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 0s and 1s
#include <stdio.h>

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

Run on IDE

Output:

```
0 to 5
```

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 and 1s
#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 subarray
// with equal number of 0s and 1s. Also returns the size of such subarray.
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 array
    // 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] = sumleft[j].
    // The idea is to create a hash table to store indexes of all visited values.
    // If you see a value again, that it is a case of sumleft[i] = sumleft[j]. Check
    // if this j-i is more than maxsize.
    // The optimum size of hash will be max-min+1 as these many different values
    // of sumleft[i] are possible. Since we use optimum size, we need to shift
    // all values in sumleft[] by min before using them as an index in hash[].
    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 it
        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;
}
```

Run on IDE

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

114 Comments Category: Arrays Hash

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)

4.2 Average Difficulty : **4.2/5.0**
Based on **10** vote(s)

Add to TODO List

Mark as DONE

Like Share 17 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.

114 Comments GeeksforGeeks 1 Login

Recommend 2 Share Sort by Newest

Join the discussion

Maulik Solanki • a month ago
http://code.geeksforgeeks.org/...
• Reply • Share ›

dwaijam • 2 months ago
package com.madmax.programs.array;

import java.util.Arrays;
import java.util.HashMap;

/**
* Created by dwaipaya on 11/10/15.
*/
public class LargestSubarrayWithEqualNumberOf0sAnd1s1 {
int[] array = {1, 0 ,0, 0,0, 1, 0, 1, 1, 0, 0, 1, 1,0 ,0 ,0};

private void evaluate(int[] array){

HashMap<integer,integer> hash = new HashMap(array.length);

int sum = 0;
int max = 0;
int maxs=0, maxe=0;

see more

• Reply • Share ›

Anuj Mahajan • 2 months ago
I think this solution will work too.

1) Create a structure to store count of zeros and ones.
struct count
{
int zero;
int one;
};

2) Auxilary array (aux) of type struct count and size = n ,will be used to store the count of zeros and ones at index i. Initialize this array with 0.

Variable start and end will be used to store starting pos and ending pos of max subarray.

3) Traverse input array and update the count of zero and one in the auxilary array(aux).

4) While traversing the array and after updating count at index i.

a) If count of zeros and ones is equal and i > end , update start = 0 & end = i.

see more

^ | ˅ • Reply • Share ›

shaunak • 3 months ago

how about this logic??
I tried this its working but im confused that whether its correct or not

^ | ˅ • Reply • Share ›

shaunak • 3 months ago

```
#include<stdio.h>
void printmax(int*,int,int);
int calculatemax(int*,int,int);
int main()
{
int a[1000];
int n,start,end,i;
printf("enter a length\n");
scanf("%d",&n);
printf("enter the elements\n");
for(i=0;i<n;i++) scanf("%d",&a[i]);="" start="0;" end="n-1;" printmax(a,start,end);="" }=""
void="" printmax(int*a,int="" start,int="" end)="" {="" int="" max;=""
max="calculatemax(a,start,end);" printf("the="" maximum="" length="" is%d",max);="" }=""
int="" calculatemax(int*a,int="" start,int="" end)="" {="" int="" i,j;="" int="" count0="0;" int=""
count1="0;" int="" mid,counta,counta1,countb,countb1;="" if(start<="end)" {=""
for(i="start;i&lt;=end;i++)" {="" if(a[i]="=1)" count1++;="" else="" count0++;="" }=""
if(count1="=count0)" return="" (count0+count1);="" else="" if(count1="">count0)
{
```

see more

^ | ˅ • Reply • Share ›

shaunak • 3 months ago

```
#include<stdio.h>
void printmax(int*,int,int);
int calculatemax(int*,int,int);
int main()
{
int a[1000];
int n,start,end,i;
```

