1 The problem

This is a problem 80 from Project Euler.

2 What am I going to do

We are going to implement the square root calculation.

I'm not sure how to do it... I found a module somewhere:

```
import Fraction (Fraction ((: − :)), decimal′, sqrt′)
import Data.Char (digitToInt)


digitCnt :: Integer
digitCnt = 200


sqrtDigSum :: (Integral a) ⇒ a → Int
sqrtDigSum n = sum $ map (digitToInt) $ take 100 $ filter (≢ '.') $ decSqrt n


decSqrt :: (Integral a) ⇒ a → [Char]
decSqrt n = decimal′ digitCnt $ sqrt′ (1 : − : (10 ↑ 200)) ((fromIntegral n) : − : 1)


sols :: (Integral a) ⇒ a → [Int]
sols prior_to = map sqrtDigSum [1 . . prior_to]


solve :: Int → [(Int, Int)]
solve prior_to = filter (λ(n, sq) → sq ↑ 2 ≢ n) $ zip [1 . . prior_to] $ sols prior_to
```