Python's integer square root algorithm

Mark Dickinson

July 6, 2019

Abstract

TBD

1 Introduction

In this paper we present a simple algorithm to compute the integer square root of a nonnegative number. This algorithm has been implemented for Python's math module and released in Python 3.8 as math.isqrt.

2 Definitions

Definition 1. Suppose n is a positive integer. Call a positive integer a a *near square root* of n if $(a-1)^2 < n < (a+1)^2$.

Equivalently, a is a near square root of n if a is either $\lfloor\sqrt{n}\rfloor$ or $\lceil\sqrt{n}\rceil$. In particular, if $n = a^2$ is a square then a is the only near square root of n.

Definition 2. For a nonnegative integer n, the *integer square root* of n is the unique nonnegative integer a satifying $a^2 \leq n < (a+1)^2$.

Given an algorithm to compute near square roots, it's easy to compute integer square roots: suppose that n is a positive integer and a is a near square root of n. If $a^2 \leq n$ then $a^2 \leq n < (a+1)^2$ and so a is the integer square root of n, if not, then $(a-1)^2 < n < a^2$ and so $a-1$ is the integer square root of n.

3 Key lemma

The key idea is that given a near square root d of $\lfloor n/k^2\rfloor$ for suitable k, dk is then an approximation to $\sqrt{n}$ and a single iteration of Newton's method gives an improved approximation. If we're careful not to choose k too large with respect to n, that improved approximation will again be a near square root. The following lemma makes this precise.

Lemma 1. *Suppose that n is a positive integer, and choose a positive integer M satisfying $4M^4 \leq n$. Given a near square root d of $\lfloor n/4M^2\rfloor$, define a by*

$$a = Md + \left\lfloor \frac{n}{4Md} \right\rfloor.$$

Then a is a near square root of n.

Proof. By definition, we have

$$(d-1)^2 < \left\lfloor \frac{n}{4M^2} \right\rfloor < (d+1)^2.$$

Since $(d+1)^2$ is an integer, we can remove the floor brackets to obtain

$$(d-1)^2 < \frac{n}{4M^2} < (d+1)^2.$$

Taking square roots throughout, multiplying by $2M$, and then rearranging gives

$$|2Md - \sqrt{n}| < 2M.$$

Squaring and dividing through by $4Md$ gives

$$0 \leq Md + \frac{n}{4Md} - \sqrt{n} < \frac{M}{d}.$$

Now we can replace $n/4Md$ with its floor to give

$$-1 < a - \sqrt{n} < M/d$$

We claim that $M \leq d$. Indeed, from the assumption that $4M^4 \leq n$ we have $M^2 \leq n/4M^2 < (d+1)^2$, so $M < d+1$.

So now

$$-1 < a - \sqrt{n} < 1$$

from which $(a-1)^2 < n < (a+1)^2$, as required. □

4 Implementation

Like many arbitrary-precision integer implementations, Python's own big integer implementation is based on binary, so multiplications and divisions by powers of 2 can be performed efficiently by shifting. If we take M to be the largest power of 2 satisfying $4M^4 \leq n$ at each stage, we get the following recursive integer square root implementation.

```
def near_sqrt(n):
    """A near square root of a positive integer."""





def isqrt(n):
    """Integer square root of a nonnegative integer."""
```