Indian Institute of Technology Delhi

COL 226 - Programming Languages

Assignment 1

Integer Square Root using Long Division

Garv Nagori
2021CS10549

Contents

1 Introduction

The objective of this assignment was to implement a square root function for arbitrarily large number using the long division method in *StandardML*. The algorithm is specified in the *'Pseudocode'* section and the proof of correctness of the algorithm is also specified.

2 Pseudocode

Algorithm 2.1

$\textsc{integer square root}\ (Num) \stackrel{df}{=}$

$$\textbf{let rec fun } \textsc{checkMultiply}\ (N, d, q, k) \stackrel{df}{=}$$

$$\begin{cases} k < 0 & \rightarrow \perp \\ k = 10 & \rightarrow (\textsc{Subtract}\ (N, \textsc{Multiply}\ (d :: 9, 9)), \textsc{Add}\ (d :: 9, 9), d :: 9) \\ \textbf{else} & \rightarrow \begin{cases} \textbf{let} \\ sub := \textsc{Subtract}\ (N, \textsc{Multiply}\ (d :: k, k)) \\ olds := \textsc{Subtract}\ (N, \textsc{Multiply}\ (d :: (k-1), (k-1))) \\ \textbf{in} \\ sub = [] \quad \rightarrow (sub, \textsc{Add}\ (d :: k, k)), 0 \\ sub = [-1] \quad \rightarrow olds, \textsc{Add}\ (d :: (k-1), k-1), d :: (k-1) \\ \textbf{else} \quad \rightarrow \textsc{checkMultiply}\ (N, d, q, k+1) \end{cases} \\ \textbf{where} & \\ \multicolumn{2}{l}{\textsc{Multiply}\ \text{(m, n) multiplies a number 'm' with a single digit 'n'}} \\ \multicolumn{2}{l}{\textsc{Subtract}\ \text{(m, n) subtracts two numbers (m-n) and returns -1 if m is smaller}} \\ \multicolumn{2}{l}{\textsc{Add}\ \text{(m, n) adds two numbers (m+n).}} \end{cases}$$

$$\textbf{let rec fun } \textsc{isqrtld\_rec}\ (n, D, d, q) \stackrel{df}{=}$$

$$\begin{cases} \textbf{let} \\ newN := D :: \textsc{first\_two\_digits}\ (n) \\ (new\_D, new\_d, new\_q) := \textsc{checkMultiply}\ (newN, d, q, 1) \\ \textbf{in} \\ \textsc{after\_two\_digits}\ (n) = [] \quad \rightarrow (new\_q, new\_D) \\ \textbf{else} \quad \rightarrow \textsc{isqrtld}\ (\textsc{after\_two\_digits}\ (n), new\_D, new\_d, new\_q) \end{cases}$$

$$\textbf{let fun } \textsc{isqrtld}\ (Num) \stackrel{df}{=}$$

$$\begin{cases} \textbf{let} \\ \textsc{Length}\ (Num) \text{ is odd} \quad \rightarrow n := 0 :: Num \\ \textbf{else} \quad \rightarrow n := Num \\ \textbf{in} \\ \textsc{isqrtld\_rec}\ (n, [], [], []) \end{cases}$$

3 Proof of Correctness

The function ***isqrtld*** is the main function to which a number ***Num*** is passed. It checks if the number of digits is odd, in which case it appends a zero at the start to make it even. It then calls the function *isqrtld_rec* with the parameters *D, d and q* as empty because it is the first recursion step.
The function ***isqrtld_rec*** is a recursive function which calls the function *checkMultiply*. ***checkMultiply*** returns the next digit in the quotient. It then outputs the ***new Dividend, Divisor and Quotient*** by appending k to the *quotient* and *divisor*. The *New dividend* is calculated by multiplying both of them and subtracting it from the original *Dividend.*

isqrtld_rec first removes a pair of digits from n and appending it to $\boldsymbol{D}$, which is the Dividend. It calls *checkMultiply* to calculate the ***new Dividend, Divisor and Quotient***. It then checks if after removing the two digits from *num*, if it is empty. In that case it returns the *quotient* obtained from
textitcheckMultiply and the *new Dividend* as the ***remainder***. Otherwise, it recursively calls itself removing the two digits at the start of *num*, and with the *new Dividend, Divisor and Quotient.*

3.1 *checkMultiply*

This function finds the largest digit k such that $N - Multiply(d :: k, k) > 0$.

1. $k < 0$: This case is redundant since a single digit lies between 0 and 9. Here, the function terminates.

2. k = *10*: In this case, we append 9 to the divisor, to get and multiply it with 9 and subtract from the dividend, to get the new Dividend. We also append 9 to the quotient and add 9 to the divisor (after appending 9) to get new Quotient and new Divisor. Here also, the function terminates

3. $0 < k < 10$: We try to calculate the new Dividend as in the previous case (by appending k). If it is negative, we conclude our search and the new digit is *k-1*. We return the new Divided, Quotient and Divisor. If the new Dividend is 0, the new digit is k and we return the same. If the new Dividend is positive, we try the same with *k+1* by recursively calling itself.
   In the case of 0 or negative, the function terminates. If it is positive we call it with *k+1* as parameter. Since $0 < k < 10$, $1 < k + 1 < 11$ and the process repeats. Eventually *k+1* either terminates by new Dividend becoming 0 or negative or it reaches 10 at that point it terminates due to the second condition.

