SE220 Lecture 3&4 Notes

I. Asymptotic Notation

i. *O*-Notation

$O(g(n))$ { $f(n)$: **there exist positive constants** c and n_0 such that $f(n)\epsilon[0, cg(n)]$ **for all** $n \geq n_0$ }.

ii. Ω-Notation

$\Omega(g(n))$ ={ $f(n)$: **there exist positive constants** c and n_0 such that $cg(n)\epsilon[0, f(n)]$ **for all** $n \geq n_0$ }

iii. Θ-Notation

$\Theta(g(n))$ ={ $f(n)$: **there exists positive constants** c_1 , c_2 and n_0 such that $f(n)\epsilon[c_1 g(n),\ c_2 g(n)]$ **for all** $n \geq n_0$ }

II. Divide-and-Conquer Example – **Square Matrices Multiplication**

Choose two 2×2 matrices A and B as example to illustrate how this algorithm is applied to compute $C = AB$ that matrix C is also a square one.

We have known that for an arbitrary entry in $n \times n$ matrix C, it has the formula

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

If we compute the multiplication directly, which needs 3 **for** loop, and the running time is $T(n) = \Theta(n^3)$. However, we have got an idea of partitioning the matrix to compute multiplication in linear algebra, which can be also considered an another way to solve this problem.

If we have $A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$, and $B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$, and $C = AB$ is supposed to be:

$$C = \begin{pmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{pmatrix}$$

Let C have the form $C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$, which corresponds to the form we just computed:

$$C_{11} = A_{11}B_{11} + A_{12}B_{21}$$
$$C_{12} = A_{11}B_{12} + A_{12}B_{22}$$
$$C_{21} = A_{21}B_{11} + A_{22}B_{21}$$
$$C_{22} = A_{21}B_{12} + A_{22}B_{22}$$

We can consider each entry in matrix A and B as a submatrix with size $\frac{n}{2}$, and what we need to do is to compute eight $\frac{n}{2} \times \frac{n}{2}$ multiplications and four additions, which can be completed with divide-and-conquer algorithm by recursion(call the function itself). Thus, we can obtain the running time

$$T(n) = \begin{cases} \Theta(1) & n = 1 \\ 8T\left(\frac{n}{2}\right) + \Theta(1) & n > 1 \end{cases} = \begin{cases} c_0 & n = 1 \\ 8T\left(\frac{n}{2}\right) + c_1 & n > 1 \end{cases}$$

$$\begin{aligned} T(n) &= c_1 + \sum_{i=1}^{\log_2 n - 1} 8^i \cdot \frac{c_1}{2} + 8^{\log_2 n} \cdot c_0 \\ &= c_1 + \frac{1}{2} \sum_{i=1}^{\log_2 n - 1} 8^i c_1 + c_0 n^3 \\ &= \Theta(n^3) \end{aligned}$$

It illustrates that there is no difference in time complexity in terms of both ways.

III. Substitution method to solve Recurrence Equation

This method have two different versions:

1. Additive term that is an explicit expression:

S1: **Guess the solution** S2: **Use induction to confirm the solution**

Example: For a given recurrence

$$T(n) = \begin{cases} 1 & n = 1 \\ 2T\left(\frac{n}{2}\right) + n & n > 1 \end{cases}$$

Solution:

S1: **Guess** $T(n) = n\log_2 n + n$

S2: **Induction**

When $n=1$, $T(1)=1\cdot\log_2 1+1=1$, the solution holds.

Assume $T\left(\frac{n}{2}\right)=\frac{n}{2}\log_2\frac{n}{2}+\frac{n}{2}$ holds.

Thus,

$$T(n)=2T\left(\frac{n}{2}\right)+n=2\left(\frac{n}{2}\log_2\frac{n}{2}+\frac{n}{2}\right)+n$$

$$=n\log_2\frac{n}{2}+2n$$

$$=n(\log_2 n-1)+2n$$

$$=n\log_2 n+n$$

which indicates the solution is correct.

