Given a sorted array and a number x, find the pair in array whose sum is closest to x

Given a sorted array and a number x, find a pair in array whose sum is closest to x.

Examples:

```
Input: arr[] = {10, 22, 28, 29, 30, 40}, x = 54
Output: 22 and 30

Input: arr[] = {1, 3, 4, 7, 10}, x = 15
Output: 4 and 10
```

A simple solution is to consider every pair and keep track of closest pair (absolute difference between pair sum and x is minimum). Finally print the closest pair. Time complexity of this solution is $O(n^2)$

An efficient solution can find the pair in O(n) time. The idea is similar to method 2 of this post. Following is detailed algorithm.

```
1) Initialize a variable diff as infinite (Diff is used to store the
   difference between pair and x).  We need to find the minimum diff.
2) Initialize two index variables l and r in the given sorted array.
       (a) Initialize first to the leftmost index:  l = 0
       (b) Initialize second  the rightmost index:  r = n-1
3) Loop while l < r.
       (a) If  abs(arr[l] + arr[r] - sum) < diff  then
           update diff and result
       (b) Else if(arr[l] + arr[r] <  sum )  then l++
       (c) Else r--
```

Following is C++ implementation of above algorithm.

```
// Simple C++ program to find the pair with sum cloest to a given no.
#include <iostream>
#include <climits>
#include <cstdlib>
using namespace std;

// Prints the pair with sum cloest to x
void printClosest(int arr[], int n, int x)
{
    int res_l, res_r;  // To store indexes of result pair

    // Initialize left and right indexes and difference between
    // pair sum and x
    int l = 0, r = n-1, diff = INT_MAX;

    // While there are elements between l and r
    while (r > l)
    {
       // Check if this pair is closer than the closest pair so far
       if (abs(arr[l] + arr[r] - x) < diff)
       {
           res_l = l;
           res_r = r;
           diff = abs(arr[l] + arr[r] - x);
       }

       // If this pair has more sum, move to smaller values.
       if (arr[l] + arr[r] > x)
           r--;
       else // Move to larger values
           l++;
    }

    cout <<" The closest pair is " << arr[res_l] << " and " << arr[res_r];
}

// Driver program to test above functions
int main()
{
    int arr[] =  {10, 22, 28, 29, 30, 40}, x = 54;
    int n = sizeof(arr)/sizeof(arr[0]);
    printClosest(arr, n, x);
    return 0;
}
```

Output:

```
The closest pair is 22 and 30
```

This article is contributed by **Harsh**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Category: Algorithms

akash • 3 months ago

It is giving incorrect output for array-{10,22,33,45,12,18,21,45} and x=32

o/p=10 and 21

expected o/p=10 and 22

http://ideone.com/A60eWQ

The Karicodist akash • 3 months ago

The expected input is a sorted array, not an unsorted one.

Mahmoud Emam • 5 months ago

update the pseudo code to not have ELSE in 3-b

Noha • 6 months ago

We can simplify this a little by avoiding the check: if (arr[l] + arr[r] > x)

Here is a c# solution:

```
static void PrintClosest(int[] arr, int x){
int l = 0;
int r = arr.Length - 1 ;
int min = int.MaxValue;
int min_l = -1;
int min_r = -1;
int abs;
while(l < r)
{
```

```
abs = Math.Abs( x - (arr[l] + arr[r]));
if(abs < min)
{
min = abs;
min_l = l;
```

see more

• Reply • Share ›

Noha Noha • 6 months ago

Never mind, I actually found a counter example where this doesn't work

• Reply • Share ›

Deepesh • 6 months ago

Use of choosing smallest and largest number in an array.Please give me reason for justfication.

• Reply • Share ›

GeeksforGeeks Mod Deepesh • 6 months ago

Deepesh, this is a simple variation of Meet in the Middle Algorithm

• Reply • Share ›

Deepesh GeeksforGeeks • 6 months ago

I know this is naive question but please elaborate so that i can understand.

Thanks in advance

• Reply • Share ›

Kartik Deepesh • 6 months ago

The idea is to do in linear time without missing any valid pair.

We start from minimum and maximum element so that we can move either of two corner indexes without missing any valid pair.

When sum of arr[l] + arr[r] is more than x, we are sure that there won't be any pair of on right side of r as the values on right side are greater than arr[r]. So we safely do r--.
Similarly when arr[l] + arr[r] is less than x, we can safely do l++.

Hope this helps.

4 • Reply • Share ›

karan • 6 months ago

In the algorithm's step 3. a), there should be "<" instead of ">".

• Reply • Share ›

GeeksforGeeks Mod ➦ karan • 6 months ago

Thanks for pointing this out. We have updated the post.

• Reply • Share ›

Harleen • a month ago

```
#include<stdio.h>
#include<conio.h>
main()
{
int sum=0,i,j,n,a[]={20,5,65,10,55},diff=10000000,d,x,y;
int sum1;
printf("ENTER x:\n");
scanf("%d",&x);
for(i=4;i>=0;i--)
{
for(j=0;j<i;j++) {="" sum1="a[i]+a[j];" d="x-sum1;" printf("sum="" of="" a[%d]="" and="" a[%d]="" is="" %d="" and="" therir="" difference="" is="" %d",i,j,sum1,abs(d));="" printf("\n");="" if(abs(d)<diff)="" {="" diff="abs(d);" n="i;" y="j;" }="" }="" }="" printf("\n%d\n",diff);="" printf("pair="" is:="" %d,%d\n",a[n],a[y]);="" }="">
```

• Reply • Share ›

Subscribe Add Disqus to your site Privacy

Iconic One Theme | Powered by Wordpress