Lecture 6

Lecturer: David Avis *Scribe: Kazuhisa Seto*

Is the sieve of Eratosthenes efficient?

Here is a discussion of this question, given as an exercise on April 14. The sieve method is as follows :

Algorithm 1

Input : n (positive integer)
Output : all primes up to n

1: Make a list $1, 2, \ldots, n$
2: Delete 1
3: Set a be the next number prime
4: Delete all multiples of a
5: Continue $3 - 4$ until $a = \sqrt{n}$

We focus on two problems to discuss the efficiency of Algorithm 1.

Problem 1. Is n a prime number ? Yes or No ?

Problem 2. Generate a list of all primes less than n

Now, we will estimate the running time of the sieve. Consider a simple implementation when $n = 400$. We proceed as follows:

1. Make a list $1, 2, \cdots, n$ time $\cdots$ n
2. Delete multiples of 2 time $\cdots$ $\frac{n}{2}$
3. Delete multiples of 3 time $\cdots$ $\frac{n}{3}$
4. Delete multiples of 5 time $\cdots$ $\frac{n}{5}$

   $\cdots\cdots$
9. Delete multiples of 19 time $\cdots$ $\frac{n}{19}$

In this case, the number of iterations for loop is 8. In general, this will be the number of primes $\leq \sqrt{n}$.

In general

$f(n) =$ the number of primes $\leq n$

total number of steps = number of steps per iteration $*$ number of iterations $\sim nf(\sqrt{n})$

The number of primes is estimated by the following famous result, known as the prime number theorem. For more information, please consult the corresponding wiki page.

$$f(n) \sim \frac{n}{\log n}$$

(Note: Throughout this note we take all logarithms to base 10. Logarithms to different bases differ only by a constant factor.) This result is very important for cryptography since it means that there are lots of primes. We note that $N = \log n$ is the number of digits in n. If we choose a number at random in the range $1, 2, \cdots, n$, the probability that it is prime is about $\frac{f(n)}{n} = \frac{1}{\log n} = \frac{1}{N}$. So, if we want a 200 digits prime and guess one at random, then $\frac{1}{200}$ is roughly the probability that it is prime. If you choose $300 - 400$ numbers, you almost surely get a prime.

Now, we come back to the discussion of the running time of the sieve. From the above discussion we see that Algorithm 1 has running time upper bounded by:

$$nf(\sqrt{n}) \leq n \cdot n^{\frac{1}{2}} \sim n^{\frac{3}{2}}$$

We are ready to answer the question "Is the sieve efficient?" In general, we say that an algorithm is efficient when the running time of it is polynomial in the size of input and output. The input size of the sieve is the number of digits in n, so it is $N = \log n (n = 10^N)$. The size of output is *constant* in Problem 1 and is the number of primes $\leq n$ ($\sim \frac{n}{\log n} = \frac{10^N}{N}$) in Problem 2. The running time of the sieve is $n^{\frac{3}{2}} = 10^{\frac{3N}{2}}$. Thus, the time complexity of these two problems is as follows.

For Problem 1 : Input size $= N$, the run time is $10^{\frac{3N}{2}}$. It is inefficient.

For Problem 2 : Output size $= M = \dfrac{10^N}{N}$,

the run time is about $M^{\frac{3}{2}}$ which is polynomial in the output size, so it is efficient.