2. Additive term is expressed with asymptotic notation

S1: **Guess the solution**

S2: **Prove the solution directly with the definition of asymptotic notation**

Example 1: For a given recurrence

$$T(n)=2T\left(\frac{n}{2}\right)+\Theta(n)$$

Solution:

S1: **Guess** $T(n)=\Theta(n\log_2 n)$

S2: **Prove** the solution directly(induction)

Since we would like to prove a solution with Θ-notation, we need to prove $T(n)=\Omega(n\log_2 n)$ first and then $T(n)=O(n\log_2 n)$.

(1) Lower bound

Hypothesis: $T(n)\geq d_1 n\log_2 n$ (d_1 is a constant)

Assume $T\left(\frac{n}{2}\right)\geq d_1\cdot\frac{n}{2}\log_2\frac{n}{2}$ holds.

$$
\begin{aligned}
T(n) &= 2T\left(\frac{n}{2}\right) + \Theta(n) \\
&\geqslant 2T\left(\frac{n}{2}\right) + c_1 n \\
&\geqslant 2\left(d_1 \cdot \frac{n}{2}\log_2\frac{n}{2}\right) + c_1 n \\
&= d_1 n(\log_2 n - 1) + c_1 n \\
&= d_1 n\log_2 n + (c_1 - d_1)n \\
&\geqslant d_1 n\log_2 n (\exists \ \ c_1 \geqslant d_1)
\end{aligned}
$$

Thus, $T(n) = \Omega(n\log_2 n)$

(2) Upper bound(likewise)

Hypothesis: $T(n) \leqslant d_2 n\log_2 n$ (d_2 is a constant)

Assume $T\left(\frac{n}{2}\right) \leqslant d_2 \cdot \frac{n}{2}\log_2\frac{n}{2}$ holds

$$
\begin{aligned}
T(n) &= 2T\left(\frac{n}{2}\right) + \Theta(n) \\
&\leqslant 2T\left(\frac{n}{2}\right) + c_2 n \\
&\leqslant 2\left(d_2 \cdot \frac{n}{2}\log_2\frac{n}{2}\right) + c_2 n \\
&= d_2 n(\log_2 n - 1) + c_2 n \\
&= d_2 n\log_2 n + (c_2 - d_2)n \\
&\leqslant d_2 n\log_2 n (\exists \ \ c_2 \leqslant d_2)
\end{aligned}
$$

Thus, $T(n) = O(n\log_2 n)$

Finally, we can prove that $T(n) = \Theta(n\log_2 n)$, which indicates that the solution is correct.

(**Tips:** The inequality begins with the definition of Θ-notation)

Example 2: Given the recurrence

$$T(n) = 2T\left(\frac{n}{2}\right) + \Theta(1)\ (n > 1)$$

Prove: $T(n) = O(n)$

Proof Let's complete this proof with the definition of Θ-notation.

Hypothesis: $T(n) \leqslant cn$ (c is a constant)

Assume $T\left(\frac{n}{2}\right) \leqslant c \cdot \frac{n}{2}$ holds

$$\begin{aligned} T(n) &= 2T\left(\frac{n}{2}\right) + \Theta(1) \\ &\leqslant 2T\left(\frac{n}{2}\right) + c_1 \\ &\leqslant 2\left(c \cdot \frac{n}{2}\right) + c_1 \\ &= cn + c_1 \end{aligned}$$

The inequality above cannot prove the hypothesis, but since the conclusion given in the question is correct, we need to change our view for this.

Please notice that we have obtained $T(n) \leqslant cn + c_1$. If we want to prove $T(n) \leqslant cn$, we can do some subtraction for the hypothesis, thus it becomes:

$T(n) \leqslant cn - kd$ (k is constant)

Assume $T\left(\frac{n}{2}\right) \leqslant c \cdot \frac{n}{2} - kd$ holds

$$\begin{aligned} T(n) &= 2T\left(\frac{n}{2}\right) + \Theta(1) \\ &\leqslant 2T\left(\frac{n}{2}\right) + c_1 \\ &\leqslant 2\left(c \cdot \frac{n}{2} - kd\right) + c_1 \\ &= cn + c_1 - 2kd \\ &\leqslant cn (\exists\ \ c_1 \leqslant 2kd) \end{aligned}$$

which indicates that the conclusion is correct.