```
printf("enter a length\n");
scanf("%d",&n);
printf("enter the elements\n");
for(i=0;i<n;i++) scanf("%d",&a[i]);="" start="0;" end="n-1;" printmax(a,start,end);="" }=""
void="" printmax(int*a,int="" start,int="" end)="" {="" int="" max;=""
max="calculatemax(a,start,end);" printf("the="" maximum="" length="" is%d",max);="" }=""
int="" calculatemax(int*a,int="" start,int="" end)="" {="" int="" i,j;="" int="" count0="0;" int=""
count1="0;" int="" mid,counta,counta1,countb,countb1;="" if(start<="end)" {=""
for(i="start;i&lt;=end;i++)" {="" if(a[i]="=1)" count1++;="" else="" count0++;="" }=""
if(count1="=count0)" return="" (count0+count1);="" else="" if(count1="">count0)
{
```

see more

• Reply • Share ›

payal • 4 months ago

This can be solved in O(n) time complexity and O(1) space complexity.

1. First, scan array from left to right and keep a counter 'excess' if arr[i] == 1, increment excess, and if arr[i] == 0 then, decrease the 'excess' count. At last, you have the count how many excess 1 do you have. Same, you can do if your start from 0. excess will keep the count more 0 than 1.
2. Then, suppose excess = 2, i.e. you have two extra 1's in the whole array, then prune the array from left or right, where it contains two 1s so that pruning size is minimum. So, left subarray after pruning is the largest subarray conatining equal number of 0s and 1s.

• Reply • Share ›

piyush jain payal • 3 months ago

sumleft[] in the above algorithm signifies the same things, the exccess amt of 0s and 1s

• Reply • Share ›

payal piyush jain • 2 months ago

yea, It signifies the same thing, but sumleft array is redundant. Problem can be solved without using an extra space and in O(n) time complexity.

• Reply • Share ›

Billionaire • 5 months ago

1. Replace 0 with -1 to make things easier

2. Prefix sum

3. Use hash table to find two prefix sum that has same value

More explanations:

http://stackoverflow.com/quest...

2 • Reply • Share ›

QILI Billionaire • 5 months ago

succinct

• Reply • Share ›

Dheeraj Aggarwal • 5 months ago

https://ideone.com/pC9qzE

• Reply • Share ›

dhiru • 6 months ago

can we do it like this count the number of zeros and ones and give the minimum of two?
Is it correct ?
Any suggestions are welcomed

• Reply • Share ›

Avatar dhiru • 5 months ago

No
0 1 1 1 1 1 1 1 1 0 0 is test case for you

• Reply • Share ›

prashant jha • 6 months ago

an interval tree will also server the purpose
where a tree node will store the count of 1's and 0's in that range

• Reply • Share ›

Dman • 6 months ago

Implemented using hashing
https://ideone.com/btIw87

1 • Reply • Share ›

Abhishek • 6 months ago

Here is good explanation of this problem's solution.
fhttp://stackoverflow.com/ques...

• Reply • Share ›

Gaurav Sehgal • 6 months ago

do we need to store the left sum in an array?I tried calculating the sum on the go.Please tell if any test case fails.

http://ideone.com/zN3Qbo

• Reply • Share ›

Karan Kapoor • 6 months ago

O(n) time
space O(1)

i=0;
while(i<size) {="" inc="" cnt0="" while="" there="" is="" 0="" inc="" cnt1="" while="" there="" is="" 1="" take="" min="" and="" store="" in="" temp="" temp="temp&lt;&lt;1;" res="max(res,temp);" }="">

• Reply • Share ›

jeff • 6 months ago

Time: O(n), Space: O(1)

http://ideone.com/LMCDZa

Have Guts to prove it wrong!!!?

• Reply • Share ›

jeff → jeff • 6 months ago

Damn, got guts to prove it wrong!!!! lol :)
failure case:{1,1,0,0,0,0,0,0,1,1,1,0,0,0,0,0,1};

1 • Reply • Share ›

Dipankar Bhardwaj • 6 months ago

Nice Solution http://code.geeksforgeeks.org/...

• Reply • Share ›

Ashish • 6 months ago

Hey can anyone review this code please.. thanx in advance
http://ideone.com/0ZLgu0

• Reply • Share ›

KURT WAGNER • 7 months ago

cool!!

• Reply • Share ›

Sunil Sharma • 8 months ago

simple implementation of second method with explanation.
https://ideone.com/4POW7D

• Reply • Share ›

Narendra • a year ago

@geeksforgeek

o(n) complexity
Please have a look at my implementation
http://ideone.com/E3TGyD

• Reply • Share ›

Tomas Krcka Narendra • a year ago
what about 1,1,0,0,0,0,0,1,1
It doesn't work.

1 • Reply • Share ›

Guest • a year ago
@geeeksforgeeks

Please have a look at my implementation

O(n) complexity
http://ideone.com/HSFMJs

• Reply • Share ›

amine • a year ago
O(n) time, O(1) space:

void maxLenghtSubarray01(int A[], int n)

{

int sum = 0;

