Problem Set 7 - Math

Question B

Question B: Math, took a pretty big hint from one of my friends to get. The initial approach that I was exploring was something completely wrong… It's a little bit embarrassing - but I hadn't touched number theory since 3u and 4u in highschool so fairs.

I initially thought of reversing the operations, and testing each number from 1 up until the actual number, and whether we could reach the original number with squaring and dividing. That way, the only way that you can go up is by squaring, and the only way you can go down is by dividing. With factorisations, we can find out whether a number can reach another by division by an integer (this is before I knew the Sieve only found prime factors). So for each number 1 to n, I would square, and then check the factors. If the original number was a factor, then win. Problem is I didn't know where to stop squaring etc.

Ultimately, I realised that for something to be square rooted into an integer, all of the multiplicities of its primes factors would need to be even, as rooting is just raising a number to the power of (½). Thus, a way we could go about this question is multiplying by prime factors until all powers are even, and then square rooting, checking if powers are even, multiplying, rooting etc. However, this wouldn't get us the least amount of moves down to the base number.

First, note that the lowest number possible is just the product of all the prime factors - this makes MORE sense now since I have a (slightly) better understanding of number theory. And also, since we can multiply by any number, we can get rid of all of the excess checkings and multiplications in the first step. The way to do so is raising all prime factors to the same power. And that power is the next power of 2 up from the highest multiplicity prime. This can just be calculated by getting the ceiling of log2 of the highest prime.

Implementation wise, something funky I did was change the factorisation function to return a map rather than a vector, which just helped to speed up operations, get multiplicities quickly.