Find position of the only set bit

Given a number having only one ’1′ and all other ’0′s in its binary representation, find position of the only set bit. Source: Microsoft Interview | 18

The idea is to start from rightmost bit and one by one check value of every bit. Following is detailed algorithm.

1) If number is power of two then and then only its binary representation contains only one ’1′. That’s why check whether given number is power of 2 or not. If given number is not power of 2, then print error message and exit.

2) Initialize two variables; i = 1 (for looping) and pos = 1 (to find position of set bit)

3) Inside loop, do bitwise AND of i and number ‘N’. If value of this operation is true, then “pos” bit is set, so break the loop and return position. Otherwise, increment “pos” by 1 and left shift i by 1 and repeat the procedure.

```
// C program to find position of only set bit in a given number
#include <stdio.h>

// A utility function to check whether n is power of 2 or not. See htt
int isPowerOfTwo(unsigned n)
{  return n && (! (n & (n-1)) ); }

// Returns position of the only set bit in 'n'
int findPosition(unsigned n)
{
    if (!isPowerOfTwo(n))
        return -1;

    unsigned i = 1, pos = 1;
```


```
    // Iterate through bits of n till we find a set bit
    // i&n will be non-zero only when 'i' and 'n' have a set bit
    // at same position
    while (!(i & n))
    {
        // Unset current bit and set the next bit in 'i'
        i = i << 1;

        // increment position
        ++pos;
    }

    return pos;
}

// Driver program to test above function
int main(void)
{
    int n = 16;
    int pos = findPosition(n);
    (pos == -1)? printf("n = %d, Invalid number\n", n):
                 printf("n = %d, Position %d \n", n, pos);

    n = 12;
    pos = findPosition(n);
    (pos == -1)? printf("n = %d, Invalid number\n", n):
                 printf("n = %d, Position %d \n", n, pos);

    n = 128;
    pos = findPosition(n);
    (pos == -1)? printf("n = %d, Invalid number\n", n):
                 printf("n = %d, Position %d \n", n, pos);

    return 0;
}
```

Output:

```
n = 16, Position 5
n = 12, Invalid number
n = 128, Position 8
```

Following is **another method** for this problem. The idea is to one by one right shift the set bit of given number ‘n’ until ‘n’ becomes 0. Count how many times we shifted to make ‘n’ zero. The

Geometric Algorithms

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

final count is position of the set bit.

```
// C program to find position of only set bit in a given number
#include <stdio.h>

// A utility function to check whether n is power of 2 or not
int isPowerOfTwo(unsigned n)
{  return n && (! (n & (n-1)) ); }

// Returns position of the only set bit in 'n'
int findPosition(unsigned n)
{
    if (!isPowerOfTwo(n))
        return -1;

    unsigned count = 0;

    // One by one move the only set bit to right till it reaches end
    while (n)
    {
        n = n >> 1;

        // increment count of shifts
        ++count;
    }

    return count;
}

// Driver program to test above function
int main(void)
{
    int n = 0;
    int pos = findPosition(n);
    (pos == -1)? printf("n = %d, Invalid number\n", n):
                 printf("n = %d, Position %d \n", n, pos);

    n = 12;
    pos = findPosition(n);
    (pos == -1)? printf("n = %d, Invalid number\n", n):
                 printf("n = %d, Position %d \n", n, pos);

    n = 128;
    pos = findPosition(n);
    (pos == -1)? printf("n = %d, Invalid number\n", n):
                 printf("n = %d, Position %d \n", n, pos);
```

```
    return 0;
}
```

Output:

```
n = 0, Invalid number
n = 12, Invalid number
n = 128, Position 8
```

We can also use log base 2 to find the position. Thanks to Arunkumar for suggesting this solution.

```
#include <stdio.h>

unsigned int Log2n(unsigned int n)
{
   return (n > 1)? 1 + Log2n(n/2): 0;
}

int isPowerOfTwo(unsigned n)
{
    return n && (! (n & (n-1)) );
}

int findPosition(unsigned n)
{
    if (!isPowerOfTwo(n))
        return -1;
    return Log2n(n) + 1;
}

// Driver program to test above function
int main(void)
{
    int n = 0;
    int pos = findPosition(n);
    (pos == -1)? printf("n = %d, Invalid number\n", n):
                 printf("n = %d, Position %d \n", n, pos);

    n = 12;
    pos = findPosition(n);
    (pos == -1)? printf("n = %d, Invalid number\n", n):
                 printf("n = %d, Position %d \n", n, pos);

    n = 128;
```


