Largest subarray with equal number of 0s and 1s

Given an array containing only 0s and 1s, find the largest subarray which contain equal no of 0s and 1s. Expected time complexity is O(n).

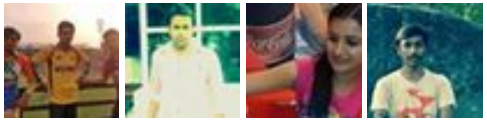
Examples:

```
Input: arr[] = {1, 0, 1, 1, 1, 0, 0}
Output: 1 to 6 (Starting and Ending indexes of output subarray)

Input: arr[] = {1, 1, 1, 1}
Output: No such subarray

Input: arr[] = {0, 0, 1, 1, 0}
Output: 0 to 3 Or 1 to 4
```

Source: Largest subarray with equal number of 0s and 1s

Method 1 (Simple)
A simple method is to use two nested loops. The outer loop picks a starting point i. The inner loop considers all subarrays starting from i. If size of a subarray is greater than maximum size so far, then update the maximum size.
In the below code, 0s are considered as -1 and sum of all values from i to j is calculated. If sum becomes 0, then size of this subarray is compared with largest size so far.

```
// A simple program to find the largest subarray with equal number of
#include <stdio.h>

// This function Prints the starting and ending indexes of the largest
// with equal number of 0s and 1s. Also returns the size of such subar
```

```
int findSubArray(int arr[], int n)
{
    int sum = 0;
    int maxsize = -1, startindex;

    // Pick a starting point as i
    for (int i = 0; i < n-1; i++)
    {
        sum = (arr[i] == 0)? -1 : 1;

        // Consider all subarrays starting from i
        for (int j = i+1; j < n; j++)
        {
            (arr[j] == 0)? sum += -1: sum += 1;

            // If this is a 0 sum subarray, then compare it with
            // maximum size subarray calculated so far
            if(sum == 0 && maxsize < j-i+1)
            {
                maxsize = j - i + 1;
                startindex = i;
            }
        }
    }
    if ( maxsize == -1 )
        printf("No such subarray");
    else
        printf("%d to %d", startindex, startindex+maxsize-1);

    return maxsize;
}

/* Driver program to test above functions*/
int main()
{
    int arr[] =  {1, 0, 0, 1, 0, 1, 1};
    int size = sizeof(arr)/sizeof(arr[0]);

    findSubArray(arr, size);
    return 0;
}
```

Output:

```
0 to 5
```

Time Complexity: O(n^2)

Time Complexity: O(n^2)
Auxiliary Space: O(1)

