Given an array A[] and a number x, check for pair in A[] with sum as x

Write a C program that, given an array A[] of n numbers and another number x, determines whether or not there exist two elements in S whose sum is exactly x.

METHOD 1 (Use Sorting)

Algorithm:

```
hasArrayTwoCandidates (A[], ar_size, sum)
1) Sort the array in non-decreasing order.
2) Initialize two index variables to find the candidate
   elements in the sorted array.
       (a) Initialize first to the leftmost index: l = 0
       (b) Initialize second  the rightmost index:  r = ar_size-1
3) Loop while l < r.
       (a) If (A[l] + A[r] == sum)  then return 1
       (b) Else if( A[l] + A[r] <  sum )  then l++
       (c) Else r--
4) No candidates in whole array - return 0
```

Time Complexity: Depends on what sorting algorithm we use. If we use Merge Sort or Heap Sort then (-)(nlogn) in worst case. If we use Quick Sort then O(n^2) in worst case.
Auxiliary Space : Again, depends on sorting algorithm. For example auxiliary space is O(n) for merge sort and O(1) for Heap Sort.

Example:
Let Array be {1, 4, 45, 6, 10, -8} and sum to find be 16

Sort the array
A = {-8, 1, 4, 6, 10, 45}

Initialize l = 0, r = 5
A[l] + A[r] (-8 + 45) > 16 => decrement r. Now r = 10
A[l] + A[r] (-8 + 10) < 2 => increment l. Now l = 1
A[l] + A[r] (1 + 10) < 16 => increment l. Now l = 2
A[l] + A[r] (4 + 10) < 14 => increment l. Now l = 3

A[l] + A[r] (6 + 10) == 16 => Found candidates (return 1)

Note: If there are more than one pair having the given sum then this algorithm reports only one. Can be easily extended for this though.

Implementation:

C

```
# include <stdio.h>
# define bool int

void quickSort(int *, int, int);

bool hasArrayTwoCandidates(int A[], int arr_size, int sum)
{
    int l, r;

    /* Sort the elements */
    quickSort(A, 0, arr_size-1);

    /* Now look for the two candidates in the sorted
       array*/
    l = 0;
    r = arr_size-1;
    while (l < r)
    {
         if(A[l] + A[r] == sum)
              return 1;
         else if(A[l] + A[r] < sum)
              l++;
         else // A[i] + A[j] > sum
              r--;
    }
    return 0;
}

/* Driver program to test above function */
int main()
{
    int A[] = {1, 4, 45, 6, 10, -8};
    int n = 16;
    int arr_size = 6;

    if( hasArrayTwoCandidates(A, arr_size, n))
        printf("Array has two elements with sum 16");
    else
        printf("Array doesn't have two elements with sum 16 ");
    getchar();
    return 0;
}

/* FOLLOWING FUNCTIONS ARE ONLY FOR SORTING
    PURPOSE */
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);
    }
}
```

Python

```
# Python program to check for the sum condition to be satisified
def hasArrayTwoCandidates(A,arr_size,sum):

    # sort the array
    quickSort(A,0,arr_size-1)
    l = 0
    r = arr_size-1

    # traverse the array for the two elements
    while l<r:
        if (A[l] + A[r] == sum):
            return 1
        elif (A[l] + A[r] < sum):
            l += 1
        else:
            r -= 1
    return 0

# Implementation of Quick Sort
# A[] --> Array to be sorted
# si  --> Starting index
# ei  --> Ending index
def quickSort(A, si, ei):
    if si < ei:
        pi=partition(A,si,ei)
        quickSort(A,si,pi-1)
        quickSort(A,pi+1,ei)

# Utility function for partitioning the array(used in quick sort)
def partition(A, si, ei):
    x = A[ei]
    i = (si-1)
    for j in range(si,ei):
        if A[j] <= x:
            i += 1

            # This operation is used to swap two variables is python
            A[i], A[j] = A[j], A[i]

        A[i+1], A[ei] = A[ei], A[i+1]

    return i+1


# Driver program to test the functions
A = [1,4,45,6,10,-8]
n = 16
if (hasArrayTwoCandidates(A, len(A), n)):
    print("Array has two elements with the given sum")
else:
    print("Array doesn't have two elements with the given sum")

## This code is contributed by __Devesh Agrawal__
```

Output:

```
Array has two elements with the given sum
```

METHOD 2 (Use Hash Map)

Thanks to Bindu for suggesting this method and thanks to Shekhu for providing code.

