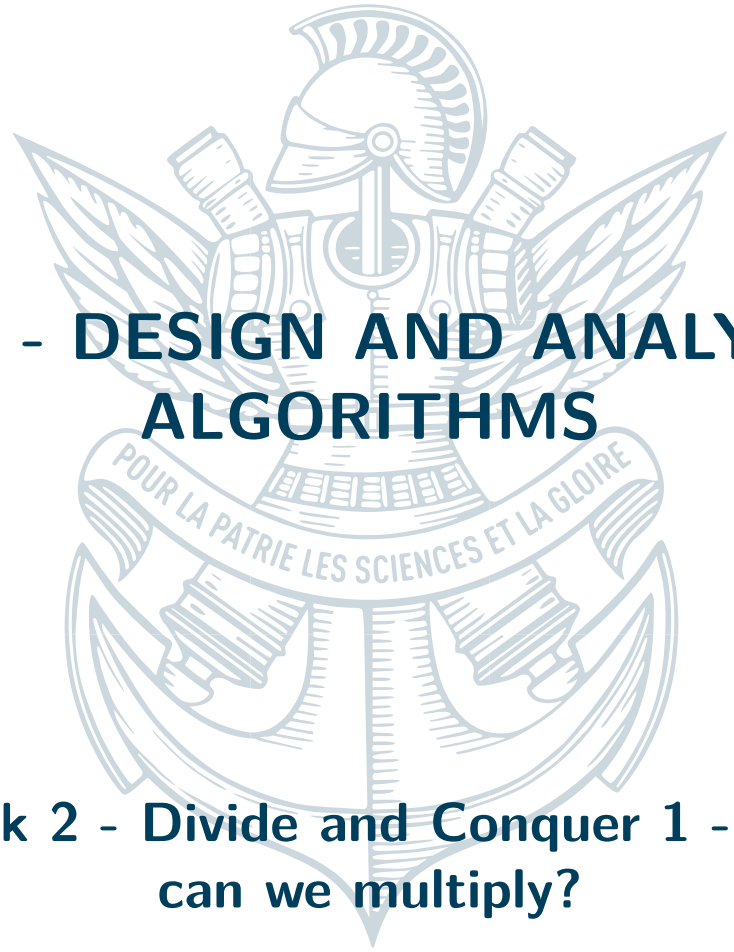

CSE202 - DESIGN AND ANALYSIS OF ALGORITHMS

Homework 2 - Divide and Conquer 1 - How fast can we multiply?

October 3, 2022

Yubo Cai

QUESTION 1

In order to apply similar method of **Karatsuba's algorithms**, we split the polynomial into two input polynomials F and G. We first consider the situation of the degree of polynomials is 1 and have $F = F_0 + TF_1$ and $G = G_0 + TG_1$ where for both F and G the degree is 1. We plug in $T = -1, 0, 1$ we have

1. $H_0 = F_0G_0$
2. $\tilde{H}_1 = (F_0 + F_1)(G_0 + G_1)$
3. $\tilde{H}_{-1} = (F_0 - F_1)(G_0 - G_1)$

Therefore we have the formula that

$$H = FG = (F_0 + TF_1)(G_0 + TG_1) = \frac{1}{2}(2H_0 + (\tilde{H}_1 - \tilde{H}_{-1})T + (\tilde{H}_1 - 2H_0 + \tilde{H}_{-1})T^2)$$

Then we can compute the product of $H = FG$ with $T = x^k$. we have

1. if n is small, use the naive multiplication
2. Let $k := \lceil n/2 \rceil$
3. Split $F = F_0 + x^kF_1, G = G_0 + x^kG_1$ with F_0, F_1, G_0, G_1 of degree $< k$
4. Compute recursively from 1
5. return $H = FG = (F_0 + TF_1)(G_0 + TG_1) = \frac{1}{2}(2H_0 + (\tilde{H}_1 - \tilde{H}_{-1})T + (\tilde{H}_1 - 2H_0 + \tilde{H}_{-1})T^2)$

Where we have the complexity is $C(n) \leq 3C(\lceil n/2 \rceil) + \lambda n$ and λ is an constant here. Then we try to deduce the complexity of computation.

