69. Sqrt(x)

Hint

Easy 7.2K 4.3K

Companies

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`

Accepted **1.6M** | Submissions **4.2M** | Acceptance Rate **37.8%**

Brute force: Start checking from 1 to x and at every step check if $i^2 == x$ or not.

```
for (i : 1 to x)
{
  if (i*i == x) return i
  else if (i*i > x) return i-1
}
```

else if (r*r > x) return (r-1)
}

Optimized approach: Using Binary Search

As we are checking on sorted integers from 1 to x, instead of linear search we can use binary search.

```
l = 1
r = x
while (l <= r)
{
    m = l + (r-l)/2
    m = l + r/2
    m = (l+r)/2
    if (m*m == x) return m
    else if (m*m < x)   l = m+1
    else                r = m-1
}
return r
```

In both the above approaches the logic is correct but calculating m*m for large input x we get integer overflow. To avoid that we can either use

① unsigned long long int l, m, r

(or)

② calculate m using different formula

$$m = l + \frac{r}{2} - \frac{l}{2}$$

$= l + \frac{(r-l)}{2}$

and instead of checking $m * m == x$

check if $m == x/m$

and

$m < x/m$

$T(n) = O(\log n)$

Takeaway:

dealing with overflow using simple techniques on mathematical expressions

note: To check if x is a perfect square or not

→ A perfect square is formed by adding consecutive odd numbers.

i.e. $1 + 3 = 4$

$1 + 3 + 5 = 9$

$1 + 3 + 5 + 7 = 16$

$1 + 3 + 5 + 7 + 9 = 25$

1+3+5+7+9 = 25

⋮

So

```
a = 1
while (a < x)
{
    a = a + (a+2)
    if (a == x) return true
}
return false
```