Next Power of 2

Write a function that, for a given no n, finds a number p which is greater than or equal to n and is a power of 2.

```
IP 5
OP 8

IP 17
OP 32

IP 32
OP 32
```

There are plenty of solutions for this. Let us take the example of 17 to explain some of them.

Method 1(Using Log of the number)

```
1.  Calculate Position of set bit in p(next power of 2):
    pos =  ceil(lgn)  (ceiling of log n with base 2)
2.  Now calculate p:
    p   = pow(2, pos)
```

Example

```
Let us try for 17
        pos = 5
        p   = 32
```


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

Method 2 (By getting the position of only set bit in result)

```
    /* If n is a power of 2 then return n */
    1  If (n & !(n&(n-1))) then return n
    2  Else keep right shifting n until it becomes zero
        and count no of shifts
        a. Initialize: count = 0
        b. While n ! = 0
                n = n>>1
                count = count + 1

     /* Now count has the position of set bit in result */
    3  Return (1 << count)
```

Example:

```
    Let us try for 17
                 count = 5
                 p     = 32
```

```
unsigned int nextPowerOf2(unsigned int n)
{
  unsigned count = 0;

  /* First n in the below condition is for the case where n is 0*/
  if (n && !(n&(n-1)))
    return n;

  while( n != 0)
  {
    n  >>= 1;
    count += 1;
  }

  return 1<<count;
}

/* Driver program to test above function */
int main()
{
```


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

```
  unsigned int n = 0;
  printf("%d", nextPowerOf2(n));

  getchar();
  return 0;
}
```

Method 3(Shift result one by one)

Thanks to coderyogi for suggesting this method . This method is a variation of method 2 where instead of getting count, we shift the result one by one in a loop.

```
unsigned int nextPowerOf2(unsigned int n)
{
    unsigned int p = 1;
    if (n && !(n & (n - 1)))
        return n;

    while (p < n) {
        p <<= 1;
    }
    return p;
}

/* Driver program to test above function */
int main()
{
  unsigned int n = 5;
  printf("%d", nextPowerOf2(n));

  getchar();
  return 0;
}
```

Time Complexity: O(lgn)

Method 4(Customized and Fast)

```
   1. Subtract n by 1
      n = n -1


   2. Set all bits after the leftmost set bit
```

```
    2. Set all bits after the leftmost set bit.

    /* Below solution works only if integer is 32 bits */
                n = n | (n >> 1);
                n = n | (n >> 2);
                n = n | (n >> 4);
                n = n | (n >> 8);
                n = n | (n >> 16);
    3. Return n + 1
```

Example:

```
Steps 1 & 3 of above algorithm are to handle cases
of power of 2 numbers e.g., 1, 2, 4, 8, 16,

    Let us try for 17(10001)
    step 1
       n = n - 1 = 16 (10000)
    step 2
       n = n | n >> 1
       n = 10000 | 01000
       n = 11000
       n = n | n >> 2
       n = 11000 | 00110
       n = 11110
       n = n | n >> 4
       n = 11110 | 00001
       n = 11111
       n = n | n >> 8
       n = 11111 | 00000
       n = 11111
       n = n | n >> 16
       n = 11110 | 00000
       n = 11111
```


Recent Comments

Abhi You live US or India?

Google (Mountain View) interview · 27 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

```
    n = 11111

   step 3: Return n+1
    We get n + 1 as 100000 (32)
```

Program:

```
# include <stdio.h>

/* Finds next power of two for n. If n itself
   is a power of two then returns n*/

unsigned int nextPowerOf2(unsigned int n)
{
    n--;
    n |= n >> 1;
    n |= n >> 2;
    n |= n >> 4;
    n |= n >> 8;
    n |= n >> 16;
    n++;
    return n;
}

/* Driver program to test above function */
int main()
{
    unsigned int n = 5;
    printf("%d", nextPowerOf2(n));

    getchar();
    return 0;

}
```

Time Complexity: O(lgn)

References:

http://en.wikipedia.org/wiki/Power_of_2

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.

56 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

KB • a month ago

```
#include <iostream>
using namespace std;

void nextpow2(int n)
{
int ans=1;
while(ans<n) {ans="ans&lt;&lt;1;}" cout<<ans<<"="" ";="" }="" int="" main()="" nextpow2(2);="" nextpow2(3);="" nextpow2(7);="" nextpow2(16);="" return="" (
```

• Reply • Share ›

Gajendra Khatri • a month ago

Please explain the working of 4th method?