$$\begin{aligned}
C(n) &\leq 3C(\lceil n/2 \rceil) + \lambda n \\
&\leq \lambda n + 3\lambda \lceil n/2 \rceil + 9C\left(\lceil n/2 \rceil_2\right) \\
&\leq \lambda N \left(1 + \frac{3}{2} + \cdots + \left(\frac{3}{2}\right)^{k-1}\right) + 3^k C\left(\lceil n/2 \rceil_k\right) \\
&\leq \lambda N (\frac{1 - (\frac{3}{2})^k}{1 - \frac{3}{2}}) + 3^k C\left(\lceil n/2 \rceil_k\right) \\
&\leq \lambda N (2(\frac{3}{2}^k - 1)) + 3^k C\left(\lceil n/2 \rceil_k\right) \\
&\leq 3^k \left(2\lambda \frac{N}{2^k} + C\left(\lceil n/2 \rceil_k\right)\right)
\end{aligned}$$

Since we have $k = \lceil log_2 n \rceil$ and N we denote as power of 2 s.t. $n \leq N < 2n$, so we have $1 \leq \frac{N}{2^k} < 2$ and $C\left(\lceil n/2 \rceil_k\right) = 1$. Therefore we have

$$C(n) \leq (2\lambda + 1)3^{\lceil \log_2 n \rceil} = (2\lambda + 1)2^{\lceil \log_2 n \rceil * log_2 3}$$

Its complexity is $C(n) = O(n^{log_2 3})$

QUESTION 2

$$V(-2,-1,0,1,2)^{-1} = \frac{1}{24}\begin{pmatrix} 0 & 0 & 24 & 0 & 0 \\ 2 & -16 & 0 & 16 & -2 \\ -1 & 16 & -30 & 16 & -1 \\ -2 & 4 & 0 & -4 & 2 \\ 1 & -4 & 6 & -4 & 1 \end{pmatrix}$$

We apply the same method from **question 1**. We set up two polynomials of $F = F_0+F_1T+F_2T^2$ and $G = G_0+G_1T+G_2T^2$ where for both F and G the degree is 2. We plug in $T = -2,-1,0,1,2$ we have

1. $\tilde{H}_{-2} = (F_0 - 2F_1 + 4F_2)\,(G_0 - 2G_1 + 4G_2)$
2. $\tilde{H}_{-1} = (F_0 - F_1 + F_2)\,(G_0 - G_1 + G_2)$
3. $H_0 = F_0G_0$
4. $\tilde{H}_1 = (F_0 + F_1 + F_2)\,(G_0 + G_1 + G_2)$
5. $\tilde{H}_2 = (F_0 + 2F_1 + 4F_2)\,(G_0 + 2G_1 + 4G_2)$

Therefore we have the formula that

$$\begin{aligned} H &= FG \\ &= (F_0 + F_1T + F_2T^2)(G_0 + G_1T + G_2T^2) \\ &= \frac{1}{24}(h_0 + h_1T + h_2T^2 + h_3T^3 + h_4T^4) \end{aligned}$$

where $h_0 = 24H_0, h_1 = 2\tilde{H}_{-2}-16\tilde{H}_{-1}+16\tilde{H}_1-2\tilde{H}_2, h_2 = -\tilde{H}_{-2}+16\tilde{H}_{-1}-30H_0+16\tilde{H}_1-\tilde{H}_2, h_3 = -2\tilde{H}_{-2} + 4\tilde{H}_{-1} - 4\tilde{H}_1 + 2\tilde{H}_2, h_4 = \tilde{H}_{-2} - 4\tilde{H}_{-1} + 6H_0 - 4\tilde{H}_1 + \tilde{H}_2$

