Find the point where a monotonically increasing function becomes positive first time

Given a function 'int f(unsigned int x)' which takes a **non-negative integer** 'x' as input and returns an **integer** as output. The function is monotonically increasing with respect to value of x, i.e., the value of f(x+1) is greater than f(x) for every input x. Find the value 'n' where f() becomes positive for the first time. Since f() is monotonically increasing, values of f(n+1), f(n+2),... must be positive and values of f(n-2), f(n-3), .. must be negative.
Find n in O(logn) time, you may assume that f(x) can be evaluated in O(1) time for any input x.

A **simple solution** is to start from i equals to 0 and one by one calculate value of f(i) for 1, 2, 3, 4 .. etc until we find a positive f(i). This works, but takes O(n) time.

Can we apply Binary Search to find n in O(Logn) time? We can't directly apply Binary Search as we don't have an upper limit or high index. The idea is to do repeated doubling until we find a positive value, i.e., check values of f() for following values until f(i) becomes positive.

```
  f(0)
  f(1)
  f(2)
  f(4)
  f(8)
  f(16)
  f(32)
  ....
  ....
  f(high)
Let 'high' be the value of i when f() becomes positive for first time.
```


Can we apply Binary Search to find n after finding ‘high’? We can apply Binary Search now, we can use ‘high/2’ as low and ‘high’ as high indexes in binary search. The result n must lie between ‘high/2’ and ‘high’.

Number of steps for finding ‘high’ is O(Logn). So we can find ‘high’ in O(Logn) time. What about time taken by Binary Search between high/2 and high? The value of ‘high’ must be less than 2*n. The number of elements between high/2 and high must be O(n). Therefore, time complexity of Binary Search is O(Logn) and overall time complexity is 2*O(Logn) which is O(Logn).

```
#include <stdio.h>
int binarySearch(int low, int high); // prototype

// Let's take an example function as f(x) = x^2 - 10*x - 20
// Note that f(x) can be any monotonocally increasing function
int f(int x) { return (x*x - 10*x - 20); }
```

```
// Returns the value x where above function f() becomes positive
// first time.
int findFirstPositive()
{
    // When first value itself is positive
    if (f(0) > 0)
        return 0;

    // Find 'high' for binary search by repeated doubling
    int i = 1;
    while (f(i) <= 0)
        i = i*2;

    //  Call binary search
    return binarySearch(i/2, i);
}

// Searches first positive value of f(i) where low <= i <= high
int binarySearch(int low, int high)
{
    if (high >= low)
    {
        int mid = low + (high - low)/2; /* mid = (low + high)/2 */

        // If f(mid) is greater than 0 and one of the following two
        // conditions is true:
        // a) mid is equal to low
        // b) f(mid-1) is negative
```


Popular Posts

```
        if (f(mid) > 0 && (mid == low || f(mid-1) <= 0))
            return mid;

        // If f(mid) is smaller than or equal to 0
        if (f(mid) <= 0)
            return binarySearch((mid + 1), high);
        else // f(mid) > 0
            return binarySearch(low, (mid -1));
    }

    /* Return -1 if there is no positive value in given range */
    return -1;
}

/* Driver program to check above functions */
int main()
{
    printf("The value n where f() becomes positive first is %d",
           findFirstPositive());
    return 0;
}
```

Output:

```
The value n where f() becomes positive first is 12
```

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.

17 Comments GeeksforGeeks

Sort by Newest

Join the discussion

Recent Comments

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

Write a C program to Delete a Tree. · 4 minutes ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 8 minutes ago

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

Root to leaf path sum equal to a given number · 32 minutes ago

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

Count trailing zeroes in factorial of a number · 34 minutes ago

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

Find subarray with given sum · 59 minutes ago

Join the discussion…

zzer • a month ago

```
int binary_search(int low,int high)
{
int candidate= -1;
int mid;
while(low <= high)
{
mid = low +(high-low)/2;
if(f(mid) >0 )
{
candidate = mid;
high = mid-1;
}
else
low = mid+1;
}
return candidate;
}
```

• Reply • Share ›

Tarzan • 2 months ago

so if the function does not return any positive value for input up till INT_MAX, th
?

