Master Method

For solving recurrences

Introduction

- “Cookbook” for solving recurrences
- No guessing, tree construction or calculation required
- Has three cases, that are used to decide the form of solution
- Given a recurrence equation, you need to decide which category it falls among the three in order to find the bounds on the running time

General Recurrence relation

- For a recurrence described below:

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

- Here each time, the original problem is divided into ‘a’ sub-problems
- Each sub-problem is solved in T(n/b) time
- Both ‘a’ and ‘b’ are positive constants
- f(n) is the time of combining the results of the a sub-problems generated at a stage

Master Method

- We have three cases for the previously described recurrence:

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \theta(n^{\log_b a})$

2. If $f(n) = \theta(n^{\log_b a})$, then $T(n) = \theta(n^{\log_b a} \lg n)$

3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af\left(\frac{n}{b}\right) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \theta(f(n))$

Example 1

$$T(n) = 9T\left(\frac{n}{3}\right) + n$$

$$n^{\log_b a} = n^{\log_3 9} = n^2$$

$$f(n) = O(n)$$

Therefore, case 1 applies and we have:

$$T(n) = \theta\left(n^{\log_b a}\right) = \theta(n^2)$$

Example 2

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

$$n^{\log_b a} = n^{\log_3 2} = n^{0.631}$$

$$f(n) = O(1)$$

Therefore, case 1 applies and we get

$$T(n) = \theta(n^{\log_3 2})$$

Example 3

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

$$n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$$

$$f(n) = \theta(1)$$

Therefore, case 2 applies and we have

$$T(n) = \theta(\lg n)$$

Example 4

$$T(n) = 3T\left(\frac{n}{4}\right) + n \lg n$$

$$n^{\log_b a} = n^{\log_4 3} = n^{0.792}$$

$$n^{\log_b a+\epsilon} = n^{\log_4 3+\epsilon} = n \; for \; \epsilon = 1$$

$$\Rightarrow f(n) = \Omega(n^{\log_4 3+\epsilon})$$

Therefore, case 4 will apply if regularity condition holds

- $af\left(\frac{n}{b}\right) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n

$$af\left(\frac{n}{b}\right) \Rightarrow 3\left(\frac{n}{4}\right)\lg\left(\frac{n}{4}\right)$$

$$= \left(\frac{3}{4}\right)nlgn - \left(\frac{3}{4}\right)nlg4$$

$$= \left(\frac{3}{4}\right)f(n) - 1.5n$$

Thus, the condition holds for c = 3/4

Therefore, case 3 applies and we have

$$T(n) = \theta(n\lg n)$$

Example 5

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

$$n^{\log_b a} = n^{\log_2 2} = n$$

$$\log_b a + \epsilon \ = \log_2 2 + \epsilon \Rightarrow n.n^{\epsilon 1}$$

$$\frac{f(n)}{\log_b a + \epsilon} = \frac{nlogn}{n.n^{\epsilon}} = \frac{lgn}{n^{\epsilon}}$$

- Above is not an asymptotic lower bound on nlgn as logn is not asymptotically larger than n^{ϵ} for any positive ϵ
- This might not be obvious for $0 < \epsilon < 1$ but holds true for sufficiently large n in this case also
- Therefore, case 3 does not hold
- We can find the asymptotic bound using recursion tree and prove it using substitution method

Proof using L’Hospital’s Rule

Suppose for any two functions f(x) and g(x) and some real number ‘a’ or $\pm infinity$, we want to find

$$\lim_{x \to a} \frac{f(x)}{g(x)}$$

And one of the following cases occurs:

$$\lim_{x \to a} \frac{f(x)}{g(x)} = \frac{0}{0} \quad \text{or}$$

$$\lim_{x \to a} \frac{f(x)}{g(x)} = \frac{\pm\infty}{\pm\infty}$$

L'Hospital's Rule

$$\text{if} \lim_{x\to a} \frac{f(x)}{g(x)} = \frac{0}{0} \quad \text{or} \quad \lim_{x\to a} \frac{f(x)}{g(x)} = \frac{\pm\infty}{\pm\infty}$$

Then,

$$\lim_{x\to a} \frac{f(x)}{g(x)} = \lim_{x\to a} \frac{f'(x)}{g'(x)}$$

In the previous example we have,

$$\lim_{n\to\infty} \frac{lgn}{n^{\epsilon}} = \frac{\infty}{\infty}$$

Thus L'Hospital's rule applies and we get,

$$\lim_{n\to\infty} \frac{lgn}{n^{\epsilon}} = \lim_{n\to\infty} \frac{\frac{1}{\mathrm{n}}}{\epsilon n^{\epsilon-1}} = \frac{0}{\infty} = 0$$

$$\lim_{n\to\infty} \frac{lgn}{n^{\epsilon}} = 0 \Rightarrow \text{that}\, n^{\epsilon}\ is\ an\ upper\ bound\ on\ lgn$$

Example 6

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

$$n^{\log_b a} = n$$

$$\Rightarrow f(n) = \theta(n^{\log_b a})$$

Therefore, case 2 applies and we have

$$T(n) = \theta(nlgn)$$

Example 7

$$T(n) = 8T\left(\frac{n}{2}\right) + \theta(n^2)$$

$$n^{\log_b a} = n^3$$

Above is polynomially larger than $\theta(n^2)$ i.e. $f(n) = O(n^{3-\epsilon})$

Therefore, case 1 applies and we have

$$T(n) = \theta(n^3)$$

Example 8

$$T(n) = 7T\left(\frac{n}{2}\right) + \theta(n^2)$$

$$n^{\log_b a} = n^{\log_2 7} = n^{2.81}$$

$$\Rightarrow f(n) = O(n^{2.81-\epsilon}) for\ \epsilon = 0.8$$

Therefore, case 1 applies and we have

$$T(n) = \theta(n^{lg7})$$

Example 9

Professor Caesar wishes to develop a matrix-multiplication algorithm that is asymptotically faster than Strassen’s algorithm. His algorithm will use the divide and-conquer method, dividing each matrix into pieces of size n/4 x n/4, and the divide and combine steps together will take $\theta(n^2)$ time. He needs to determine how many subproblems his algorithm has to create in order to beat Strassen’s algorithm. If his algorithm creates ‘a’ subproblems, then the recurrence for the running time T(n) becomes $T(n) = aT\left(\frac{n}{4}\right) + \theta(n^2)$.What is the largest integer value of a for which Professor Caesar’s algorithm would be asymptotically faster than Strassen’s algorithm?

$$n^{\log_b a} = n^{\log_4 a}$$

Time complexity of Strassen's Method = $O\left(n^{lg7}\right) = O(n^{2.81})$

Using the Master method, the largest value of 'a' is possible when case 1 applies

For the time complexity of Professor's algorithm to be less than $\mathrm{O}(n^{2.81})$ we need to have

$$2 \le \log_4 a < 2.81$$

$$\log_4 a < \log_2 7$$

$$\frac{\log_2 a}{2} < \log_2 7$$

$$\log_2 a < 2\log_2 7$$

$$\log_2 a < \log_2 7^2$$

$$a < 49$$

Therefore, the maximum possible value of ‘a’ is 48.