Homework 4

Liwen Ouyang

March 16th 2020

Divide-and-conquer Multiplication

(a) Let x be a number of n digits, and y be a number of m digits in base $r = 10$. We know $n \geq m$. In order to apply Karatsuba's algorithm we can rewrite x as:

$$x = a_1 r^{n-m} + a_2 r^{n-2m} + \cdots + a_{\lfloor \frac{n}{m} \rfloor} r^{n - \lfloor \frac{n}{m} \rfloor m} + a_{n\%m} = \sum_{i=1}^{\lfloor \frac{n}{m} \rfloor} a_i r^{n-im} + a_{n\%m}$$

where a_i is a number of m digits based on our construct. We can express the multiplication as:

$$x \times y = \sum_{i=1}^{\lfloor \frac{n}{m} \rfloor} a_i y r^{n-im} + a_{n\%m} y \tag{1}$$

Now we can compute $a_i \times y$ using Karatsuba's algorithm because they both have m digits. Since $r = 10$ is the base, multiply its powers with another number is trivial and hence negligible.

procedure M$(x, y, r = 10)$ ▷ x, y are integers of n, m digits, respectively, in base 10
  if x, y both have 1 digit **then**
    return xy ▷ Base case
  end if
  rewrite x as $\sum_{i=1}^{\lfloor \frac{n}{m} \rfloor} a_i r^{n-im} + a_{n\%m}$ ▷ This is $O(\frac{n}{m})$
  for all a_i **do** ▷ $\lfloor \frac{n}{m} \rfloor$ loop
    $m_i \leftarrow K(a_i, y) \times r^{n-im}$ ▷ Karatsuba multiplication: $O(m^{\log_2 3})$
  end for
  $m_s \leftarrow M(a_{n\%m}, y)$ ▷ Recursively compute the remaining product
  return $\sum_i m_i + m_s$ ▷ Linear time summation: $O(n + m) = O(n)$
end procedure

The algorithm returns the desired result as shown by (1). For its running time we have the following recurrence:

$$T(n, m) = T(m, p = n\%m) + O(\frac{n}{m} m^{\log_2 3}) + O(\frac{n}{m}) + O(n)$$

Which can be simplified, since $\frac{n}{m} m^{\log_2 3} \geq \frac{n}{m} \geq n$, to:

$$T(n, m) = T(m, p = n\%m) + O(nm^{\log_2 3 - 1})$$

Assuming $T(n, m) \leq cnm^{\log_2 3 - 1}$ we want to show:

$$T(n, m) \leq cmp^{\log_2 3 - 1} + dnm^{\log_2 3 - 1} \leq cnm^{\log_2 3 - 1}$$

Dividing both sides by $cm^{\log_2 3 - 1}$ yields:

$$n \geq m^{2 - \log_2 3} p^{\log_2 3 - 1} + \frac{d}{c}$$

Note that $p < m \to p^{\log_2 3 - 1} < m^{\log_2 3 - 1}$:

$$m^{2-\log_2 3} p^{\log_2 3 - 1} + \frac{d}{c} < m^{2-\log_2 3} m^{\log_2 3 - 1} + \frac{d}{c} = m + \frac{d}{c}$$

If we choose c such that $m + \frac{d}{c} \leq n$, our inductive step holds. And for base case we have $T(1,1) = O(1)$, which completes the proof of the algorithm's running time $O(nm^{\log_2 3 - 1})$

(b) Using the algorithm from (a) we can define a recursive algorithm to calculate 2^n based on the fact that:

$$a^n = a^{\lfloor \frac{n}{2} \rfloor} \times a^{n - \lfloor \frac{n}{2} \rfloor} \tag{2}$$

procedure F($n, a = 2$) ▷ Given integer $n \geq 0$, calculate a^n
  if $n = 0$ **then**
    return 0
  else if $n = 1$ **then**
    return a ▷ Base Case
  else
    $n_1 \leftarrow \lfloor \frac{n}{2} \rfloor$
    $n_2 \leftarrow n - a$
    return $M(F(n_1, a), F(n_2, a))$ ▷ running time related to the number of digits of a^{n_1}, a^{n_2}
  end if
end procedure

Using induction it is trivial to show that the algorithm is correct based on (2). The number of digits of a^{n_1}, a^{n_2} can be estimated by $\frac{n}{2} \log a$. Therefore, the running time of the algorithm follows the recurrence:

$$T(n) = 2T(\frac{n}{2}) + O((\frac{n}{2} \log a)^{\log_2 3})$$

Which by Master Theorem solves to $T(n) = \Theta(n^{\log_2 3})$.

(c) It is impossible to calculate the decimal representation of any n-bit number in the same asymptotic time. We need to call $F(n_0)$ for all $n_0 \leq n$ and sum the result together to get the decimal representation for a n bit number , which will result in a longer running time.