Integer Root Function

Ben Braniff

Sep 18, 2024

1.0 Problem Statement

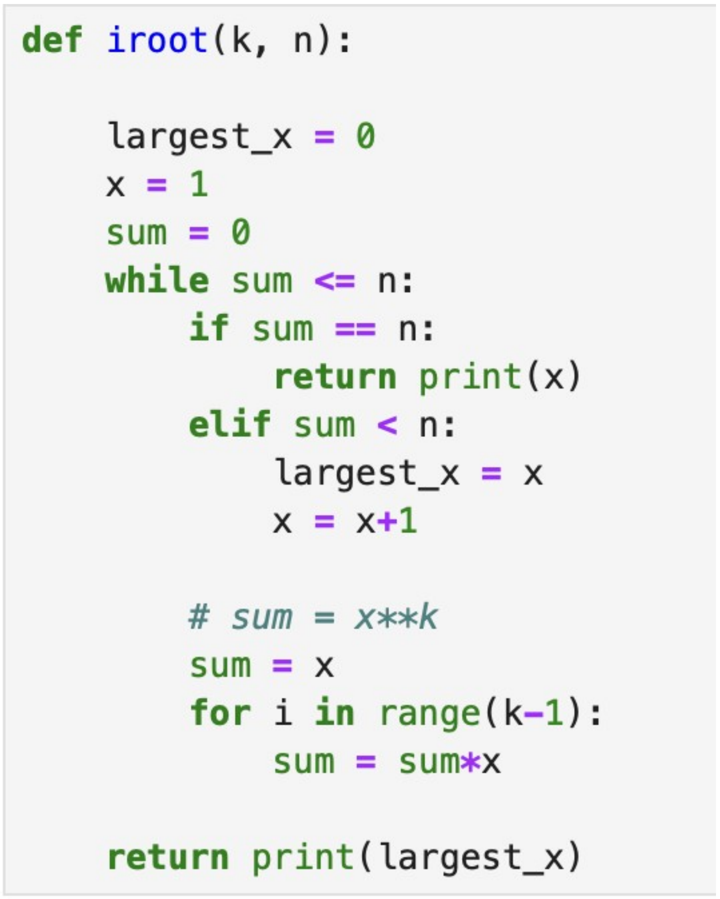
The integer root function: *iroot(k, n)* returns the largest integer x such that x^k does not exceed n, assuming k and n are both positive integers. (Of course, you must remain in the integer realm all the time; you are not allowed to simply call a floating point library and do floor(sqrt(k,n)).)

You are allowed only the four basic arithmetic operations. If you need exponentiation, then do a version with it, then replace the exponentiation with a function of your own which uses only the four basic arithmetic operations.

1.1 General Idea

Test every x value from 1 to k-root of n until we find an x value where x^k equal n or is the closest x without exceeding n.

1.2 The Code



```python
def iroot(k, n):

    largest_x = 0
    x = 1
    sum = 0
    while sum <= n:
        if sum == n:
            return print(x)
        elif sum < n:
            largest_x = x
            x = x+1

        # sum = x**k
        sum = x
        for i in range(k-1):
            sum = sum*x

    return print(largest_x)
```

1.3 Tests

```
[14]: iroot(1, 1)

      1
```

```
[15]: iroot(2, 1)

      1
```

```
[16]: iroot(3, 1)

      1
```

```
[17]: iroot(3, 125)

      5
```

```
[18]: iroot(2, 37)

      6
```

```
[19]: iroot(2,99)

      9
```

1.4 Proof of Correctness

Starting at x = 1, we iterate positively through all integer values of x until x^k equals n or exceeds it.

If x^k equals n, we return x.

If x^k is less than n, we keep going.

If x^k exceeds the value of n, we return the last value of x that did not exceed n (i.e. largest_x).

1.5 Runtime

Since we are using only while and for loops, we will only use summations and not base case and repetition case. The while loop checks does not iterate linearly. Since we are checking all values of x up until the supposed k-root of n, the summation will be $\sum_{i=1}^{n^{1/k}}$

The summation for the for loop inside the while loop will simply be $\sum_{j=0}^{k-2} 1$

So together $T(n) = \sum_{i=1}^{n^{1/k}} \sum_{j=0}^{k-2} 1$

$= \sum_{i=1}^{n^{1/k}} ((k-2)-0+1) \cdot 1$

$= \sum_{i=1}^{n^{1/k}} k-1$

$= (n^{1/k} -1 + 1) \cdot (k-1)$

$= k \cdot n^{1/k} - k$

since $k \cdot n^{1/k}$ grows faster than k, we only care about the first term so this simplifies to $T(n) = k \cdot n^{1/k}$