(**Tips:** The core of proving upper or lower bound is to prove that the recurrence align with the definition in the end. Thus, it is possible to adjust the hypothesis by subtracting a lower-order term.)

IV. Recursion Tree

1. Regular Recurrence

Example: Given the recurrence

$$T(n) = 3T\left(\frac{n}{4}\right) + \Theta(n^2)\ (n > 1)$$

Solution:

Since we know that $T(n) = constant$ when $n = 1$, rewrite the recurrence

$$T(n) = \begin{cases} c_0 & n = 1 \\ 3T\left(\frac{n}{4}\right) + c_1 n^2 & n > 1 \end{cases}$$

$$\begin{aligned} T(n) &= \sum_{i=0}^{\log_4 n - 1} \left(\frac{3}{16}\right)^i c_1 n^2 + c_0 \cdot 3^{\log_4 n} \\ &= c_1 n^2 \sum_{i=0}^{\log_4 n - 1} \left(\frac{3}{16}\right)^i + c_0 \cdot 3^{\log_4 n} \\ &= \frac{16}{13} c_1 n^2 \left[1 - \left(\frac{3}{16}\right)^{\log_4 n}\right] + c_0 \cdot 3^{\log_4 n} \\ &= \frac{16}{13} c_1 n^2 \left[1 - n^{\frac{1}{2}\log_2 3 - 2}\right] + c_0 n^{\frac{1}{2}\log_2 3} \\ &= \frac{16}{13} c_1 n^2 + \left(c_0 - \frac{16}{13} c_1\right) n^{\frac{1}{2}\log_2 3} \\ &= O(n^2) \end{aligned}$$

Do the proof for the result to ensure it is correct

Assume $T(n) \leqslant c_2 n^2$ holds for $n \geqslant n_0$

$$\Leftrightarrow \frac{16}{13} c_1 n^2 + \left(c_0 - \frac{16}{13} c_1\right) n^{\frac{1}{2}\log_2 3} \leqslant c_2 n^2 (n \geqslant n_0)$$

$$\Leftrightarrow c_2 \geqslant \frac{16}{13} c_1 + \left(c_0 - \frac{16}{13} c_1\right) n^{\frac{1}{2}\log_2 3 - 2} (n \geqslant n_0)$$

Use math, we can obtain that the inequality can hold with specified c_0 and c_1, since

terms $n^{\frac{1}{2}\log_2 3-2}$ is decreasing, which proves the $T(n)=O(n^2)$ is correct.

Or we can do like this

$$\begin{aligned}
T(n) &= \sum_{i=0}^{\log_4 n-1}\left(\frac{3}{16}\right)^i c_1 n^2 + c_0\cdot 3^{\log_4 n} \\
&= c_1 n^2 \sum_{i=0}^{\log_4 n-1}\left(\frac{3}{16}\right)^i + c_0\cdot 3^{\log_4 n} \\
&< c_1 n^2 \sum_{i=0}^{\infty}\left(\frac{3}{16}\right)^i + c_0\cdot 3^{\log_4 n} \qquad (\sum_{i=0}^{\infty} x^i = \frac{1}{1-x}\,(|x|<1)) \\
&= \frac{16}{13} c_1 n^2 + c_0\cdot 3^{\log_4 n} \\
&= O(n^2)
\end{aligned}$$

Try to explore if $T(n)=\Omega(n^2)$ also holds:

Assume $T(n)\geq c_3 n^2$ holds for $n\geq n_0$

$$\Leftrightarrow c_3 \leq \frac{16}{13}c_1 + \left(c_0 - \frac{16}{13}c_1\right) n^{\frac{1}{2}\log_2 3-2} \,(n\geq n_0)$$

Although term $n^{\frac{1}{2}\log_2 3-2}$ is decreasing, and $\lim_{n\to\infty} n^{\frac{1}{2}\log_2 3-2}=0$, we can choose specified c_1 to ensure the inequality to hold, which means $T(n)=\Omega(n^2)$ is correct.

