Previous (/articles/largest-divisible-subset/) Next (/articles/remove-k-digits/)

342. Power of Four (/problems/power-of-four/)

Sept. 4, 2019 | 10.3K views

Average Rating: 4.79 (19 votes)

Given an integer (signed 32 bits), write a function to check whether it is a power of 4.

Example 1:

```
Input: 16
Output: true
```

Example 2:

```
Input: 5
Output: false
```

Follow up: Could you solve it without loops/recursion?

Solution

Overview

Prerequisites

This bitwise trick will be used as something already known:

- How to check if the number is a power of two : `x > 0 and x & (x - 1) == 0`.

  Please check the article Power of Two (https://leetcode.com/articles/power-of-two/) for the

detailed explanation.

Intuition

There is an obvious $\mathcal{O}(\log N)$ time solution and we're not going to discuss it here.

Java Python Copy

```
class Solution(object):
    def isPowerOfTwo(self, n):
        if n == 0:
            return False
        while n % 4 == 0:
            n /= 4
        return n == 1
```

Let's discuss $\mathcal{O}(1)$ time and $\mathcal{O}(1)$ space solutions only.

Approach 1: Brute Force + Precomputations

Let's precompute all possible answers, as we once did for the problem Nth Tribonacci Number (https://leetcode.com/articles/n-th-tribonacci-number/).

Input number is known to be signed 32 bits integer (https://en.wikipedia.org/wiki/Integer_(computer_science)#Common_long_integer_sizes), i.e. $x \leq 2^{31} - 1$. Hence the max power of four to be considered is $[\log_4 (2^{31} - 1)] = 15$. Voila, here is all 16 possible answers: 4^0, 4^1, 4^2, ..., 4^{15}. Let's precompute them all, and then during the runtime just check if input number is in the list of answers.

Java Python Copy

Articles > 342. Power

```python
class Powers:
    def __init__(self):
        max_power = 15
        self.nums = nums = [1] * (max_power + 1)
        for i in range(1, max_power + 1):
            nums[i] = 4 * nums[i - 1]

class Solution:
    p = Powers()
    def isPowerOfFour(self, num: int) -> bool:
        return num in self.p.nums
```

Complexity Analysis

- Time complexity: $\mathcal{O}(1)$.
- Space complexity: $\mathcal{O}(1)$.

Approach 2: Math

If num is a power of four $x = 4^a$, then $a = \log_4 x = \frac{1}{2} \log_2 x$ is an integer. Hence let's simply check if $\log_2 x$ is an even number.

Java Python Copy

```python
from math import log2
class Solution:
    def isPowerOfFour(self, num: int) -> bool:
        return num > 0 and log2(num) % 2 == 0
```

