

Two elements whose sum is closest to zero

Question: An Array of integers is given, both +ve and -ve. You need to find the two elements such that their sum is closest to zero.

For the below array, program should print -80 and 85.

1	60	-10	70	-80	85
---	----	-----	----	-----	----

METHOD 1 (Simple)

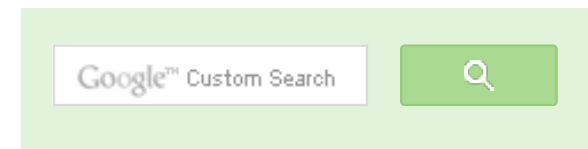
For each element, find the sum of it with every other element in the array and compare sums. Finally, return the minimum sum.

Implementation

```
# include <stdio.h>
# include <stdlib.h> /* for abs() */
# include <math.h>
void minAbsSumPair(int arr[], int arr_size)
{
    int inv_count = 0;
    int l, r, min_sum, sum, min_l, min_r;

    /* Array should have at least two elements*/
    if(arr_size < 2)
    {
        printf("Invalid Input");
        return;
    }

    /* Initialization of values */
```



GeeksforGeeks



53,522 people like [GeeksforGeeks](#).



[Interview Experiences](#)

[Advanced Data Structures](#)

[Dynamic Programming](#)

[Greedy Algorithms](#)

[Backtracking](#)

[Pattern Searching](#)

[Divide & Conquer](#)

[Mathematical Algorithms](#)

[Recursion](#)

[Geometric Algorithms](#)

```

min_l = 0;
min_r = 1;
min_sum = arr[0] + arr[1];

for(l = 0; l < arr_size - 1; l++)
{
    for(r = l+1; r < arr_size; r++)
    {
        sum = arr[l] + arr[r];
        if(abs(min_sum) > abs(sum))
        {
            min_sum = sum;
            min_l = l;
            min_r = r;
        }
    }
}

printf(" The two elements whose sum is minimum are %d and %d",
        arr[min_l], arr[min_r]);
}

/* Driver program to test above function */
int main()
{
    int arr[] = {1, 60, -10, 70, -80, 85};
    minAbsSumPair(arr, 6);
    getchar();
    return 0;
}

```



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

Time complexity: $O(n^2)$

METHOD 2 (Use Sorting)

Thanks to baskin for suggesting this approach. We recommend to read [this post](#) for background of this approach.

Algorithm

- 1) Sort all the elements of the input array.
- 2) Use two index variables l and r to traverse from left and right ends respectively. Initialize l as 0 and r as n-1.
- 3) $sum = a[l] + a[r]$
- 4) If sum is -ve, then $l++$

- 5) If sum is +ve, then r-
- 6) Keep track of abs min sum.
- 7) Repeat steps 3, 4, 5 and 6 while l < r

Implementation

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

void quickSort(int *, int, int);

/* Function to print pair of elements having minimum sum */
void minAbsSumPair(int arr[], int n)
{
    // Variables to keep track of current sum and minimum sum
    int sum, min_sum = INT_MAX;

    // left and right index variables
    int l = 0, r = n-1;

    // variable to keep track of the left and right pair for min_sum
    int min_l = l, min_r = n-1;

    /* Array should have at least two elements*/
    if(n < 2)
    {
        printf("Invalid Input");
        return;
    }

    /* Sort the elements */
    quickSort(arr, l, r);

    while(l < r)
    {
        sum = arr[l] + arr[r];

        /*If abs(sum) is less then update the result items*/
        if(abs(sum) < abs(min_sum))
        {
            min_sum = sum;
            min_l = l;
            min_r = r;
        }
    }
    if(sum < 0)
```

Free C# Code Generator



Download Today!

IRON SPEED®



```

        l++;
    else
        r--;
}

printf(" The two elements whose sum is minimum are %d and %d",
       arr[min_l], arr[min_r]);
}

/* Driver program to test above function */
int main()
{
    int arr[] = {1, 60, -10, 70, -80, 85};
    int n = sizeof(arr)/sizeof(arr[0]);
    minAbsSumPair(arr, n);
    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 arr[], int si, int ei)
{
    int x = arr[ei];
    int i = (si - 1);
    int j;

    for (j = si; j <= ei - 1; j++)
    {
        if(arr[j] <= x)
        {
            i++;
            exchange(&arr[i], &arr[j]);
        }
    }

    exchange (&arr[i + 1], &arr[ei]);
    return (i + 1);
}

```

Recent Comments

Aman Hi, Why arent we checking for conditions...

Write a C program to Delete a Tree. · 22 minutes ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 25 minutes ago

Sanjay Agarwal bool

tree::Root_to_leaf_path_given_sum(tree...

Root to leaf path sum equal to a given number · 50 minutes ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...


Count trailing zeroes in factorial of a number · 52 minutes ago

newCoder3006 If the array contains negative numbers also. We...

Find subarray with given sum · 1 hour ago

newCoder3006 Code without using while loop. We can do it..

Find subarray with given sum · 1 hour ago

AdChoices 

► [SUM Function](#)

► [The SUM of All](#)

► [SUM Program](#)

[► Try SUM](#)[► Check SUM](#)[► SUM Time](#)[► SUM To](#)[► SUM SUM](#)[► Int C++](#)

