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
```

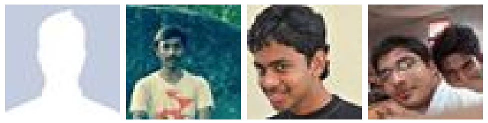
Time Complexity: Depends on what sorting algorithm we use. If we use Merge Sort or Heap Sort then (-)(nlogn) in worst case. If we use Quick Sort then O(n^2) in worst case.
Auxiliary Space : Again, depends on sorting algorithm. For example auxiliary space is O(n) for merge sort and O(1) for Heap Sort.

Example:

Google™ Custom Search

GeeksforGeeks

53,521 people like GeeksforGeeks.



Interview Experiences
Advanced Data Structures
Dynamic Programming
Greedy Algorithms
Backtracking
Pattern Searching
Divide & Conquer
Mathematical Algorithms
Recursion

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:

```
# 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;
}
```

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

```c
/* 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);
    }
}
```

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]]
```

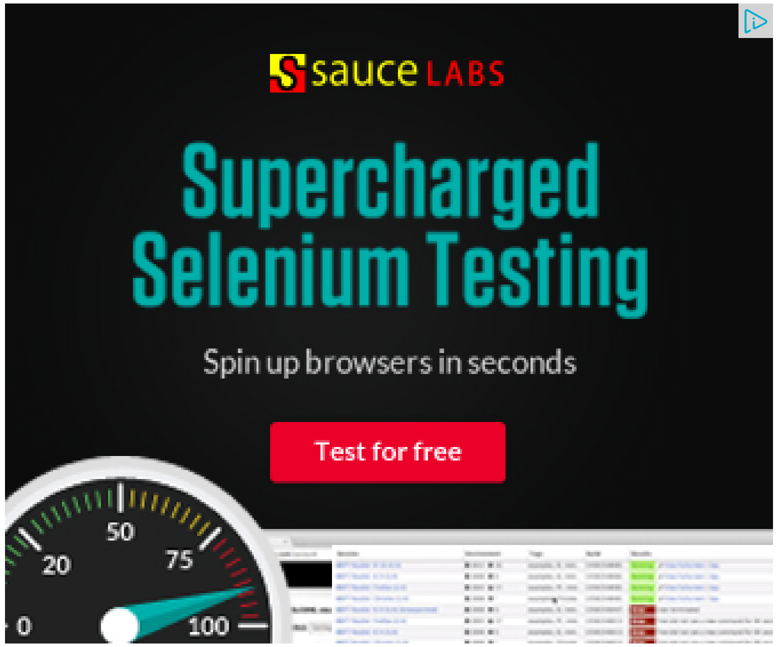
Implementation:

```
#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], tem
    }
    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 = 6;

    printPairs(A, arr_size, n);

    getchar();
    return 0;
}
```

Time Complexity: O(n)
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.



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

21 Tweet 0 4

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

86 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

nani • 6 days ago

Write a C++ program considering the following:
1- Define a 1 dimensional array A of 10 integers.
DisplayArray is a function that accepts an array and displays the array elemen
3- ArraySum is a function that returns the sum of the array elements.
4- displayRever is a function that accepts an array and displays the array elements in reverse order

Reply • Share ›

Anil kumar • 15 days ago

Method 2 has a bug. What if temp <0

Reply • Share ›

Abhinav Choudhury • 24 days ago

Second method: Can't we just use a hash table and say the average time con
case, we do not need to worry about the range of numbers, if we use a simple
worse case time complexity may well be O(n^2), but auxiliary space will be O(

Reply • Share ›

abhismart92 • a month ago

i didn't understand why code is depended on range of numbers?????

