Data Structures Algorithms Interview Preparation Topic-wise Practice C++ Java Python Competitive Programming Mac

Given two unsorted arrays, find all pairs whose sum is x

Difficulty Level : Easy • Last Updated : 25 Jan, 2022

Given two unsorted arrays of distinct elements, the task is to find all pairs from both arrays whose sum is equal to **X**.

Examples:

```
Input :  arr1[] = {-1, -2, 4, -6, 5, 7}
         arr2[] = {6, 3, 4, 0}
         x = 8
Output : 4 4
         5 3

Input : arr1[] = {1, 2, 4, 5, 7}
        arr2[] = {5, 6, 3, 4, 8}
        x = 9
Output : 1 8
```

Recommended: Please solve it on "***PRACTICE***" first, before moving on to the solution.

A **Naive approach** is to simply run two loops and pick elements from both arrays. One by one check that both elements sum is equal to given value x or not.

C++

```
// C++ program to find all pairs in both arrays
// whose sum is equal to given value x
#include <bits/stdc++.h>

using namespace std;

// Function to print all pairs in both arrays
// whose sum is equal to given value x
void findPairs(int arr1[], int arr2[], int n,
               int m, int x)
{
    for (int i = 0; i < n; i++)
        for (int j = 0; j < m; j++)
            if (arr1[i] + arr2[j] == x)
                cout << arr1[i] << " "
                     << arr2[j] << endl;
    int arr1[] = { 1, 2, 3, 7, 5, 4 };
    int arr2[] = { 0, 7, 4, 3, 2, 1 };
    int n = sizeof(arr1) / sizeof(int);
    int m = sizeof(arr2) / sizeof(int);
    int x = 8;
    findPairs(arr1, arr2, n, m, x);
    return 0;
}
```

Java

```
// Java program to find all pairs in both arrays
// whose sum is equal to given value x
import java.io.*;

class GFG {

    // Function to print all pairs in both arrays
    // whose sum is equal to given value x
    static void findPairs(int arr1[], int arr2[], int n,
                          int m, int x)
    {
        for (int i = 0; i < n; i++)
            for (int j = 0; j < m; j++)
                if (arr1[i] + arr2[j] == x)
                    System.out.println(arr1[i] + " "
                                       + arr2[j]);
    }
```

```
        int arr2[] = { 0, 7, 4, 3, 2, 1 };
        int x = 8;
        findPairs(arr1, arr2, arr1.length, arr2.length, x);
    }
}

// This code is contributed
// by sunnysingh
```

Python3

```
# Python 3 program to find all
# pairs in both arrays whose
# sum is equal to given value x

# Function to print all pairs
# in both arrays whose sum is
# equal to given value x
def findPairs(arr1, arr2, n, m, x):

    for i in range(0, n):
        for j in range(0, m):
            if (arr1[i] + arr2[j] == x):
                print(arr1[i], arr2[j])

# Driver code
arr1 = [1, 2, 3, 7, 5, 4]
arr2 = [0, 7, 4, 3, 2, 1]
```

```
// This code is contributed by Smitha Dinesh Semwal
```

C#

```
// C# program to find all
// pairs in both arrays
// whose sum is equal to
// given value x
using System;

class GFG {

    // Function to print all
    // pairs in both arrays
    // whose sum is equal to
    // given value x
    static void findPairs(int[] arr1, int[] arr2,
                          int n, int m, int x)
    {
        for (int i = 0; i < n; i++)
            for (int j = 0; j < m; j++)
                if (arr1[i] + arr2[j] == x)
                    Console.WriteLine(arr1[i] + " " + arr2[j]);
    }

    // Driver code
    static void Main()
    {
```

```
                  arr2.length, x);
    }
}

// This code is contributed
// by Sam007
```

PHP

```
<?php
// PHP program to find all pairs
// in both arrays whose sum is
// equal to given value x

// Function to print all pairs
// in both arrays whose sum is
// equal to given value x
function findPairs($arr1, $arr2,
                   $n, $m, $x)
{
    for ($i = 0; $i < $n; $i++)
        for ($j = 0; $j < $m; $j++)
            if ($arr1[$i] + $arr2[$j] == $x)
                echo $arr1[$i] . " " .
                     $arr2[$j] . "\n";
}

// Driver code
```

```
findPairs($arr1, $arr2,
          $n, $m, $x);

// This code is contributed
// by Sam007
?>
```

