Find whether a given number is a power of 4 or not

Asked by Ajay

1. A simple method is to take log of the given number on base 4, and if we get an integer then number is power of 4.

2. Another solution is to keep dividing the number by 4, i.e, do n = n/4 iteratively. In any iteration, if n%4 becomes non-zero and n is not 1 then n is not a power of 4, otherwise n is a power of 4.

```
#include<stdio.h>
#define bool int

/* Function to check if x is power of 4*/
bool isPowerOfFour(int n)
{
  if(n == 0)
    return 0;
  while(n != 1)
  {
   if(n%4 != 0)
      return 0;
    n = n/4;
  }
  return 1;
}

/*Driver program to test above function*/
int main()
{
  int test_no = 64;
  if(isPowerOfFour(test_no))
    printf("%d is a power of 4", test_no);
  else
```

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

```
    printf("%d is not a power of 4", test_no);
  getchar();
}
```

3. A number n is a power of 4 if following conditions are met.
a) There is only one bit set in the binary representation of n (or n is a power of 2)
b) The count of zero bits before the (only) set bit is even.

For example: 16 (10000) is power of 4 because there is only one bit set and count of 0s before the set bit is 4 which is even.

Thanks to Geek4u for suggesting the approach and providing the code.

```
#include<stdio.h>
#define bool int

bool isPowerOfFour(unsigned int n)
{
  int count = 0;

  /*Check if there is only one bit set in n*/
  if ( n && !(n&(n-1)) )
  {
     /* count 0 bits before set bit */
     while(n > 1)
     {
       n  >>= 1;
       count += 1;
     }

    /*If count is even then return true else false*/
    return (count%2 == 0)? 1 :0;
  }

  /* If there are more than 1 bit set
    then n is not a power of 4*/
  return 0;
}

/*Driver program to test above function*/
int main()
{
   int test_no = 64;
   if(isPowerOfFour(test_no))
     printf("%d is a power of 4", test_no);
```

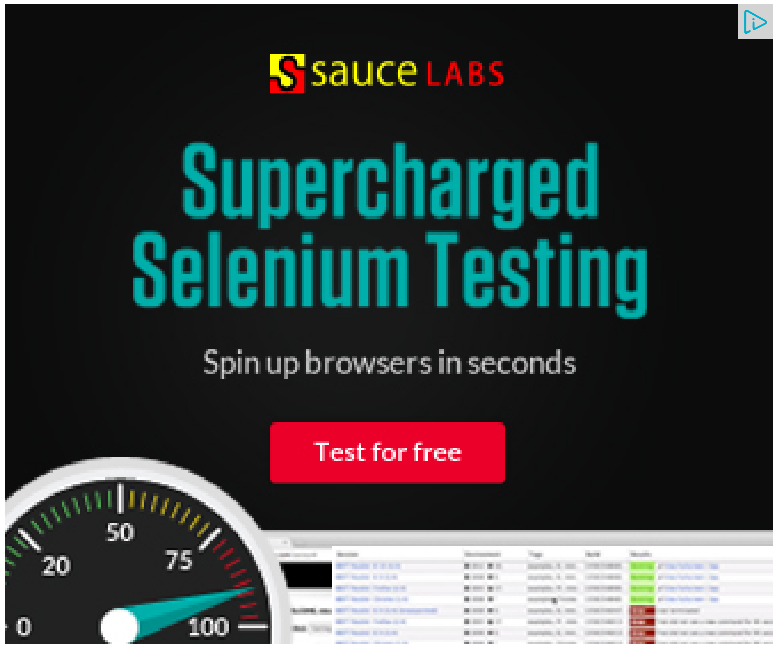
Geometric Algorithms

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

```
  else
    printf("%d is not a power of 4", test_no);
  getchar();
}
```

Please write comments if you find any of the above codes/algorithms incorrect, or find other ways to solve the same problem.



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

3 Tweet 0 0

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

64 Comments GeeksforGeeks

Sort by Newest ▾

Join the discussion…

Anon · 6 months ago

When a number is a power of 4 , there wouldn't be any bit in odd position, for e
4^0 = 000 ... 00001 , one at position 0
4^1 = 000 ... 00100 , one at position 2
4^2 = 000 ... 10000 , one at position 4
and so on.
So we just check if only one 1 bit in the number and also that bit shouldn't be i
To check if it has only 1 bit set in the number, we can do : (n&(n-1)) == 0
To check no bit should be in odd position : (n&0xAAAAAAAA)==0

So solution is :

bool check(unsigned int n){
return (n&(n-1)) == 0 && (n&0xAAAAAAAA) == 0;
}

5 ^ | v ·

Suryabhan Singh · 7 months ago

another solution-

```
int fun(int n)
{
```

705

Subscribe

Recent Comments

Abhi You live US or India?