Reply • Share ›

Aishwarya Kr Singh abhismart92 • a month ago

let's say n_Max is maximum among all input numbers and let's assum
are going to put a[n_Max]=1 to notify later that this element is present i
array of size n_Max+1 and hence space complexity will be O(n_Max) c
integers

Reply • Share ›

Rachit Chawla • 3 months ago

I didn't understand the second method.. please explain the if condition in the fc

Reply • Share ›

Jonathan Chen Rachit Chawla • 2 months ago

So let's say we found a value A. We want a value B such that A+B=Su

The if statement is saying look at index B of our HashMap. If the value i
B before (earlier in the looping). Thus, we are done, return A and B.

Otherwise, we haven't come across B before. So, instead, we will cha
HashMap from 0 to 1 to indicate we have an A. In case we come acros

Sumit · 3 months ago

Method 2 stating that if range of numbers are given,,,
If the range is known then we can sort them in O(n) time using counting or rad
O(n) time.

Correct me if i m wrong.

^ | ⌄ · Reply · Share ›

Jonathan Chen ➦ Sumit · 2 months ago

Method 2 doesn't use any sorting.

If we know the range of numbers, we can simply create (in this case) a
of numbers. Each index in the hashmap corresponds to that number. (
in our problem).

We loop through the given array A.
Do 'Sum' - 'A[i]' = (Sum - A[i]) <- whatever the value is.

So, for A[i] + 'some number' to equals 'Sum', we need 'some number'

Check index 'Sum - A[i]' in the hashmap to see if we have ever come a
index 'Sum - A[i]' is set to 1, then we have the two numbers necessary

If not, then we mark index 'A[i]' from 0 to 1 to mark that we have such a
across 'Sum - A[i]' in the future).

My question is: Do we even need to know the range of numbers (espe
super familiar with the langauge).

see more

1 ^ | ⌄ · Reply · Share ›

joenjoin · 4 months ago

What if the pair is not limited to 2 numbers, but as many numbers as possible
Thanks.

· Reply · Share ›

realsteel joenjoin · a month ago

meet in middle algorithm !!!

· Reply · Share ›

deepak · 5 months ago

a. User input 11.5, 12, -2.5
and 0.0 (in textarea in 4 lines) b. User input 23.5 in the text field. c. User clicks
d. The output would be 11.5, 12, -2.5 and 0.0 (in div in 4 lines) with line numbe
(which are 11.5 and 12) because they can add up to 23.5. e. If user input 12 in
line 2 (which is 12) meaning 12 itself one line only. f. If user input 3, highlight n
"No numbers can add up to this" beside the text field.

1 · Reply · Share ›

Amit Baghel · 6 months ago

Interesting :D

· Reply · Share ›

abhishek · 6 months ago

// handling of duplicates included

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

void swap(int A[], int p, int r) {
int temp = A[p];
A[p] = A[r];
A[r] = temp;
```

```
}

int partition(int A[], int p, int r) {
int i = p+1;
int j = r;

while(i < j) {
while(i <= r && A[i] <= A[p])
i++;
```

see more

^ | ⌄ • Reply • Share ›

Harjit Singh • 6 months ago

Here is the working code for all the cases. Key is to use sorting and hashing.

```
#include<iostream>
#include<conio.h>
using namespace std;
int compare (const void * a, const void * b)
{
return ( *(short int*)a - *(short int*)b );
}
void display(int A[],int n)
{
for(int i=0;i<n;i++) cout<<a[i]<<"="" ";="" }="" void="" add_hash(int="" h[],int=""
check_hash(int="" h[],int="" num)="" {="" return(h[num]);="" }="" int="" min(int=
return b;
else
return a;
}
void find_pairs(int A[],int n,int x)
```

see more

^ | ⌄ • Reply • Share ›

kkk • 7 months ago

what if elements are equal.

^ | ⌄ • Reply • Share ›

MayankSwarnkar • 7 months ago

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