```
/* Implementation of Quick Sort
arr[] --> Array to be sorted
si  --> Starting index
ei  --> Ending index
*/
void quickSort(int arr[], int si, int ei)
{
    int pi;    /* Partitioning index */
    if(si < ei)
    {
        pi = partition(arr, si, ei);
        quickSort(arr, si, pi - 1);
        quickSort(arr, pi + 1, ei);
    }
}
```

Time Complexity: complexity to sort + complexity of finding the optimum pair = $O(n \log n) + O(n)$
= $O(n \log n)$

Asked by Vineet

Please write comments if you find any bug in the above program/algorithm or other ways to solve the same problem.

Curve-Fitting for MSD

 miraibio.com

Fully Supports 4PL & 5PL Models.
Download 14-Day Free Trial Today!



Related Tpoics:

- Remove minimum elements from either side such that $2 \times \text{min}$ 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



2



Tweet

0



0

Writing code in comment? Please use [ideone.com](https://www.ideone.com) and share the link here.

42 Comments

GeeksforGeeks

Sort by Newest ▼





with the discussion...



adude · 7 months ago

Quicksort has a worst case of $O(n^2)$, which I feel is worth noting. Your time c

^ | v · Reply · Share ›



ashu · 9 months ago

This can be further optimized to $O(n \log n + \log n)$.

Do a binary search for 0 in the sorted array.

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

^ | v · Reply · Share ›



neham · a year ago

what if there are ties between the pairs i.e. two pairs with same lowest sum s
for example (10,8,3,5,-9,-7,6)

here there are three pairs $((10,-9) = 1), ((8,-9) = -1), ((8,-7) = 1)$ whose abs(sum

Acc. to above algo it print (10,-9) pair as it comes first in sorted array.

what should be the right output in this case?

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

1 ^ | v · Reply · Share ›



Rodrex Lee · a year ago

what if there are ties between the pairs i.e. two pairs with same lowest sum s
for example (10,8,3,5,-9,-7,6).

here there are three pairs $((10,-9) = 1), ((8,-9) = -1), ((8,-7) = 1)$ whose abs(sum

Acc. to above algo it print (10,-9) pair as it comes first in sorted array.

what should be the right output in this case?

^ | v • Reply • Share ›



ajiteshpathak • a year ago

Here is another try for solution. Thought is to calculate the sum for each comb element

```
int SumCloseToZero(int *arr, int n)
{
    int x, y; // Variables to store the indexes of the two numbers
    int i = 0, j = 1;

    int min_sum = INT_MAX;
    int min_curr;

    while (i < n - 1)
    {
        min_curr = arr[i] + arr[j];

        if (abs(min_curr) < min_sum)
        {
            min_sum = abs(min_curr);
            x = i;
```

[see more](#)

^ | v • Reply • Share ›



asd • 3 years ago

sorry the previous algo needs a bit of changes...

Here is the updated one.

If there are minimum of 2 numbers in the array {

a1, a2 = (1) Find the positive first and second minimum from the array.

b1, b2 = (2) Find the negative first max and second max from the array.

Do all four combinations of additions and then output that value whose absolut

}

else

Print the no. present

^ | v • Reply • Share ›



amitp49 → asd • 2 years ago

Will it work on {-300,-2,-1,5,6,300} ?

i guess ans should be -300 & 300 , but ur algo would give -2,-1 as ans

^ | v • Reply • Share ›



ronny → amitp49 • 2 years ago

You are right @ amitp49..the above code by @ asd would not v

^ | v • Reply • Share ›



asd • 3 years ago

If there are minimum of 2 numbers in the array {

(1) Find the positive first and second minimum from the array.

(2) Find the negative first max and second max from the array.

if there is a positive minimum and there is a positive maximum then add those

else if there is a no positive minimum then output the sum of first negative ma:

else if there is a no negative maximum then output the sum of first negative m

else

Print the no. present

^ | v • Reply • Share ›



sourabh • 3 years ago

method 2 will fail when sum=0. It will get stuck in the while loop. There should
== 0) {print arr[l] and arr[r]; return;}

^ | v • Reply • Share ›



Sandeep → sourabh • 3 years ago

@sourabh: Please take a closer look at the solution, it works for all cas
then please provide an example for which it fails.

^ | v • Reply • Share ›



Dreamer • 3 years ago

For Method 2::

What about {-100, -30, 1, 2, 7} for instance. The answer should be 1 and 2 sin

^ | v • Reply • Share ›



Sandeep → Dreamer • 3 years ago

@baskin: Thanks for suggesting this. This is simpler and seems to wo
original post to use this method.

@Dreamer: We have method 2. The updated method works well for th

^ | v • Reply • Share ›



Sandeep → Dreamer • 3 years ago

@Dreamer: The method doesn't seem to work for this example. We w
this case.

^ | v • Reply • Share ›



baskin → Sandeep · 3 years ago

Sandeep, how is it different from the well known question of f
Here $k = 0$. We try our best to get as close to k as possible.