• Reply • Share ›

Ivan Povalyukhin • 2 months ago

2 ** (Math.log2 N).ceil

• Reply • Share ›

arnie • 2 months ago

How do I get the closest power of 2 below a given number?
Example.
Input:
2
3
7
16
Output:
2
2

4
16

• Reply • Share ›

Manoj Kumar Regar → arnie • 2 months ago

I improved the same concept...you will get it:)

```
#include<stdio.h>
unsigned int nextPowerOf2(unsigned int n)
{
unsigned count = 0;

/* First n in the below condition is for the case where n is 0*/
if (!(n&(n-1)))
return n;

while( n != 0)
{
n >>= 1;
count += 1;
}

return 1<<(count-1);
}
```

see more

• Reply • Share ›

rupam • 2 months ago

2*(n&n+1)

• Reply • Share ›

Amit Kumar • 4 months ago

Also n&(n-1) ==0;

• Reply • Share ›

Amit Kumar • 4 months ago

void main()
{
int value=19,i=1;

while (i<value){ i="i<<1;" }="" printf("="" value:="" %d",="" i);="" }="">

• Reply • Share ›

Raghav Agrawal • 8 months ago

This method gets the result in (no. of bits set) steps. It first checks if the numb
gets the most significant bit and shifts it by 1 to get the next higher power of 2.

```
public static int nextPowerOf2(int n) {
if(n <= 0) throw new IllegalArgumentException("n should be > 0");

if (getLeastSignificantBit(n) == n) return n; //check if n is itself power of 2
return getMostSignificantBit(n) << 1;
}

private static int getLeastSignificantBit(int n) {
if(n <= 0) throw new IllegalArgumentException("n should be > 0");

return n&(~(n-1));
}

private static int getMostSignificantBit(int n) {
if(n <= 0) throw new IllegalArgumentException("n should be > 0");

int temp = n;
while(temp != 0) {
```

```
n = temp;
temp = temp^(getLeastSignificantBit(temp));
}
return n;
}
```

• Reply • Share ›

Shiva Shankar Anumula • 8 months ago

An another solution for this..

```
int count =1; //No of Left shifts required
while(x >1)
{
x>>=1;count ++;
}

x<<=count;
```

• Reply • Share ›

chandni ➦ Shiva Shankar Anumula • 8 months ago

add the following check to it so that If n itself
is a power of two then returns n!

```
if (x&& !(x&(x-1)))
{
printf ("%d",x);
return;
}
```

also, your solution fails for x=0.
add if (!x) x++; before bit shifting x in the final stmt.

So, here's your darn little program:

```
int x;
if (x&& !(x&(x-1)))
{
printf ("%d",x);
```

see more

Reply • Share ›

chandni ➦ chandni • 8 months ago

ignore the last closing brace for main()

Reply • Share ›

Ankita • 10 months ago

```
 // Keep it Simple !!!!
#include<stdio.h>
#include<conio.h>
#include<math.h>
int main()
{
    int i=0,result=1,num=23;
    if(num<0)
            return 1;
    while(result<num)
    result=(pow(2,i++));
    printf(" aSF :: %d",result);
    getch();
    return 0;
}
```

Reply • Share ›

jugal • 10 months ago

complexity of method 4 is not log(n). because we are performing constant ope
time. please check.

1 • Reply • Share ›

ministar • 11 months ago

```
 int getNext2Power(unsigned num){
        int count=0,c=0; //count : contains no.of bits of given numbe
        for(;num;count++,c+=!(num%2),num/=2);
        if(c==(count-1)) return 1<<c;
        return 1<<count;
}

int main(){
        printf("%d\n",getNext2Power(17));
        printf("%d\n",getNext2Power(32));
        return 0;
}
//O(logn) time
```

• Reply • Share ›

Varun Kumar • 11 months ago

```
 int main()
{
    int i,p=0;
    for(i=0;p>>=1;i++); // shifting the number to the right till it i

     int ans= pow((double)2,(double)(i+1));
      printf("%d",ans);
    return 0;
```

```
}
```

• Reply • Share ›

GeeksforGeeks • 11 months ago

Thanks for pointing this out. There was a typo in the code. There was bitwise a
corrected it now. The code should work find for all cases.

