Find whether an array is subset of another array | Added Method 3

Given two arrays: arr1[0..m-1] and arr2[0..n-1]. Find whether arr2[] is a subset of arr1[] or not. Both the arrays are not in sorted order.

Examples:
Input: arr1[] = {11, 1, 13, 21, 3, 7}, arr2[] = {11, 3, 7, 1}
Output: arr2[] is a subset of arr1[]

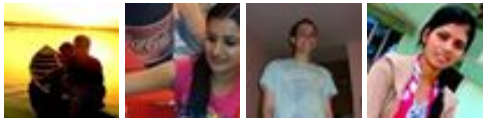
Input: arr1[] = {1, 2, 3, 4, 5, 6}, arr2[] = {1, 2, 4}
Output: arr2[] is a subset of arr1[]

Input: arr1[] = {10, 5, 2, 23, 19}, arr2[] = {19, 5, 3}
Output: arr2[] is not a subset of arr1[]

Method 1 (Simple)
Use two loops: The outer loop picks all the elements of arr2[] one by one. The inner loop linearly searches for the element picked by outer loop. If all elements are found then return 1, else return 0.

```
#include<stdio.h>

/* Return 1 if arr2[] is a subset of arr1[] */
bool isSubset(int arr1[], int arr2[], int m, int n)
{
    int i = 0;
    int j = 0;
    for (i=0; i<n; i++)
    {
        for (j = 0; j<m; j++)
```

```
        {
            if(arr2[i] == arr1[j])
                break;
        }

        /* If the above inner loop was not broken at all then
           arr2[i] is not present in arr1[] */
        if (j == m)
            return 0;
    }

    /* If we reach here then all elements of arr2[]
      are present in arr1[] */
    return 1;
}

int main()
{
    int arr1[] = {11, 1, 13, 21, 3, 7};
    int arr2[] = {11, 3, 7, 1};

    int m = sizeof(arr1)/sizeof(arr1[0]);
    int n = sizeof(arr2)/sizeof(arr2[0]);

    if(isSubset(arr1, arr2, m, n))
      printf("arr2[] is subset of arr1[] ");
    else
      printf("arr2[] is not a subset of arr1[]");

    getchar();
    return 0;
}
```

Time Complexity: O(m*n)

Method 2 (Use Sorting and Binary Search)

```
1) Sort arr1[] O(mLogm)
2) For each element of arr2[], do binary search for it in sorted arr1[].
        a) If the element is not found then return 0.
3) If all elements are present then return 1.
```

```
#include<stdio.h>

/* Fucntion prototypes */
```

Geometric Algorithms

Popular Posts

```
void quickSort(int *arr, int si, int ei);
int binarySearch(int arr[], int low, int high, int x);

/* Return 1 if arr2[] is a subset of arr1[] */
bool isSubset(int arr1[], int arr2[], int m, int n)
{
    int i = 0;

    quickSort(arr1, 0, m-1);
    for (i=0; i<n; i++)
    {
        if (binarySearch(arr1, 0, m-1, arr2[i]) == -1)
           return 0;
    }

    /* If we reach here then all elements of arr2[]
      are present in arr1[] */
    return 1;
}

/* FOLLOWING FUNCTIONS ARE ONLY FOR SEARCHING AND SORTING PURPOSE */
/* Standard Binary Search function*/
int binarySearch(int arr[], int low, int high, int x)
{
  if(high >= low)
  {
    int mid = (low + high)/2;  /*low + (high - low)/2;*/

    /* Check if arr[mid] is the first occurrence of x.
        arr[mid] is first occurrence if x is one of the following
        is true:
        (i)  mid == 0 and arr[mid] == x
        (ii) arr[mid-1] < x and arr[mid] == x
     */
    if(( mid == 0 || x > arr[mid-1]) && (arr[mid] == x))
      return mid;
    else if(x > arr[mid])
      return binarySearch(arr, (mid + 1), high, x);
    else
      return binarySearch(arr, low, (mid -1), x);
  }
 return -1;
}

void exchange(int *a, int *b)
{
    int temp;
```

```
    temp = *a;
    *a   = *b;
    *b   = temp;
}

int partition(int A[], int si, int ei)
{
    int x = A[ei];
    int i = (si - 1);
    int j;

    for (j = si; j <= ei - 1; j++)
    {
        if(A[j] <= x)
        {
            i++;
            exchange(&A[i], &A[j]);
        }
    }
    exchange (&A[i + 1], &A[ei]);
    return (i + 1);
}

/* Implementation of Quick Sort
A[] --> Array to be sorted
si  --> Starting index
ei  --> Ending index
*/
void quickSort(int A[], int si, int ei)
{
    int pi;    /* Partitioning index */
    if(si < ei)
    {
        pi = partition(A, si, ei);
        quickSort(A, si, pi - 1);
        quickSort(A, pi + 1, ei);
    }
}

/*Driver program to test above functions */
int main()
{
    int arr1[] = {11, 1, 13, 21, 3, 7};
    int arr2[] = {11, 3, 7, 1};

    int m = sizeof(arr1)/sizeof(arr1[0]);
    int n = sizeof(arr2)/sizeof(arr2[0]);
```


