Integer square-root by long division

Aman Kumar, 2019CS10324

January 22, 2023

Integer square-root by long division

Algorithm The recursive pseudocode (SML like) is as follows:

```
isqrtld(S)
    // Input: a string of digits representing the number in decimal, S
    // Output: a pair of string of digits, where the first string is
    // the square root of the given number an in integer form and
    // the second string is the remainder r in digit form

    val s_list = divide_in_pairs(S)
    val result = helper(s_list)
    // return the first and last elements of result
    return (hd result, last result)
```

```
divide_in_pairs(S)
    // Input: a string of digits representing the number in decimal, S
    // Output: List of digits of the number into pairs of segments
    // starting with the digit in the units place (in reversed order)
    // For example: "12356" => ["56", "23", "01"]

    if size of string is odd:
        S := "0" ^ S
    val n = length S
    result = []
    fun loop (i > 0) =
        if(i < n-1) then
            return loop(i+2)@[substr(S, i, 2)]
        else
            return []

    return loop(0)
```

```
helper(s_list)
    // Explained below

    if size of s_list is 0 then
        return [0,0]
    else
        val curr_val = hd s_list

        val last_call = helper(tl x_list)
        val last_quotient = hd last_call
        val last_remainder = hd (tl last_call)

        val curr_dividend = last_remainder * 100 + value
        val curr_divisor = last_quotient * 20
        val new_digit = nextDigit [curr_dividend, curr_divisor]
        val curr_divisor = curr_divisor + new_digit
        val curr_remainder = curr_dividend - curr_divisor*new_digit
        val curr_quotient = last_quotient * 10 + new_digit
        return [curr_quotient, curr_remainder]
```

```
nextDigit(dividend, divisor)

    fun loop(i) =
        if i = 10 then
            return 9
        else if i*(divisor+i) <= dividend then
            return loop(i+1)
        else
            return i-1

    return loop(0)
```

Explanation

1. Start with the unit digit and divide the number into pairs.
2. From the first pair, the divisor and quotient are the greatest number whose square s less than equal to the first segment.
3. To calculate the next dividend, subtract the square of the divisor from the first segment and lower the next segment to the right of the remainder.
4. Now, take two times the previous quotient and concatenate it with an appropriate digit that is also the next digit of the quotient, chosen so that the product of the new divisor and this digit is equal to or just less than the new dividend.

5. Repeat until every segment is covered up.

By setting the initial values of the divisor, quotient, and remainder to 0, we can combine steps (2) and (4).

Example (Wikipedia)

```
Find the square root of 152.2756.

          1  2 3  4
        /------------
      \/  01 52 27 56

          01                    1*1 <= 1 < 2*2                x = 1
          01                      y = x*x = 1*1 = 1
          00 52                 22*2 <= 52 < 23*3             x = 2
          00 44                   y = (20+x)*x = 22*2 = 44
             08 27              243*3 <= 827 < 244*4          x = 3
             07 29               y = (240+x)*x = 243*3 = 729
                98 56           2464*4 <= 9856 < 2465*5       x = 4
                98 56             y = (2460+x)*x = 2464*4 = 9856
                00 00           Algorithm terminates: Answer is 1234
```

Proof of correctness : Proof by construction

Given: A number N a whose digits are $a_{n1}, ..., a_0$ in decimal for $n > 0$
Let guessed result from the algorithm be M.
Identity: $(10x + y)^2 = 100x^2 + 20xy + y^2$

Observation 1: Adding another digit $(10x + y)$ in x add two more digits in its square.
Let

$$
\begin{aligned}
N &= (10^m * b_0 + 10^{m-1} * b_1 + \cdots 10^1 * b_{m-1} + b_m)^2 \\
&= (10^m b_0)^2 + 2 * 10^m b_0 10^{m-1} b_1 + (10^{m-1} b_1)^2 + 2(10^m b_0 + 10^{m-1} b_1) 10^{m-2} b_2 + (10^{m-2} b_2)^2 \\
&\quad + \cdots + (10 b_{n-1})^2 + 2 \left(\sum_{i=1}^{n-1} 10^{m-i} b_i \right) b_n + b_n^2 \\
&= (10^m b_0)^2 + [2 * 10^m b_0 + 10^{m-1} b_1] 10^{m-1} b_1 + \cdots + \left[2 \left(\sum_{i=1}^{n-1} 10^{m-i} b_i \right) + b_n \right] b_n.
\end{aligned}
$$

Here $b_i (\forall\, 0 \le i < m)$ is an integer in range $[0, 9]$.

Now, this algorithm finds the value of each b_i in order.
Suppose we already found the values of $b_0, b_1, b_2, \cdots, b_k$. and let $\hat{N} = b_0 b_1 \cdots bk$. Then $\hat{N}^2 \le$ (digits we have processed. Say N_2). If we add another digit b_{k+1}, then we will multiply $\hat{N}$ by 10, and we will add an extra term, say, X.

$$
X = \left[2 \left(\sum_{i=1}^{k} 10^{m-i} b_i \right) + 10^{m-(k+1)} b_{k+1} \right] 10^{m-(k+1)} b_{k+1} = \left[20 \left(\sum_{i=1}^{n-1} 10^{k-i} b_i \right) + b_{k+1} \right] 10^{2(m-(k+1))} b_{k+1}
$$

This can be simplified as (20*quotient + new digit)* new digit

Let R_i represent how far we are from the actual digits after i operations $i.e. N_2 - \hat{N}$. After adding one digit, we know that two digits will be added to its square (observation 1). R_i = remainder from the last iteration + next two digits.

Now, we need to guess the maximum possible value of b_{k+1} such that $X \leq R_{k+1}$ else, the remainder will always be negative. Proved.

Acknowledgements

- Wikipedia: To understand the working of the algorithm.