Then we can compute the product of $H = FG$ with $T = x^k$. we have

1. if $n < 3$, use the naive multiplication
2. Let $k := \lceil n/3 \rceil$
3. Split $F = F_0 + x^kF_1 + x^{2k}F_2, G = G_0 + x^kG_1 + x^{2k}G_1$ with $F_0, F_1, F_2, G_0, G_1, G_2$ of degree less than k
4. Compute recursively in the equation above
5. return $H = FG = \frac{1}{24}(h_0 + h_1x^k + h_2x^{2k} + h_3x^{3k} + h_4x^{4k})$ where

$$\begin{cases} h_0 = 24H_0 \\ h_1 = 2\tilde{H}_{-2} - 16\tilde{H}_{-1} + 16\tilde{H}_1 - 2\tilde{H}_2 \\ h_2 = -\tilde{H}_{-2} + 16\tilde{H}_{-1} - 30H_0 + 16\tilde{H}_1 - \tilde{H}_2 \\ h_3 = -2\tilde{H}_{-2} + 4\tilde{H}_{-1} - 4\tilde{H}_1 + 2\tilde{H}_2 \\ h_4 = \tilde{H}_{-2} - 4\tilde{H}_{-1} + 6H_0 - 4\tilde{H}_1 + \tilde{H}_2 \end{cases}$$

The algorithms have 5 times recursive computation in $C(\lceil n/3 \rceil)$ and a number of linear computations in n where we consider it as λn. Therefore we have the complexity of $C(n) \leq 5C(\lceil n/3 \rceil) + \lambda n$. For the similar method in question 1 we have

$$\begin{aligned}
C(n) &\leq 5C(\lceil n/3 \rceil) + \lambda n \\
&\leq \lambda n + 5\lambda \lceil n/3 \rceil + 25C\left(\lceil n/3 \rceil_2\right) \\
&\leq \lambda N \left(1 + \frac{5}{3} + \cdots + \left(\frac{5}{3}\right)^{k-1}\right) + 5^k C\left(\lceil n/3 \rceil_k\right) \\
&\leq \lambda N (\frac{1 - (\frac{5}{3})^k}{1 - \frac{5}{3}}) + 3^k C\left(\lceil n/2 \rceil_k\right) \\
&\leq \lambda N (\frac{3}{2}(\frac{5^k}{3} - 1)) + 5^k C\left(\lceil n/3 \rceil_k\right) \\
&\leq 5^k \left(\frac{3}{2}\lambda \frac{N}{3^k} + C\left(\lceil n/3 \rceil_k\right)\right)
\end{aligned}$$

Since we have $k = \lceil log_3 n \rceil$ and N we denote as power of 3 s.t. $n \leq N < 3n$, so we have $1 \leq \frac{N}{2^k} < 3$ and $C\left(\lceil n/3 \rceil_k\right) = 1$. Therefore we have

$$C(n) = O(n^{log_3 5})$$

Since $log_3 5 < log_2 3$. Therefore the second algorithms is better than the Karatsuba's algorithm from question 1.

QUESTION 3

From the method in question 1 and 2, we can deduce the generalization in the following:

1. if $n < m$, use the naive multiplication
2. Let $k := \lceil n/m \rceil$
3. Split F and G into the form of $F = F_0 + x^k F_1 + ... + x^{(m-1)k} F_{m-1}$ and $G = G_0 + x^k G_1 + ... + x^{(m-1)k} G_{m-1}$ where for all G_i, F_j the degree is less than k.
4. Compute recursively F and G with the linear combination at $(-m, -m+1, ..., m-1, m)$
5. We return $H = FG$ with

$$V^{-1}(-m, ..., m) \begin{pmatrix} P(a_0) \\ \vdots \\ P(a_k) \end{pmatrix} = \begin{pmatrix} p_0 \\ \vdots \\ p_k \end{pmatrix}$$

   where we can find out p_0 to p_k with matrix computation.

For similar reason we have the complexity of the algorithm

$$C(n) \leq (2m-1)C(\lceil n/m \rceil) + \lambda n$$

Therefore we have the $C(n)$ in **big O** that

$$C(n) = O(n^{log_m(2m-1)})$$

Then we compute the limit of $f(m)$ as $m \to \infty$ where we have

$$log_m(2m-1) = \frac{ln(2m-1)}{ln(m)} \sim \frac{ln(2m)}{ln(m)} = \frac{ln2 + ln(m)}{ln(m)} = 1 + \frac{ln2}{ln(m)}$$

as $m \to \infty$

$$1 + \frac{ln2}{ln(m)} \to 1$$

Therefore the asymptotical complexity of the algorithms can be express as $O(n)$