Recent Comments

```
    if(isSubset(arr1, arr2, m, n))
      printf("arr2[] is subset of arr1[] ");
    else
      printf("arr2[] is not a subset of arr1[] ");

    getchar();
    return 0;
}
```

Time Complexity: O(mLogm + nLogm). Please note that this will be the complexity if an mLogm algorithm is used for sorting which is not the case in above code. In above code Quick Sort is sued and worst case time complexity of Quick Sort is O(m^2)

Method 3 (Use Sorting and Merging)

1) Sort both arrays: arr1[] and arr2[] O(mLogm + nLogn)

2) Use Merge type of process to see if all elements of sorted arr2[] are present in sorted arr1[].

Thanks to Parthsarthi for suggesting this method.

```
/* Return 1 if arr2[] is a subset of arr1[] */
bool isSubset(int arr1[], int arr2[], int m, int n)
{
    int i = 0, j = 0;

    if(m < n)
       return 0;

    quickSort(arr1, 0, m-1);
    quickSort(arr2, 0, n-1);
    while( i < n && j < m )
    {
        if( arr1[j] <arr2[i] )
            j++;
        else if( arr1[j] == arr2[i] )
        {
            j++;
            i++;
        }
        else if( arr1[j] > arr2[i] )
            return 0;
    }

    if( i < n )
        return 0;
```

```
    else
        return 1;
}
```

Time Complexity: O(mLogm + nLogn) which is better than method 2. Please note that this will be the complexity if an nLogn algorithm is used for sorting both arrays which is not the case in above code. In above code Quick Sort is sued and worst case time complexity of Quick Sort is O(n^2)

Method 4 (Use Hashing)
1) Create a Hash Table for all the elements of arr1[].
2) Traverse arr2[] and search for each element of arr2[] in the Hash Table. If element is not found then return 0.
3) If all elements are found then return 1.

Note that method 1, method 2 and method 4 don't handle the cases when we have duplicates in arr2[]. For example, {1, 4, 4, 2} is not a subset of {1, 4, 2}, but these methods will print it as a subset.

Source: http://geeksforgeeks.org/forum/topic/if-an-array-is-subset-of-another

Please write comments if you find the above codes/algorithms incorrect, or find other ways to solve the same problem.

Related Tpoics:

- Remove minimum elements from either side such that 2*min becomes more than max
- Divide and Conquer | Set 6 (Search in a Row-wise and Column-wise Sorted 2D Array)
- Bucket Sort
- Kth smallest element in a row-wise and column-wise sorted 2D array | Set 1
- Find the number of zeroes
- Find if there is a subarray with 0 sum
- Divide and Conquer | Set 5 (Strassen's Matrix Multiplication)
- Count all possible groups of size 2 or 3 that have sum as multiple of 3

Writing code in comment? Please use **ideone.com** and share the link here.

48 Comments GeeksforGeeks

Sort by Newest

Guest • 3 months ago

@GeeksforGeeks Kindly have a look at this. Thanks.

```
int isSubset(int a[],int b[],int sizeA,int sizeB)
{
int j=0;
int i;
int count = 0;
if(sizeA >=sizeB)
{
for(i=0;i<sizea;i++) {="" if(b[j]="a[i])" {="" count++;="" j++;="" i="0;" }="" if(j="sizeb)="" printf("b[]="" is="" subset="" of="" a[]\n");="" else="" printf("b[]="" is="" }="">
```

Reply • Share ›

wasseypuriyan • 7 months ago

Method 3 falis when we have repeated elements in arr1[].
Correct me if I'm wrong ;)

Reply • Share ›

OP Coder ➦ wasseypuriyan • 12 days ago

A set cannot have duplicates.