```
#include<iostream>
#include<vector>
#include<cstdlib>
#include<algorithm>

using namespace std;

int main()
{
vector<int> v;
vector<int>::iterator pos1,pos2;
int i,n;
for(i=1;i<101;i++)
v.push_back(rand()%100);
for(pos1=v.begin();pos1!=v.end();pos1++)
cout<<*pos1<<endl; sort(v.begin(),v.end());="" cout<<"enter="" the="" numbe
for(pos1=v.begin();pos1!=v.end();pos1++)
{
for(pos2=pos1;pos2!=v.end();pos2++)
{
if(*pos1+*pos2==n)
```

cout<<"One of the pair whose sum is equal to the given number is-"<<endl; c
}="" }="" }="" return="" 0;="" }="">

Reply • Share ›

Harjit Singh MayankSwarnkar • 6 months ago

```
#include<iostream>
#include<conio.h>
using namespace std;
int compare (const void * a, const void * b)
{
  return ( *(short int*)a - *(short int*)b );
}
void display(int A[],int n)
{
     for(int i=0;i<n;i++) cout<<a[i]<<"="" ";="" }="" void="" a
       return b;
    else
       return a;
}
void find_pairs(int A[],int n,int x)
{


         int h[x/2+1];
```

see more

Reply • Share ›

mahi2 • 7 months ago

One of the approach is ...if n is the number
for(i=0;i<n;i++) {num="n-a[i]" {for(j="i+1;j&lt;n;j++)" {search="" for="" num..if="
the="" result!="" }="" }="">

din • 8 months ago

can this be done in O(n) without hashmap??

• Reply • Share ›

Varunvats • 10 months ago

[sourcecode language="C++"]
void printPairs(int A[], int n, int x) {
map<int,int> m_map;
for(int i=0; i<n; i++){
if(m_map[A[i]]){
cout<<"index:"<<m_map[A[i]]<<" "<<i;
break;
}
else
m_map[x-A[i]]=i;
}
}

2 • Reply • Share ›

Abhay • 10 months ago

Another approach, (PS: Not read the comments , so sorry if already mentione
array linearly , searching for x-a[i]. If found return true , else false.

```
 import java.util.Arrays;
class Solver
{

    public boolean SolverUtil(int a[],int x)
    {
        Arrays.sort(a);
        int cnt=0;
        for(int i=0;i<End-1;i++)
        {
            if(Binsearch(a,x-a[i],0,End))
                    return true;
        }
        return false;
```

see more

^ | v • Reply • Share ›

Ronny ➦ Abhay • 9 months ago

But what if the sum to be found is twice a number present but that num
Eg. arr[]={1,5,9,12,24};
and sum to be searched is 18. your procedure will return true for this c

Refer http://ideone.com/KCFI4j

2 ^ | v • Reply • Share ›

Kartik ➦ Ronny • 3 months ago

This case can be easily handled by augmenting Binsearch rout
http://ideone.com/wZF7vl

^ | v • Reply • Share ›

Dinesh Kumar M • 11 months ago

thz

^ | v • Reply • Share ›

Mahendra Mundru • 11 months ago

//O(max(range, n)) time ; O(range) auxiliary space.
int pairForSumXv2(int a[], int length, int x){.

```
int hashMap[range+1], i,is=0;
for(i=0;i<range+1;i++) hashMap[i]=0;
for(i=0;i<length;i++){
if(hashMap[x-a[i]]){
printf("The elements with sum %d are (%d,%d)n", x,a[i], x-a[i]);.
hashMap[x-a[i]]=hashMap[a[i]]=0;
is=1;
}
else
hashMap[a[i]]=1;
}
if(is==1)
return 1;
return 0;
}
```

see more

• Reply • Share ›

rkl • a year ago