Complexity Analysis

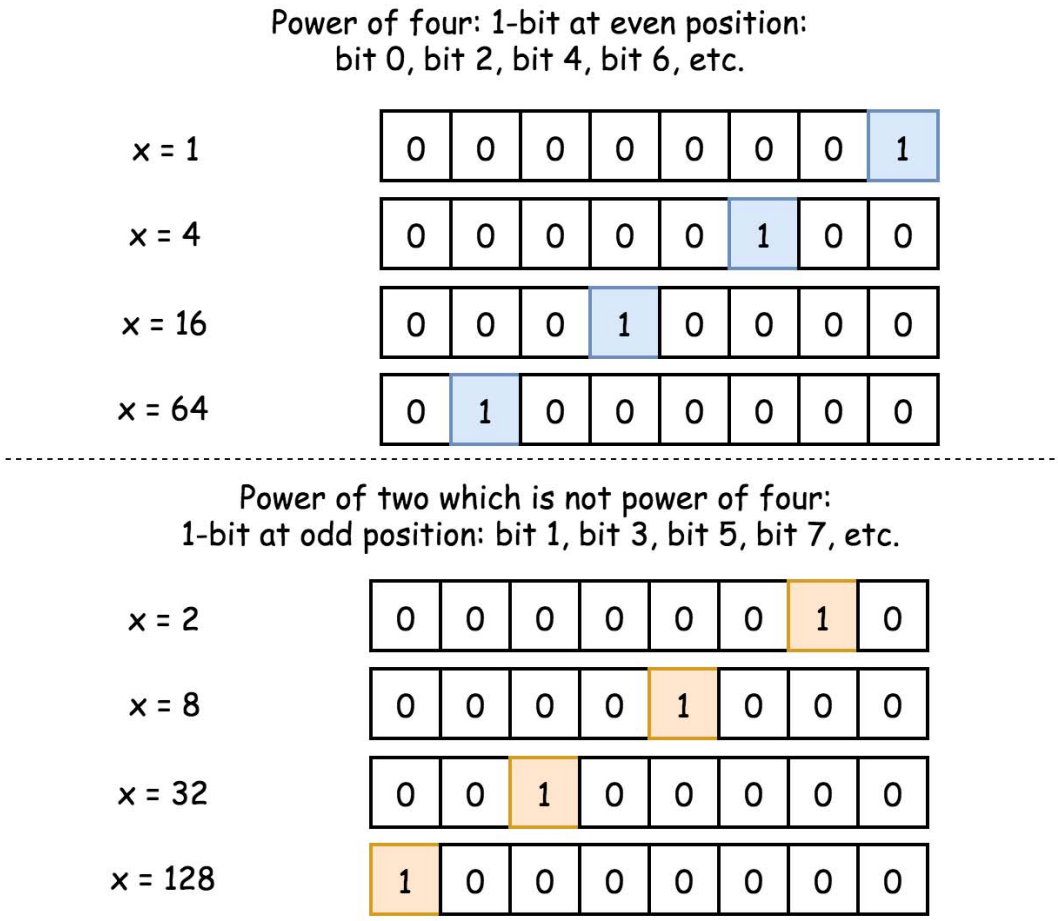
- Time complexity : $\mathcal{O}(1)$.
- Space complexity : $\mathcal{O}(1)$.

Approach 3: Bit Manipulation

Let's first check if num is a power of two: `x > 0 and x & (x - 1) == 0`.

Now the problem is to distinguish between even powers of two (when x is a power of four) and odd powers of two (when x is *not* a power of four). In binary representation both cases are single 1-bit followed by zeros.

> What is the difference? In the first case (power of four), 1-bit is at even position: bit 0, bit 2, bit 4, etc. In the second case, at odd position.



Hence power of four would make a zero in a bitwise AND with number $(101010...10)_2$:

$$4^a \wedge (101010...10)_2 == 0$$

> How long should be $(101010...10)_2$ if x is a signed integer? 32 bits. To write shorter, in 8 charaters instead of 32, it's common to use hexadecimal (https://en.wikipedia.org/wiki/Hexadecimal) representation: $(101010...10)_2 = (aaaaaaaa)_{16}$.

$x \wedge (aaaaaaaa)_{16} == 0$

Java | Python — Copy

```
class Solution:
    def isPowerOfFour(self, num: int) -> bool:
        return num > 0 and num & (num - 1) == 0 and num & 0xaaaaaaaa == 0
```

Complexity Analysis

- Time complexity : $\mathcal{O}(1)$.
- Space complexity : $\mathcal{O}(1)$.

Approach 4: Bit Manipulation + Math

Let's first check if x is a power of two: `x > 0` and `x & (x - 1) == 0`. Now one could be sure that $x = 2^a$. Though x is a power of four only if a is even.

Next step is to consider both cases $a = 2k$ and $a = 2k + 1$, and to compute x modulo after division by three:

$$(2^{2k} \mod 3) = (4^k \mod 3) = ((3+1)^k \mod 3) = 1$$

$$((2^{2k+1}) \mod 3) = ((2 \times 4^k) \mod 3) = ((2 \times (3+1)^k) \mod 3) = 2$$

> If x is a power of two and `x % 3 == 1`, then x *is* a power of four.

Java | Python — Copy

```
class Solution:
    def isPowerOfFour(self, num: int) -> bool:
        return num > 0 and num & (num - 1) == 0 and num % 3 == 1
```

How this works: mod arithmetic

To show the idea, let's compute $x = 2^{2k} \mod 3$.