```
// sort array
// Start ptr i from left end and j from right end.
// s = a[i] + a[j]
// if s is too -ve, i++
// if s is too ve, j--
// keep track of abs min.
// break when i >= j
```

^ | v · Reply · Share ›



Venki · 3 years ago

Logical error at " $-10 + 1 = 9$ " and next few steps of given example.

^ | v · Reply · Share ›



Sandeep → Venki · 3 years ago

@venki: Thanks for pointing this out. We have corrected it.

^ | v · Reply · Share ›



balaji_ramani · 3 years ago

Doesn't method #2 fail for the following input: $\{-500, -300, 1, 2\}$

While the expected pair is $\{1, 2\}$, the answer we get is $\{-300, 2\}$

We need to handle the case where the two absolute min elements are of the s
finding l and r should do I guess.

^ | v · Reply · Share ›



mAc → balaji_ramani · 3 years ago

Besides doing the thing already mentioned...

we can also store the sum of the first two elements of the positive sorted array.

Then the overall minimum would be:

`min(min(sum_positive_array,sum_negative_array),Value_already_four`

^ | v • Reply • Share ›



Rider • 4 years ago

In Method 2 :

`while(a[r] < -0 && r< arrsize-1)`

the condition `r < arrsize -1` should come first otherwise for all negative it will ca

^ | v • Reply • Share ›



Rider → Rider • 4 years ago

sry for the above comment

code is ok.I forget to see r goes from 0 to arr_size-2 not arr_size - 1

^ | v • Reply • Share ›



Venki • 4 years ago

@Moderators, In METHOD 1, What is the significance of the following stateme

`min_l = l;`

`min_r = r;`

In both the methods, we can optimize the for loops by checking for sentinel ze
be less than zero.

^ | v • Reply • Share ›



GeeksforGeeks → Venki • 4 years ago

@Venki: Thanks for pointing out the typo. We have corrected it.

Could you please provide more details (or code if possible) for optimiz

^ | v • Reply • Share ›



Venki → GeeksforGeeks · 4 years ago

In the code, the expression "if(abs(min_sum) > abs(sum))" will "min_sum" becomes zero. The inner and outer loops will iterate with other method.

Let me know, if I am missing anything to understand.

^ | v · Reply · Share ›



Venki → Venki · 3 years ago

Optimized loop,

```
for(l = 0; l < (arr_size - 1) && (0 != min_sum)
{
    for(r = l+1; (r < arr_size) && (0 != min_
{
    sum = arr[l] + arr[r];

    count++;

    if( abs(min_sum) > abs(sum) )
    {
        min_sum = sum;
        min_l = l;
        min_r = r;
    }
}
}
```

^ | v · Reply · Share ›



Arif Ali Saiyed · 4 years ago

Well! I have another suggestion here correct if I am wrong.

it's not very efficient but still not listed here.

Lets say number of elements in given array are N, then I will create a matrix of
and store the sum of elements

For given exmaple

{1, 60, -10, 70, -80, 85};

my sparse matrix will look like this

