Proving the Correctness of an Iterative Implementation of the Function `isqrt`

Karl Stroetmann

April 26, 2021

1 Mathematical Preliminaries

The integer square root $\texttt{isqrt}(n)$ of a natural number n is defined as the largest natural number r such that r^2 is less or equal than n:

$$\texttt{isqrt}(n) := \max\bigl(\{r \in \mathbb{N} \mid\; r^2 \leq n\}\bigr).$$

In order to understand the implications of this definition we define for a given number $n \in \mathbb{N}$ the set

$$S(n) := \bigl\{r \in \mathbb{N} \mid\; r^2 \leq n\bigr\}.$$

Then $\texttt{isqrt}(n)$ is the maximum of the set $S(n)$. This implies that both

$$\texttt{isqrt}(n) \in S(n) \quad \text{and} \quad \bigl(\texttt{isqrt}(n) + 1\bigr) \notin S(n)$$

holds. Therefore we have

$$\bigl(\texttt{isqrt}(n)\bigr)^2 \leq n \quad \text{and} \quad n < \bigl(\texttt{isqrt}(n) + 1\bigr)^2$$

If $r \in \mathbb{N}$ satisfies both

$$r^2 \leq n \quad \text{and} \quad n < (r+1)^2,$$

then r is the maximum of the set $S(n)$ and hence $r = \texttt{isqrt}(n)$. Therefore the following relation holds for all $r \in \mathbb{N}$:

$$r^2 \leq n \wedge n < (r+1)^2 \;\Leftrightarrow\; r = \texttt{isqrt}(n).$$

In order to compute $\texttt{isqrt}(n)$ we need two propositions that can be used to reduce $\texttt{isqrt}(n)$ to $\texttt{isqrt}(n \mathbin{/\!/} 4)$, where the operator "$/\!/$" denotes integer division, i.e. we have

$$n = 4 \cdot n \mathbin{/\!/} 4 + n \,\%\, 4.$$

Proposition 1 For every $n \in \mathbb{N}$ we have

$$\texttt{isqrt}(n) \leq 2 \cdot \texttt{isqrt}(n \mathbin{/\!/} 4) + 1.$$

Proof: The inequation $n \mathbin{/\!/} 4 < \bigl(\texttt{isqrt}(n \mathbin{/\!/} 4) + 1\bigr)^2$ is equivalent to

$$n \mathbin{/\!/} 4 + 1 \leq \bigl(\texttt{isqrt}(n \mathbin{/\!/} 4) + 1\bigr)^2.$$

Multipling this inequation with 4 yields:

$$4 \cdot (n \mathbin{/\!/} 4) + 4 \leq \bigl(2 \cdot \texttt{isqrt}(n \mathbin{/\!/} 4) + 2\bigr)^2.$$

Since $n \,\%\, 4 < 4$ we have

$$n = 4 \cdot n \mathbin{/\!/} 4 + n \,\%\, 4 < 4 \cdot n \mathbin{/\!/} 4 + 4 \leq \bigl(2 \cdot \texttt{isqrt}(n \mathbin{/\!/} 4) + 2\bigr)^2,$$

i.e. we have $n < \bigl(2 \cdot \texttt{isqrt}(n \mathbin{/\!/} 4) + 2\bigr)^2$ and this implies

$$\texttt{isqrt}(n) < 2 \cdot \texttt{isqrt}(n \,//\, 4) + 2.$$

Therefore we have

$$\texttt{isqrt}(n) \leq 2 \cdot \texttt{isqrt}(n \,//\, 4) + 1. \qquad \Box$$

The next proposition provides a lower bound for $\texttt{isqrt}(n \,//\, 4)$.