Method 2 (Tricky)
Following is a solution that uses O(n) extra space and solves the problem in O(n) time complexity.
Let input array be arr[] of size n and maxsize be the size of output subarray.
1) Consider all 0 values as -1. The problem now reduces to find out the maximum length subarray with sum = 0.
2) Create a temporary array sumleft[] of size n. Store the sum of all elements from arr[0] to arr[i] in sumleft[i]. This can be done in O(n) time.
3) There are two cases, the output subarray may start from 0th index or may start from some other index. We will return the max of the values obtained by two cases.
4) To find the maximum length subarray starting from 0th index, scan the sumleft[] and find the maximum i where sumleft[i] = 0.
5) Now, we need to find the subarray where subarray sum is 0 and start index is not 0. This problem is equivalent to finding two indexes i & j in sumleft[] such that sumleft[i] = sumleft[j] and j-i is maximum. To solve this, we can create a hash table with size = max-min+1 where min is the minimum value in the sumleft[] and max is the maximum value in the sumleft[]. The idea is to hash the leftmost occurrences of all different values in sumleft[]. The size of hash is chosen as max-min+1 because there can be these many different possible values in sumleft[]. Initialize all values in hash as -1
6) To fill and use hash[], traverse sumleft[] from 0 to n-1. If a value is not present in hash[], then store its index in hash. If the value is present, then calculate the difference of current index of sumleft[] and previously stored value in hash[]. If this difference is more than maxsize, then update the maxsize.
7) To handle corner cases (all 1s and all 0s), we initialize maxsize as -1. If the maxsize remains -1, then print there is no such subarray.

```
// A O(n) program to find the largest subarray with equal number of 0s
#include <stdio.h>
#include <stdlib.h>

// A utility function to get maximum of two integers
int max(int a, int b) { return a>b? a: b; }

// This function Prints the starting and ending indexes of the largest
```

```
// with equal number of 0s and 1s. Also returns the size of such subar
int findSubArray(int arr[], int n)
{
    int maxsize = -1, startindex;  // variables to store result values

    // Create an auxiliary array sunmleft[]. sumleft[i] will be sum of
    // elements from arr[0] to arr[i]
    int sumleft[n];
    int min, max; // For min and max values in sumleft[]
    int i;

    // Fill sumleft array and get min and max values in it.
    // Consider 0 values in arr[] as -1
    sumleft[0] = ((arr[0] == 0)? -1: 1);
    min = arr[0]; max = arr[0];
    for (i=1; i<n; i++)
    {
        sumleft[i] = sumleft[i-1] + ((arr[i] == 0)? -1: 1);
        if (sumleft[i] < min)
            min = sumleft[i];
        if (sumleft[i] > max)
            max = sumleft[i];
    }

    // Now calculate the max value of j - i such that sumleft[i] = sum
    // The idea is to create a hash table to store indexes of all visi
    // If you see a value again, that it is a case of sumleft[i] = sum
    // if this j-i is more than maxsize.
    // The optimum size of hash will be max-min+1 as these many differ
    // of sumleft[i] are possible. Since we use optimum size, we need
    // all values in sumleft[] by min before using them as an index in
    int hash[max-min+1];

    // Initialize hash table
    for (i=0; i<max-min+1; i++)
        hash[i] = -1;

    for (i=0; i<n; i++)
    {
        // Case 1: when the subarray starts from index 0
        if (sumleft[i] == 0)
        {
           maxsize = i+1;
           startindex = 0;
        }

        // Case 2: fill hash table value. If already filled, then use
```


Recent Comments

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

Write a C program to Delete a Tree. · 18 minutes ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 22 minutes ago

Sanjay Agarwal bool tree::Root_to_leaf_path_given_sum(tree...

Root to leaf path sum equal to a given number · 47 minutes ago

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

Count trailing zeroes in factorial of a number · 48 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

```
        if (hash[sumleft[i]-min] == -1)
            hash[sumleft[i]-min] = i;
        else
        {
            if ( (i - hash[sumleft[i]-min]) > maxsize )
            {
                maxsize = i - hash[sumleft[i]-min];
                startindex = hash[sumleft[i]-min] + 1;
            }
        }
    }
    if ( maxsize == -1 )
        printf("No such subarray");
    else
        printf("%d to %d", startindex, startindex+maxsize-1);

    return maxsize;
}

/* Driver program to test above functions */
int main()
{
    int arr[] =  {1, 0, 0, 1, 0, 1, 1};
    int size = sizeof(arr)/sizeof(arr[0]);

    findSubArray(arr, size);
    return 0;
}
```

Output:

```
0 to 5
```

Time Complexity: O(n)
Auxiliary Space: O(n)

Thanks to Aashish Barnwal for suggesting this solution.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Related Tpoics:

- Remove minimum elements from either side such that 2*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

Writing code in comment? Please use **ideone.com** and share the link here.

AlienOnEarth • 13 days ago

Excellent solution :)

  • Reply • Share ›

SunilVA • 3 months ago

Another solution

a. If no .of 0s equals no. of 1s then it is the entire array.

b. if no. of 0s is lesser than no. of 1s then choose start and ending positions to
1s(whichever is lesser).

c. If either of them is 0, then no sub-array.