• Reply • Share ›

randeep hooda • 11 months ago

```
 /*
i think sandeep is right....
sandeep keep it up...
haryana k lagte ho..
 */
```

• Reply • Share ›

Sandeep Yadav • 11 months ago

according to question if input is 2 then output should be 2 not 4.
with this ur given condition it vl print 4 as output.
but with my suggested condition it vl give output as 2.
and when your input is 0 you have to make one more if for handleing that case

• Reply • Share ›

GeeksforGeeks • 11 months ago

Doesn&#039t seem to be a bug. Note that the condition "!(a&(a-1))" would be

• Reply • Share ›

Sandeep Yadav • 11 months ago

method 2 also have bug.

if(!(a&(a-1)))

corect me if I am wrong.

• Reply • Share ›

Hanish Bansal • 11 months ago

Method 4 does not work for n=0.

• Reply • Share ›

Hanish • a year ago

How is the complexity of method 4 O(log n) ??
Since there are fixed no. of instructions, should it not be O(1) ??

• Reply • Share ›

Abhi ➦ Hanish • a year ago

If 4 is considered as O(1) then 3 must also be considered as O(1) sinc
number of shifts(since there is a limit on number of bits)

• Reply • Share ›

Nishant Kumar • a year ago

works with GCC compiler.

```
 #include<stdio.h>
int nextPow2(int x){
    if(x == 0)
        return 1;
    if(x & x-1)
        return 1 << (sizeof(x)*8-__builtin_clz(x));
    else
        return x;
}
```

```
int main(){
    int x = 1071741824;
    printf("%d",nextPow2(x));
}
```

• Reply • Share ›

Nishant Kumar • a year ago

I think it will be little bit faster than others in some cases as it iterates only upto

```
 int x = new Scanner(System.in).nextInt();
int count = 0;
int tmp = x;
while(x > 0){
tmp = x;
x&=x-1;
count++;
}
if(count == 1)
System.out.println(tmp);
else if(tmp == 0)
System.out.println("1");
else
System.out.println(tmp << 1);
```

• Reply • Share ›

RAUNAK • a year ago

can we do it by
counting the no of bits and the ans will be 1 followed by count no bits
example :for 17

no of bits in 17 will be 10001
so the ans will be 100000

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

• Reply • Share ›

Sourabh Goyal • 2 years ago

In method 2:
if (n & !(n&(n-1)))
return n;
It does seem to be working at all.The if statement always return false whether
What is the use of this portion of code.

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

• Reply • Share ›

kafee → Sourabh Goyal • 2 years ago

This code
if (n & !(n&(n-1)))
return n;
Which is used for determining exact power of 2 is not correct, it shoulc
if(!(n&(n-1)))
return n;

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

• Reply • Share ›

anurag → kafee • 2 years ago

(n & !(n&(n-1)))

should be

(n && !(n&(n-1)))

see the method 4 of http://www.geeksforgeeks.org/a...

```
/* Paste your code here (You may delete these lines if
```

• Reply • Share ›

Hanish Bansal ➦ anurag • 11 months ago

(y)

• Reply • Share ›

Ankita ➦ Hanish Bansal • 10 months ago

```
 I am not getting the purpose of using this chunk
if (n && !(n & (n - 1)))
        return n;
```

• Reply • Share ›

Manoj Kumar Regar ➦ Ankita • 2 months ago

if n is a power of 2 (in binary 10,100,1000...) , we shoulc
!n&(n-1) is used...how this is working ...let us see with 8
8 : 1000
(8-1): 111
8&(8-1) :0000
!8&(8-1) :1111
if statement encounters true and returns true..
here n&& ! n&(n-1) also determines whether n is zero c
execute....
and while will not execute...finaly returns 1<<0 which is

:)

• Reply • Share ›

Bohemia → Ankita • 9 months ago

Basically it sets the right most 1-bit to 0 :) So a power o
it is set to 0 ,thus n&&!(n-1))) should become zero,if it is
2,else not

• Reply • Share ›

swati → Ankita • 9 months ago

this is to check if n is power of 2.. if yes return the numb

• Reply • Share ›

Dhaval Patel • 2 years ago

[sourcecode language="java"]

```
public class NextPowerOf2 {
public static void main(String[] args) {
int input = 32;
int output = 1;

for (int i=0;i<input;i++) {

output = output << 1;

if (output >= input) {
break;
}

}

System.out.println(output);
```

}

Reply • Share ›

crazypro • 2 years ago

```
#include
#include

int findnextpow2(int);

int main()
{
int num;

int nextnum;

printf("Enter the number whose next power of 2 number is to be find..\n");
scanf("%d",&num);

nextnum = findnextpow2(num);

printf("next power of two of the given nunmber %d is %d\n",num,nextnum);

getch();
return 0;
```

see more

Reply • Share ›

saurabh • 2 years ago

```
 #include<stdio.h>
#include<conio.h>
#include<math.h>
int main()
```

```
{
int a,i=1;
printf("NEXT POWER OF 2\n");
printf("Enter NO. :");
scanf("%d",&a);
if(a==0)
printf("1");
else
{
while( ((int)pow(2,i)) <= a)
{
i++;
}
printf("NEXT POWER of 2 : %d",(int)pow(2,i));
}
getch();
}
```

