Find a triplet that sum to a given value

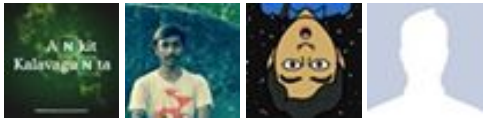
Given an array and a value, find if there is a triplet in array whose sum is equal to the given value. If there is such a triplet present in array, then print the triplet and return true. Else return false. For example, if the given array is {12, 3, 4, 1, 6, 9} and given sum is 24, then there is a triplet (12, 3 and 9) present in array whose sum is 24.

Method 1 (Naive)
A simple method is to generate all possible triplets and compare the sum of every triplet with the given value. The following code implements this simple method using three nested loops.

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

```
// returns true if there is triplet with sum equal
// to 'sum' present in A[]. Also, prints the triplet
bool find3Numbers(int A[], int arr_size, int sum)
{
    int l, r;

    // Fix the first element as A[i]
    for (int i = 0; i < arr_size-2; i++)
    {
        // Fix the second element as A[j]
        for (int j = i+1; j < arr_size-1; j++)
        {
            // Now look for the third number
            for (int k = j+1; k < arr_size; k++)
            {
                if (A[i] + A[j] + A[k] == sum)
                {
                  printf("Triplet is %d, %d, %d", A[i], A[j], A[k]);
                  return true;
                }
```

```
            }
        }
    }

    // If we reach here, then no triplet was found
    return false;
}

/* Driver program to test above function */
int main()
{
    int A[] = {1, 4, 45, 6, 10, 8};
    int sum = 22;
    int arr_size = sizeof(A)/sizeof(A[0]);

    find3Numbers(A, arr_size, sum);

    getchar();
    return 0;
}
```

Output:

```
Triplet is 4, 10, 8
```

Time Complexity: O(n^3)

Method 2 (Use Sorting)

Time complexity of the method 1 is O(n^3). The complexity can be reduced to O(n^2) by sorting the array first, and then using method 1 of this post in a loop.

1) Sort the input array.

2) Fix the first element as A[i] where i is from 0 to array size – 2. After fixing the first element of triplet, find the other two elements using method 1 of this post.

```
# include <stdio.h>

// A utility function to sort an array using Quicksort
void quickSort(int *, int, int);

// returns true if there is triplet with sum equal
// to 'sum' present in A[]. Also, prints the triplet
bool find3Numbers(int A[], int arr_size, int sum)
{
```

Geometric Algorithms

Popular Posts

```
    int l, r;

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

    /* Now fix the first element one by one and find the
       other two elements */
    for (int i = 0; i < arr_size - 2; i++)
    {

        // To find the other two elements, start two index variables
        // from two corners of the array and move them toward each
        // other
        l = i + 1; // index of the first element in the remaining elem
        r = arr_size-1; // index of the last element
        while (l < r)
        {
            if( A[i] + A[l] + A[r] == sum)
            {
                printf("Triplet is %d, %d, %d", A[i], A[l], A[r]);
                return true;
            }
            else if (A[i] + A[l] + A[r] < sum)
                l++;
            else // A[i] + A[l] + A[r] > sum
                r--;
        }
    }

    // If we reach here, then no triplet was found
    return false;
}

/* FOLLOWING 2 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);
    }
}

/* Driver program to test above function */
int main()
{
    int A[] = {1, 4, 45, 6, 10, 8};
    int sum = 22;
    int arr_size = sizeof(A)/sizeof(A[0]);

    find3Numbers(A, arr_size, sum);

    getchar();
    return 0;
```


Recent Comments

```
}
```

Output:

```
Triplet is 4, 8, 10
```

Time Complexity: O(n^2)

Note that there can be more than one triplet with the given sum. We can easily modify the above methods to print all triplets.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

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.

Prateek Sharma • 4 months ago

Simple O(nlogn) solution..

```
import java.util.Arrays;

public class TripletSumToAGivenValue {

int arr[]= new int[]{2,3,5,3,5};

int value = 10;

boolean flag = false;

void tripletSumToValue(){

System.out.print("Triplets are");

Arrays.sort(arr);

int i = 0;
```

see more

1 ^ | v • Reply • Share ›

chaks • 5 months ago

How about using this map technique for solving in O(n^2) ?

```
#include<iostream>
#include<map>
void findTriplet(int arr[], int n, int sum) {
std::map<int, bool=""> triplet;
int flag = 0;
for(int i = 0; i < n; i++) {
for(int j = i+1; j < n; j++) {
triplet[arr[j]] = false;
}
for(int j = i+1; j < n; j++) {
if(triplet[sum-arr[i]-arr[j]] == true) {
std::cout << "(" << arr[i] << "," << arr[j] << "," << sum-arr[i]-arr[j] << ")";
flag = 1;
goto outer;
}
triplet[arr[j]] = true;
```

see more

^ | v • Reply • Share ›

Guest • 6 months ago

recursive implementation

```
int isTriplet(int *a, int n, int sum , int count)
{
```

```
 return 1;
if(count >3 || n<=0)
 return 0
else
 return isTriplet(a,n-1,sum-a[n],count+1) || isTriplet(a,n-1,sum,cou
}
```

