Indian Institute of Technology Delhi

COL 226 - Programming Languages

Assignment 1

Integer Square Root by Long Division

Aryan Sharma
2021CS10553

Contents

1 Introduction

The objective of this assignment is to find the integer square root along with the remainder by long division method for large integers(of course, for small too).

2 Algorithm

2.1 Psuedocode

The function isqrtld takes as input a list of integers representing a number, whose integer square root is to be calculated.

Algorithm 2.1

$$\textsc{isqrtld}\ (num) \stackrel{df}{=}$$

$$\textbf{let rec fun}\ \textsc{isqrtld\_helper}\ (num, rem, isqrt) \stackrel{df}{=}$$

$$\begin{cases} num.is\_empty() & \rightarrow \begin{cases} rem.is\_empty() & \rightarrow \quad (isqrt, [0]) \\ \textbf{else} & \rightarrow \quad (isqrt, rem) \end{cases} \\ \textbf{else} & \rightarrow \begin{cases} \textbf{let} \begin{cases} ans := \textsc{mult}\ (isqrt, 2) \\ rem := rem@[hd(num), hd(tl(num))] \\ prod := \textsc{find}\ (ans, rem) \\ rem := \textsc{sub}\ (rem, \textsc{mult}\ (ans@prod, prod)) \\ isqrt := prod :: isqrt \end{cases} \\ \textbf{in} \\ \textsc{isqrtld\_helper}\ (tl(tl(num)), rem, ans\_rt) \end{cases} \end{cases}$$

$$\textbf{in} \begin{cases} length(num)\ is\ even & \rightarrow \quad \textsc{isqrtld\_helper}\ (num, [], []) \\ \textbf{else} & \rightarrow \begin{cases} \textbf{let} \begin{cases} x := \textsc{find}\ ([], [hd(num)]) \\ rem := \textsc{sub}\ ([hd(num)], \textsc{mult}\ ([x], x)) \end{cases} \\ \textbf{in} \\ \textsc{isqrt\_helper}\ (tl(num), rem, [x]) \end{cases} \end{cases}$$

[1]

[1] ADD(*num1* : int list, *num2* : int)* , MULT(*num1* : int list, *num2* : int)* , SUB(*num1* : int list, *num2* : int list)* , FIND(*num* : int list, *rem* : int list)* := this returns an largest integer less than 10 such that when added to the end of num and multiplied by the returned integer to be less than rem.

2.2 Proof Of Correctness

Claim : Algorithm 2.1 gives correct answer for integer square root and remainder of a number with digits $\{a_1, a_2, ... \ a_{n-1}, a_n\}$, $\forall \ a_i \in \{0, 1, 2...9\}$,$\forall \ i \in [n]$,$\forall \ n \in N$, $a_1 \neq 0$

Proof :
In each recursive call of the function ISQRTLD_HELPER the function consumes two leading digits of num, the square root of the digits consumed so far is stored in the variable *isqrt* and remainder in *rem*.
We Prove the correctness of Algorithm 2.1 in two cases,

Case 1 : if n is **odd**.

CLAIM : On calling *ISQRTLD*, after the end of k calls of *ISQRTLD_HELPER*, isqrt and rem store the square root and remainder of 2k + 1 most significant digits. Exactly (n-1)/2 + 1 calls are made for *ISQRTLD_HELPER*.

Proof Of above Claim :

The above Claim has two parts,
1.1 After the end of k calls of ISQRTLD_HELPER, isqrt and rem store the square root and remainder of 2k + 1 most significant digits.
1.2 Exactly (n+1)/2 calls are made for *ISQRTLD_HELPER*.

Proof of **(1.1)** :

Proof By Induction on k.

Inductive Hypothesis(P(k)) : After the end of k calls, isqrt and rem store the square root and remainder of 2k + 1 most significant digits.

Base Case(k = 0) : Before the first call the square root of the most significant digit(2k + 1 = 1) of the number is calculated in the function *ISQRTLD*, thus P(0) is true.

Inductive Step :
Let, P(k-1) be true, then, isqrt is the integer square root of $a_1a_2...a_{2k-1}(= \text{num})$ and rem is equal to num - $isqrt^2$. Thus,

$$isqrt^2 \leq num \leq (1 + isqrt)^2$$
$$isqrt^2 + rem = num$$

Let the next two digits consumed be a and b, in that order, thus the number consumed so far is $100num + 10a + b$ and the new value of rem becomes $100rem + 10a + b$. In the algorithm, a single integer prod is found such that $(20isqrt + prod) * prod \leq 100rem + 10a + b$. The value of isqrt at the end of this call would be $(10isqrt + prod)$. Now we show that this new value of isqrt is integer square root of new value of num(i.e. $100num + 10a + b$)

Thus, from above,

$$(20isqrt + prod) * prod \leq 100rem + 10a + b = 100(num - isqrt^2) + 10a + b$$
$$(10isqrt + prod)^2 \leq 100num + 10a + b$$

The value of rem at the end of this call is $100rem + 10a + b - (10isqrt + prod) * prod$, which is equal to the (new value of num) - (new value of isqrt).

