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;
}
```

Run on IDE

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)
{
    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 elements
        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;
}
```

Run on IDE

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.

71 Comments Category: Arrays

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

(Login to Rate and Mark)

3.5 Average Difficulty : **3.5/5.0**
Based on **7** vote(s)

Add to TODO List

Mark as DONE

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