](#)
- [C](#)
- [C++](#)
- [Java](#)
- [Books](#)
- [Contribute](#)
- [Ask a Q](#)
- [About](#)

[Array](#)

[Bit Magic](#)

[C/C++](#)

[Articles](#)

[GFacts](#)

[Linked List](#)

[MCQ](#)

[Misc](#)

[Output](#)

[String](#)

[Tree](#)

[Graph](#)

## 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](#) and store the intersection in a temporary array, then find the intersection of third array and temporary array. Time complexity of this solution is  $O(n_1 + n_2 + n_3)$  where  $n_1$ ,  $n_2$  and  $n_3$  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](#). 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.

Following is C++ implementation of the above idea.

```
// C++ program to print common elements in three arrays
#include <iostream>
using namespace std;

// This function prints common elements in ar1
int 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 array
        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;
}
```

Output:

Common Elements are 20 80

Time complexity of the above solution is  $O(n1 + n2 + n3)$ . In worst case, the largest sized array may have all small elements and middle sized array has all middle elements.

This article is compiled by **Rahul Gupta** Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

## Related Topics:

- [Find Union and Intersection of two unsorted arrays](#)
- [Pythagorean Triplet in an array](#)
- [Maximum profit by buying and selling a share at most twice](#)
- [Design a data structure that supports insert, delete, search and getRandom in constant time](#)
- [Print missing elements that lie in range 0 – 99](#)
- [Iterative Merge Sort](#)
- [Group multiple occurrence of array elements ordered by first occurrence](#)
- [Given a sorted and rotated array, find if there is a pair with a given sum](#)



Tweet

< 3

g+1

< 2

Writing code in comment? Please use [ideone.com](http://ideone.com) and share the link here.

29 Comments

GeeksforGeeks

1

Login ▾

♥ Recommend

🔗 Share

Sort by Newest ▾



Join the discussion...



**Kenneth** • 4 months ago

My codes similar to the given one, but is extended to any N arrays (i.e., N is not just 3).

<http://ideone.com/cGVIQ4>

^ | v • Reply • Share >

**Naveen Khatri** • 6 months ago

nice algorithm

1 ^ | v • Reply • Share ›

**rohit\_90** • 7 months ago

I coded this problem in following way. I think it is more generalized. Any thoughts?

class FindCommonElementsInThreeSortedArrays {

public static void main(String[] args) {

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};

findCommonElementsInThreeSortedArrays(ar1, ar2, ar3);

}

public static void findCommonElementsInThreeSortedArrays(int[] arr1, int[] arr2, int[] arr3){

```
for(int i1=0,i2=0,i3 =0 ;i1<arr1.length&& i2<arr2.length&& i3<arr3.length;){ if(arr1[i1]==arr2[i2]"
&&="" arr2[i2]==arr3[i3]){ " system.out.print(arr1[i1]+"="" );="" i1++;="" i2++;="" i3++;=""
}else{="" int="" max="max(arr1[i1]," arr2[i2],="" arr3[i3]);="" i1="moveForward(arr1," i1,=""
max);="" i2="moveForward(arr2," i2,="" max);="" i3="moveForward(arr3," i3,="" max);="" }=""
}="" system.out.println();="" }="" private="" static="" int="" moveforward(int[]="" arr,="" int="" i,=""
int="" max){="" while(arr[i]<max)="" i++;="" return="" i;="" }="" private="" static="" int=""
max(int="" a,="" int="" b,="" int="" c){="" return="" a="">b?(c>a?c:a):(c>b?c:b);
}
```

}

1 ^ | v • Reply • Share ›

**rohit\_90** → rohit\_90 • 7 months ago<http://ideone.com/W7fzZQ>

1 ^ | v • Reply • Share ›

**Anil Kumar K K** → rohit\_90 • 7 months ago

@rohit\_90 , Instead of moveForward(), You can do the same thing with :

while(arr1[i1] &lt; max) i1++;

while(arr2[i2] &lt; max) i2++;


while(arr3[i3] &lt; max) i3++;

This reduces lots of expensive stack operations.

2 ^ | v • Reply • Share ›

**rohit\_90** → Anil Kumar K K • 7 months ago

@Anil Kumar K K yes, In that way we can reduce method call

 **@Amit Kumar R N**, yes, in that way we can reduce method call overheads.

^ | v • Reply • Share ›



**Nitin Nayyar** • 7 months ago

Awesome logic!

^ | v • Reply • Share ›



**vaibhav shukla** • 7 months ago

couldn't it go like this : Pick one element from array A, binary search it on array B, "if and only if" found, binary search it on array C, otherwise it won't be common in all three. This way, i guess the worst case complexity would be  $O(n \log n^2)$  ?? Any thoughts ?