$$100(num - isqrt^2) + 10a + b - 20isqrt * prod - prod^2$$
$$= 100num + 10a + b - (10isqrt + d)^2$$
$$= num - isqrt^2$$

We also show here that $(10isqrt + prod + 1)^2 \geq 100num + 10a + b$. It is sufficient to show that new value of rem is less than $2 * (10isqrt + prod) + 1$. This is ensured as prod os the maximal such integer that satisfies, $(20isqrt + prod) * prod \leq 100rem + 10a + b$.
Thus,

$$new\_rem + (20isqrt + prod) * prod < (20isqrt + prod + 1) * (prod + 1)$$
$$new\_rem < 2 * (10isqrt + prod) + 1$$

Hence, We have shown that if P(k-1) is true then P(k) is true $\forall k \geq 1$ that can be acheived.
Hence, Proved by Induction.

Proof of **(1.2)** :

This follows quite trivially, in each call 2 digits of num are consumed starting with 1 digit already consumed before the first call. When num becomes null, *ISQRTLD_HELPER* returns the tuple (isqrt, rem). Thus after k calls, 2k + 1 digits have been consumed, thus after (n-1)/2 calls, num becomes empty and one final call is made which returns the value of (isqrt, rem) at the end of (n-1)/2 calls.

This completes the Proof for Case 1.

Case 2 : if n is **even**.

CLAIM : On calling ISQRTLD, after the end of k calls of ISQRTLD_HELPER, isqrt and rem store the square root and remainder of 2k most significant digits. Exactly n/2 + 1 calls are made for ISQRTLD_HELPER.

Proof Of above Claim :

The above Claim has two parts,
2.1 After the end of k calls of ISQRTLD_HELPER, isqrt and rem store the square root and remainder of 2k most significant digits.
2.2 Exactly n/2 + 1 calls are made for *ISQRTLD_HELPER*.

Proof of **(2.1)** :

Proof By Induction on k.

Inductive Hypothesis(P(k)) : After the end of k calls, isqrt and rem store the square root and remainder of 2k most significant digits.

Base Case(k = 0) : Before the first call the list isqrt is empty which is interpreted as 0 in the function *ISQRTLD_HELPER*, thus P(0) is true.

Inductive Step :

Let, P(k-1) be true, then, isqrt is the integer square root of $a_1a_2...a_{2k}(= \text{num})$ and rem is equal to num - $isqrt^2$. Thus,

$$isqrt^2 \leq num \leq (1 + isqrt)^2$$
$$isqrt^2 + rem = num$$

Let the next two digits consumed be a and b, in that order, thus the number consumed so far is $100num + 10a + b$ and the new value of rem becomes $100rem + 10a + b$. In the algorithm, a single integer prod is found such that $(20isqrt + prod) * prod \leq 100rem + 10a + b$. The value of isqrt at the end of this call would be $(10isqrt + prod)$. Now we show that this new value of isqrt is integer square root of new value of num(i.e. $100num + 10a + b$)

Thus, from above,

$$(20isqrt + prod) * prod \leq 100rem + 10a + b = 100(num - isqrt^2) + 10a + b$$
$$(10isqrt + prod)^2 \leq 100num + 10a + b$$

The value of rem at the end of this call is $100rem + 10a + b - (10isqrt + prod) * prod$, which is equal to the (new value of num) - (new value of isqrt).

$$100(num - isqrt^2) + 10a + b - 20isqrt * prod - prod^2$$
$$= 100num + 10a + b - (10isqrt + d)^2$$
$$= num - isqrt^2$$

We also show here that $(10isqrt + prod + 1)^2 \geq 100num + 10a + b$. It is sufficient to show that new value of rem is less than $2 * (10isqrt + prod) + 1$. This is ensured as prod os the maximal such integer that satisfies, $(20isqrt + prod) * prod \leq 100rem + 10a + b$.
Thus,

$$new\_rem + (20isqrt + prod) * prod < (20isqrt + prod + 1) * (prod + 1)$$
$$new\_rem < 2 * (10isqrt + prod) + 1$$

Hence, We have shown that if P(k-1) is true then P(k) is true $\forall k \geq 1$ that can be acheived.
Hence, Proved by Induction.

Proof of **(2.2)** :

This follows quite trivially, in each call 2 digits of num are consumed starting with 1 digit already consumed before the first call. When num becomes null, *ISQRTLD_HELPER* returns the tuple (isqrt, rem). Thus after k calls, 2k digits have been consumed, thus after n/2 calls, num becomes empty and one final call is made which returns the value of (isqrt, rem) at the end of n/2 calls.

This completes the Proof for Case 2.

Hence, we have shown that Algorithm 2.1 consumes all the digits of the input number and computes the integers square root and remainder correctly.

3 Design Choices

The following Design Choices have been made in different functions :
1. Empty list everywhere has been interpreted as the number 0.
2. remove_zeros function has been implemented to remove the leading zeros in the list representation format for numbers.
3. Multiplication, Addition and Subtraction are performed by reversing the list representation of the number for ease in recursive function.

4 Conclusion

In this Assignment we have implemented a program in StandardML using the long division algorithm to find integer square root of an integer along with the remainder, we provided with a pseudocode and proof of correctness of the long division algorithm.