CSL100: Introduction to Computers and Computer Science

I semester 2013-14

Instructors: Subhashis Banerjee and S. Arun-Kumar

Euclidean composites refer to composite numbers derived from Euclid's proof of the infinitude of primes. Go back to your elementary number theory and look up this proof.

It is a proof by contradiction which starts with the assumption that there are only a finite number $n > 0$ of primes, say $\{p_1, p_2, \ldots, p_n\}$. Then a number q is constructed which is 1 more than the product of $p_1, \ldots p_n$. The proof then shows that since $q = p_1 \times \cdots \times p_n + 1$, q leaves a remainder of 1 when divided by any of the primes $p_1, \ldots p_n$. Hence q must be a prime which shows that there are at least $n + 1$ primes, contradicting the assumption that there are only n primes. Hence the set of primes is infinite.

Let $q_n = 1 + p_1 \times \cdots \times p_n$. A little examination shows that since $p_1 = 2$ is even, each q_n is odd. Further the first few of these viz. $q_1 = 3$, $q_2 = 7$, $q_3 = 31$, are all odd primes. We call q_n a **euclidean composite** if q_n is not a prime.

Are there any euclidean composites? If so are there only a finite number of them or are they also infinite? These are questions that naturally arise.

The assignment

1. Develop an algorithm by *successive refinement* to compute all the euclidean composites less than the value of the largest integer in the SML implementation.
2. Develop the proof of correctness of your algorithm alongside the development of your algorithm
3. Compute the running time and the space complexity of your algorithm in terms of the number of invocations of the functions that are defined in your algorithm.
4. Transform as many of your recursive functions to tail-recursive functions and remove duplications of computations to make your algorithm as space and time efficient as possible.
5. Translate your algorithm into SML and execute it to find all the euclidean composites between 2 and the largest integer allowed in the SML implementation (given by $valOf(Int.maxInt)$)