```
public class Sum {

        public static void main(String[] args) {
                int[] arr = {3, -2, 5, 6, 1, 9, 7, 8, 12, 4};

                printPairs(arr, 10);
        }

        private static void printPairs(int[] arr, int sum) {

                HashMap<Integer, Boolean> map = new HashMap<Integer, I
                for(int i = 0; i < 10; i++) {
                        if (map.containsKey(sum-arr[i])) {
                                System.out.println(arr[i] + " " + (su
                        } else {
                                map.put(arr[i], true);
                        }
```

see more

1 ^ | ˅ • Reply • Share ›

Ashish Saxena ➦ rkl • 6 months ago

This is a java version of method 2. Here are few comments

For High N (Array Size). We also need to initialize HashMap with high c

By default the capacity is 16.

Please correct me if I am wrong.

^ | ˅ • Reply • Share ›

ebcdic666 • a year ago

For the hash table solution, we need not know the range of the numbers, we c
elements seen.

2 ^ | ˅ • Reply • Share ›

Ankush Acharyya • a year ago

(Y)

^ | ˅ • Reply • Share ›

pawan • a year ago

#include

int arr[6]={3,5,18,7,9,-12};

```
int sort(int arr_len)
{
int i,j;
printf("2\n");
for(i=0;i<arr_len;i++)
{
for(j=0;jarr[j+1])
{
int temp=arr[j];
arr[j]=arr[j+1];
arr[j+1]=temp;
}
}
}
```

see more

Reply • Share ›

abhishek08aug • a year ago

Here is the simple quadratic O(n^2) solution:

```
 #include<stdio.h>
int find_two_elements_with_given_sum(int array[], int sum, int array_s
{
  int i, j;
  for(i=0; i<array_size; i++) {
    for(j=i+1; j<array_size; j++) {
      if(array[i]+array[j]==sum) {
        return 1;
      }
    }
  }
  return 0;
}

int main()
{
  int array[] = {1, 4, 45, 5, 11, -8};
  int sum = 16;
  int array_size = sizeof(array)/sizeof(array[0]);
  printf("%d\n", find_two_elements_with_given_sum(array, sum, array_s
  return 0;
}
```

Reply • Share ›

Chinmaya • a year ago

Alternatively

1 - Sort the array - O(nlogn)
2 - for each element in the array,perform binary search for the element(sum-A
3 - If found, print success.

[sourcecode language="C++"]
find_sum_elements(vector<int> a[],int sum)
{
sort(a.begin(),a.end());
for(int i=0;i<a.size();i++)
{
if(binary_search(a.begin()+i+1,a.end(),sum - a[i]))
return SUCCESS;
}
}

^ | ˅ • Reply • Share ›

krishnx • a year ago

```
 #include<iostream>
#include<cstdlib>

using namespace std;

void findPair(int a[], int, int);
int compare(const void *a, const void *b);

int main()
{
  int a[] = {1, 4, 45, 6, 10, -8};
  int length = sizeof(a)/sizeof(a[0]);
  cout << length << endl;

  int sum = 0;
  cout << "Enter the sum: \n";
  cin >> sum;
```

see more

^ | ˅ • Reply • Share ›

Nikin Kumar Jain • a year ago

Please check out Little more optimized code for performing Quicksort.

```
 // QSortProject.cpp : Defines the entry point for the console applic
//

#include "stdafx.h"
```

```
using namespace std;


int partition(int arr[], int si, int ei)
{
        int i = 0;
        for(int j = 0; j < ei; j++)
        {
                if(arr[i] <= arr[ei])
                {
                        if(i!=j)
```

see more

^ | ⌄ • Reply • Share ›

Aman • a year ago

what if there are duplicate elements in second method

^ | ⌄ • Reply • Share ›

binary001 → Aman • a year ago

if there are duplicate a you want to print all the pairs ..
1) sort the array
2) create 2d array (int ar[n][2]) which store the no and its frequency .
3) now u got duplicate free array
4) use the normal algo. and at the time of printing use loop
for i =1 to f_X
for j =1 to f_Y
print x,y;

f_x and f_y are the frequency of X and Y

^ | ⌄ • Reply • Share ›

alien ➦ Aman • a year ago

doesn't matter even if there are duplicates. Put only first occurrence of because ultimately you have to find pair of numbers not indexes.

^ | ˅ • Reply • Share ›

alien • a year ago

another approach:

1.) put all the elements in a hash table

2.) Use that hashtable to find element sum-a[i] from i to N

