# CS161 – Fall 2016 — Homework 4

Jenny Zeng, SID 52082740, zhaohuaz@uci.edu

R-11.4, R-24.5, R-24.10, C-24.8

## R-11.4

A complex number $a + b\mathbf{i}$, where $\mathbf{i} = \sqrt{-1}$, can be represented by the pair $(a, b)$. Describe a method performing only three real-number multiplications to compute the pair $(e, f)$ representing the product of $a + b\mathbf{i}$ and $c + d\mathbf{i}$.

**Answer:**

$$\begin{aligned}(a + bi) * (c + di) &= ac + adi + cbi + bdi^2 \\ &= ac + (ad + cb)i - bd\end{aligned}$$

let $p = ac$, $q = bd$, $r = (a + b) * (c + d)$, so r is

$$\begin{aligned}r &= ac + bc + ad + bd \\ &= p + q + (ad + bd)\end{aligned}$$

so

$$(ad + bd) = r - p - q$$

and

$$\begin{aligned}(a + bi) * (c + di) &= ac + (ad + cb)i - bd \\ &= p - q + (r - p - q)i\end{aligned}$$

Because each $p, q, r$ is one real-number multiplication, it can perform only three real-nnumber multiplication to compute $(e, f)$.

## R-24.5

Show the execution of method `FastExponentiation`$(5, 12, 13)$ by constructing a table similar to Table 24.6.

| $p$ | 12 | 6 | 3 | 1 | 0 |
|---|---|---|---|---|---|
| $r$ | 1 | 12 | 8 | 2 | 1 |

**Table 24.6:** Example of an execution of the repeated squaring algorithm for modular exponentiation. For each recursive invocation of `FastExponentiation`$(2, 12, 13)$, we show the second argument, $p$, and the output value $r = 2^p \bmod 13$.

**Answer:**

| $p$ | 12 | 6 | 3 | 1 | 0 |
|---|---|---|---|---|---|
| $r$ | 1 | 12 | 8 | 5 | 1 |

## R-24.10

Construct a table showing an example of the RSA cryptosystem with parameters $p = 17$, $q = 19$, and $e = 5$. The table should have two rows, one for the plaintext $M$ and the other for the ciphertext $C$. The columns should correspond to integer values in the range $[10, 20]$ for $M$.

**Answer:**

| $M$ | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $C$ | 193 | 197 | 122 | 166 | 29 | 2 | 118 | 272 | 18 | 304 | 39 |

## C-24.8

Suppose the primes $p$ and $q$ used in the RSA cryptosystem, to define $n = pq$, are in the range $[\sqrt{n} - \log n, \sqrt{n} + \log n]$. Explain how you can efficiently factor $n$ using this information.

**Answer:**
Because $n = \sqrt{n} * \sqrt{n}$, and if $\sqrt{n}$ is integer, then $p = q = \sqrt{n}$.
However, either $p$ or $q \leq \sqrt{n}$, and the other one will be $\geq \sqrt{n}$
Therefore, we can simply do:

**function** FIND()
  **for** $i = \sqrt{n} - \log n$, $i \leq \sqrt{n}$, i++ **do**
    **if** n mod i == 0 **then**
      **return** (i, n/i)

we can find p or q in range $[\sqrt{n} - \log n, i \leq \sqrt{n}]$
time complexity:
For loop: $O(\log n) = O(k)$ because for a k-bits number, $n = 2^k$ so $log(n) = k$
inside loop:
mod operation would take $O(k^2)$
total time complexity is $O(k^2) * O(k) = O(k^3)$