   Hence, the function *checkMultiply* is algorithmically correct.

3.2 *isqrtld_rec*

This function recursively calculates the square root of a number n. The length of n is always even since the non-recursive function *isqrtld* handles that by appending '0' at the start if it is odd. Let the length of n be given by *2l*. We will prove that this function is correct by induction on l and also give the proof of termination for an a number of arbitrary size.

Base Case: If l is 1. Since D was empty it appends the two digits to it. d and q are also empty, so *checkMultiply* finds k such that $k^2 \leq d < (k+1)^2$. It returns new Quotient as k and the new Dividend as $d - k^2$. *After_two_digits(n)* returns empty so the function terminates and returns the quotient k and remainder $d - k^2$. The new divisor is $2k$ which is $2q$.

Induction Hypothesis: Let the function return the square root and remainder of a number correctly for $l = L$ and the function terminates. That is, it returns q and r such that $q^2 \leq n < (q+1)^2$ and $r = n - q^2$.
The new divisor d is such that $d + k = 2q$.

Induction Step: We have to prove that the function returns the correct quotient and remainder for a number with $l = L + 1$. Since, the function considers two digits at a time, it computes the first $2L$ digits correctly, by our induction hypothesis. So, now we have new Dividend as r, quotient as q and divisor as d. Let the last two digits be represented by a single integer p. Hence, the whole number becomes $100n + p$, and we have calculated q and r for n. We find k' such using *checkMultiply*.

We first prove the claim that $d + k = 2q$ at any iteration. We assumed it to be true for $l = L$ and now prove it for $l = L + 1$.
The new divisor is $d' = 10(d + k) + k' = 10(2q) + k' = 20q + k'$
The new quotient is $q' = 10q + k'$.
Therefore $d' + k' = 20q + k' + k' = 2(10q + k') = 2q'$. Hence, proved.

Now, we prove the quotient is actually the square root of $100n + p$. The new divisor is $20q + k'$ and we multiply it with k' and subtract it from $100r + p$. The new quotient is $10q + k'$. We prove both sides of inequality one-by-one.-
$(10q + k')^2 = 100q^2 + 20qk' + k'^2$
$\leq 100n + 20qk' + k'^2$ (By our induction hypothesis)
$\leq 100n + 100r + p$ (Since divisor $\times$ quotient $\leq 100r + p$)
$\leq 100n + p$

$(10q + k' + 1)^2 = 100q^2 + k'^2 + 1 + 20qk' + 20q + 2k'$
$= 100q^2 + (20q + k' + 1)(k' + 1)$ (The right expression is if $k + 1$ was chosen)
$> 100q^2 + 100r + p$ (Since k is the largest such digit)
$= 100n + p$ (Since $q^2 + r = n$)

We also have to prove $1 \leq k' \leq 9$ since *checkMultiply* only uses a digit between 0 and 9. This can be proven by *contradiction*. Assume $k = 10$ then $(10q + 10)^2 = 100n + p$
$=> 100q^2 + 200q + 100 = 100q^2 + 100r + p$ (Since $q^2 + r = n$) $=> p = 100(2q - r + 1)$ This shows that p is a multiple of 100 since q and r are integers, but this contradicts our assumption that $0 \leq p \leq 99$. Hence, proved
Since both sides are proved, it is proved that $10q + k'$ is the integer square root of $100n + p$.

Now, to prove the remainder,
Remainder is given by $100n + p - (10q + k')^2$
The new Dividend which is calculated at the $L + 1^{th}$ iteration is
$100r + p - (20q + k')k' = 100n - 100q^2 + p - 20qk' - k'^2$ (Since $r = n - q^2$)
$= 100n + p - (10q + k')^2$, which is equal to the remainder.
Since after processing p, the *num* becomes empty the new Dividend is the remainder, which is what we expected. Hence this algorithm also gives the remainder correctly

After processing the next two digits i.e., p the function *After_two_digits (n)* returns an empty list, at that stage it returns the quotient and the remainder. Thus we have shown that if the algorithm terminates for $l = L$, it also terminates for $l = L + 1$.

Invoking the ***principle of mathematical induction***, since we have shown that if the algorithm returns square root and remainder for *l=L* correctly, it also does for *l=L+1*, and since the input will always be of finite length, this algorithm works for *a number with any arbitrary finite length.*

4 Design Choices

1. Except for the list which stores the actual number, all other numbers have been stored in reverse in a list. Eg 124 will be stored as [4,2,1]. This makes the multiplication by a digit, addition and subtraction of two listss functions easier as we can just use *tl* function and recursion to keep track of carry-overs.

2. All the functions multiply, add and subtract remove the trailing zeroes if any so the return value is never [0,0] etc but instead [], an empty list.

3. The subtract function additionally returns only the list [-1] (represented as [$\tilde{1}$]) the subtraction results in a negative answer. This is done because it will be easier to check if the subtraction is negative by just looking at the head of the list.