Javascript

```
<script>

// Javascript program to find all pairs in both arrays
// whose sum is equal to given value x

// Function to print all pairs in both arrays
// whose sum is equal to given value x
function findPairs(arr1, arr2, n, m, x)
{
    for (let i = 0; i < n; i++)
        for (let j = 0; j < m; j++)
            if (arr1[i] + arr2[j] == x)
                document.write(arr1[i] + " "
                     + arr2[j] + "<br>");
}

// Driver code
    let arr1 = [ 1, 2, 3, 7, 5, 4 ];
    let arr2 = [ 0, 7, 4, 3, 2, 1 ];
```

```
// This code is contributed by Surbhi Tyagi.

</script>
```

Output:

```
1 7
7 1
5 3
4 4
```

Time Complexity : O(n^2)

Auxiliary Space : O(1)

Searching Approach : As we know sorting algorithms can sort data in O (n log n) time. So we will choose a O (n log n) time algorithm like : Quick Sort or Heap Sort. For each element of second array , we will subtract it from K and search it in the first array.

Steps:

1. First sort the given array using a O(n log n) algorithm like Heap Sort or Quick Sort.
2. Run a loop for each element of array-B (0 to n).
3. Inside the loop, use a temporary variable say temp, and temp = K - B[i].

C++

```
#include<bits/stdc++.h>
using namespace std;

void heapify(int a[] , int n , int i)
{
    int rootLargest = i;
    int lchild = 2 * i;
    int rchild = (2 * i) + 1;

    if (lchild < n && a[lchild] > a[rootLargest])
        rootLargest = lchild;

    if (rchild < n && a[rchild] > a[rootLargest])
        rootLargest = rchild;

    if (rootLargest != i)
    {
        swap(a[i] , a[rootLargest]);

        //Recursion
        heapify(a , n , rootLargest);
    }
}

int binarySearch(int a[] , int l , int r , int x)
```

```
            return m;

        if (a[m] < x)
            l = m + 1;

        else
            r = m - 1;
    }
    return -1;
}

int main()
{
    int A[] = {1,2,1,3,4};
    int B[] = {3,1,5,1,2};

    int K = 8;

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

    // Building the heap
    for (int i = n / 2 - 1 ; i >= 1; i--)
        heapify(A , n , i);

    for(int i=0 ; i<n ; i++)                      //O(n)
    {
        int temp = K - B[i];                      //O(1)

        if(binarySearch(A , 0 , n-1 , temp))      //O(logn)
    return 0;
}
```

Java

```
import java.util.*;

class GFG{
  static    int A[] = {1,2,1,3,4};
  static void heapify( int n , int i)
  {
    int rootLargest = i;
    int lchild = 2 * i;
    int rchild = (2 * i) + 1;

    if (lchild < n && A[lchild] > A[rootLargest])
      rootLargest = lchild;

    if (rchild < n && A[rchild] > A[rootLargest])
      rootLargest = rchild;

    if (rootLargest != i)
    {
      int t = A[i];
      A[i] = A[rootLargest];
      A[rootLargest] = t;
      //Recursion
```

```
{
  while (l <= r)
  {
    int m = l + (r - l) / 2;

    if (A[m] == x)
      return m;

    if (A[m] < x)
      l = m + 1;

    else
      r = m - 1;
  }
  return -1;
}

public static void main(String[] args)
{

  int B[] = {3,1,5,1,2};

  int K = 8;

  int n = A.length;

  // Building the heap
  for (int i = n / 2 - 1 ; i >= 1; i--)
    heapify( n , i);
```

```
      {
        System.out.print("\nFound the elements.\n");
        break;
      }
    }
  }
}

// This code is contributed by Rajput-Ji
```

Python3

```
A = [ 1, 2, 1, 3, 4 ];

def heapify(n, i):
    rootLargest = i;
    lchild = 2 * i;
    rchild = (2 * i) + 1;

    if (lchild < n and A[lchild] > A[rootLargest]):
        rootLargest = lchild;

    if (rchild < n and A[rchild] > A[rootLargest]):
        rootLargest = rchild;

    if (rootLargest != i):
        t = A[i];
        A[i] = A[rootLargest];
```

```
def binarySearch(l, r, x):
    while (l <= r):
        m = l + (r - l) // 2;

        if (A[m] == x):
            return m;

        if (A[m] < x):
            l = m + 1;

        else:
            r = m - 1;

    return -1;

if __name__ == '__main__':

    B = [ 3, 1, 5, 1, 2 ];

    K = 8;

    n = len(A);

    # Building the heap
    for i in range(n// 2 - 1,0, -1):
        heapify(n, i);

    for i in range(n):
        temp = K - B[i];
```

