Power of Four

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

Example:
Given num = 16, return true. Given num = 5, return false.

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

Credits:
Special thanks to @yukuairoy for adding this problem and creating all test cases.

Solution 1

```
    public boolean isPowerOfFour(int num) {
        return num > 0 && (num&(num-1)) == 0 && (num & 0x55555555) != 0;
        //0x55555555 is to get rid of those power of 2 but not power of 4
        //so that the single 1 bit always appears at the odd position
    }
```

written by aiscong original link here

Solution 2

```
bool isPowerOfFour(int num) {
    return num > 0 && (num & (num - 1)) == 0 && (num - 1) % 3 == 0;
}
```

written by Beibeibei original link here

Solution 3

```
public class Solution {
    public boolean isPowerOfFour(int num) {
        return (num > 0) && ((num & (num - 1)) == 0) && ((num & 0x55555555) == nu
m);
    }
}
```

The basic idea is from power of 2, We can use "n&(n-1) == 0" to determine if n is power of 2. For power of 4, the additional restriction is that in binary form, the only "1" should always located at the odd position. For example, 4^0 = 1, 4^1 = 100, 4^2 = 10000. So we can use "num & 0x55555555==num" to check if "1" is located at the odd position.

written by diegozeng original link here