1 The problem

This is a problem 87 from Project Euler.

2 Definitions

```
import Control.Monad (guard)
import Math.Sieve.Factor
import qualified Data.Set as Set
```

```
max_n = 50000000
```

Let's generate a global sieve:

```
si = sieve max_n
```

And use it to get some primes:

```
primes = filter (isPrime si) [1 .. max_n `div` 10]
```

3 Straight solution

We can just enumerate all triples (p_1, p_2, p_3):

```
enumerate_triples :: [(Int, Int, Int)]
enumerate_triples = do p1 ← takeWhile (λp1 → (p1 ↑ 4 < max_n)) primes
  p2 ← takeWhile (λp2 → (p1 ↑ 4 + p2 ↑ 3 < max_n)) primes
  p3 ← takeWhile (λp3 → (p1 ↑ 4 + p2 ↑ 3 + p3 ↑ 2 < max_n)) primes
  return (p1, p2, p3)
```

```
convert_triple :: (Int, Int, Int) → Int
convert_triple (p1, p2, p3) = p1 ↑ 4 + p2 ↑ 3 + p3 ↑ 2
```

```
main :: IO ()
main = do print $ length primes
  print $ Set.size $ Set.fromList $ map convert_triple enumerate_triples
```

I'm using a *Set* because *nub* is too slow. 30 seconds - not to be ashamed.