^ | v • Reply • Share ›



**Gaurav Kumar** → vaibhav shukla • 6 months ago

$O(n \log n^2)$  if all three arrays are same ... othwise  $O(n1 * (\log N2 + \log N3))$  .

^ | v • Reply • Share ›



**vaibhav shukla** → Gaurav Kumar • 6 months ago

oh yes Gaurav ... thnx for pointing out.

^ | v • Reply • Share ›



**Mahendra** → vaibhav shukla • 7 months ago

sucks. why do you want to go for  $n \log n^2$  when you already have  $O(n)$  solution.

^ | v • Reply • Share ›



**young** • 7 months ago

nice algo... at first i thought of Merge sort with little changes which is little same with this algo

^ | v • Reply • Share ›



**MrJack** • 7 months ago

Your "simple" solution is still the best one. Pass two array as input and get an array of common element is what candidate should answer. He should mention that this solution is generic and make a production code some sense. In this case temp space should not be show stopper.

2 ^ | v • Reply • Share ›



**vaibhav shukla** → MrJack • 7 months ago

Indeed. That way, the function would just take  $O(n1+n2)$  time and would be called twice and the array returned after second call would contain the intersection of all three arrays.

^ | v • Reply • Share ›



**kaushik Lele** • 7 months ago

We should not iust check the two numbers but we should find maximum of three numbers.

We should not just check the two numbers but we should find maximum of three numbers.

e.g. if current numbers are 1,6,3. Then even if BOTH array1 and array3 has 2,3,4,5 in them it is of no use as second array will NOT have it.

So code should be as follows :-

```
public static void findCommonOfThreeArrays() {

    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};

    ArrayList<integer> result = new ArrayList<integer>();
    int x=0;
    int i=0,j=0,k=0;
    while(i<ar1.length &&" j<ar2.length=" &&" k<=" ar3.length)=" {" if(ar1[i]="=ar2[j],
    }
    }
}
```

3 ^ | v • Reply • Share ›



**kaushik Lele** → kaushik Lele • 7 months ago

<http://ideone.com/KlgPgH>

2 ^ | v • Reply • Share ›



**Dheeraj Singhal** • 8 months ago

consider three array and three variable as a[],b[],c[]

variable as i1=0,i2=0,i3=0 and size are s1 is size of a[],s2 of b[],s3 of c[] now

```
while(i1< s1 && i2<s2 &&" i3<s3)=" {" if(a[i1]<b[i2]=" ||=" a[i1]<c[i3]=" i1++;=" if(b[i2]
<a[i1]=" ||=" b[i2]<c[i3]=" i2++;=" if(c[i3]<a[i1]=" ||=" c[i3]<b[i2]=" i3++;=" if((a[i1]="=b[i2])"
&&" (a[i1]="=c[i3])) {" printf("%d",a[i1]);=" i1++;=" i2++;=" i3++;=" }=" }=">
```

^ | v • Reply • Share ›



**dinesh** • 8 months ago

As arrays are sorted we can very well use Binary searches and make our process faster compared to normal searches as done above.

3 ^ | v • Reply • Share ›



**gdb** → dinesh • 7 months ago

Your approach will take  $O(n1 \cdot \log(n2) \cdot \log(n3))$  and it is far more than  $O(n1 + n2 + n3)$ .

2 ^ | v • Reply • Share ›



**Smita** • 8 months ago



What about using Hash Tables? It will store number of occurrences for each number in it. At the end if the count for any number is 3 then it means it exists in each set. The only problem i see here is that set should not have duplicates.

^ | v • Reply • Share ›



**vicharak** → Smita • 5 months ago

see , it is given that it is a non decreasing array so obviously any no can repeat any no of time in any array so considering the fact that you are using hash table you will end up counting the same number from a single array and think that you found it in all three..... what i mean is

eg..... arr1 = 1,3,4,6,6,6,7

arr2 = 1,3,5,5,7,8

arr3 = 1,5,8,9

here in this example...acc to your suggested method 5 and 6 both should be common but they are not actually

^ | v • Reply • Share ›



**igaauravsehrawat** → Smita • 8 months ago

that will require extra space and extra loops . Not optimized .