• Reply • Share ›

saurabh ➦ saurabh • 2 years ago

above code Method 4 does not work well for 1 as inp

• Reply • Share ›

kartik ➦ saurabh • 2 years ago

@saurabh: It works fine. It produces 1 as output which is corre

```
# include <stdio.h>

/* Finds next power of two for n. If n itself
```

```
unsigned int nextPowerOf2(unsigned int n)
{
    n--;
    n |= n >> 1;
    n |= n >> 2;
    n |= n >> 4;
    n |= n >> 8;
    n |= n >> 16;
    n++;
    return n;
}
```

see more

• Reply • Share ›

saurabh • 2 years ago

int i=1;
while(((int)pow(2,i)) < inp)
{
i++;
}
printf("NEXT POWER of 2 : %d",(int)pow(2,i));

• Reply • Share ›

abhishek • 3 years ago

Get the first set bit of number n then left shift to get the result
or,
if the number is itself is power of 2 return number itself

```
 int count = 0;
while(n != 0)
```

```
{
  //to get the last set bit
  x = n & (~n+1);
  //unset last bit
  n = n & (n-1);
  count++;
}
//if n is power of 2
if(count == 1) return x;
else
 //otherwise left shift the first set bit of n
 return x = x<<1;
```

Please let me know about my approach.

• Reply • Share ›

Abhirup Ghosh • 3 years ago

```
 int next_2_power (int n)
{
  return 1<<((int)log2(n)+1);
}
```

• Reply • Share ›

puneet saraswat • 3 years ago

```
 int next_power_of2(int n)
{
    if (!(n & (n - 1)))
    {
```

```
    }

    while(n & (n - 1))
    {
        n &= n-1;
    }
    return n << 1;
}
```

• Reply • Share ›

Raghu • 4 years ago

An Easier solution would be ,

```
int power(int i){
        int count=0;
        int previ=0;
        previ=i;
        while(i!=1) // making all the bits zero except the last
        {
                i>>=1;
                count++;
        }
        if((previ & (previ-1))==0)// if i is a power of 2 just left sl
                i<<=count;
        else
                i<<=(count+1);

         return i;
}
```

• Reply • Share ›

Venki • 4 years ago

There is another way. The number will be power of 2 if there only leftmost bit is leading zeros in the bit pattern of the number. Assume we have a function to c architectures like ARM provides direct instruction to count leading zeros). We bit using this leading zeros count. An example is given,

x = 0x12345678

In binary we can write it as

00010010001101000101011001111000

We will get number of leading zero count as 3. To get next higher power of 2 v and reset all other bits. This can be achieved easily,

1 << (32 - leadingZeros(x))

However, we need to check that x is not exact power of 2, in which case the re easily (x & (x-1)).

• Reply • Share ›

Sambasiva • 4 years ago

Complete Solution...

```
 int nextpow2(int n)
{
        int m;

        if(!n) return 1;

        int sign = (n < 0) ? (n = -n), -1 : 1;
```

```
        if(!(n & n-1))
                return sign * n;

        while(m = n, n &= n-1);

        return sign * (m<<1);
}
```

• Reply • Share ›

Sambasiva • 4 years ago

```
 int nextpow2(int n)
{
        int m;

        if(!( n & n-1))
                return n;
        while(n)
        {
                m = n;
                n = n & n-1;
        }
        return m<<1;
}
```

• Reply • Share ›

Shekhu ➦ Sambasiva • 4 years ago

Sambasiva' s solution doesn't work for n = 0. I guess initializing m = 1

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

```
# include <conio.h>

int nextpow2(int n)
{
        int m;

        if(!( n & n-1))
                return n;
        while(n)
        {
                m = n;
                n = n & n-1;
        }
        return m<<1;
}
```

see more

• Reply • Share ›

geek4u ➦ Sambasiva • 4 years ago

I think you missed some brackets in your code. The code should be.

```
 int nextpow2(int n)
{
        int m;

        if(!( n & (n-1)))
                return n;
        while(n)
        {
                m = n;
                n = n & (n-1);
```

```
        }
        return m<<1;
}
```

• Reply • Share ›

Load more comments

Subscribe Add Disqus to your site