Count set bits in an integer

Write an efficient program to count number of 1s in binary representation of an integer.

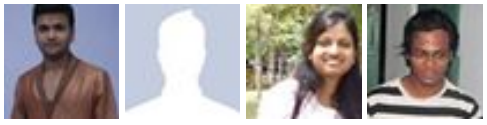
1. Simple Method Loop through all bits in an integer, check if a bit is set and if it is then increment the set bit count. See below program.

```
/* Function to get no of set bits in binary
   representation of passed binary no. */
int countSetBits(unsigned int n)
{
  unsigned int count = 0;
  while(n)
  {
    count += n & 1;
    n >>= 1;
  }
  return count;
}

/* Program to test function countSetBits */
int main()
{
    int i = 9;
    printf("%d", countSetBits(i));
    getchar();
    return 0;
}
```

Time Complexity: (-)(logn) (Theta of logn)

2. Brian Kernighan's Algorithm:
Subtraction of 1 from a number toggles all the bits (from right to left) till the rightmost set



Interview Experiences
Advanced Data Structures
Dynamic Programming
Greedy Algorithms
Backtracking
Pattern Searching
Divide & Conquer
Mathematical Algorithms
Recursion

bit(including the righmost set bit). So if we subtract a number by 1 and do bitwise & with itself (n & (n-1)), we unset the righmost set bit. If we do n & (n-1) in a loop and count the no of times loop executes we get the set bit count.
Beauty of the this solution is number of times it loops is equal to the number of set bits in a given integer.

```
1  Initialize count: = 0
2  If integer n is not zero
   (a) Do bitwise & with (n-1) and assign the value back to n
       n: = n&(n-1)
   (b) Increment count by 1
   (c) go to step 2
3  Else return count
```

Implementation of Brian Kernighan's Algorithm:

```
#include<stdio.h>

/* Function to get no of set bits in binary
   representation of passed binary no. */
int countSetBits(int n)
{
    unsigned int count = 0;
    while (n)
    {
      n &= (n-1) ;
      count++;
    }
    return count;
}

/* Program to test function countSetBits */
int main()
{
    int i = 9;
    printf("%d", countSetBits(i));
    getchar();
    return 0;
}
```

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

Example for Brian Kernighan's Algorithm:

```
n =  9 (1001)
count = 0

Since 9 > 0, subtract by 1 and do bitwise & with (9-1)
n = 9&8  (1001 & 1000)
n = 8
count  = 1

Since 8 > 0, subtract by 1 and do bitwise & with (8-1)
n = 8&7  (1000 & 0111)
n = 0
count = 2

Since n = 0, return count which is 2 now.
```

Time Complexity: O(logn)

3. Using Lookup table: We can count bits in O(1) time using lookup table. Please see http://graphics.stanford.edu/~seander/bithacks.html#CountBitsSetTable for details.

You can find one use of counting set bits at http://geeksforgeeks.org/?p=1465

References:

http://graphics.stanford.edu/~seander/bithacks.html#CountBitsSetNaive

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)
- Find position of the only set bit
- Swap all odd and even bits
- Add two bit strings
- Write your own strcmp that ignores cases
- Binary representation of a given number

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

31 Comments GeeksforGeeks

Sort by Newest

Recent Comments

Abhi You live US or India?

Google (Mountain View) interview · 28 minutes ago

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

Write a C program to Delete a Tree. · 1 hour ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 1 hour ago

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

Root to leaf path sum equal to a given number · 1 hour ago

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

Count trailing zeroes in factorial of a number · 1 hour ago

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

Join the discussion…

abhinav • 4 days ago

doing n-1 doesnt take O(N) time???, i highly doubt ....

 • Reply • Share ›

AlienOnEarth • 6 days ago

Explanation for 2nd method is very nice.

 • Reply • Share ›

Sapan Kumar Das • 5 months ago

checkout this link ... it gives some good logic and explanation as well

http://www.codextream.com/?p=6...