This method works in O(n) time if range of numbers is known.

Let sum be the given sum and A[] be the array in which we need to find pair.

```
1) Initialize Binary Hash Map M[] = {0, 0, ...}
2) Do following for each element A[i] in A[]
   (a)  If M[x - A[i]] is set then print the pair (A[i], x - A[i])
   (b)  Set M[A[i]]
```

Implementation:

C/C++

```
#include <stdio.h>
#define MAX 100000

void printPairs(int arr[], int arr_size, int sum)
{
  int i, temp;
  bool binMap[MAX] = {0}; /*initialize hash map as 0*/

  for (i = 0; i < arr_size; i++)
  {
      temp = sum - arr[i];
      if (temp >= 0 && binMap[temp] == 1)
         printf("Pair with given sum %d is (%d, %d) \n",
                 sum, arr[i], temp);
      binMap[arr[i]] = 1;
  }
}

/* Driver program to test above function */
int main()
{
    int A[] = {1, 4, 45, 6, 10, 8};
    int n = 16;
    int arr_size = sizeof(A)/sizeof(A[0]);

    printPairs(A, arr_size, n);

    getchar();
    return 0;
}
```

Java

```
// Java implementation using Hashing
import java.io.*;

class PairSum
{
    private static final int MAX = 100000; // Max size of Hashmap

    static void printpairs(int arr[],int sum)
    {
        // Declares and initializes the whole array as false
        boolean[] binmap = new boolean[MAX];

        for (int i=0; i<arr.length; ++i)
        {
            int temp = sum-arr[i];

            // checking for condition
            if (temp>=0 && binmap[temp])
            {
                System.out.println("Pair with given sum " +
                                   sum + " is (" + arr[i] +
                                   ", "+temp+")");
            }
            binmap[arr[i]] = true;
        }
    }

    // Main to test the above function
    public static void main (String[] args)
    {
        int A[] = {1, 4, 45, 6, 10, 8};
        int n = 16;
        printpairs(A,  n);
    }
}

// This article is contributed by Aakash Hasija
```

Python

```
# Python program to find if there are two elements wtih given sum
CONST_MAX = 100000

# function to check for the given sum in the array
def printPairs(arr, arr_size, sum):

    # initialize hash map as 0
    binmap = [0]*CONST_MAX

    for i in range(0,arr_size):
        temp = sum-arr[i]
        if (temp>=0 and binmap[temp]==1):
            print "Pair with the given sum is", arr[i], "and", temp
        binmap[arr[i]]=1

# driver program to check the above function
A = [1,4,45,6,10,-8]
n = 16
printPairs(A, len(A), n)

# This code is contributed by __Devesh Agrawal__
```

Time Complexity: O(n)
Output:

```
Pair with given sum 16 is (10, 6)
```

Auxiliary Space: O(R) where R is range of integers.

If range of numbers include negative numbers then also it works. All we have to do for negative numbers is to make everything positive by adding the absolute value of smallest negative integer to all numbers.

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

311 Comments Category: Arrays Hash Tags: Hashing

Related Posts:

- Longest Span with same Sum in two Binary arrays
- Count Inversions of size three in a give array
- Find the subarray with least average
- Count triplets with sum smaller than a given value
- Find zeroes to be flipped so that number of consecutive 1's is maximized
- Reorder an array according to given indexes
- Find maximum value of Sum(i*arr[i]) with only rotations on given array allowed
- Find maximum average subarray of k length

Writing code in comment? Please use code.geeksforgeeks.org, generate link and share the link here.