Recent Comments

```
    pos = findPosition(n);
    (pos == -1)? printf("n = %d, Invalid number\n", n):
                 printf("n = %d, Position %d \n", n, pos);

    return 0;
}
```

Output:

```
n = 0, Invalid number
n = 12, Invalid number
n = 128, Position 8
```

This article is compiled by **Narendra Kangralkar**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Related Tpoics:

- Check if a number is multiple of 9 using bitwise operators
- How to swap two numbers without using a temporary variable?
- Divide and Conquer | Set 4 (Karatsuba algorithm for fast multiplication)
- Swap all odd and even bits

- Add two bit strings
- Write your own strcmp that ignores cases
- Binary representation of a given number
- Find the element that appears once

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

28 Comments GeeksforGeeks

Sort by Newest

Ravi Kumar • 3 months ago

this question is same as finding whether number of multiple of 2 or not

```
check(int n)
{
return n&!(n&(n-1)) ;
}
```


Hardhik Mallipeddi Ravi Kumar • 2 months ago

Are you sure whether it was for finding whether a number is a multiple number is a multiple of 2 or not you can just do n%2 and check if it's 0

If it were for finding whether a number is a power of 2, you can do n&(r were 0 it implies it is a power of two and not otherwise.

Prashanth S • 8 months ago

cout<<x&-x ;="">

1 ^ | v • Reply • Share ›

asunel • 8 months ago

This problem is equivalent to finding the rightmost set bit....

3 ^ | v • Reply • Share ›

Shradha Agrawal • 9 months ago

Divide and Conquer approach:
let size of integer is 32 bits.
nb = 32/2.
1. left half of number = number % 2^nb.
2. right half of number = number /2 ^nb.
3.as it has only one bit set so that bit will either be in left half of right half.
so , number = number_left==0? number_right : number_left;.
4. If we have chosen right half then our position of set bit has been moved by r
so, if(number_left == 0).
pos += nb.
5.nb = nb/2.
6. if(nb==0) stop else go to step 1.
Code is:

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

int pos = 0;.
```

see more

^ | v • Reply • Share ›

Deepak Kushwah • 9 months ago

GeeksforGeeks this was the q which was asked in qualcomm interview also.
search karanpreet has given that solution add this solution also :)

Karanpreet Singh Aulakh • 10 months ago

Another approach can be to use something similar to Binary Search.
In this way we can reduce the complexity to log(k) where k is the number of bi
The idea is to calculate the middle bit index, see if the given number, say num
and accordingly recurse with the suitable part of the number.

```
#include<stdio.h>
int func(int num, int start, int end).
{

int mid=(start+end)/2;.

if((1<<mid)>num).

return func(num, start, mid-1);.

else if((1<<mid)<num).

return func(num, mid+1, end);.

else.
```

see more

maruti kutre ➦ Karanpreet Singh Aulakh • 7 months ago

Check for this value int n =1048576;// (int) Math.pow(2, 20); binary way

atiq • 10 months ago

I couldn't find any use of last method as Finding Log is inefficient ... so i did it b
program.

```
#include<iostream>
using namespace std;
int PowerofTwo(long int n)
{
return n&&(!(n&n-1))
}
int setBit(long int n)
{
if(!PowerofTwo(n))
return -1;
long int mid,left,right;
left=0,right=31;
long int k=1;
int temp=1;
while(k)
```

see more

• Reply • Share ›

Robin40 • 11 months ago

this is a O(1) solution:

```
 #include <stdio.h>


inline bool isPowerOfTwo(unsigned int x)
{
    return x &&! (x & (x - 1));
}

int findPosition(unsigned int x)
{
```

```
    if (isPowerOfTwo(x))
        return hashPos[x % 37];

    return -1;
}
```

see more

• Reply • Share ›

Akil ➦ Robin40 • 7 months ago

how to get the hashpos[] array value

• Reply • Share ›

punjabi kudi • 11 months ago