 • Reply • Share ›

Ritesh • a year ago

We can use the gcc buitin method

int __builtin_popcount (unsigned int x)

Returns the number of 1-bits in x.

7 • Reply • Share ›

G.Prasath • a year ago

```
#include

main()
{
int a,count=0,i;
printf("\nEnter the number! ");
scanf("%d",&a);
```

Find subarray with given sum · 2 hours ago

```
for( i=0 ; i>=1,i++ )
count+= a&1;

if(a==-1)
printf("\n Number is negative! Total 1's = %d ",a);
else
printf("\n Total 1's = %d ",count);

}
```

Reply • Share ›

G.Prasath ➦ G.Prasath • a year ago

The for loop statement is :

for(i=0 ; i>=1,i++)

SOrry guys!

Reply • Share ›

nikhil • a year ago

i thought of this solution..

```
int setBits(int number){
int count=0;
while(number!=0){
if(number%2!=0)
count++;
number=number/2;
}
return count;
}
```

comments are most welcome!!

Reply • Share ›

seeker7 • 2 years ago

how is the complexity of the first algorithm logn * theta(logn).
Please explain ...

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

• Reply • Share ›

Mahesh • 2 years ago

#include

int main()
{
int num,cnt=0,i=1;
pf("Enter your num value:");
sf("%d",&num);

while(i<=32)
{
if(num&(1<<i))
cnt++;
i++;
}
pf("total set bits are:%d",cnt);
retrun 0;
}

• Reply • Share ›

Satish Yadav ➚ Mahesh • 9 months ago

What if number is not represented in 32 bit?

• Reply • Share ›

Sourabh mehrotra • 2 years ago

```
#include
#include

int main(int argc,char **str)
{
int num=17,i,sum=0,aux,b;
b=countbits(num);
b--;
sum=b*pow(2,b-1);
aux=1<<b;
printf("%d\n",aux);
while(aux0)
{
if(n&0x1)
cnt++;
n=n>>1;
}
return cnt;
```

see more

• Reply • Share ›

Mahesh ➦ Sourabh mehrotra • 2 years ago

```
#include

    int main()
    {
    int num,cnt=0,i=1;
    pf("#nEnter your num value:");
    sf("%d",&num);
```

```
while(i<=32)
{
if(num&(i<<i))
cnt++;
i++;
}
pf("#ntotal set bits are:%d",cnt);
retrun 0;
}
```

• Reply • Share ›

Amit Gupta • 3 years ago

Hi!

I think the complexity of solution 1 is O(logN*logN) and the second solution is (

In solution 1, each bitwise AND costs logN, and this is to be done logN times.

In solution 2, we do same bitwise AND, only this time, it's required to be done (

Do people agree on this?

• Reply • Share ›

raghu • 3 years ago

@Sandeep,Kartik...GEEKSFORGEEKS..& all geeks ..can any1 explain me thi
set bits in a number using look up table..

http://graphics.stanford.edu/~... for details

Please help & reply ASAP.

• Reply • Share ›

abc ➚ raghu • 3 years ago

Given is a 32 bit number...that means a 4 byte integer.
p[0], p[1], p[2], p[3] are the 4 bytes of the integer
so for example:

```
 v = 10010100100010001011101100111111
the 4 bytes are:
p[0] = 00111111 = 63
p[1] = 10111011 = 187
p[2] = 10001000 = 136
p[3] = 10010100 = 148

so number of bits set in v = number of bits set in p[0] +
                                                      numbe
                                                      numbe
                                                      numbe

If BitSetsTable is an array where BitSetsTable[i] = number of s
```

see more

• Reply • Share ›

vpshastry ➚ abc • 2 years ago

can some one explain this initialization??

```
 static const unsigned char BitsSetTable256[256] =
{
#   define B2(n) n,     n+1,     n+1,     n+2
#   define B4(n) B2(n), B2(n+1), B2(n+1), B2(n+2)
#   define B6(n) B4(n), B4(n+1), B4(n+1), B4(n+2)
    B6(0), B6(1), B6(1), B6(2)
};
```

from
http://graphics.stanford.edu/~...

