Sieve of Eratosthenes (Scala)

From LiteratePrograms

Other implementations: Bash | C++ | Forth | Haskell | Python | Python, arrays | **Scala**

The Sieve of Eratosthenes (http://en.wikipedia.org/wiki/Sieve_of_Eratosthenes) is an algorithm for rapidly locating all the prime numbers in a certain range of integers. It operates by marking as composite all nontrivial multiples of each prime in sequence until only primes remain unmarked. It is most well-suited to implementation using arrays, which are compact and feature random access.

Scala uses an eager evaluation strategy by default. Lazy evaluation, which is the default in languages such as Haskell and Clean, can sometimes be used to express algorithms in a more natural manner; such is the case here.

To cater to this kind of algorithms, we need to use the built-in *Stream* class. A Stream object of type A consists of a head of type A, and a tail which is a *suspension* : its value is not computed until the user forces it by using it. It then evaluates to a new Stream of type A.

To begin with, we need a Stream of all integers >= 2. The following function computes that.

```
<<ints>>=
  def ints(n: Int): Stream[Int] =
    Stream.cons(n, ints(n+1))
```

We can now write the *primes* function. It takes a stream of number as input. The first number in that stream is a prime number (trivial in the base case, since 2 is a prime. It will be seen shortly that this requirement always holds)

The output of *primes* is the stream of prime numbers. Given that we now have one prime number, i.e. the first number in the stream, we now have to remove all its multiples from the rest of the stream. This filtered stream can now be passed to primes, which will continue the process *ad infinitum*.

```
<<primes>>=
  def primes(nums: Stream[Int]): Stream[Int] =
    Stream.cons(nums.head,
                primes ((nums tail) filter (x => x % nums.head != 0)) )
```

Now we're ready to write the driver function for this. Note how *primes* is called: it's passed a stream of integers [2..], and from the resulting stream we want to extract the first *n* terms. This is then converted to a list and printed to the console.

```
<<Sieve.scala>>=
object Sieve {

ints

primes

  def main(args: Array[String]): Unit = {
    val n = Integer.parseInt(args(0))
```

```
    System.out.println(primes(ints(2)) take n toList)
  }
}
```

Download code (http://en.literateprograms.org/index.php?title=Special:Downloadcode/Sieve_of_Eratosthenes_(Scala)&oldid=8087)

Categories: Sieve of Eratosthenes | Programming language:Scala | Environment:Portable

---