  • Reply • Share ›

Tarzan ➚ SunilVA • 2 months ago

You dont explain how to solve point b. when the entire question is abou

1   • Reply • Share ›

Thanh Nguyen • 4 months ago

{{{
// This solution does 1 and only 1 pass through the array. So, it is a true O(n).
// Basically, it compares the count of 0's and 1's up to the current indx.
// 0's is count as -1 and 1's is count as +1 (ie. the different of count 0's and
// count 1's
// ex: if A[indx] = -3, it means that up to this point there are 3 more 0's than
// 1'
// So, as we traverse down the array, we look back (use hash table)
// to see if at one
// point before we also has the same count as the current point
// (ie. A[indx]) then

// (ie. A[indx]) then
// it must be true that we have the same number of 0's and 1's from the
// last time (prev+1) until now (indx).
// So, we just find the max between indexes everytime we encounter this
// condition.

void LargestSubArray(int A[], int len){
Hash sumTbl;

see more

• Reply • Share ›

Thanh Nguyen Thanh Nguyen • 4 months ago

at the end right before the "return", do:

cout << "max: "<<max<<"from i="<<i<<" to="" j="<<j<<

• Reply • Share ›

dazzling420 • 5 months ago

/*Correct me if any of the test cases fail*/
/*---*/
/*----Programming Language -- C--------*/
/*-----TIME COMPLEXITY --- O(n)--------*/
/*------SPACE COMPLEXITY--O(1)-------*/
/*----Coder Saurabh Sharma-------------*/
/*--Mail me at dazzling420@gmail.com--*/
/*---*/
#include <stdio.h>
#include<malloc.h>
#include <string.h>
#include<math.h>
int main()
{

```
int count1=0,count0=0,i,st=-1,end=-1,l,flag;
char *str;
str = (char*)malloc(sizeof(char)*200);
scanf("%s",str);
```

see more

Reply • Share ›

dazzling420 • 5 months ago

the answer to the above problem for the array 1001011 could be 0 to 5 or 1 to correct me if i am wrong

Reply • Share ›

spmahale • 6 months ago

//Little different approach see if anything is wrong

```
#include <iostream>
using namespace std;

int maxSubArray(int *a, int size){
int ones = 0;
int zeros = 0;

//Keep a count of one's and zero's
for(int i=0; i<size; i++){="" if(a[i]="=" 1)="" ones++;="" else="" zeros++;="" }="" minelement="1;" length="" of="" subarray="" is="" twice="" the="" count="" of=
if(ones="">= zeros){
maxLength = 2*zeros;
minElement = 0;
} else maxLength = 2*ones; //Minimum Element is 1 implicitly

//Now based on the count of minimum element search the low and high index
if(maxLength == 0) return 0;
```

see more

• Reply • Share ›

spmahale • 6 months ago

//Little different approach see if anything is wrong

#include <iostream>
using namespace std;

int maxSubArray(int *a, int size){
int ones = 0;
int zeros = 0;

//Keep a count of one's and zero's
for(int i=0; i<size; i++){="" if(a[i]="=" 1)="" ones++;="" else="" zeros++;="" }="" minelement="1;" length="" of="" subarray="" is="" twice="" the="" count="" of=
if(ones="">= zeros){
maxLength = 2*zeros;
minElement = 0;
} else maxLength = 2*ones; //Minimum Element is 1 implicitly

//Now based on the count of minimum element search the low and high index
if(maxLength == 0) return 0;

see more

• Reply • Share ›

B.Srikanth • 9 months ago

#include "stdafx.h"
#include
#include
#include
void find(int *a, int *l, int *h)

```
void find(int *a,int *l,int *h)
{
int n=0,i=0,c=0,d=0;
for(i=0;i<=*h;i++)
{
if(a[i]==0)
c++;
else
d++;
}
while(c!=d && *ld)
{
if(a[*l]==0 || a[*h]==0)
```

see more

• Reply • Share ›

Gandeevan • 9 months ago

why do we shift by min?

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

• Reply • Share ›

wasseypuriyan ➚ Gandeevan • 7 months ago

Suppose we get the min & max value as -2 & 2 respectively, then we b
5.