Finally, we can obtain $T(n)=\Theta(n^2)$

2. Irregular Recurrence

Example: Given the recurrence

$$T(n)=T\left(\frac{n}{3}\right)+T\left(\frac{2n}{3}\right)+\Theta(n)\,(n>1)$$

Find the upper and lower bound of $T(n)$.

Solution:

Rewrite the recurrence $T(n)=T\left(\frac{n}{3}\right)+T\left(\frac{2n}{3}\right)+cn\,(n>1)$

If we draw the recursion tree, we will find that the tree is asymmetric, which means it has two paths to the base layer. We can solve the heights respectively.

$$\frac{n}{3^{h_1}}=1 \Rightarrow h_1=\log_3 n,\ \ \frac{n}{\left(\frac{3}{2}\right)^{h_2}}=1 \Rightarrow h_2=\log_{\frac{3}{2}} n$$

From the math, we can know that $h_2 > h_1 (n \geqslant n_0)$, which means h_2 is the longest path from the root. When $h \leqslant h_1$, the tree is symmetric, the sum of each layer is cn. However, when $h_1 < h \leqslant h_2$, the tree is asymmetric. Since the number of nodes has **reduced**, the sum of each layer is less than cn.

To find lower bound, let $h \leqslant h_1$ and the number of the rest of the nodes be m, thus

$$T(n)=h_1 cn+m=cn\log_3 n+m \geqslant cn\log_3 n$$

Thus, $T(n)=\Omega(n\log_3 n)$

Meanwhile, we can also obtain that $m \leqslant (h_2-h_1)cn$

$$T(n)=h_1 cn+m \leqslant h_1 cn+(h_2-h_1)cn=h_2 cn=cn\log_{\frac{3}{2}} n$$

Thus, $T(n)=O\left(n\log_{\frac{3}{2}} n\right)$

According to the lower and upper bound obtained above, we can know that $T(n)=O(n\log_2 n)$ is also correct. Likewise, we can know that $T(n)=\Omega(n\log_2 n)$ is also correct.

V. Most-common-used **Master Theorem**

Proof For a given recurrence equation of the form $T(n)=aT\left(\frac{n}{b}\right)+cn^d$, where $a>0$, $b>1$, c and d are constants. Use recursion tree to solve this equation, we have:

$$T(n) = cn^d + \sum_{i=1}^{\log_b n} cn^d \left(\frac{a}{b^d}\right)^i$$
$$= cn^d \left[1 + \sum_{i=1}^{\log_b n} \left(\frac{a}{b^d}\right)^i\right]$$

When $a = b^d (d = \log_b a)$, $T(n) = cn^d (1 + \log_b n) = \Theta(n^d \log_b n)$;

When $a \neq b^d (d \neq \log_b a)$,

$$T(n) = cn^d \left\{1 + \frac{\frac{a}{b^d}\left[\left(\frac{a}{b^d}\right)^{\log_b n} - 1\right]}{\frac{a}{b^d} - 1}\right\}$$
$$= cn^d \left[1 + \frac{\frac{a}{b^d}\left(n^{\log_b a - d} - 1\right)}{\frac{a}{b^d} - 1}\right]$$

If $a < b^d (d > \log_b a)$, $n^{\log_b a - d} - 1 < 0$, $\frac{a}{b^d} - 1 < 0 \Rightarrow 1 + \frac{a}{b^d} \cdot \frac{n^{\log_b a - d} - 1}{\frac{a}{b^d} - 1} > 0$

Thus, $T(n) = \Theta(n^d)$.

If $a > b^d (d < \log_b a)$, $n^{\log_b a} > n^d$

Thus, $T(n) = \Theta\left(n^{\log_b a}\right)$

Finally, this most-common master theorem can be restated as:

$$T(n) = aT\left(\frac{n}{b}\right) + cn^d = \begin{cases} \Theta(n^d \log_b n), & a = b^d \\ \Theta(n^d), & a < b^d \\ \Theta\left(n^{\log_b a}\right), & a > b^d \end{cases}$$