367. 有效的完全平方数

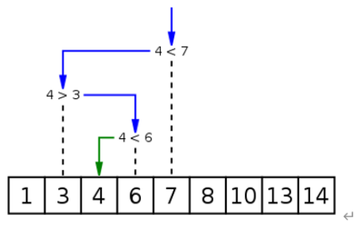
题目

给定一个正整数 num，编写一个函数，如果 num 是一个完全平方数，则返回 True，否则返回 False。

说明：不要使用任何内置的库函数，如 sqrt。

知识点

二分搜索是一**种在有序数组中查找某一特定元素的搜索算法**。搜索过程从数组的中间元素开始，如果中间元素正好是要查找的元素，则搜索过程结束；如果某一特定元素大于或者小于中间元素，则在数组大于或小于中间元素的那一半中查找，而且跟开始一样从中间元素开始比较。如果在某一步骤数组为空，则代表找不到。这种搜索算法每一次比较都使搜索范围缩小一半。

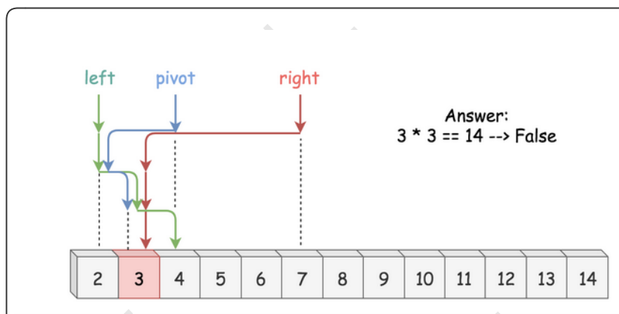


方法（官方解法）

二分查找

递归

- 若 `num < 2`，返回 `true`。
- 设置左边界为 `2`，右边界为 `num/2`。
- 当 `left <= right`：
  - 令 `x = (left + right) / 2` 作为一个猜测，计算 `guess_squared = x * x` 与 `num` 做比较：
    - 如果 `guess_squared == num`，则 `num` 是一个完全平方数，返回 `true`。
    - 如果 `guess_squared > num` ，设置右边界 `right = x-1`。
    - 否则设置左边界为 `left = x+1`。
- 如果在循环体内没有找到，则说明 `num` 不是完全平方数，返回 `false`。



Python | Java

```python
class Solution:
    def isPerfectSquare(self, num: int) -> bool:
        if num < 2:
            return True

        left, right = 2, num // 2

        while left <= right:
            x = left + (right - left) // 2
            guess_squared = x * x
            if guess_squared == num:
                return True
            if guess_squared > num:
                right = x - 1
            else:
                left = x + 1

        return False
```

时间复杂度：$\mathcal{O}(\log N)$。

空间复杂度：$\mathcal{O}(1)$。

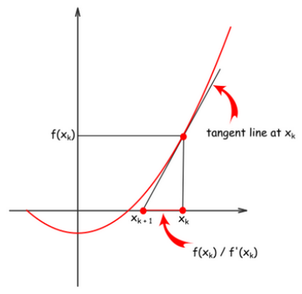
x*x 有可能会导致溢出的
应该写 mid == num / mid && num % mid == 0

牛顿迭代法

牛顿迭代法：公式是如何推导的呢？让我们做一个非常粗略的推导。

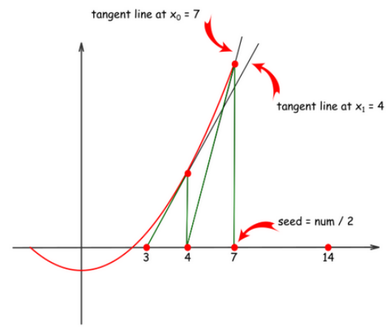
问题是找出：$f(x) = x^2 - \text{num} = 0$ 的根。

牛顿迭代法的思想是从一个初始近似值开始，然后作一系列改进的逼近根的过程。



算法：

- 我们取 `num/2` 作为初始近似值。
- 当 `x * x > num`，用牛顿迭代法取计算下一个近似值：$x = \frac{1}{2}\left(x + \frac{\text{num}}{x}\right)$。
- 返回 `x*x == num`。



Python | Java

```python
class Solution:
    def isPerfectSquare(self, num: int) -> bool:
        if num < 2:
            return True

        x = num // 2
        while x * x > num:
            x = (x + num // x) // 2
        return x * x == num
```

时间复杂度：$\mathcal{O}(\log N)$。

空间复杂度：$\mathcal{O}(1)$。

我的解法

```python
class Solution:
    def isPerfectSquare(self, num: int) -> bool:
        # 开方
        return  num**(1/2) == int(num**(1/2))
```

违背了题目的要求，相当于使用了sqrt

sqrt() 方法返回数字x的平方根

XMind | 试用模式