```
0 | 61 | 9 | 71 | -79 | 86
0 | 0 | 50 | 130 | 20 | 145
0 | 0 | 0 | 60 | -90 | 75
0 | 0 | 0 | 0 | 10 | 155
0 | 0 | 0 | 0 | 0 | 5
```

while creating this sparse matrix... keep a track of min_element & element_on

[see more](#)

^ | v • Reply • Share ›



Jing → Arif Ali Saiyed • 4 years ago

This is basically METHOD 1 in the post.

^ | v • Reply • Share ›



Laxmikant • 4 years ago

Hi,

I tried method 2 on {-30, 2, 3, 7} and its giving me sum as -23 and elements as
answer should be 5 and elements as 2 and 3

^ | v • Reply • Share ›



Jing → Laxmikant • 4 years ago



You are right. The code does not consider the case where the 2 numb

^ | v • Reply • Share ›



Jing → Jing • 4 years ago

A little add-on will fix it. After sorting, check the sum of the 2 larg
the 2 smallest positive numbers. Use the smaller one as the ini

^ | v • Reply • Share ›



RandomSurfer → Jing • 3 years ago

Well that's a good idea. But what if the array is somethin
2 negative numbers. Or say [-80,-1,93] doesn't have 2 p
few cases to handle, won't there?

^ | v • Reply • Share ›



Sandeep • 4 years ago

@ims_china: Could you add few words about the approach used in the given

^ | v • Reply • Share ›



ims_china → Sandeep • 4 years ago

First I want to say there should be more space-efficient solution using

To find two elements giving minimal sum, their absolute values should
positive and another one is negative.

So the basic idea is to change the original array to an array with all pos
information separately.

Then we sort the resultant array and just scan it for find the two adja
(considering their original sign).

A more space-efficient variant could be: just using

```
static bool CustomLessFunc(int nA, int nB){
```

Are you a developer? Try out the [HTML to PDF API](#)

```
        return abs(nA) < abs(nB);  
};
```

in the STL based sorting.

In this way, there is no need for using an extra tagMegaNum structure.
didn't consider too much :)

^ | v • Reply • Share ›



ims_china → [ims_china](#) • 4 years ago

So in summary, when you sorting the array (non-STL is also ok
values , don't consider the sign :)

^ | v • Reply • Share ›



ims_china • 4 years ago

Just my 2 cents (not checked very much :)

```
int FindLeastSum(int Array[], int nSize, int& nValA, int& nValB)  
{  
    typedef struct tagMegaNum{  
  
        tagMegaNum(){  
  
            nAbsVal = 0;  
            nSignVal = 1;  
  
        };  
  
        tagMegaNum(int nNum){
```



```
if ( nNum >= 0 ){
    nAbsVal = nNum;
    nSignVal = 1;
}
```

[see more](#)

^ | v • Reply • Share ›



ims_china → ims_china • 4 years ago

Improved version:

(The code below is just for showing the basic idea, it's kind of ugly :)

```
static bool CumstomLessFunc(const int& nA, const int& nB){

    return abs(nA) < abs(nB);

};

int FindLeastSum2(int Array[], int nSize, int& nValA, int& nValB)
{
    std::vector vecVals;

    for(int k=0; k<nSize; ++k){

        vecVals.push_back(Array[k]);

    }
}
```

[see more](#)

^ | v • Reply • Share ›



GeeksforGeeks → ims_china • 4 years ago

The code has been corrected. Let us know if it is still not what y

inconvenience. I appreciate your effort.

^ | v • Reply • Share ›



ims_china → GeeksforGeeks • 4 years ago

No, it's still right :) It should be:

```
for(int k=1; k < nSize; ++k ){

    int nSum = vecVals[k] + vecVals[k+1];

    if(abs(nSum) < abs(nMinSum) ){

        nMinSum = nSum;
        nIndexA = k;
        nIndexB = k+1;
    }
}
```

(Wish this time the code can be posted correctly :)

Sorry for making the comments messy. If possible, please correct the final version :)

^ | v • Reply • Share ›



abhishek • 4 years ago

the code provided in method 2 worked just fine,
it gives the minimum absolute sum, not the minimum sum as doubted by some

as far as viresh's method goes, it fails for {-6,-2, 0, 1, 2, 3} as the inputted array

^ | v • Reply • Share ›



rv_10987 · 4 years ago

@viresh chaudhari-

Dat won't print the sum closest to '0' bt instead will print out the minimum sum

^ | v · Reply · Share ›



Viresh Chaudhari · 4 years ago

```
int LeastSum(int a[], UInt n)
{
    //=== Find min value
    int min = a[0];
    int index = 0;
    for(UInt i = 1; i < n; i++)
    {
        if(a[i] < min)
        {
            min = a[i];
            index = i;
        }
    }

    int sum = INT_MAX;
    for(UInt i = 0; i < n; i++)
    {
        if( i != index)
```

see more

^ | v · Reply · Share ›



Asit → Viresh Chaudhari · 4 years ago

This will not work if the min_element of the array is not associated with 59,70,-80,85. It will still print 5, but the result is 1(diff between 60 and -5

^ | v · Reply · Share ›



Subscribe



Add Disqus to your site

@geeksforgeeks, **Some rights reserved**

Contact Us!

Powered by **WordPress** & **MooTools**, customized by geeksforgeeks team