C#

```
using System;
public class GFG {
  static int []A = { 1, 2, 1, 3, 4 };

  static void heapify(int n, int i) {
    int rootLargest = i;
    int lchild = 2 * i;
    int rchild = (2 * i) + 1;

    if (lchild < n && A[lchild] > A[rootLargest])
      rootLargest = lchild;

    if (rchild < n && A[rchild] > A[rootLargest])
      rootLargest = rchild;

    if (rootLargest != i) {
      int t = A[i];
      A[i] = A[rootLargest];
      A[rootLargest] = t;
      // Recursion
      heapify(n, rootLargest);
    }
  }

  static int binarySearch(int l, int r, int x) {
      if (A[m] < x)
        l = m + 1;

      else
        r = m - 1;
    }
    return -1;
  }

  public static void Main(String[] args) {

    int []B = { 3, 1, 5, 1, 2 };
    int K = 8;
    int n = A.Length;

    // Building the heap
    for (int i = n / 2 - 1; i >= 1; i--)
      heapify(n, i);

    for (int i = 0; i < n; i++) // O(n)
    {
      int temp = K - B[i]; // O(1)

      if (binarySearch(0, n - 1, temp - 1) != -1) // O(logn)
      {
        Console.Write("\nFound the elements.\n");
        break;
      }
    }
```

Javascript

```
<script>
function heapify(a,n,i)
{
    let rootLargest = i;
    let lchild = 2 * i;
    let rchild = (2 * i) + 1;

    if (lchild < n && a[lchild] > a[rootLargest])
        rootLargest = lchild;

    if (rchild < n && a[rchild] > a[rootLargest])
        rootLargest = rchild;

    if (rootLargest != i)
    {
        swap(a[i] , a[rootLargest]);

        //Recursion
        heapify(a , n , rootLargest);
    }
}

function binarySearch(a,l,r,x)
{
    while (l <= r)
    {
            l = m + 1;

        else
            r = m - 1;
    }
    return -1;
}
    let A = [1,2,1,3,4];
    let B = [3,1,5,1,2];

    let K = 8;

    let n = A.length;

    // Building the heap
    for (let i = n / 2 - 1 ; i >= 1; i--)
        heapify(A , n , i);

    for(let i=0 ; i<n ; i++)                 //O(n)
    {
        let temp = K - B[i];                 //O(1)

        if(binarySearch(A , 0 , n-1 , temp)) //O(logn)
        {
            document.write("\nFound the elements.\n");
            break;
        }
    }
```

```
Found the elements
```

Time Complexity: O(n logn).

O(log n + n log n)

So, Overall time complexity = O(n logn).

An **Efficient solution** of this problem is to hashing. Hash table is implemented using unordered_set in C++.

- We store all first array elements in hash table.
- For elements of second array, we subtract every element from x and check the result in hash table.
- If result is present, we print the element and key in hash (which is an element of first array).

C++

```
// C++ program to find all pair in both arrays
// whose sum is equal to given value x
#include <bits/stdc++.h>
using namespace std;
```

```
    // Insert all elements of first array in a hash
    unordered_set<int> s;
    for (int i = 0; i < n; i++)
        s.insert(arr1[i]);

    // Subtract sum from second array elements one
    // by one and check it's present in array first
    // or not
    for (int j = 0; j < m; j++)
        if (s.find(x - arr2[j]) != s.end())
            cout << x - arr2[j] << " "
                 << arr2[j] << endl;
}

// Driver code
int main()
{
    int arr1[] = { 1, 0, -4, 7, 6, 4 };
    int arr2[] = { 0, 2, 4, -3, 2, 1 };
    int x = 8;
    int n = sizeof(arr1) / sizeof(int);
    int m = sizeof(arr2) / sizeof(int);
    findPairs(arr1, arr2, n, m, x);
    return 0;
}
```

