LC 69 : square root

3种做法：

1. 牛顿法 ☆快得多，而且代码简洁不易出错.
2. 二分查找法
3. bit manipulation. 时间复杂度：O(32) = O(1)！☆

牛顿法代码： 时间复杂度：O(log n)

```
def mySqrt(self, x):
    r = x
    while r*r > x:
        r = (r + x//r)//2
    return r
```

牛顿法原理：

求 $f(x) = x^2 - num = 0$ 的解

这刚好是抛物线与 x 轴的正交点！

假设 $f(x_0) = 0$，即 x_0 是正解

那我们想让近似解无限逼近 x_0.

而 $\lim_{x \to x_0} \frac{f(x) - f(x_0)}{x - x_0} = f'(x_0)$ 即 $f(x)$ 在 x_0 附近的导数

有 $x - x_0 = \frac{f(x) - f(x_0)}{f'(x_0)}$

将 $f(x)=x^2-num$ 代入

$$x_0=x-\frac{x^2-num}{2x}=\frac{x}{2}+\frac{num}{2x}$$

由于通过不断迭代，可以使 x 逼近 X_0。

设第 $k+1$ 次迭代为

$$x_{k+1}=\frac{1}{2}\left(x_k+\frac{num}{x_k}\right)$$

∴ 代码为 $r=(r+x/r)/2$

这里为了返回值为 int，所以用了双斜杠.

一般性有：$f(x)=x^m-a$

$$f'(x)=mx^{m-1}$$

$$x_0=x-\frac{f(x)}{f'(x)}=x-\frac{x^m-a}{mx^{m-1}}=\frac{m-1}{m}x+\frac{a}{mx^{m-1}}$$

即 $x_{k+1}=\frac{m-1}{m}x_k+\frac{a}{mx_k^{m-1}}$

如果 $m=3$，则 $r=\frac{2}{3}r+\frac{a}{3r^2}$

$f(x)$ 必须2阶可导，且起始点正确（即第一次开始迭代的点）

3. bit manipulation:

难道只能死记硬背?

```
public int mySqrt(int x) {
    int res = 0;
    for (int mask = 1<<15; mask != 0; mask >>= 1) {
        int next = res | mask ; // set bit
        if (next <= x / next ) res = next;
    }
    return res;
}
```

貌似只处理这个 int 后一半(即16位)的 bit 值?

例如: 当 int = 8, 即 00000000 0000 0000 0000 0000 0000 1000

next:

10000000 00000000

01000 0 . . . . . .

⋮

00000000 00001010 (2), res = 2

1 | 2 = 3, 3 > 8/3

∴, finally, res = 2