Reply • Share ›

Guest • 7 months ago

GeeksforGeeks ,Creating hash table for integral values ,i don't think it is benefi create hash for integral if items has large gape between them .

Reply • Share ›

Sudarshan Singh • 7 months ago

@GeeksforGeeks ,Creating hash table for integral values ,i don't think it is ben
create hash for integral if items has large gape between them .

Reply • Share ›

Sudarshan Singh • 7 months ago

@GeeksForGeeks Dear ,the statement "{1, 4, 4, 2} is not a subset of {1, 4, 2},
subset" ,is itself not appropriate as {1,4.4,2} is not a set as per mathematical c

Reply • Share ›

wgpshashank • 7 months ago

In Java it can be done in single line

```
int isSubArray(int array[] , int subArray[])
{

return Collections.indexOfSubList(Arrays.asList(array), Arrays.asList(subArra

}
```

PS: of-course do some checking before you process , don't rely for it on java r

Reply • Share ›

aswathy • 8 months ago

dont you know what is subset?its just number checking

Reply • Share ›

Abhishek Choudhery • 9 months ago

What is the purpose of this &#039(mid == 0 || x > arr[mid-1])&#039 condition
Isn&#039t checking &#039 a[mid]==x&#039 sufficient?

In parition function, what is being done?

Reply • Share ›

Abhishek Choudhery • 9 months ago

What is the purpose of this &#039(mid == 0 || x > arr[mid-1])&#039 condition Isn&#039t checking &#039 a[mid]==x&#039 sufficient?

In parition function, what is being done?

Ronny • 11 months ago

Source code for the same using method 4
and it handles the duplicate case as well.
So kindly update the post.
Method 4(hashing) can very well handle the case of duplicates

```
#include<stdio.h>

#define RANGE 100

int hash[RANGE];

int check(int arr1[],int arr2[],int m,int n)
{
    int i;
    for(i=0;i<m;i++)
      hash[arr1[i]]++;
```


Ronny • 11 months ago

@geeksforgeeks @kartik @venki

Method 4 can handle duplicate case also, if we store in the hash table the cou
And whenever we search for an element in array 2, if hash[arr2[i]]>0, then we'l
hash[arr2[1]]==0 then we can return zero.

This method has O(m+n) time complexity (O(m) for creating hash and O(n) f
and space complexity is O(R) where R is the range of elements

1 ▲ | ▼ • Reply • Share ›

Ronny ➦ Ronny • 3 months ago

@geeksforgeeks Kindly have a look at this. Thanks.

▲ | ▼ • Reply • Share ›

Muthukumar Suresh • a year ago

Can we use bitwise XOR for this .

let X=XOR(array arr1) and Y=XOR(aray arr2)

Z= ((X XOR Y) XOR (X)) XOR Y

if Z is 0 it is subarray otherwise not.

(X XOR Y) gives the elements in ARR1 not in ARR2
this being XORd with X will give the elements present in Y which finally being X
element not present the XOR sum will be 0

▲ | ▼ • Reply • Share ›

hmm ➦ Muthukumar Suresh • a year ago

if ar1= {1} and ar2 ={2} then it will not work

▲ | ▼ • Reply • Share ›

4m7u1 • a year ago

in method three..
if(i < n)

return 0;
else
return 1;
does this handles all the test cases? I mean, what if i>n but j<m, then="" it="" has="" not="" been="" searched="" completely="" ,="" and="" we="" cannot="" of="" the="" array1="" or="" not,="" but="" acc="" to="" this="" code="" it="" cor and becomes true and returns 1.. can anyone explain me this..

• Reply • Share ›

GeeksforGeeks 4m7u1 • a year ago

@4m7u1: Please write code between sourcecode tags.