• Reply • Share ›

Guest • 7 months ago

In method 2 , Inner while loop can be replaced with a binary search as the arra
search complexity to nLogn. In fact we can still do little correction , when we s
right element already present, we can check weather the number to be found i
if(! (Sum - left - right) > left && (Sum - left - right) < right))
break;

1 • Reply • Share ›

Akhil • 11 months ago

```
 #include<stdio.h>
#include<stdlib.h>
int compare(const void *a, const void *b)
{
    return (*(int*)a-*(int*)b);
}
int bSearch(int a[], int low, int high, int target)
{
    while(low<=high)
    {
        int mid = (low+high)/2;
        if(a[mid]==target)
            return mid;
        else if(a[mid]>target)
```

```
            high = mid-1;
        else
            low = mid+1;
    }
```

see more

Reply • Share ›

Akhil Akhil • 11 months ago

My Bad!
Doesn't work for {1,2,3,4,5,6,7} find 6.

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

Reply • Share ›

Akhil • 11 months ago

A simple Concise O(NLogN) Approach!

[sourcecode language="C++"]
#include<stdio.h>
#include<stdlib.h>
int compare(const void *a, const void *b)
{
return (*(int*)a-*(int*)b);
}
int bSearch(int a[], int low, int high, int target)
{
while(low<=high)
{
int mid = (low+high)/2;
if(a[mid]==target)

else if(a[mid]>target)
high = mid-1;

see more

• Reply • Share ›

chirag ➚ Akhil • 3 months ago

i think u won't be able to get the triplet when-
a[]={4,5,6,8}
& the sum of 3 is 15.

• Reply • Share ›

jinzhi chen ➚ Akhil • 5 months ago

nice solution

• Reply • Share ›

venkatesh.bandaru • a year ago

```
public class Triplet {


   /**
    * @param args the command line arguments
    */
   public static void main(String[] args) {
       // TODO code application logic here


 int a[]={1,5,7,8,12,13};
 int size=6;
int i=0;
int j=i+1;
int k=size-1;
int s,sum=13;
```

```
{


if( i<k && j< k)
```

see more

Reply • Share ›

venkatesh.bandaru venkatesh.bandaru • a year ago

This is a O(n) algorithm, give me your comments if you find any bugs

Reply • Share ›

pritybhudolia venkatesh.bandaru • 11 months ago

This wont work for {1,3,5,8,12} sum=21.

Reply • Share ›

pritybhudolia venkatesh.bandaru • 11 months ago

Your solution will work only for sorted arrays.

1 Reply • Share ›

jinzhi chen pritybhudolia • 5 months ago

yeah, but for small numbers, we can use radix sort to a

Reply • Share ›

BIG BOOOBIES pritybhudolia • 11 months ago

bandar venkatesh{

bool isSolutionCorrect(){

algo!=O(n);
bugs=3+rand(n);
printf("you are an asshole");
return false;

}
}

 • Reply • Share ›

Ravi Teja Vadrevu • a year ago

There is a simple solution for this on the same lines of finding a SUM by pair o

Step 1. Store inorder traversal of a Tree into an arrayList; (sorted)

Step 2. Now traverse through arrayList once to form a HashMap of sum to pai
1+2 = 3 - > {1,2},
1+3 = 4 -> {1,3},
1+4 , 2+3 = 5 -> {1,4}, {2,3}
2+4 =6 -> {2,4}
3+4 = 7 -> {3,4}

Step 3. Now traverse through the array again and for each element try to find S
resultant pair + i is our triplet

Code :

Class BinaryTripletSum{

private List arrayList = new ArrayList();
private TreeNode root;

see more

 • Reply • Share ›

Nirdesh Mani Sharma. • a year ago

We can improve the efficiency of the Method 2 code by checking a condition a
pair of 2 because the data is in sorted order and its not possible to make a sur
negative.

coderAce • a year ago

Is O(n^2) the lower bound for this problem. Can anybody confirm on this?