Google (Mountain View) interview · 20 minutes ago

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

Write a C program to Delete a Tree. · 59 minutes 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 · 1 hour ago

AdChoices

► Math Geeks

► The Power of 2

► Numbers Number

```
    if(!n)
        return 0;
    else if(n==1)
        return 1;
    else if(n==2)
        return 0;
    else if(n&(n-1))
        return 0;
    else
        return 1;
}
```

asunel • 8 months ago

Another short efficient solution :

```
#include<stdio.h>
#include<math.h>
#define bool int

bool isPowerOfFour(unsigned int n)
{
  return n &(n-1)|(int)log2(n&(-n))&1;
}

/*Driver program to test above function*/
int main()
{
   int test_no = 16;
   if(!isPowerOfFour(test_no))
```

```
    printf("%d is a power of 4", test_no);
  else
    printf("%d is not a power of 4", test_no);
  getchar();
}
```

1

sumit5113 • 9 months ago

we don't even need to count the number of even number of zeros for number

```
boolean isPowerOfFour(int num){

        if(!num && (num&(num-1))==0){
                //this mean it is exact power of 2
                //now for power of four we must check if num>>2 is an
                while(num>1){
                        //if num = 4*(number is multiple of 4) -> the
                        num=num>>2;
                        if(num==0){
                        //num=8,32,128...
                                return false;
                        }
                }

                return true;
        }

        return false;
}
```

1

crazy • 11 months ago

In method 3 instead of counting number of zeros in the right side 1)First find th
greater than 1 and it is odd then it is a power of 4..

```
 bool isPowerOfFour(unsigned int n)
{
  /*Check if there is only one bit set in n*/
  if ( n && !(n&(n-1)) )
  {
   /*find the rightmost set bit if it is odd and greater than 1 returı
   return (n&~(n-1)>1)&&((n&~(n-1))%2);
  }

  /* If there are more than 1 bit set
    then n is not a power of 4*/
  return 0;
}
```

correct me if i am wrong...

crazy ➦ crazy • 11 months ago

oops i am sory it will not work correct the return statement into
return ((log2(n)+1)>1)&&((log2(n)+1)%2)

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

adrain • 11 months ago

the method 3 has so many fallacies

No .1 > not multiples of 4 will have only one bit set that is true for multiple of 2
example 24 (6x4) = 11000 (two bits are set although number is multiple of fou

no2.>> The idea of even number of zeros is useless
example 8 (4x2) = 1000 (it has only 3 zeros before set bit)

In fact if the number has a single bit set and has 00 before it it will always be a

please correct method 3.
Thanks

GeeksforGeeks → adrain • 11 months ago

Please take a closer look. The question is about power of 4, not multipl

Niraj • a year ago

Method 3 can be solved by this way .
Time = O(1) space= O(1)

```
int power_of_four(int n)
{
int first_set_bit =1;
if(n&& !(n&(n-1)))
{
/* find first set bit */
first_set_bit=n&(-n);
}
if(first_set_bit-1)%2==0)
return TRUE;
else
```

return FALSE;
}

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

ashish dey • a year ago

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

#include
int main()
{
int n;
printf("enter the value of n");
scanf("%d",&n);
if(!(n & (n-1)) && ((n-1)%3 == 0))
printf("power of 4");
else
printf("not a power of 4");

}

gaurav • a year ago

just check it is power of 2 .. if yes count bit position using shift operator.
void is_power_of_four(int num)
{
int count = 0;
int m= num;
int x = num&(num-1);
if(!x)// checks it is power of 2

```
{
while(m != 0)
{
m = m>>1;
count++;
}
if(count%2 != 0)
{
printf("power of 4");
}
else
```

see more

vasu • a year ago

I do not understand this :

/*Check if there is only one bit set in n*/
if (n && !(n&(n-1)))

Can someone help me?

If n = 1100 then , this will give true .is it not wrong?