1 ^ | v • Reply • Share ›



**ankit jain** • 8 months ago

```
import java.util.ArrayList;
import java.util.Stack;
```

```
public class Main {

    public static void main(String[] args) {

        int[] arr1 = { 1, 5, 10, 20, 40, 80 };
        int[] arr2 = { 6, 7, 20, 80, 100 };
        int[] arr3 = { 3, 4, 15, 20, 30, 70, 80, 120 };
        Stack<integer> st = new Stack<integer>();

        ArrayList<integer> arr = new ArrayList<integer>();

        for (int i = 0; i < arr1.length; i++) {

            arr.add(arr1[i]);
        }
        for (int i = 0; i < arr2.length; i++) {
```

---

see more

1 ^ | v • Reply • Share ›

**atul** • 8 months ago

ar1] = {3, 4, 15, 20, 30, 70, 80, 120} -- x

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

ar3[] = {1, 5, 10, 20, 40, 80} --- z

if we consider above sequence then algo will increment arr1 instead of arr3.

so to correct this, sequence should also be considered.

consider sequence such that 1st element is in increasing order; so x will be the array with least  $x = \min(\text{arr1}[0], \text{arr2}[0], \text{arr3}[0])$  followed by  $\text{middle} = \min(\text{group}\{\text{arr1}[0], \text{arr2}[0], \text{arr3}[0]\} - x)$ ..then last left.

^ | v • Reply • Share ›

**Stuti** → atul • 8 months ago

But code is working fine against your example without considering sequence.

^ | v • Reply • Share ›

**Satya Swaroop** → Stuti • 8 months ago

Code is working because if there is no element less than 3 in ar1 then any element less than 3 in other arrays is not going to be our solution. So it's fine to have such conditions and move a head. as we are checking the equals condition before that. The conditions make sure that we don't cross the common elements.

^ | v • Reply • Share ›

**Prakash Lalwani** • 8 months ago

In the fourth case, when  $x > y > z$ , the solution increments k (index of third array). Since  $x > y$  in this case, we may as well increment j (index of second array) as y can't be a common element. This way, we save a comparison in the next iteration when  $\text{ar1}[i]$ ,  $\text{ar2}[j]$  and  $\text{ar3}[k+1]$  would have been compared.

Same is the case when  $x < y < z$ . It does not sound like a big deal to save a few comparisons but consider this being a part of an application that is being used by millions of users. I believe we require a different solution for the decision-making part. I would suggest the following.

- 1) If  $x < y$ , we can move ahead in  $\text{ar1}[]$ .
- 2) Else if  $x > y$ , we can move ahead in  $\text{ar2}[]$ .
- 3) If  $y < z$ , we can move ahead in  $\text{ar2}[]$ .
- 4) Else if  $y > z$ , we can move ahead in  $\text{ar3}[]$ .
- 5) If x, y and z are same, we can simply print any of them as common element and move ahead in all three arrays.

5 ^ | v • Reply • Share ›

**shivam** • 8 months ago





nice little algorithm :)  
can we generalize it for n arrays?

^ | v • Reply • Share ›



**Szilard Mandici** → shivam • 8 months ago

Yes, it is basically a modified k-way merge of k arrays.

1 ^ | v • Reply • Share ›



Subscribe



Add Disqus to your site



Privacy

- 
- 
- 
- 

- [Interview Experiences](#)
- [Advanced Data Structures](#)
- [Dynamic Programming](#)
- [Greedy Algorithms](#)
- [Backtracking](#)
- [Pattern Searching](#)
- [Divide & Conquer](#)

- [Mathematical Algorithms](#)
- [Recursion](#)
- [Geometric Algorithms](#)

## • Popular Posts

- [All permutations of a given string](#)
- [Memory Layout of C Programs](#)
- [Understanding “extern” keyword in C](#)
- [Median of two sorted arrays](#)
- [Tree traversal without recursion and without stack!](#)
- [Structure Member Alignment, Padding and Data Packing](#)
- [Intersection point of two Linked Lists](#)
- [Lowest Common Ancestor in a BST](#)
- [Check if a binary tree is BST or not](#)
- [Sorted Linked List to Balanced BST](#)

Follow @GeeksforGeeks

## • Recent Comments

- [It\\_k](#)

i need help for coding this function in java...

[Java Programming Language](#) · [1 hour ago](#)

- [Piyush](#)

What is the purpose of else if (recStack[\*i])...

[Detect Cycle in a Directed Graph](#) · [1 hour ago](#)

- [Andy Toh](#)

My compile-time solution, which agrees with the...

[Dynamic Programming | Set 16 \(Floyd Warshall Algorithm\)](#) · [1 hour ago](#)

- [lucy](#)

because we first fill zero in first col and...

[Dynamic Programming | Set 29 \(Longest Common Substring\)](#) · [2 hours ago](#)

- [lucy](#)

@GeeksforGeeks i don't n know what is this long...

[Dynamic Programming | Set 28 \(Minimum insertions to form a palindrome\)](#) · [2 hours ago](#)

- [manish](#)

Because TAN is not a subsequence of RANT. ANT...

[Given two strings, find if first string is a subsequence of second](#) · [2 hours ago](#)

•

@geeksforgeeks, [Some rights reserved](#) — [Contact Us!](#)

Powered by [WordPress](#) & [MooTools](#), customized by geeksforgeeks team