Now we store the index value for each sumleft[index] values as followir
If sumleft[index] = 2 the we store index value in H[4]
If sumleft[index] = 1 the we store index value in H[3]
If sumleft[index] = 0 the we store index value in H[2]

If sumleft[index] = -1 the we store index value in H[1]
If sumleft[index] = -2 the we store index value in H[0]

So in order to store index value for negative sumleft_values we have to

arpit tak • 9 months ago

[sourcecode language="JAVA"]

```
public class Sol {

public static void main(String[] args)
{
int[] a= {1, 0, 0, 1, 0, 1, 1,0};

int sum =0 ,in =0, out =0;
int prevdiff =0, diff;
int i,j;

for(i=0; i<a.length ;i++)
{
in = 0;
out = 0 ;

for(j=i; j<a.length ;j++)
{
```


alveko arpit tak • 9 months ago

quadratic

Rax • 10 months ago

It is clear why this code has been added:

```
if (!check) {
oneszeros[arr[i]]--;
i++;
}
```

Reply • Share ›

Sanjay Agarwal • 10 months ago

I am not able to reach to the conclusion that there can be "max-min+1" differe
explain.

Reply • Share ›

rakitic ➦ Sanjay Agarwal • 10 months ago

min is the minimum sum in sum_left and max is the maximum sum in
check while traversing in sum_left that the current value has appeared
that means in between those indices -1 and 1 have cancelled each oth
So , basically hash should contain all the possible values of sum_left ,

Reply • Share ›

shek8034 • 11 months ago

A variant of this problem with very nice explanation in the first comment of this
http://stackoverflow.com/quest...

2 Reply • Share ›

alveko ➦ shek8034 • 9 months ago

the solution explained there requires O(n) extra memory and does not

Reply • Share ›

Aditya Ambashtha ➦ shek8034 • 10 months ago

bravo sire!

you brought something better! :)
Thanks

• Reply • Share ›

Anupam • 11 months ago

```
#include
#include
#include
#include
#include
using namespace std;
#define FOR(i,a,b) for(i=a;i=0;i--)
{
s+=arr[i];
sum[i]=s;
}
int n1,n0,end=n-1, start=0;
while(end>start)
{
//TO count the number of 1s from start index to end index
n1=sum[start]-sum[end];

if(arr[end]==1)
```

see more

• Reply • Share ›

shine • 11 months ago

O(n) time and O(1) space complexity

```
#include
#include
#include
```

```
#define M 7

int f(int,int *);
int g(int,int *);
void main()
{clrscr();int arr[]={1,0,0,1,0,1,1}; int a=0,b=0,min,m;
for(int i=0;i<M;++i)
{ if(arr[i]==0)
a++;
else b++;
}
if(a==b)
printf("length=%d",a+b);
else if((a==0)||(b==0))
```

see more

• Reply • Share ›

Ananth • a year ago

Can be done in O(n) time and O(1) space. Algo is as follows.
1. Count 0s and 1s in one pass.
2. Keep two pointers at i=1 and j=n
3. If one of these contain the max element (as in if 0s > 1s, then 0 is max), inc pointer accordingly and subtract the respective count.
4. If neither contain the max, increment i, subtract the count.
5. Repeat till either i>=j or 0s == 1s.

Let me know if there is a goof-up.

```
int[] subarrayEqual(int[] arr) {
                int[] oneszeros = new int[2];
                oneszeros[0] = oneszeros[1] = 0;
                int[] pos = new int[2];
```

```
        for (int i = 0; i < arr.length; i++) {
                oneszeros[arr[i]]++;
        }
        int i = 0;
        int j = arr.length - 1;
```

see more

• Reply • Share ›

Ananth ➦ Ananth • a year ago

Corrected code due to a bug.

```
int[] subarrayEqual(int[] arr) {
        int[] oneszeros = new int[2];
        oneszeros[0] = oneszeros[1] = 0;
        int[] pos = new int[2];
        for (int i = 0; i < arr.length; i++) {
            oneszeros[arr[i]]++;
        }
        int i = 0;
        int j = arr.length - 1;
        while (i < j) {
            if (oneszeros[0] == oneszeros[1])
                break;
            boolean check = false;
            if (oneszeros[0] > oneszeros[1]) {
                if (arr[i] == 0) {
                    check = true;
```

see more

• Reply • Share ›

alyeko ➦ Ananth • 9 months ago

Nice idea, but I'm afraid it does not work for the cases like this c
1111000000000000000011