• Reply • Share ›

4m7u1 4m7u1 • a year ago

i greater than n but j less than m

• Reply • Share ›

4m7u1 4m7u1 • a year ago

I mean, what if i>n but j<m, ie="" else="" case="" and="" becomes="" t anyone="" explain="" me="" this..="">

• Reply • Share ›

rajee • a year ago

hey method 3 would fail if array 1 has repeated elements !!

```
/* Paste your code here (You may delete these lines if not writing c
```

• Reply • Share ›

GeeksforGeeks rajee • a year ago

Could you provide more details or a sample test case for which it failec

• Reply • Share ›

Rasheed Ahamed • a year ago

find-whether-an-array-is-subset-of-another-array-set-1 and return index value

• Reply • Share ›

Sumit • a year ago

```
/* Paste your code here (You may delete these lines if not writing c
```

If the numbers are prime we can just check whether the product of bigger arra
one

• Reply • Share ›

Filmon Kidane • a year ago

thanks I am sastisfied of t his page.

• Reply • Share ›

Yusra Mehri • a year ago

I didn&#039t get any benefit from your information it seems to me complicatec

• Reply • Share ›

bunty • 2 years ago

@geeks: In method 3,
2) After sorting two arrays, we can actually find the intersection of two sorted a
array2 is sub-array of array1.

• Reply • Share ›

bunty ➦ bunty • 2 years ago

- Issue would be intersection need a little complex code.
Better would be merge, I suppose.

• Reply • Share ›

Steve • 2 years ago

```
void main()
{
int a[] = {11, 1, 13, 21, 3, 7};
int b[] = {11, 3, 7, 4};
max=maximum(a,6);
int sub=subset(a,b,6,4,max);
if(sub)
printf("B is a subset of A");
else
printf("B is not a subset of A");
}
int maximum(int a[],n)
{
max=-1;
for(i=0;i<n;i++)
{
if(max<a[i])
max=a[i];
```

see more

Reply • Share ›

Mustafa • 2 years ago

THANKYOU!!!!!!!!!!!!! :)

Reply • Share ›

v3nom • 2 years ago

can't we just read the main array in a map and then store its size...now insert
remains same then it is a sub-array otherwise not.

Reply • Share ›

kartik ↗ v3nom • 2 years ago

Your approach looks similar to method 4. If we compare it with method
be doing less number of comparisons in general. Let me know your th

• Reply • Share ›

Shailesh • 3 years ago

1) We can use BitSet instead of HashMap to keep track of elements. (Space
element again)
2) Algorithm would look like this
a) Traverse array1 and set the bit for each element in BitSet
b) Traverse array2 and for each element check if BitSet is set to true if not arr
the elements of array2 being set in BitSet so array2 is subset of array1.

To handle duplicates following algo can be used

1) Traverse array1 for each element put the (key,value) = (number,count) cou
number exist in array1
2) Traverse array2 for each element check if it exist in HashMap and count val
If found reduce the count by 1 if not then array2 is not subset of array1.

For both the above approach complexity is O(m+n)

• Reply • Share ›

Tanmay Chakrabarty • 3 years ago

I think, If we have sorted the arrays, then we can make a simple comparision l

```
 a=1,2,4,7;
b=1,4,4;


    j=0;
    for(i=0;i<3;i++)
    {
          x=0;
```

```
        for(j=j;j<4;j++)
           {
                if(b[i]==a[j])
                    {
                       x=1;
                       j++;
                       break;
                    }
           }
```

see more

• Reply • Share ›

Kshitij • 3 years ago

For method 2, this was quoted ->
"Time Complexity: O(mLogm + nLogm) If an mLogm algorithm is used for sor
code. In above code Quick Sort is sued and worst case time complexity of Qu

In Method 3 also, however, quicksort is used again and hence the above logic
the time complexity will not remain O(mLogm + nLogn).
Also a slight typo,
Time Complexity: O(mLogm + nLogn) which is better than **method 3**
instead should be method 2.

Thanks,
Kshitij

• Reply • Share ›

GeeksforGeeks ➦ Kshitij • 2 years ago

@Kshitij: Thanks for the comments. We have updated the post. Keep

• Reply • Share ›

ktanay • 3 years ago

"Note that method 1, method 2 and method 4 don't handle the cases when we
{1, 4, 4, 2} is not a subset of {1, 4, 2}, but these methods will print it as a subs

Hashing can still be used in the case of duplicates. All we need to do is to kee
{1,4,2}.
1 -> 1
4 -> 1
2 -> 1
And using arr2 and hashing the elements similarly decrement this count in cas
answer.
For the case in consideration.
1 -> 0
4 -> -1
2 -> 0
So when we encounter a count less than zero (0) we know that arr2 is not a s
mistaken.