First, $2^{2k} = 2^{2^k} = 4^k$. Second, since $4 = 3 + 1$, x could be rewritten as

$$x = ((3+1)^k \mod 3)$$

Let's decompose

$$(3+1)^k = (3+1) \times (3+1)^{k-1} = 3 \times (3+1)^{k-1} + (3+1)^{k-1}.$$

The first term is divisible by 3, i.e. $(3 \times (3+1)^{k-1}) \mod 3 = 0$. Hence

$$x = ((3+1)^{k-1} \mod 3)$$

One could continue like this k -> k - 1 -> k - 2 -> ... -> 1 and finally rewrite x as

$$x = ((3+1)^1 \mod 3) = 1.$$

The job is done. Now $y = 2^{2k+1} \mod 3$ is simple, because if $x \mod 3 = 1$, then $y \mod 3 = 2x \mod 3 = 2$.

Complexity Analysis

- Time complexity : $\mathcal{O}(1)$.
- Space complexity : $\mathcal{O}(1)$.

Rate this article:

Previous (/articles/largest-divisible-subset/) Next (/articles/remove-k-digits/)

Comments: 10 Sort By

Type comment here... (Markdown is supported)

Preview Post

bohebohe00 (/bohebohe00) ★ 4 November 15, 2019 4:47 PM

Can someone tell me why we can't use `return Math.log(num) / Math.log(4) % 1 == 0;` ?

4 Share Reply

SHOW 2 REPLIES

(/magetron)

magetron (/magetron) ★ 3 October 26, 2019 3:05 PM

might not be a very decent approach, we can also use __builtin_popcount

```
class Solution {
public:
    bool isPowerOfFour(int num) {
```

Read More

3 Share Reply

SHOW 1 REPLY

(/techzoo)

techzoo (/techzoo) ★ 4 a day ago

The bit manipulation is just not intuitive at all.

4 Share Reply

SHOW 1 REPLY

(/nmaryala)

nmaryala (/nmaryala) ★ 3 September 7, 2019 7:25 PM

Some refresher formulas for mod arithmetic would be helpful in case of approach 4 !!

2 Share Reply

SHOW 1 REPLY

(/asset311)

asset311 (/asset311) ★ 6 20 hours ago

A bit of a hacky solution, but can use the fact that powers of 4 in binary without leading zeroes, will be of odd length. And the most significant bit is 1, with all else being 0.
In Python:

```
class Solution:
```

Read More

2 Share Reply

(/evgeni-nabokov)

Evgeni-nabokov (/evgeni-nabokov) ★ 8 3 hours ago

What is the complexity of `Math.log()` ?

1 Share Reply

(/psherm85)

psherm85 (/psherm85) ★ 15 8 hours ago

One more approach: when you list out the numbers, you can see that they end either in "4" or in "6" (with "1" being the only exception). So return true when num is "1" or when both:
a) num%10 is either 4 or 6
AND
b) num is a power of 2 (i.e. num&-num == num)

Read More

0 Share Reply

(/srinivas_karre)

srinivas_karre (/srinivas_karre) ★ 0 8 hours ago

Java Solution with Bit Simple Bit Manipulation

For each power of 3,5,7,9,.... bit is enabled. We need to check whether bit enabled is odd and it should only one such bit enabled.

Read More

0 Share Reply

(/__thalaiva__)

__thalaiva__ (/__thalaiva__) ★ 91 11 hours ago

Binary search is a decent solution for interview as well, since it does not use loop or recursion. I am not sure why the author has mentioned - "There is an obvious O(logN) time solution and we're not going to discuss it here."

0 Share Reply

SHOW 2 REPLIES

(/takenpilot)

takenpilot (/takenpilot) ★ 2 May 6, 2020 3:53 PM

An Accepted JavaScript solution that converts the number to base4 and then uses regex.

```
var isPowerOfFour = function(num) {
  return num > 0 && /^10*$/.test(num.toString(4));
};
```

Read More

0 Share Reply