(1) (2) (3) 231. Power of two ***

Average Rating: 4.94 (72 votes)

Сору

Given an integer, write a function to determine if it is a power of two.

Example 1:

Aug. 17, 2019 | 20.4K views

```
Input: 1
Output: true
Explanation: 2^0 = 1
```

```
Example 2:
  Input: 16
 Output: true
  Explanation: 2^4 = 16
```

Example 3:

```
Input: 218
Output: false
```

Overview

Solution

Java Python

1 class Solution(object): def isPowerOfTwo(self, n): if n == 0: 3

We're not going to discuss here an obvious $\mathcal{O}(\log N)$ time solution

```
return False
             while n % 2 == 0:
                n /= 2
             return n == 1
Instead, the problem will be solved in \mathcal{O}(1) time with the help of bitwise operators. The idea is to discuss
such bitwise tricks as

 How to get / isolate the rightmost 1-bit: x & (-x).

 How to turn off (= set to 0) the rightmost 1-bit: x & (x - 1).
```

These tricks are often used as something obvious in more complex bit-manipulation solutions, like for N Queens problem, and it's important to recognize them to understand what is going on.

- Intuition

The idea behind both solutions will be the same: a power of two in binary representation is one 1-bit, followed by some zeros:

$4 = (00000100)_2$

 $8 = (00001000)_2$

A number which is not a power of two, has more than one 1-bit in its binary representation:

 $1 = (00000001)_2$

 $2 = (00000010)_2$

- $3 = (00000011)_2$
- $5 = (00000101)_2$
- $6 = (00000110)_2$ $7 = (00000111)_2$

The only exception is 0, which should be treated separately.

Let's first discuss why x & (-x) is a way to keep the rightmost 1-bit and to set all the other bits to 0.

Adding 1 to $\neg x$ in binary representation means to carry that 1-bit till the rightmost 0-bit in $\neg x$ and to set all the lower bits to zero. Note, that the rightmost 0-bit in $\neg x$ corresponds to the rightmost 1-bit in x.

Approach 1: Bitwise Operators : Get the Rightmost 1-bit

Two's complement: -x = -x + 1

x & (-x) keeps the rightmost 1-bit and sets all the other bits to 0

Get/Isolate the Rightmost 1-bit

In summary, -x is the same as $\neg x + 1$. This operation reverts all bits of x except the rightmost 1bit.

Basically, that works because of two's complement. In two's complement notation -x is the same as $\neg x + y$

1. In other words, to compute -x one has to revert all bits in x and then to add 1 to the result.

x = 70 0 0 0 1 1

~x + 1

x & (-x)

x = 6

1

1

1

0

0

0

0

0

0

0

0

0

0

0

1

1

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

1

0

0

0

0

0

x & (x - 1) = 0

x & (x - 1) != 0

Сору

Next **①**

Sort By ▼

Post

0

1

1

0

1

0

0

1

1

1

0

0

0

0

0

0

0

1

0

0

1

1

0

x = x & (-x)

1

1

1

0

0

x = 60 0 0 0 0 1 1 All bits are inverted except the rightmost 1-bit 0 1 1 1 0 1 1 Hence, x and -x have just one bit in common - the rightmost 1-bit. That means that x & (-x) would keep that rightmost 1-bit and set all the other bits to 0. 0 0 0 0 1 1 1 0 0 1 1 1 1 1

0 0 0 0 0 **Detect Power of Two** So let's do x & (-x) to keep the rightmost 1-bit and to set all the others bits to zero. As discussed above, for the power of two it would result in x itself, since a power of two contains just one 1-bit.

Other numbers have more than 1-bit in their binary representation and hence for them x & (-x) would

x & (-x) == x

Implementation

not be equal to x itself.

Hence a number is a power of two if x & (-x) == x.

-x = -x + 1

x & (-x)

-x = -x + 1

x & (-x)

Approach 2: Bitwise operators : Turn off the Rightmost 1-bit

Let's first discuss why x & (x - 1) is a way to set the rightmost 1-bit to zero.

x = 4

x - 1

x & (x - 1)

x & (x - 1)

x = 4

x - 1

x & (x - 1)

Type comment here... (Markdown is supported)

To subtract 1 means to change the rightmost 1-bit to 0 and to set all the lower bits to 1.

0

0

0

0

0

0

2. x & (x - 1) sets this 1-bit to zero, and hence one has to verify if the result is zero x & (x - 1) ==

0

0

0

0

0

0

1

0

0

0

0

0

0

0

0

0

0

0

0

0

```
if x is a power of two
                           x = 6
                                            0
                                                                        0
                                                0
                                                     0
                                                          0
                                                               1
                                                                   1
                                                                                   x != x & (-x)
```

0

1

0

0

0

1

0

0

0

0

```
Сору
               Python
                            C
  C++
          Java
     class Solution(object):
          def isPowerOfTwo(self, n):
                  return False
              return n & (-n) == n
Complexity Analysis
  • Time complexity : \mathcal{O}(1).
   • Space complexity : \mathcal{O}(1).
```

Now AND operator: the rightmost 1-bit will be turned off because 1 & 0 = 0, and all the lower bits as well.

Turn off

Turn off the Rightmost 1-bit

the rightmost 1-bit : $\times \& (x - 1)$ x = 6

The solution is straightforward:

1. Power of two has just one 1-bit.

Detect Power of Two

x & (x - 1) 0 0 0 0 0 0 0 0 x & (x - 1) == 0if x is a power of two x = 60 0 0 0 1 0 1 0 0 0 0 1 0

0

0

• Space complexity : $\mathcal{O}(1)$. Rate this article: * * * * *

O Previous

Comments: 16

Preview

• Time complexity : $\mathcal{O}(1)$.

Complexity Analysis

Implementation

Java

C++

3

4

Python

if n == 0:

def isPowerOfTwo(self, n):

return False return n & (n - 1) == 0

class Solution(object):

C

Ixyuan0420 * 15 @ August 18, 2019 1:09 PM Great post. Thanks ;) 13 A V C Share Reply bitsrajat 🛊 4 🗿 August 19, 2019 9:21 PM why did you convert the int value to long? 4 A V C Share Share SHOW 5 REPLIES

MartinPayne ★2 ② June 8, 2020 9:27 PM Great explanation, maybe add a part that explains why you need to cast to long (I'm sure overflow but having pictures for that too would be helpful). 2 A V 🗗 Share 🦘 Reply nits2010 🖈 623 ② August 20, 2019 6:46 PM

there are 3 zero at the end, not 2 zero.

ArizonaZervas 🖈 1 🧿 June 14, 2020 7:13 AM

1 A V 🗈 Share 🦘 Reply

Awesome explanation.! The diagrams help a lot.!

sir x = 8 binary is 1000

yendoan007 ★4 ② August 18, 2019 6:53 AM

mjolewis 🛊 3 ② August 18, 2019 8:52 PM

Great solution and explanation. 4 A V C Share Reply

3 A V 🗗 Share 🦘 Reply

Amazing solution.

and x = 8-1 = 7 has binary 111 not 011 Read More 2 A V C Share Share SHOW 1 REPLY

The C and C++ solutions for approach 1 should've been for approach 2. 1 A V Share Share Reply SHOW 1 REPLY maroochydore 🖈 1 🧿 September 10, 2019 6:13 PM

> Today I learned! Great Post! 0 ∧ ∨ ☑ Share ¬ Reply

(12)

I took granted that number is always integer type. but computer already deals with number as binary format. Handling number with bitwise gives me huge inspiration always. 1 A V 🗗 Share 🦘 Reply zzznotsomuch * 54 ② June 9, 2020 6:46 AM