Vibhu Tiwari → vasu • a year ago

No this would not give a true result as it is not a bit-wise and but it is a
conditions are checked.Here as you said that n=1100 then n-1=1011 t
result in 1000 i.e. 8 of which the not operator when applied the result is
applied checks that as one condition is wrong so the overall result is al

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

vasu Vibhu Tiwari • a year ago

Got it!! That made it very clear.Thanks !

Vibhu Tiwari • a year ago

The power of 4 can just be checked by right shifting the number by 2 bits and
number.

```
#include <stdio.h>
#include <stdlib.h>
unsigned int powof4(unsigned int n){
if((n%(n>>2))==0)
return 1;
else
return 1;
}
int main()
{
int b=powof4(16);
if(b==1)
printf("is a power of 4");
else if(b==0)
printf("is not a power of 4");
return 0;
}
```

Vibhu Tiwari Vibhu Tiwari • a year ago

Sorry here in else part return 0 instead of 1.

pr6989 • 2 years ago

```
 #include<stdio.h>
long count_bits(unsigned int);
int main()
{
unsigned int n;
printf("Enter a number : ");
scanf("%u",&n);
int set_bit_no=log2(n&-n)+1;
long x=count_bits(n);
if((x==1)&&(set_bit_no%2==1))
printf("%u is a power of 4\n",n);
else
printf("%u is not a power of 4\n",n);

return 0;
}

long count_bits(unsigned int x)
{
  unsigned int c; // c accumulates the total bits set in x
  for(c=0;x;c++)
    x&=x-1; // clear the least significant bit set
  return c;
}
```

pr6989 • 2 years ago

Rather than finding the count of zero bits before the (only) set bit and testing if whether the single set bit is at an odd position or not.For eg:

4^0=1=0001 (set bit is at position 1)
4^1=4=0100 (set bit is at position 3)
4^2=16=10000 (set bit is at position 5)

...and so on

anji.swe • 2 years ago

Is power of 2?
(n&(-n)) = n
(n&(1+~n)) = n
(n & (~(n & (n - 1)))) =n
(n & (n - 1))=0
~(n & (n - 1))=-1
!(n & (n - 1)) =1

0, 1 are treated as powers of 2, if don`t want put a condition

So... use any one of them in if condition and evaluate n...

Is power of 4?
(~ (n & (n-1)) & (n & 0x555 555 54)) == n

// anlyze the bits of powers of 4 , will get the 0x555 555 54 number.

Is power of 8?
(~ (n & (n-1)) & (n & 0x2 924 924 8)) == n

see more

Padam • 2 years ago

```
// this function return 1 if n is a power of 4
int isPowerOfFour(int n)
{
if(n == 0)
return 0;
while(n%4 == 0)
{
n = n/4;
}

if( n == 1)
return 1;
else
return 0;
}
```

Ankur • 3 years ago

Its written (n && !(n&(n-1)))

Shouldt it be only !(n&(n-1)) as if its power of 2 say 1000 then n&n-1 is 0 so !(
be 0 so wudnt go into if

PsychoCoder ➜ Ankur • 2 years ago

It is because 0 is not the power of 4

asd • 3 years ago

Here is the O(1) Solution.

bool isPowerOfFour(int n)
{
if (n && !(n & (n-1)))
{
int x = log2(n&-n);
if(x == ((x >> 1) << 1))
return true;
}
return false;
}

sudeep • 3 years ago

bool isPowerOfFour(unsigned int n)
{
int count = 0;

/*Check if there is only one bit set in n*/
if (n && !(n&(n-1)) & ((((~0)/3)& n)!=0))
return true;
return false;
}

Nitin → sudeep • 3 years ago

you'r code is wrong

sarath • 3 years ago

here is the one line

```
 if(n&&!(n&(n-1))&&!(n<=2))
printf("it is the power of 4  ");
printf( "it is the power of 4.." );
```

Manna sarath • 3 years ago

this is a code for checking whether the number is of power of 2 or not!

rajx • 3 years ago

chek this out:

```
 void is Pow_of_four( int i )
{
    if( !(i & (i-1)) && i & 0x55555554 )
        printf("yes");
    else
        printf("No");
}
```

anji.swe rajx • 2 years ago

hi
can any one explain this logic ?