We need to change this

while (f(i) <= 0)

• Reply • Share ›

raghvendra • 9 months ago

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

Find subarray with given sum · 1 hour ago

```
#include<stdio.h>
#include<iostream>
#include<cmath>
using namespace std;
#define p 1e-6
double value(double x)
{
    return 2*x+5;
}
double binary(double low,double high)
{
    double mid;
    while(abs(high-low)>p)
    {
      mid=low+(high-low)/2;
      if(abs(value(mid))<=p)return mid;
      else if(value(mid)>0)
```

see more

• Reply • Share ›

Sunil • 10 months ago

We can do a binary search for a value of mid where,
if mid satisfies the condition f(mid)*f(mid+1)<0, we return mid+1.
if mid satisfies the condition f(mid)*f(mid-1)<0, we return mid.

because (-ve)*(-ve)=(+ve) and (+ve)*(+ve)=(-ve) only at the point of transition

• Reply • Share ›

darkpassenger • 10 months ago

can you tell any case when binary search returns -1 i.e there is no element wh
function find first positive that positive element exists

function find first positive that positive element exists.

Abhinav Aggarwal • 11 months ago

If you do that, then the gap between the subsequent iterations will increase wh
now you make gap 3*i from i. Then you would need to apply binary search in tl

Anshul Gupta • 11 months ago

This is more like newton-raphson method which terminates for the first +ve f(x

Manish • 11 months ago

We can make use of f'(x)(rate of change of f(x) at x) for computing the amount
it will significantly reduce complexity.

couldnt figure out how to use it...:(

GeeksforGeeks • 11 months ago

Please tale a closer look at the article. Also, take few examples. It is simple, n
few things.
1) Function must be monotonically increasing, i.e., f(0) < f(1) < f(2) <.... f(n) <
2) We want to find out FIRST value i such that f(i) is positive where i may be a

Mukul Taneja • 11 months ago

I cannot understand two things.
why these two assumptions are made?
1. The result n must lie between ‘high/2? and ‘high’.
2. The value of ‘high’ must be less than 2*n?
Plz explain.....

Reply • Share ›

Priyank Jain • 11 months ago

why not use a higher increment?

So, instead of.

i = I * 2,

why not use something like I *= 3 or even 4?

Reply • Share ›

zzer ➦ Priyank Jain • a month ago

then the range between low and high is bigger, and we can simple use i<<2, it is faster as well

Reply • Share ›

md03 • 11 months ago

```
if (f(mid) > 0 && (mid == low || f(mid-1) <= 0))
            return mid;
```

Correct me if I am wrong admin:
Since the function is monotonically increasing, the condition:
if(f(mid)>0 && f(mid-1)<=0)

is sufficient.

mid==low is satisfied when the high is equal to low or high=low+1. Even in this mid-1 will always be non-negative since mid=0 will never be tested here, since step of the "int findFirstPositive()" function.

Reply • Share ›

kartik ➚ md03 • 11 months ago

'mid == low' is also needed. Consider the case when low = 0, high = 0

▲ | ▼ • Reply • Share ›

md03 ➚ kartik • 11 months ago

In case of low=0 and high=0, mid=0. If first positive value is at i

if (f(0) > 0)
return 0;
If the first positive value is not at index 0, thus the first condition
second(f(mid-1)< =0) shall not be checked.

In the case when low=5 and high=6, mid=5
if f(5) is the first positive, then f(4) is negative and hence conditi
enough.
if f(5) is not the the first positive then f(4) is also positive, again

▲ | ▼ • Reply • Share ›

Guest ➚ md03 • 8 months ago

agreed

▲ | ▼ • Reply • Share ›

Ishwar Jindal • 11 months ago

what about using a step variable. we will not need special binary search functi
is the code:
FindFirstPos() {

int step=0, i=1;.

if(f(0)>0) return 0;.

while(1) {.
if(f(i)<0) {.

```
if(! step) step=1;
else step*=2;
i+=step;
}.
if(f(i)>=0) {.
if(step==1) return i;.
else {
step/=2;.
i-=step;.
}
}.

}.
}
```

Please let me know if there seem some bug.