for(int i=0; i<n; i++)="" sum+="A[i];" int="" i="0;" int="" j="n-1;" while(="" i<j="")="" {="" int="" sz="(j-i+1)/2;" if(="" sz="=" sum)="" {="" std::cout<<"i:="" "<<i<<"="" j:="" " <<j<<std::endl;="" break;="" }="" if(="" a[i]<a[j])="" {="" sum-="A[j--];" }="" else="" if(="" a[i]="">A[j])

{

sum-=A[i++];

}

else if(A[i]==1 && A[i]==A[j])

see more

• Reply • Share ›

Raj • a year ago

Raj • a year ago

Probably i didnt understand correctly .. but would this second solution work for 0 0 1 1 0

as per the algo, it will give the maximum subarray as the whole array , since the sumleft
-1 -2 -1 0 -1
has sumleft[0]=sumleft[4] = -1.

but that is in correct answer

• Reply • Share ›

123 Raj • a year ago

when sumLeft[0]==sumLeft[4], the startIndex is 1, and output is [1, 4]

• Reply • Share ›

Sattu Supari • a year ago

time O(n) space O(1)
update array as sum array (sum of all elements inclusive till the element)
now traverse reverse
check value and create new index as new_index=i-sum[i]*2
if index in bound and sum just before index is 0 we found our array

• Reply • Share ›

mercs Sattu Supari • 10 months ago

check for 1 1 0 1 0 1 0
ans should b 6 bt it wud come 0 from ur algo
problem is u are starting all subarrays from 0
which is wrong as a subarray can start from any position!!

• Reply • Share ›

Sattu Supari • a year ago

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

void findSubArray(int arr[],int size)

{

int i=0,j=0;

for(i=1;i<size;i++) {="" arr[i]+="arr[i-1];" }="" for(i="0;i&lt;size;i++)" printf("%d="" ",arr[i]);="" for(i="size-1;i">=0;i--)

{

j=i-arr[i]*2;

if(j==-1){

see more

• Reply • Share ›

Vikas Gupta • a year ago

In the code

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

If the value can be either 0 or 1 so sumleft[0] would be either -1 or +1. Then, why to check this condition as sumleft[0] can never be 0?

• Reply • Share ›

Vikas Gupta → Vikas Gupta • a year ago

ohh sorry.. got it.. :)

• Reply • Share ›

Amit • a year ago

http://ideone.com/pDCbq8

1 • Reply • Share ›

Tequila → Amit • 9 months ago

1,0,1,1,1,0

gives wrong output

• Reply • Share ›

Rashid khan • a year ago

was asked in written test of ONE97

• Reply • Share ›

tushar • a year ago

@geeksforgeeks

min and max shud be initialized with sumleft[0]
not arr[0]

• Reply • Share ›

kAnN • 2 years ago

http://ideone.com/cRwDjt

Done without extra Space and at time O(n) . Better than the given answers .

Explanation and complexity analysis is given in the Code itself . Do tell me if the answer is wrong (I have checked it with many outputs :P but do challenge me !)

-kAnN

• Reply • Share ›

John kAnN • a year ago

This doesn't work for: [0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1]

• Reply • Share ›

Narendra John • a year ago

Please check http://ideone.com/E3TGyD and let me know if this implementation is okay

• Reply • Share ›

Yaduvir • 2 years ago

#include<iostream>

#include<string.h>

using namespace std;

int findSubArr(int A[],int s)

{

int B[2],upto=0;

memset(B,0,2);

for(int i=0;i<s;i++) {="" b[a[i]]++;="" if(b[0]="=B[1])" upto="i;" }="" return="" upto;="" }="" int="" main()="" {="" int="" a[]="{1,0,0,1,0,1,1},num,size;" size="sizeof(A)/sizeof(A[0]);" num="findSubArr(A,size);" if(num)="" cout<<num;="" else="" cout<<"none";="" return="" 0;="" }="">

• Reply • Share ›

newCoder3006 • 2 years ago

Let me know if I have missed any case.

1. If (no .of 0s == no. of 1s) then the entire array is the result.

2. if (no. of 0s < no. of 1s) then choose start and ending positions to include those many no. of 0s and 1s(whichever is lesser).

3. If either of the counts is 0, then no sub-array.

Have a look at the code: http://ideone.com/ERJLTC

1 • Reply • Share ›

Siya newCoder3006 • a year ago

But answer should be 1-6 but your code is giving 0-3 !!

• Reply • Share ›

Ankur Jain newCoder3006 • 2 years ago

http://ideone.com/yDqWKa

see this

• Reply • Share ›

AlienOnEarth • 2 years ago

Excellent solution :)

• Reply • Share ›

SunilVA • 2 years 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 include those many no. of 0s and 1s(whichever is lesser).

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

• Reply • Share ›

Tarzan SunilVA • 2 years ago

You dont explain how to solve point b. when the entire question is about solving that

4 • Reply • Share ›

Load more comments