Proposition 2 For all $n \in \mathbb{N}$ we have

$$2 \cdot \texttt{isqrt}(n \,//\, 4) \leq \texttt{isqrt}(n).$$

Proof: The definition of $\texttt{isqrt}(n \,//\, 4)$ implies that

$$\texttt{isqrt}(n \,//\, 4)^2 \leq n \,//\, 4$$

holds. Multiplying this inequation by 4 yields

$$4 \cdot \texttt{isqrt}(n \,//\, 4)^2 \leq 4 \cdot n \,//\, 4$$

Since we have $4 \cdot n \,//\, 4 \leq 4 \cdot n \,//\, 4 + n \,\%\, 4 = n$ this implies

$$4 \cdot \texttt{isqrt}(n \,//\, 4)^2 \leq n.$$

This can be written as

$$\bigl(2 \cdot \texttt{isqrt}(n \,//\, 4)\bigr)^2 \leq n.$$

The definition of $\texttt{isqrt}(n)$ therefore implies

$$2 \cdot \texttt{isqrt}(n \,//\, 4) \leq \texttt{isqrt}(n). \qquad \Box$$

Together, the previous propositions imply the following corollary:

Corollary 3 We have

$$\texttt{isqrt}(n) \in \bigl\{2 \cdot \texttt{isqrt}(n \,//\, 4),\; 2 \cdot \texttt{isqrt}(n \,//\, 4) + 1\bigr\}.$$

Furthermore, we have

$$\texttt{isqrt}(n) = \begin{cases} 2 \cdot \texttt{sqrt}(n \,//\, 4) + 1 & \text{if} \quad \bigl(2 \cdot \texttt{sqrt}(n \,//\, 4) + 1\bigr)^2 \leq n; \\ 2 \cdot \texttt{sqrt}(n \,//\, 4) & \text{otherwise.} \end{cases}$$

2 A Recursive Implementation

```
def rsqrt(n):
    if n == 0:
        return 0
    r = isqrt(n // 4)
    if (2 * r + 1) ** 2 <= n:
        return 2 * r + 1
    else:
        return 2 * r
```

Figure 1: A recursive implementation of `isqrt`.

Since we have

$$\texttt{isqrt}(0) = 0.$$

the previous Corollary shows that a recursive implementation of $\texttt{isqrt}(n)$ can be given as shown in Figure 1.

3 An Iterative Implementation

```
def list_of_digits(n):
    L = []
    while n > 0:
        L += [n % 4]
        n  = n // 4
    return L
```

Figure 2: Computing the base 4 digits of a number.

The recursive implementation of $\texttt{isqrt}(n)$ is based on the formula

$$\texttt{isqrt}(n) = \begin{cases} 2 \cdot \texttt{isqrt}(n \,/\!/\, 4) + 1 & \text{if } \big(2 \cdot \texttt{isqrt}(n \,/\!/\, 4) + 1\big)^2 \leq n; \\ 2 \cdot \texttt{isqrt}(n \,/\!/\, 4) & \text{otherwise.} \end{cases}$$

In each of these two cases, $\texttt{isqrt}(n)$ is computed in terms of $\texttt{isqrt}(n \,/\!/\, 4)$. The number $n \,/\!/\, 4$ results from the number n by cutting of the last two bits. If we want to transform our recursive implementation into an iterative implementation, then the iterative implementation needs to add two bits of n in every iteration. Therefore, we first implement the auxiliary function `list_of_digits` next. Given a natural number n, the $\texttt{list\_of\_digits}(n)$ returns the representation of n in base 4, i.e. it calculates a list $L = [d_0, d_1, \cdots, d_k]$ such that

$$n = \sum_{i=0}^{k} d_i \cdot 4^i \quad \text{where } 0 \leq d_i < 4.$$

The implementation of the function `list_of_digits` is shown in Figure 2.

```
def isr(n):
    L = list_of_digits(n)
    r = 0
    m = 0
    while len(L) > 0:
        m = 4 * m + L[-1]
        L = L[:-1]
        if (2 * r + 1) ** 2 <= m:
            r = 2 * r + 1
        else:
            r = 2 * r
    return r
```

Figure 3: An iterative implementation of `isqrt`.

Figure 3 shows an iterative implementation of the function `isr`. In order to understand this implementation we annotate the variables occurring in this function. Figure 4 shows the annotated version of the function `isr`.

```
def isr(n):
    L_0 = list_of_digits(n)
    r_0 = 0
    m_0 = 0
    while len(L_1) > 0:
        m_{i+1} = 4 * m_i + L_i[-1]
        L_{i+1} = L_i[:-1]
        if (2 * r_i + 1) ** 2 <= m_{i+1}:
            r_{i+1} = 2 * r_i + 1
        else:
            r_{i+1} = 2 * r_i
    return r_{k+1}
```

Figure 4: An annotated program to compute powers.