Reply • Share ›

prity ➤ Ananth • 10 months ago

good one :)

Reply • Share ›

Amroz • a year ago

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

struct ordinates{
        int y;
        int minx;
        int maxx;
        struct ordinates *next;
}*ordinate = NULL;

int length = 0;
int from,to;

void add_update(int y, int x){

struct ordinates *temp;

for(temp = ordinate;temp != NULL; temp = temp->next)
```

see more

Reply • Share ›

abhishek08aug • a year ago

Intelligent :D

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

Reply • Share ›

Amit • a year ago

it can be solved using two stacks, using the original array as a stack and other from first index on first stack and push the index in other stack and just collaps viceversa and remove the indexes as well, so in the end you will be left with in the max difference netween any indices gives you the maximum subarray with

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

Reply • Share ›

Aaman Amit • a year ago

seems good approach

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

Reply • Share ›

Rahul Singh Amit • a year ago

Can you elaborate your approach a little more with an example or provi

Reply • Share ›

Aadarsh • a year ago

Firstly since we have only two elements here, Either the first element or the la

ans.

Now traverse the array two times.
In first traversal ,consider first element to be part of ans.
Now keep on calculating count of 0's and count of 1's.
Do this

```
if(arr[i]) count1++;
else count0++;
if(count1 == count0)
{
    if(count1>max1)
        max1 = count1;
    length = 2*max1;
}
/* Paste your code here (You may delete these lines if not writing co
```

Now do the same thing for the array in reverse direction,considering arr[size-1
Which returns largest length is the ans. If length = 0 . no such array exists

Please tell me if this solution seems ok.

• Reply • Share ›

Aadarsh Aadarsh • a year ago

Correction Typo - Whichever returns largest length is the ans. If length
so basically i am talking about this
void largetSubArray(int *arr,int size)
{
int count0 = 0,count1 = 0,i,lower,higher,max1=0,max2=0,length=0;
lower = 0;
for(i=lower;imax1)

```
max1 = count1;
length = 2*max1;
}
}
count0 = 0,count1=0,higher = size-1;
for(i=higher;i>=0;i--)
{
if(arr[i]) count1++;
else count0++;
if(count1 == count0)
{
```

see more

• Reply • Share ›

Aadarsh Aadarsh • a year ago

Putting it nicely

```
void largetSubArray(int *arr,int size)
{
int count0 = 0,count1 = 0,i,lower,higher,max1=0,max2=0,lengt
lower = 0;
for(i=lower;imax1)
max1 = count1;
length = 2*max1;
}
}
count0 = 0,count1=0,higher = size-1;
for(i=higher;i>=0;i--)
{
if(arr[i]) count1++;
else count0++;
```

```
if(count1 == count0)
{
```

see more

• Reply • Share ›

koolkeshaw ➦ Aadarsh • 11 months ago

let's consider an array
{0,0,0,0,1,1,0,0,0}

Here neither first nor last element of array will be part of

• Reply • Share ›

Hara Shankar Nayak • a year ago

This problem can be solved without using extra space and O(n) time complex
Take two variables say Count0 and Count1 which keeps track of total number
Start and End variables contain the starting and ending point of the window.
Max_Start and Max_End contains the actual window starting and ending point.
max variable contains the maximum length of subarray that contains equal nu
Let N be the size of array.
Logic:-
Repeat Steps 1 to 3 upto last element.
1.>Increment the variable Count0 if the element is 0 or Count1 if the element is
2.>If the absolute value of difference of Count0 and Count1 is greater than the
are not scaned)then then decrement the Count0 or Count1 if arr[Start] is 0 or
Keep repeating Step 2 till the condition fail.
3.>If Count0=Count1 then update the max,Max_Start,Max_End.
4.>If Max_Start=-1 then print no subarray found otherwise print the index.

