

## What to use instead?

If you're checking primality → use deterministic Miller–Rabin

Works in  $O(\log n)$  and handles up to  $1e18$  safely.

If you're factoring → use Pollard Rho

Much faster than  $\sqrt{n}$ .

If you're counting divisors → use primality test + Pollard Rho.

If you're checking something mathematical → use formulas.

---

Tell me the exact problem you want to solve

and I'll give you the optimal solution pattern.



when a even number has odd divisor possible

An **even number** can have **odd divisors** — but **only the odd part** of the number can give odd divisors.

### ✓ Rule

A number  $n$  has an **odd divisor > 1** iff  
after removing all factors of 2, the remaining number is  $> 1$ .

Mathematically:

$$n = 2^k \times m$$

where  $m$  is **odd**.

- If  $m > 1 \rightarrow$  **odd divisor exists**

- If  $m = 1 \rightarrow$  NO odd divisor except 1
- 

## 🔥 Examples

### Example 1: 12

$$12 = 2^2 \times 3$$

Odd part = 3  $\rightarrow$  odd divisor exists (3).

### Example 2: 40

$$40 = 2^3 \times 5$$

Odd part = 5  $\rightarrow$  odd divisor exists (5).

### Example 3: 8

$$8 = 2^3 \times 1$$

Odd part = 1  $\rightarrow$  NO odd divisor > 1.

(8 is a pure power of 2.)

### Example 4: 64

$$64 = 2^6 \times 1$$

No odd divisor except 1.

---

## ✓ Final Check

Even numbers have odd divisors > 1 unless they are pure powers of 2.

So:

- 6  $\rightarrow$  yes (odd divisor 3)
- 10  $\rightarrow$  yes (odd divisor 5)
- 14  $\rightarrow$  yes (odd divisor 7)
- 18  $\rightarrow$  yes (odd divisor 3, 9)

- $2 \rightarrow$  no (only 1)
  - $4 \rightarrow$  no (only 1)
  - $8 \rightarrow$  no
  - $16 \rightarrow$  no
  - $32 \rightarrow$  no
- 

If you want, I can give you:

- Code to check if an even number has an odd divisor
- A faster mathematical trick depending on your competitive programming problem.



do both

Below are **both solutions**:

1. Mathematical trick (fastest)
  2. Code (Python first, then C++ as you prefer for coding problems)
- 

## Mathematical Trick

An even number  $n$  has an **odd divisor  $> 1$**  iff:

👉  $n$  is NOT a power of 2

Because:

$$n = 2^k \times m$$

- If  $m > 1$ , then  $m$  is an odd divisor.
- If  $m = 1$ , then the number is **pure power of 2**  $\rightarrow$  **no odd divisor except 1**.

✓ So the check becomes:

If  $(n \& (n - 1)) == 0 \rightarrow n$  is power of 2  $\rightarrow$  NO odd divisor

Else → Odd divisor exists.

---

## ✓ Python Code (your preference for ML/quick scripts)

python

 Copy code

```
def has_odd_divisor(n):
    if n % 2 != 0:
        return True # odd numbers always have odd divisors > 1

    # remove all powers of 2
    while n % 2 == 0:
        n //= 2

    return n > 1 # if odd part > 1, odd divisor exists
```

## Fast version (bit trick):

python

 Copy code

```
def has_odd_divisor(n):
    return (n & (n - 1)) != 0 # True if NOT a power of 2
```

## ✓ C++ Code (since you want both for coding problems)

cpp

 Copy code

```
bool hasOddDivisor(long long n) {
    if (n % 2 != 0) return true; // odd number => odd divisor exists

    // remove all factors of 2
    while (n % 2 == 0) n /= 2;

    return n > 1; // odd part > 1 => odd divisor exists
}
```