```
#include
#include
#define MAX 6
#define MIN 0
#define LIMIT 100

void print(int arr[])
{
int i;
for(i=MIN;i<11;i++)
{
printf("%d, ",arr[i]);
}
}
```

see more

^ | ˅ • Reply • Share ›

Rohan • a year ago

I ran this code for int a[] = {-4,-4,-4,-4,-4,1,2,3,4,3,5,6,10};

even though this array contains -ve values HASH[-ve integer] is not giving any and code is running fine without adding any offset...

why is this happening??

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


int HASH[10000]={0};

int bin_map_solution(int a[])
{
        int i =0;
        int number = 6;
    int count=1;
```

see more

4 • Reply • Share ›

batfan47 • a year ago

Correcting Bad formatting in the previous comment

```
 #include <iostream>
#include <cmath>
#define MAX 100000

using namespace::std;
typedef long element;
typedef unsigned long counter;

long findmin(element term[], counter count, int sum)
{
```

```
counter i;
element min;
for(i = 1, min = term[0]; i != count; i++)
  if(term[i] < min)
    min = term [i];
if (min < 0)
```

see more

^ | ⌄ • Reply • Share ›

Kartik ➦ batfan47 • a year ago

This method seems to be similar to method 2 in the above post.

^ | ⌄ • Reply • Share ›

Ankit Malhotra ➦ Kartik • a year ago

It is exactly the same method but it has been changed to handl
sum.

^ | ⌄ • Reply • Share ›

batfan47 • a year ago

For bitmap using negative integers, twice the absolute value of the least negat
the value of the sum itself can also be negative and should be considered whil
Following is the code that includes this and solves the problem for negative va

//[sourcecode language="C++"]
#include <iostream>
#include <cmath>
#define MAX 100000

int findmin(int arr[], int arr_size, int sum)
{
int i, min;

```
for(i = 1, min = arr[0]; i < arr_size; i++)
{
if(arr[i] < min)
{
min = arr [i];
}
}
```

see more

1 ^ | ⌄ • Reply • Share ›

bunty • 2 years ago
@Geeks: Correction in second method's last comments,

Auxiliary Space: O(R) where R is range of integers.
" R can not be just the range of number given, as the sum of two numbers fro
the range of that numbers"
Ex; range could be {1,2,3,4,5} and sum is 9,
then Aux array must be at least 9 elements long.

Also these method does not need the info of the range of number.

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

^ | ⌄ • Reply • Share ›

Spock • 2 years ago
Well we can simply use array instead of hashmap.
1 ^ | ⌄ • Reply • Share ›

Anuj • 2 years ago
Can you elaborate the Method 2 ?
What is Hash map and why its subscript is arr[i]

Shyam • 2 years ago
What does "If range of numbers include negative numbers then also it works.
is scale everything with reference to the smallest negative integer in the given
explain this statement?

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

^ | ⌄ • Reply • Share ›

GeeksforGeeks ➦ Shyam • 2 years ago
@Shyam: We have changed the language so that it makes sense now

^ | ⌄ • Reply • Share ›

Nihal • 2 years ago
This one works well. No extra space required. And Time Complexity = o(n)

```
 #include <stdio.h>
#include<conio.h>
using namespace std;
int checkpair(int[] ,int, int);
int main()
{
    int a[5] = {6,1,18,3,4};
    bool result = checkpair(a,24,0);
    if(result)
    printf("pair exists");
    else
    printf("pair does not exists");
    getch();
    return 0;
}
```

see more

Reply • Share ›

Load more comments

Subscribe Add Disqus to your site