```
void print_largest_subarray(int arr[],int size)
{
```

see more

• Reply • Share ›

Bharart ➦ Hara Shankar Nayak • a year ago

for input {1,1,1,1,0,1,1}.. u'r code is giving "no subarray found". actual s

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

• Reply • Share ›

Hara Shankar Nayak ➦ Hara Shankar Nayak • a year ago

One thing is missing in the algorithm part.
While repeating steps 1 to 3 for each element,increment end++ after s
In the code everything is correct only forgotten to write in the logic part.
Thanks

• Reply • Share ›

Cheekooooo • a year ago

This problem can be solved in O(n) by using Moore voting algorithm.

For array of size n, cancel each occurance of 0 with 1. If no more cancellation the CONTINUOUS cancelled element which will be the solution.

• Reply • Share ›

coderAce ➦ Cheekooooo • a year ago

Can you elaborate a little?

• Reply • Share ›

Arpit • 2 years ago

#include
#include

```
using namespace std;
struct lowhigh
{
int a;
int b;
};
typedef struct lowhigh lh;

int main()
{
int arr1[]={1,0,1,0,1,1,1};
int size=7;
for(int i=0;i<size;i++)
{
if(arr1[i]==0)
```

see more

• Reply • Share ›

Arpit • 2 years ago

[sourcecode language="C++"]

```
/* Paste your code here (You may delete these lines if not writing code) */
#include<iostream>
#include<limits.h>

using namespace std;
struct lowhigh
{
int a;
int b;
};
```

```
typedef struct lowhigh lh;

int main()
{
int arr1[]={1,0,1,0,1,1,1};
int size=7;
for(int i=0;i<size;i++)
{
```

see more

• Reply • Share ›

sreeram • 2 years ago

This is taken from stackoverflow and did few edits ...
pls check it

```
#include
#define M 15

using namespace std;

void getSum(int arr[],int prefixsum[],int size) {
int i;
prefixsum[0]=arr[0]=0;
prefixsum[1]=arr[1];
for (i=2;i<=size;i++) {
prefixsum[i]=prefixsum[i-1]+arr[i];
}
}

void find(int a[],int &start,int &end,int arr[]) {
while(start < end) {
cout <<"in loop "<< start <<" "<<end < 2 * (a[end] - a[start-1])) {
```

see more

sreeram sreeram • 2 years ago

sry...to calculate element at particular position i we can use ps[i+1]-ps|

ICANTH • 2 years ago

may useful.~

```
pair<int ,int> max_equal(const int arr[], const int N)
{
        int *c =  new int[N];
        // initialize c arr.
        c[0] = arr[0];
        for(int i = 1; i < N; ++i)
                c[i] = arr[i] + c[i-1];
        // try to move the first or second to make more "balance".
        int i = 0, j = N - 1;
        while(i < j)
        {
                if (2 * (c[j] - c[i] + arr[i]) ==  j - i + 1)
                { // if '1' num is equal than '0's num.
                        return make_pair<int, int>(i, j);
                }
```

see more

rohit • 2 years ago

http://ideone.com/HKa6K

rohit • 2 years ago

what about this code

```
/* #include<stdio.h>
#include<conio.h>
main()
{
      int arr[8]={1,0,1,1,0,0,1,1};
      int i,max=0,count0=0,count1=0,index;
      for(i=0;i<8;i++)
      {
      if(arr[i]==0)
      count0++;
      else
      count1++;
      if(count1==count0)
      {
      max=2*count0;
      index=i;
      }
      }
      printf("length=%d,starting %d,end %d",max,max-(index+1),index);
      getch();
      }
*/
```

rohit rohit • 2 years ago

i got it ,this is wrong

Dipanjan • 2 years ago

```
public static String getSubset(int[] a, int n) {
                int startIndex = 0, endIndex = 0;
                int zeroCounts = 0, oneCounts = 0;
                int sum = 0;
                StringBuffer result = new StringBuffer();
                // Convert all 0's to -1 and count the number of zeros
                for (int i = 0; i < n; i++) {
                        if (a[i] == 0) {
                                a[i] = -1;
                                zeroCounts++;
                        } else {
                                oneCounts++;
                        }
                }
                if (zeroCounts == 0 || oneCounts == 0 || a.length == (
                        return "no subset possible";
                }
```

see more

Dipanjan ➦ Dipanjan • 2 years ago

It take O(n) time with no extra space

Load more comments