My 2 cents.
Thanks,
Kshitij

  • Reply • Share ›

Ankit Mehta • 3 years ago

Hi, This can be done using the Set data structure also. Create an instance of s
larger array into it. Then add the elements of the smaller array and check if the
increased that means it is not the subset, if size is same that would mean it is

For implementation we can use HashSet class in Java. Is this approach corre

  • Reply • Share ›

rajcools ➦ Ankit Mehta • 3 years ago

yes it is..... but this is a kind of hashing and included in method 4....

but yes your approach is correct

rajcools • 3 years ago

code for method 3
a simple code for hashing.... here i have used index as the key value..

```
 #include<stdio.h>
#include<stdlib.h>

#define TABLESIZE 100
typedef int KEYTYPE;
typedef int VALUETYPE;

//for hashtable i am using key as index only
struct record
{
// KEYTYPE k; //we can easily use complex hash function and
// then use ey correspondingly
    VALUETYPE r; //this is the value
} table[TABLESIZE];
```


Nitish ➚ rajcools • 3 years ago

Nice solution

There is one more problem, It will be using a lot of space that is depen

Can we use hash function to resolve this problem.

Parthsarthi • 3 years ago

Why do binary search after sorting??
Since both arrays will be available in sorted form, we can just use two pointers

```
 bool isSubset(int arr1[], int arr2[], int m, int n)
{
    int i = 0, j = 0;

    quickSort(arr1, 0, m-1);
    quickSort(arr2, 0, n-1);
    while( i < n && j < m )
    {
        if( arr1[j] <arr2[i] )
            j++;
        else if( arr1[j] == arr2[i] )
        {
            j++;
            i++;
        }
```

see more

• Reply • Share ›

GeeksforGeeks ➦ Parthsarthi • 3 years ago

@Parthsarthi: Thanks for suggesting a new method. We have added it

• Reply • Share ›

kartik ➦ Parthsarthi • 3 years ago

@Parthsarthi: It doesn't work for following example.

```
/*Driver program to test above functions */
```

```
int main()
{
    int arr1[] = {1, 2, 4};
    int arr2[] = {1, 2, 3, 4, 5};

    int m = sizeof(arr1)/sizeof(arr1[0]);
    int n = sizeof(arr2)/sizeof(arr2[0]);

    if(isSubset(arr1, arr2, m, n))
      printf("arr2[] is subset of arr1[] ");
    else
      printf("arr2[] is not a subset of arr1[] ");

    getchar();
    return 0;
}
```

• Reply • Share ›

rajcools ➚ kartik • 3 years ago

@kartik where is the problem works just fine...one more thing a
should be a check
if(m<n)
return FALSE;

• Reply • Share ›

kartik ➚ rajcools • 3 years ago

@Parthsarthi: There was some confusion from my side
all cases.

• Reply • Share ›

Parthsarthi ➚ kartik • 3 years ago

it prints "arr2[] is not a subset of arr1[]" on my machine, which i

It prints "arr2[] is not a subset of arr1[]" on my machine..which i

Reply • Share ›

Arun ➔ Parthsarthi • 3 years ago

In both the cases your arrays dont contain duplicates. C

int arr1[]={1,1,1,2,3,3}
int arr2[]={1,1,2,2,3,3}

Reply • Share ›

Tanmay Chakrabarty ➔ Arun • 3 years ago

Would you plz notice this,

```
 /*
Find whether an array is subset of another array
Link: http://geeksforgeeks.org/?p=12926
*/

#include<iostream.h>

int main()
{
    int a[6],b[6],i,j,x;

    a[0]=1;    a[1]=1;    a[2]=1;
    a[3]=2;    a[4]=3;    a[5]=3;

    b[0]=1;    b[1]=1;    b[2]=2;
    b[3]=2;    b[4]=3;    b[5]=3;
```

see more

Reply • Share ›