Equilibrium index of an array

Equilibrium index of an array is an index such that the sum of elements at lower indexes is equal to the sum of elements at higher indexes. For example, in an arrya A:

A[0] = -7, A[1] = 1, A[2] = 5, A[3] = 2, A[4] = -4, A[5] = 3, A[6]=0

3 is an equilibrium index, because:
A[0] + A[1] + A[2] = A[4] + A[5] + A[6]

6 is also an equilibrium index, because sum of zero elements is zero, i.e., A[0] + A[1] + A[2] + A[3] + A[4] + A[5]=0

7 is not an equilibrium index, because it is not a valid index of array A.

Write a function *int equilibrium(int[] arr, int n)*; that given a sequence arr[] of size n, returns an equilibrium index (if any) or -1 if no equilibrium indexes exist.

Method 1 (Simple but inefficient)
Use two loops. Outer loop iterates through all the element and inner loop finds out whether the current index picked by the outer loop is equilibrium index or not. Time complexity of this solution is O(n^2).

```
#include <stdio.h>

int equilibrium(int arr[], int n)
{
  int i, j;
  int leftsum, rightsum;

  /* Check for indexes one by one until an equilibrium
    index is found */
  for ( i = 0; i < n; ++i)
  {
    leftsum = 0;  // initialize left sum for current index i
    rightsum = 0; // initialize right sum for current index i

    /* get left sum */
    for ( j = 0; j < i; j++)
      leftsum  += arr[j];

    /* get right sum */
    for( j = i+1; j < n; j++)
      rightsum += arr[j];

    /* if leftsum and rightsum are same, then we are done */
    if (leftsum == rightsum)
      return i;
    }

  /* return -1 if no equilibrium index is found */
  return -1;
}

int main()
{
  int arr[] = {-7, 1, 5, 2, -4, 3, 0};
  int arr_size = sizeof(arr)/sizeof(arr[0]);
  printf("%d\n", equilibrium(arr, arr_size));

  getchar();
  return 0;
}
```

Time Complexity: O(n^2)

Method 2 (Tricky and Efficient)

The idea is to get total sum of array first. Then Iterate through the array and keep updating the left sum which is initialized as zero. In the loop, we can get right sum by subtracting the elements one by one. Thanks to Sambasiva for suggesting this solution and providing code for this.

```
1) Initialize leftsum  as 0
2) Get the total sum of the array as sum
3) Iterate through the array and for each index i, do following.
    a)  Update sum to get the right sum.
          sum = sum - arr[i]
       // sum is now right sum
    b) If leftsum is equal to sum, then return current index.
    c) leftsum = leftsum + arr[i] // update leftsum for next iteration.
4) return -1 // If we come out of loop without returning then
             // there is no equilibrium index
```

```
#include <stdio.h>

int equilibrium(int arr[], int n)
{
   int sum = 0;      // initialize sum of whole array
   int leftsum = 0; // initialize leftsum
```

```
    int i;

    /* Find sum of the whole array */
    for (i = 0; i < n; ++i)
         sum += arr[i];

    for( i = 0; i < n; ++i)
    {
       sum -= arr[i]; // sum is now right sum for index i

       if(leftsum == sum)
         return i;

       leftsum += arr[i];
    }

     /* If no equilibrium index found, then return 0 */
     return -1;
}

int main()
{
  int arr[] = {-7, 1, 5, 2, -4, 3, 0};
  int arr_size = sizeof(arr)/sizeof(arr[0]);
  printf("First equilibrium index is %d\n", equilibrium(arr, arr_size));

  getchar();
  return 0;
}
```

Time Complexity: O(n)

As pointed out by Sameer, we can remove the return statement and add a print statement to print all equilibrium indexes instead of returning only one.

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

108 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 Average Difficulty : **2/5.0**
Based on **11** vote(s)

Add to TODO List

Mark as DONE

Like Share 5 people like this. Be the first of your friends.

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