69. Sqrt(x)

Solved

Easy | Topics | Companies | Hint

Given a non-negative integer `x`, return *the square root of* `x` *rounded down to the nearest integer*. The returned integer should be **non-negative** as well.

You **must not use** any built-in exponent function or operator.

- For example, do not use `pow(x, 0.5)` in c++ or `x ** 0.5` in python.

Example 1:

```
Input: x = 4
Output: 2
Explanation: The square root of 4 is 2, so we return 2.
```

Example 2:

```
Input: x = 8
Output: 2
Explanation: The square root of 8 is 2.82842..., and since we round it down to the nearest integer, 2 is returned.
```

Constraints:

- $0 <= x <= 2^{31} - 1$

Python:

```
class Solution:
    def mySqrt(self, x: int) -> int:
        # Special cases for 0 and 1
        if x < 2:
```

```
        return x

    left, right = 1, x // 2  # sqrt(x) is always <= x//2 for x >= 2

    while left <= right:
        mid = (left + right) // 2
        if mid * mid == x:  # exact square root
            return mid
        elif mid * mid < x:
            left = mid + 1   # move right
        else:
            right = mid - 1  # move left

    return right  # right will be floor(sqrt(x))
```

JavaScript:

```
/**
 * @param {number} x
 * @return {number}
 */
var mySqrt = function(x) {
    if (x < 2) return x;  // sqrt(0) = 0, sqrt(1) = 1

    let left = 1, right = Math.floor(x / 2), ans = 0;

    while (left <= right) {
        let mid = Math.floor((left + right) / 2);

        if (mid * mid === x) {
            return mid; // perfect square
        } else if (mid * mid < x) {
            ans = mid;     // store the best possible answer so far
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }

    return ans;
};
```

Java:

```
class Solution {
    public int mySqrt(int x) {
        // Special cases
```

```
        if (x < 2) {
            return x; // sqrt(0)=0, sqrt(1)=1
        }

        int left = 1, right = x / 2;  // sqrt(x) cannot be more than x/2 for x >= 2
        int ans = 0;

        while (left <= right) {
            int mid = left + (right - left) / 2;

            // Use long to avoid integer overflow when multiplying
            long square = (long) mid * mid;

            if (square == x) {
                return mid;  // Exact square root found
            } else if (square < x) {
                ans = mid;   // Possible answer, but keep searching right
                left = mid + 1;
            } else {
                right = mid - 1; // Square too large, move left
            }
        }

        return ans; // The floor of sqrt(x)
    }
}
```