[Click on the logo to find **Problem Statement**]

Intuition

The code implements the square root function using binary search. It starts by handling the base cases where the input is 0 or 1, returning the input itself in those cases. Then, it initializes a search range from 1 to half of the input. Within a while loop, it iteratively narrows down this range by adjusting the start and end points based on whether the square of the midpoint is less than, equal to, or greater than the input. If the square of the midpoint is equal to the input, the midpoint is returned as the square root. If not, the search range is updated accordingly. This process continues until the search range collapses, and the endpoint of the range is returned as the integer square root of the input.

Approach

1. Handle base cases: Check if the input is 0 or 1. If so, return the input itself, as the square root of 0 or 1 is the number itself.
2. Initialize the search range: Set the start point to 1 and the end point to half of the input value (x//2). This is because the square root of any number x cannot be greater than x//2.
3. Binary search: Use a while loop to iteratively narrow down the search range. Calculate the midpoint of the current range.
4. Check if the square of the midpoint equals the input value: If the square of the midpoint equals x, return the midpoint as the integer square root.
5. Update the search range based on comparison: If the square of the midpoint is less than x, adjust the start point to be the midpoint + 1. If the square of the midpoint is greater than x, adjust the end point to be the midpoint - 1.
6. Continue the binary search until the start point exceeds the end point. At this point, return the endpoint as the integer square root of the input value.

Complexity

- Time complexity: O(log(x))

- Space complexity: O(1)

Code

```
class Solution:
    def mySqrt(self, x: int) -> int:
        if x <= 1:
            return x
        start = 1
        end = x//2
        while start <= end:
            mid = (end - start) // 2 + start
            if mid * mid == x:
                return mid
            elif mid * mid < x:
                start = mid + 1
            else:
                end = mid - 1
        return end
```

If you want to see more solutions to coding problems, you can visit: