Find four elements that sum to a given value | Set 1 (n^3 solution)

Given an array of integers, find all combination of four elements in the array whose sum is equal to a given value X.

For example, if the given array is {10, 2, 3, 4, 5, 9, 7, 8} and X = 23, then your function should print “3 5 7 8″ (3 + 5 + 7 + 8 = 23).

Sources: Find Specific Sum and Amazon Interview Question

A **Naive Solution** is to generate all possible quadruples and compare the sum of every quadruple with X. The following code implements this simple method using four nested loops

```
#include <stdio.h>

/* A naive solution to print all combination of 4 elements in A[]
  with sum equal to X */
void findFourElements(int A[], int n, int X)
{
  // Fix the first element and find other three
  for (int i = 0; i < n-3; i++)
  {
    // Fix the second element and find other two
    for (int j = i+1; j < n-2; j++)
    {
      // Fix the third element and find the fourth
      for (int k = j+1; k < n-1; k++)
      {
        // find the fourth
        for (int l = k+1; l < n; l++)
           if (A[i] + A[j] + A[k] + A[l] == X)
              printf("%d, %d, %d, %d", A[i], A[j], A[k], A[l]);
      }
    }
  }
}

// Driver program to test above funtion
int main()
```

```
{
    int A[] = {10, 20, 30, 40, 1, 2};
    int n = sizeof(A) / sizeof(A[0]);
    int X = 91;
    findFourElements (A, n, X);
    return 0;
}
```

Output:

```
20, 30, 40, 1
```

Time Complexity: O(n^4)

*The time complexity can be improved to O(n^3) with the **use of sorting** as a preprocessing step, and then using method 1 of this post to reduce a loop.*

Following are the detailed steps.
1) Sort the input array.
2) Fix the first element as A[i] where i is from 0 to n–3. After fixing the first element of quadruple, fix the second element as A[j] where j varies from i+1 to n-2. Find remaining two elements in O(n) time, using the method 1 of this post

Following is C implementation of O(n^3) solution.

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

/* Following function is needed for library function qsort(). Refer
   http://www.cplusplus.com/reference/clibrary/cstdlib/qsort/ */
int compare (const void *a, const void * b)
{  return ( *(int *)a - *(int *)b ); }

/* A sorting based solution to print all combination of 4 elements in A[]
   with sum equal to X */
void find4Numbers(int A[], int n, int X)
{
    int l, r;

    // Sort the array in increasing order, using library
    // function for quick sort
    qsort (A, n, sizeof(A[0]), compare);

    /* Now fix the first 2 elements one by one and find
       the other two elements */
    for (int i = 0; i < n - 3; i++)
    {
        for (int j = i+1; j < n - 2; j++)
        {
```

```
            // Initialize two variables as indexes of the first and last
            // elements in the remaining elements
            l = j + 1;
            r = n-1;

            // To find the remaining two elements, move the index
            // variables (l & r) toward each other.
            while (l < r)
            {
                if( A[i] + A[j] + A[l] + A[r] == X)
                {
                   printf("%d, %d, %d, %d", A[i], A[j],
                                           A[l], A[r]);
                   l++; r--;
                }
                else if (A[i] + A[j] + A[l] + A[r] < X)
                    l++;
                else // A[i] + A[j] + A[l] + A[r] > X
                    r--;
            } // end of while
        } // end of inner for loop
    } // end of outer for loop
}

/* Driver program to test above function */
int main()
{
    int A[] = {1, 4, 45, 6, 10, 12};
    int X = 21;
    int n = sizeof(A)/sizeof(A[0]);
    find4Numbers(A, n, X);
    return 0;
}
```

Output:

```
1, 4, 6, 10
```

Time Complexity: O(n^3)

This problem can also be solved in O(n^2Logn) complexity. We will soon be publishing the O(n^2Logn) solution as a separate post.

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

29 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)

2.5 Average Difficulty : **2.5/5.0**
Based on **2** 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.