Raja Sriram anji.swe • 2 years ago

1010101010101010101010100 is the binary representation c
operated with any power of 2(but not power of 4 like 8,32) answ
answer is itself..

coz here all the alternate bits are set . .

I do not thing rajx • 2 years ago

This is good

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

jagannath rajx • 3 years ago

good one

Nitin jagannath • 3 years ago

nice code ....

Tulley • 3 years ago

```
 int isPowerOf4 (int num)
{
    int num1 = num >> 2;
    if ((num1 << 2) == num))
    {
        return 1; /*TRUE*/
    }
    return 0; /*FALSE*/
}
```

kartik Tulley • 3 years ago

Looks like a program to check for multiple of 4, not power of 4. For exa
not a power of 4.

Vikas • 3 years ago

Given a number say n, find next power of 4.

e.g.,
input = 7 output = 16
input = 16 output = 64

Venki → Vikas • 3 years ago

@Vikas, a simple approach, initialize an unsigned integer with 1 and sl
resultant shift is less than n.

Suresh • 3 years ago

```
 //Check if a number is a Power of 4
#include<stdio.h>
char PowerOf4(int x)
{
        while(x>1)
        {
                //check if x is power of 2
                if(x&(x-1)!=0)
                        return 'Y';
                x>>=2;
                //right shifting will eventually result in x=4
                if(x==4)
                        return 'N';
        }
        return 0;
}
```

Suresh Paldia → Suresh • 3 years ago

Moderator.. The program above has a small mistake.

"Y" AND "N" are interchanged. below is the correct code that I actually

```
 #include<stdio.h>
char PowerOf4(int x)
{
    while(x>1)
    {
        //if x is not power of 2, return N
        if(x&(x-1)!=0)
            return 'N';
        x>>=2;
        //right shift by 2 eachtime eventually results in x=4
        if(x==4)
            return 'Y';
    }
    return N;
}
```

Nitin → Suresh Paldia • 3 years ago

dude @suresh : you are checking only power of 2 ....then wt ab

ajit → Suresh Paldia • 3 years ago

check with 18, 66, 72 , it gives 'Y'

Saket · 4 years ago

One simple thing which i can think of is :

Steps:-

1) Find out if the number is power of 2 or not.
that is num = 2 ^n;

code :-

```
     //check for a non zero number to be 2 ^ n
    if ((num != 0) &&(num & (num - 1))== 0)
    {
       //finding the value of power
       for (i=1;i++)
       {
        num = num >> 1;
        if (num == 0)
        break;
       }
```

see more

Pranshu · 4 years ago

```
   #include <stdio.h>
  // Returns the number of set bits in x
  int pop(unsigned x) {
     x = x - ((x >> 1) & 0x55555555);
     x = (x & 0x33333333) + ((x >> 2) & 0x33333333);
     x = (x + (x >> 4)) & 0x0F0F0F0F;
     x = x + (x >> 8);
```

```
    return x & 0x0000003F;
}

int main() {

    unsigned n,c_zeros;
    scanf("%u",&n);

    // sets all the bits after the rightmost set bit, and resets the
    c_zeros = -n & ( n -1 );
```

see more

kartik Pranshu • 4 years ago

Looks like the program doesn't work for 8, 32, 128, ... It prints them as

Aishwarya Singh • 4 years ago

it is not working for 81

3 power 4 = 81

Aishwarya Singh Aishwarya Singh • 4 years ago

also for 625 etc

Kunal Aishwarya Singh • 4 years ago

The program is to check whether given number is "POWER O

whether a number is 4th power of some base..

Ashirs • 4 years ago

4 = 100
8 = 1000
12 = 1100

the last two digits of binary representation should be zero.

So efficient solution would be
bool IsPowerOfFour_V1(int number)
{
return (number & 3 == 0);
}

Ashirs ➦ Ashirs • 4 years ago

Sorry, I mistaken it as amultiple of four

chao • 4 years ago

```
 bool isPowerOfFour(int i)
{
        while((i>>2)<<2==i && i>4){
                i = i>>2;
        }

        if(i==4) return true;
        else return false;

}
int _tmain(int argc, _TCHAR* argv[])
{
        int test_no = 128;
        if(isPowerOfFour(test_no))
                printf("%d is a power of 4\n", test_no);
        else
                printf("%d is not a power of 4\n", test_no);

        return 0;
}
```

Load more comments

Subscribe   Add Disqus to your site