# Count total set bits in all numbers from 1 to n

Given a positive integer n, count the total number of set bits in binary representation of all numbers from 1 to n.

Examples:

```
Input: n = 3
Output:  4

Input: n = 6
Output: 9

Input: n = 7
Output: 12

Input: n = 8
Output: 13
```

Source: Amazon Interview Question

**Method 1 (Simple)**
A simple solution is to run a loop from 1 to n and sum the count of set bits in all numbers from 1 to n.

```
// A simple program to count set bits in all numbers from 1 to n.
#include <stdio.h>

// A utility function to count set bits in a number x
unsigned int countSetBitsUtil(unsigned int x);
```

Google™ Custom Search

GeeksforGeeks
53,527 people like GeeksforGeeks.

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

```
// Returns count of set bits present in all numbers from 1 to n
unsigned int countSetBits(unsigned int n)
{
    int bitCount = 0; // initialize the result

    for(int i = 1; i <= n; i++)
       bitCount += countSetBitsUtil(i);

    return bitCount;
}

// A utility function to count set bits in a number x
unsigned int countSetBitsUtil(unsigned int x)
{
    if (x <= 0)
        return 0;
    return (x %2 == 0? 0: 1) + countSetBitsUtil (x/2);
}

// Driver program to test above functions
int main()
{
   int n = 4;
   printf ("Total set bit count is %d", countSetBits(n));
   return 0;
}
```

Output:

```
Total set bit count is 6
```

Time Complexity: O(nLogn)

**Method 2 (Tricky)**
If the input number is of the form 2^b -1 e.g., 1,3,7,15.. etc, the number of set bits is b * 2^(b-1). This is because for all the numbers 0 to (2^b)-1, if you complement and flip the list you end up with the same list (half the bits are on, half off).

If the number does not have all set bits, then some position m is the position of leftmost set bit. The number of set bits in that position is n – (1 << m) + 1. The remaining set bits are in two parts:

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

1) The bits in the (m-1) positions down to the point where the leftmost bit becomes 0, and
2) The 2^(m-1) numbers below that point, which is the closed form above.

An easy way to look at it is to consider the number 6:

```
0|0 0
0|0 1
0|1 0
0|1 1
-|-
1|0 0
1|0 1
1|1 0
```

The leftmost set bit is in position 2 (positions are considered starting from 0). If we mask that off what remains is 2 (the "1 0" in the right part of the last row.) So the number of bits in the 2nd position (the lower left box) is 3 (that is, 2 + 1). The set bits from 0-3 (the upper right box above) is 2*2^(2-1) = 4. The box in the lower right is the remaining bits we haven't yet counted, and is the number of set bits for all the numbers up to 2 (the value of the last entry in the lower right box) which can be figured recursively.

```
// A O(Logn) complexity program to count set bits in all numbers from
#include <stdio.h>

/* Returns position of leftmost set bit. The rightmost
   position is considered as 0 */
unsigned int getLeftmostBit (int n)
{
   int m = 0;
   while (n  > 1)
   {
      n = n >> 1;
      m++;
   }
   return m;
}

/* Given the position of previous leftmost set bit in n (or an upper
   bound on leftmost position) returns the new position of leftmost
   set bit in n  */
unsigned int getNextLeftmostBit (int n, int m)
{
   unsigned int temp = 1 << m;
   while (n  < temp)
   {
      temp = temp >> 1;
      m--;
   }
   return m;
}

// The main recursive function used by countSetBits()
unsigned int _countSetBits(unsigned int n, int m);

// Returns count of set bits present in all numbers from 1 to n
unsigned int countSetBits(unsigned int n)
{
   // Get the position of leftmost set bit in n. This will be
   // used as an upper bound for next set bit function
   int m = getLeftmostBit (n);

   // Use the position
   return _countSetBits (n, m);
}

unsigned int _countSetBits(unsigned int n, int m)
{
    // Base Case: if n is 0, then set bit count is 0
    if (n == 0)
       return 0;

    /* get position of next leftmost set bit */
    m = getNextLeftmostBit(n, m);

    // If n is of the form 2^x-1, i.e., if n is like 1, 3, 7, 15, 31,.
    // then we are done.
    // Since positions are considered starting from 0, 1 is added to m
    if (n == ((unsigned int)1<<(m+1))-1)
        return (unsigned int)(m+1)*(1<<m);

    // update n for next recursive call
    n = n - (1<<m);
    return (n+1) + countSetBits(n) + m*(1<<(m-1));
}

// Driver program to test above functions
int main()
{
   int n = 17;
```

Subscribe

## Recent Comments

Abhi You live US or India?

Google (Mountain View) interview · 31 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...

Find subarray with given sum · 2 hours ago

```
    printf ("Total set bit count is %d", countSetBits(n));
    return 0;
}
```

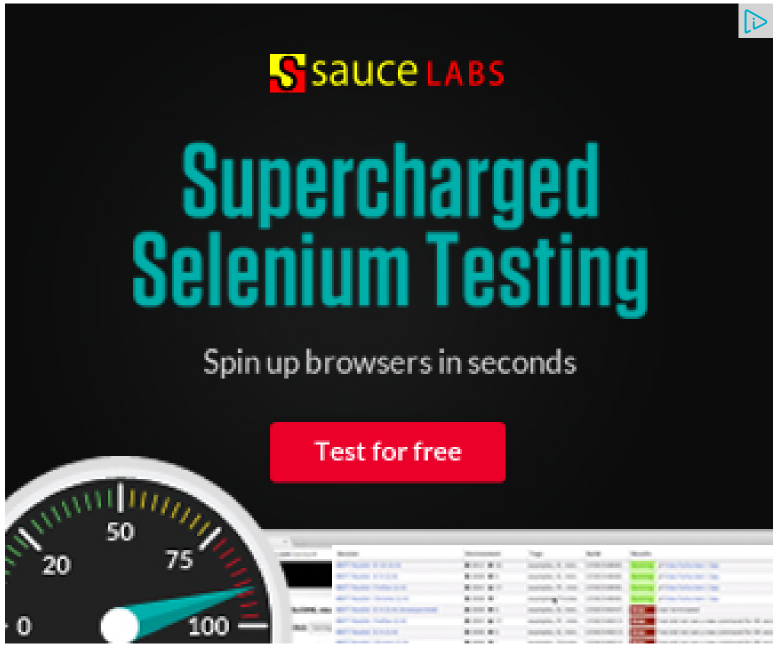
```
Total set bit count is 35
```

Time Complexity: O(Logn). From the first look at the implementation, time complexity looks more. But if we take a closer look, statements inside while loop of getNextLeftmostBit() are executed for all 0 bits in n. And the number of times recursion is executed is less than or equal to set bits in n. In other words, if the control goes inside while loop of getNextLeftmostBit(), then it skips those many bits in recursion.

Thanks to agatsu and IC for suggesting this solution.

**See this for another solution suggested by Piyush Kapoor.**

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)
- 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

9 Tweet 0 0

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