Reply • Share ›

Venki vpshastry • 2 years ago

Read the following post http://www.geeksforgeeks.org/a

Reply • Share ›

abc • 3 years ago

This can be done in O(1):

```
 for 32 bit numbers:
int countSetBits(unsigned int u)
{
    unsigned int uCount = 0;
    uCount = u - ((u >> 1) && 033333333333) - ((u >> 2) && 0111111111
    uCount = ((uCount + (uCount >> 3)) && 030707070707)%63;
    return uCount;
}

for 64 bit numbers:
int countSetBits(unsigned long long u)
{
    unsigned long long uCount = 0;
    uCount = u - ((u >> 1) && 0x7777777777777777) - ((u >> 2) && 0x33
    uCount = ((uCount + (uCount >> 4)) && 0x0F0F0F0F0F0F0F0F)%255;
    return uCount;
}
```

1 • Reply • Share ›

Rishabh abc • 7 months ago

can anyone explain this method ? Why dont we often "cite" this metho

• Reply • Share ›

Venki • 4 years ago

Another method is documented in the following link

http://math-puzzles-computing....

• Reply • Share ›

sreemathi kumar • 4 years ago

This will also work out.

```
 int countbits(int n)
{
int count=0;
int m=1;
while(n)
{
 if(n & m)
 {
  count++;
  n=n^m;   //unset the bit set(from rightmost)
 }
 m<<=1;
}
return count;
}
```

1 • Reply • Share ›

hariom • 4 years ago

How come the time complexity of Brian Kernighan's Algorithm is O(logn). Shouldn't it be O(no_of_set_bits) ??

Reply • Share ›

Venki hariom • 4 years ago

@hariom, It is worst case complexity. Since atmost the number can ha

Reply • Share ›

hariom Venki • 4 years ago

@venki
I suppose ,I am unable to understand your point here.
The max no of set bits are 32 or 64(i.e. N) depending on the no
How can it be log(N)??

Reply • Share ›

Venki hariom • 4 years ago

@hariom, As you mentioned the complexity is O(no_of_ proportional to log(number), hence the complexity. This will iterate atmost [log(number) + 1] times.

Reply • Share ›

vkjk89 Venki • 2 years ago

@Hariom:-
Just to explain you by example.
Say no of bits = 4
hence Max N = 15 (1111)
so loop will execute 4 times
and logN=log15=4(approx).
and worst case complexity=logN

Reply • Share ›

The solution one go into infinite loop when value of n is negative.
The correct solution should be like this:

```
int countSetBits(int n)
{
   int mask=1;
   unsigned int count=0;
   while (mask)
   {
       if (n&mask)
           count++;
       mask<<=1;
   }
   return count;
}
```

please let me know if there is any problem in this solution

Venki rka143 • 4 years ago

Good catch RKA. The following change will correct the issue,

```
/* Function to get no of set bits in binary
   representation of passed binary no. */
int countSetBits(unsigned int n)
{
  unsigned int count = 0;
  while(n)
```

```
  {
    count += n & 1;
    n >>= 1;
  }
  return count;
}

/* Program to test function countSetBits */
int main()
{
    int i = -9;
    printf("%d", countSetBits(i));
    getchar();
    return 0;
}
```

Reply • Share ›

David ➚ Venki • 11 months ago

No changes are necessary, I think. "Unsigned int" argument pre
way to GEEK out on the original algorithms. That is the only wa

```
UNSIGNED int n
```

Reply • Share ›

GeeksforGeeks ➚ Venki • 4 years ago

@rka143 & @Venki

Thanks for the correction guys. We have updated the post.

Reply • Share ›

Sharad Chandra • 4 years ago

Check this:
http://gurmeetsingh.wordpress....
http://stackoverflow.com/quest...

• Reply • Share ›

Subscribe Add Disqus to your site