To understand this implementation of `isqrt`, we note its invariants. To this end assume that in base 4 the number n is given as

$$n = \sum_{j=0}^{k} d_j \cdot 4^j \quad \text{where } 0 \leq d_j < 4$$

Let us denote by m_i and r_i the values of the variable `m` and `r` at the beginning of the $(i+1)^{\text{th}}$ iteration of the `while`-loop. Then the following invariants hold:

(a) $L_i = [d_0, d_1, \cdots, d_{k-i}]$.

(b) $m_i = n \mathbin{//} 4^{k+1-i} = \sum\limits_{j=k+1-i}^{k} d_j \cdot 4^{j+i-(k+1)}$.

Proof: Since we have $L_0 = [d_0, d_1, \cdots, d_k] = [d_0, d_1, \cdots, d_{k-0}]$ and every iteration of the while loop chops of one element of L the first invariant should be obvious. We prove the second invariant by induction on i.

B.C.: $i = 0$.

$m_0 = 0$ and we also have

$$n \mathbin{//} 4^{k+1-0} = 0, \text{ since } n < 4^{k+1}.$$

I.S.: $i \mapsto i + 1$.

We have

$$\begin{aligned}
m_{i+1} &= 4 \cdot m_i + L_i[-1] \\
&\stackrel{\text{ih}}{=} 4 \cdot (n \mathbin{//} 4^{k+1-i}) + d_{k-i} \\
&= 4 \cdot \left(\sum_{j=k+1-i}^{k} d_j \cdot 4^{j+i-(k+1)} \right) + d_{k-i} \\
&= \left(\sum_{j=k+1-i}^{k} d_j \cdot 4^{j+(i+1)-(k+1)} \right) + d_{(k+1)-(i+1)} \cdot 4^{k+1-(i+1)+(i+1)-(k+1)} \\
&= \sum_{j=k+1-(i+1)}^{k} d_j \cdot 4^{j+(i+1)-(k+1)} \\
&= n \mathbin{//} 4^{k+1-(i+1)}
\end{aligned}$$

The invariants imply the following relation connecting m_i and m_{i+1}

$$m_{i+1} = 4 \cdot m_i + d_{k+1-(i+1)}.$$

Therefore, we have that

$$m_i = m_{i+1} \mathbin{//} 4.$$

Now we are ready to state and prove the crucial invariant:

$$r_i = \texttt{isqrt}(m_i).$$

We prove this invariant by induction on i.

B.C.: $i = 0$. We have

$$\texttt{isqrt}(m_0) = \texttt{isqrt}(0) = 0 = r_0.$$

I.S.: $i \mapsto i + 1$. Since we have

$$m_i = m_{i+1} \mathbin{//} 4 \quad \text{and} \quad r_i = \texttt{isqrt}(m_i)$$

we know that

$$\texttt{isqrt}(m_{i+1}) = 2 \cdot r_i \quad \text{or} \quad \texttt{isqrt}(m_{i+1}) = 2 \cdot r_i + 1.$$

We perform a case distinction analogous to the case distinction that is present in the program.

(a) $(2 \cdot r_i + 1)^2 \leq m_{i+1}$.
Then we have $\texttt{isqrt}(m_{i+1}) = 2 \cdot r_i + 1$. In this case we have

$$r_{i+1} = 2 \cdot r_i + 1 = \texttt{isqrt}(m_{i+1}).$$

(b) $m_{i+1} < (2 \cdot r_i + 1)^2$.
Then we have $\texttt{isqrt}(m_{i+1}) = 2 \cdot r_i$. In this case we have

$$r_{i+1} = 2 \cdot r_i = \texttt{isqrt}(m_{i+1}).$$

Therefore, we have $\texttt{isqrt}(m_{i+1}) = r_{i+1}$ in ever case.

To finish the proof we note that the loop ends after $k + 1$ iterations. Therefore the beginning of the $(k + 2)^{\text{th}}$ iteration the final value of `r` is

$$\texttt{r} = r_{k+1} = \texttt{isqrt}(m_{k+1}) = \texttt{isqrt}(n \mathbin{//} 4^{k+1-(k+1)}) = \texttt{isqrt}(n).$$

Since the `while`-loop obviously terminates and the function $\texttt{isr}(n)$ returns `r` we have shown that

$$\texttt{isr}(n) = \texttt{isqrt}(n). \qquad \square$$