Java

```
    // Function to find all pairs in both arrays
    // whose sum is equal to given value x
    public static void findPairs(int arr1[], int arr2[],
                                int n, int m, int x)
    {
        // Insert all elements of first array in a hash
        HashMap<Integer, Integer> s = new HashMap<Integer, Integer>();

        for (int i = 0; i < n; i++)
            s.put(arr1[i], 0);

        // Subtract sum from second array elements one
        // by one and check it's present in array first
        // or not
        for (int j = 0; j < m; j++)
            if (s.containsKey(x - arr2[j]))
                System.out.println(x - arr2[j] + " " + arr2[j]);
    }

    /* Driver program to test above function */
    public static void main(String[] args)
    {
        int arr1[] = { 1, 0, -4, 7, 6, 4 };
        int arr2[] = { 0, 2, 4, -3, 2, 1 };
        int x = 8;

        findPairs(arr1, arr2, arr1.length, arr2.length, x);
    }
}
```

```python
# Python3 program to find all
# pair in both arrays whose
# sum is equal to given value x

# Function to find all pairs
# in both arrays whose sum is
# equal to given value x
def findPairs(arr1, arr2, n, m, x):

    # Insert all elements of
    # first array in a hash
    s = set()
    for i in range (0, n):
        s.add(arr1[i])

    # Subtract sum from second
    # array elements one by one
    # and check it's present in
    # array first or not
    for j in range(0, m):
        if ((x - arr2[j]) in s):
            print((x - arr2[j]), '', arr2[j])

# Driver code
arr1 = [1, 0, -4, 7, 6, 4]
arr2 = [0, 2, 4, -3, 2, 1]
x = 8

n = len(arr1)
```

C#

```
// C# Code for Given two unsorted arrays,
// find all pairs whose sum is x
using System;
using System.Collections.Generic;

class GFG {

    // Function to find all pairs in
    // both arrays whose sum is equal
    // to given value x
    public static void findPairs(int[] arr1, int[] arr2,
                                int n, int m, int x)
    {
        // Insert all elements of first
        // array in a hash
        Dictionary<int,
                   int>
            s = new Dictionary<int,
                              int>();

        for (int i = 0; i < n; i++) {
            s[arr1[i]] = 0;
        }

        // Subtract sum from second array
```

```
            }
        }
    }

    // Driver Code
    public static void Main(string[] args)
    {
        int[] arr1 = new int[] { 1, 0, -4, 7, 6, 4 };
        int[] arr2 = new int[] { 0, 2, 4, -3, 2, 1 };
        int x = 8;

        findPairs(arr1, arr2, arr1.Length,
                  arr2.Length, x);
    }
}

// This code is contributed by Shrikant13
```

Javascript

```
<script>

//  Javascript Code for Given two unsorted arrays,
// find all pairs whose sum is x

    // Function to find all pairs in both arrays
    // whose sum is equal to given value x
```

```
      s.set(arr1[i], 0);

    // Subtract sum from second array elements one
    // by one and check it's present in array first
    // or not
    for (let j = 0; j < m; j++)
      if (s.has(x - arr2[j]))
        document.write(x - arr2[j] + " " + arr2[j] + "<br/>");
  }

  // Driver code

  let arr1 = [ 1, 0, -4, 7, 6, 4 ];
    let arr2 = [ 0, 2, 4, -3, 2, 1 ];
    let x = 8;

    findPairs(arr1, arr2, arr1.length, arr2.length, x);

</script>
```

Output:

```
6 2
4 4
```

Auxiliary Space: O(n)

This article is contributed by **DANISH_RAZA** . If you like GeeksforGeeks and would like to contribute, you can also write an article using write.geeksforgeeks.org or mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.
Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Like 30

RECOMMENDED ARTICLES

Page : 1 2 3

01 Given two arrays count all pairs whose sum is an odd number
26, Apr 19

02 Count pairs from two sorted arrays whose sum is equal to a given value x
23, Jun 17

03 Find Union and Intersection of two unsorted arrays
01, May 15

04 Print all pairs in an unsorted array with equal sum
22, Nov 18

05 Find XOR sum of Bitwise AND of all pairs from given two Arrays
21, Jun 21

06 Smallest Difference pair of values between two unsorted Arrays
11, May 17

07 Merging two unsorted arrays in sorted order
29, Jul 17

08 Count pairs from two arrays whose modulo operation yields K
22, Nov 18

GeeksforGeeks

Vote for difficulty

Current difficulty : Easy

Easy Normal Medium Hard Expert

Improved By : Sam007, namankatuva, ihritik, shrikanth13, Mayuri Gupta 4, sp999, RishabhAgnihotri, surbhityagi15, imsushant12, avijitmondal1998, vaibhavrabadiya3, Rajput-Ji

Article Tags : Amazon, Facebook, Arrays, Hash

Practice Tags : Amazon, Facebook, Arrays, Hash

Improve Article Report Issue

GeeksforGeeks

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org

Company

About Us

Careers

Privacy Policy

Contact Us

Copyright Policy

Learn

Algorithms

Data Structures

Languages

CS Subjects

Video Tutorials

Web Development

Web Tutorials

HTML

CSS

JavaScript

Contribute

Write an Article

Write Interview Experience

Internships

Videos