```
 /*
// check out this sol....no recursive call

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

int findpos(int n){
    if(n&&(n&(n-1)))
    return -1;

    return log2(n);
}
int main(){
    int a=31;
    a=findpos(a);
    if(a==-1)
```

```
    printf("not possible\n");
    else
    printf("%d\n",a);
}

 */
```

Reply • Share ›

abhishek08aug • a year ago

And here comes the recursive solution. :D

```
#include<stdio.h>
int find_position(int num, int pos) {
  if(num%2!=0 && num!=1) {
    return -1;
  }
  if(num==1) {
    return pos;
  } else {
    return find_position(num>>1, ++pos);
  }
}

int main() {
  printf("set bit position in %d is %d\n", 13, find_position(13,1));
  printf("set bit position in %d is %d\n", 1, find_position(1,1));
  printf("set bit position in %d is %d\n", 8, find_position(8,1));
```

see more

Reply • Share ›

GeeksforGeeks • a year ago

Arunkumar Somalinga: Thanks for suggesting a new method. We have added

 • Reply • Share ›

arena_zp • a year ago

Your algorithm can be improved to be O(log log N).
In short, the successful of x|(2^1), x|(2^2), x|(2^3),..... is monotone!

 • Reply • Share ›

Peng Zhang • a year ago

How do you think the log operator works in the implementation of library? :-)

 • Reply • Share ›

Avi • a year ago

You can skip counting a lot of 1s by using the fact that the number is a power
do it with just one if condition in the function.

```
 #include<iostream>
using namespace std;
bool CheckIfMoreThanAndReduce(unsigned long long int* n, int shift) {
  if ((1UL<<shift) <= *n) {
    (*n) >>= shift;
    return true;
  } else {
    return false;
  }
}
int main() {
  unsigned long long int n;
  cin >> n;
  int count = 0;
  for (unsigned int i = 32; i > 0 ; i/=2) {
```

see more

• Reply • Share ›

Avinash Singh • a year ago

```
#include<stdio.h>
#include<conio.h>
int ispow(int a).
{
int count=0;.
while(a%2==0 && a>2).

{.

a=a/2;.

count++;.

}.
if(a==2).

return ++count;.
else.

return -1;
```

see more

• Reply • Share ›

ammy • a year ago

i think,a better solution is possible
The given solutions are o(logn) and i think the below solution may work in log(l

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

```
#include<stdio.h>
int main()
{
    unsigned int n,i,j;
    printf("enter the number::\n");
    scanf("%u",&n);
    i=16,j=8;
    if(n&(n-1))
    {
        printf("number has more than one sit bit\n");
        return 0;
    }
    while(j>0)
```

see more

Reply • Share ›

jordan • a year ago

I did not get the question. Please clarify the question, I needed to open this pag

Reply • Share ›

atul • a year ago

@geeksforgeeks: you below previous post will solve this problem after checki
http://www.geeksforgeeks.org/p...

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

Reply • Share ›

Rakesh Rk • a year ago

yup

 • Reply • Share ›

Sreenivas Doosa • a year ago

Nice approaches duude..

I am adding one point here if n is singed integer.
findPosition() method will be as follows...

```
int findPosition(int n)
{

        if(n == INT_MIN)
                return true;

        if(!isPowerOfTwo(n))
                return false;

        ...
        ...
}
```

see more

 • Reply • Share ›

Arunkumar Somalinga • a year ago

yes

 • Reply • Share ›

Rupesh • a year ago

Log(x)/Log(2)

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

• Reply • Share ›

Rupesh Singh Paikara • a year ago

log of base 2?

• Reply • Share ›

Arunkumar Somalinga • a year ago

Hi.I have an another solution for this problem.First check if the number is powe
return then log value of the number(base 2) it will give the correct answer.

• Reply • Share ›

arunkumar267 • a year ago

Hi.I have an another solution for this problem.First check if the number is powe
return then log value of the number(base 2) it will give the correct answer

• Reply • Share ›