Nice solution though :)

• Reply • Share ›

nikhil coderAce • a year ago

its upper bound..
fix 1st element.. then inner loop will run for n-1 time
fix 2nd element.. then inner loop will run for n-2 times
...
finally = n-1 + n-2 +.... + 1 = n62

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

• Reply • Share ›

Amit • a year ago

```
 /* Paste your code here (You may delete these lines if not writing c
#include <stdio.h>
int compare(const void *a, const void *b){
    return *(int *)a - *(int *)b;
}
int getTriptel(int ar[], int ar_size,int n)
{
     qsort(ar,ar_size,sizeof(ar[0]),compare);
     int temp[ar_size],i=0,k=ar_size-1,p=0,flag = 0;
     for(i = 0;i<ar_size;i++){
           temp[i] = n - ar[i];
     }
     i = 0;
```

```
    while((i<ar_size && k>=0)&& p<ar_size &&(i<=k) ){
          if(ar[i]+ar[k] == temp[p] ){
                      if( i != k && i != p && p != k){
                      printf("Equal %d %d %d \n",ar[i],ar[k],ar[p])
                      printf("\n");
```

see more

1 ^ | v • Reply • Share ›

joker • a year ago

```
 bool solve(vector<int> a,int k)
{
   for(int i=0;i<a.size()-3;i++)
   for(int j=a.size()-1;j>i;j--)
   if(binary_search(a.begin()+i,a.begin()+j,k-(a[i]+a[j])))
   return true;
   return false;
}
main()
{
    int n;
    int b[]={1, 4, 45, 6, 10, 8};
    vector<int> a(b,b+sizeof(b)/sizeof(int));
    sort(a.begin(),a.end());
    int k=22;
    puts(solve(a,k)?"yes":"no");
    system("pause");
    return 0;
}
```

^ | v • Reply • Share ›

Bismoy Zicrué Murasing • a year ago

I feel sometimes this website though they present things well but unnecessaril
simple way and except that thing this website is just wonderful but obviously n
kind of complicated very much sometimes...

• Reply • Share ›

sid • 2 years ago

second method is very good and to print all the triplets the only need to change
type

if(A[i] + A[l] + A[r] == sum)
{
printf("Triplet is %d, %d, %d", A[i],A[l],A[r]);
l++;
r--;
}
instead of
if(A[i] + A[l] + A[r] == sum)
{
printf("Triplet is %d, %d, %d", A[i],A[l],A[r]);
return true;
}

• Reply • Share ›

arvind • 2 years ago

I think u should add a constraint at this point in method2,

```
if( A[i] + A[l] + A[r] == sum && i!=l && i!=r && l!=r)     <---
          {
              printf("Triplet is %d, %d, %d", A[i], A[l], A[r]);
              return true;
```

```
        }
```

wat say?

Reply • Share ›

Ravi • 2 years ago

```
#include
void merge(int x[],int n)
{
int aux[n],i=0,j=0,k=0,l1=0,l2=0,u1=0,u2=0;
int size=1;
while(size<n)
{
l1=0,k=0;
while(l1+size<n)
{
l2=l1+size;
u1=l2-1;
u2=(l2+size-1<n)?l2+size-1:n-1;
for(i=l1,j=l2;i<=u1&&j<=u2;k++)
{
if(x[i]<=x[j])
aux[k]=x[i++];
else
```

see more

Reply • Share ›

Ravi • 2 years ago

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

```
void merge(int x[],int n)
{
    int aux[n],i=0,j=0,k=0,l1=0,l2=0,u1=0,u2=0;
    int size=1;
    while(size<n)
    {
        l1=0,k=0;
        while(l1+size<n)
        {
            l2=l1+size;
            u1=l2-1;
            u2=(l2+size-1<n)?l2+size-1:n-1;
            for(i=l1,j=l2;i<=u1&&j<=u2;k++)
            {
                if(x[i]<=x[j])
                    aux[k]=x[i++];
                else
```

see more

• Reply • Share ›

luv • 2 years ago

Here's an method for finding the triplets.

It uses Backtracking.

```
#include<stdio.h>

int S=3;                //SUM for which elements are to be found

int sumcheck(int el[],int si,int ei,int arr[],int cnt)
{
```

```
        if(cnt==2)
        {
        int sum=arr[0]+arr[1]+arr[2];

        if(sum==S)
        printf("(%d, %d, %d) ", arr[0], arr[1], arr[2]);
```

see more

Reply • Share ›

Sravan Vurapalli • 2 years ago

How about constructing BST with array.

Then recussively calculate the combination at root level .

I am not sure if this works , let me try and come back to you.

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

Reply • Share ›

udp • 2 years ago

A O(n^2Logn) solution
1) Sort A[]
2) Generate all pairs of two elements from A[]. Do following for every pair.
.....a) Calculate sum of the current pair.
.....b) Binary Search for sum minus sum of current pair. If the difference is fou
3) Return false.

Reply • Share ›

Yogesh Batra udp • 2 years ago

counter example:- array have distinct elements and you are looking for
(6,3) now remaining sum left is 3 and you are searching 3 in the array
included it, it will give true.

and why are implemeting O(n^2Logn)approach if O(n^2) solution has a

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

• Reply • Share ›

Subscribe Add Disqus to your site