ECE Algorithm Homework 1

1. Prove the *Symmetry* property of $\Theta(\cdot)$, i.e. $f(n) = \Theta(g(n))$ if and only if $g(n) = \Theta(f(n))$.

Solution:

f(n) = Θ(g(n)) ==> g(n) = Θ(f(n))

Suppose that f(n) = Θ(g(n)). We got f(n) = O(g(n)) and f(n) = Ω (g(n)).

Then, there exist A, B > 0 such that Ag(n) ≤ f(n) ≤ Bg(n) for sufficiently large n.

Since f(n) ≤ Bg(n) ==> (1/B) f(n) ≤ g(n) and Ag(n) ≤ f(n) ==> g(n) ≤ (1/A) f(n),

we have (1/B) g(n) ≤ g(n) ≤ (1/A) g(n) for sufficiently large n.

Since 1/A, 1/B > 0, We got g(n) = O(f(n)) and g(n) = Ω (f(n)). we conclude that g(n) = Θ(f(n)).

g(n) = Θ(f(n)) ==> f(n) = Θ(g(n))

Can proof using the same way we proof f(n) = Θ(g(n)) ==> g(n) = Θ(f(n)).

We can conclude that f(n) = Θ (g(n)) if and only if g(n) = Θ (f(n)).

2. Problem 3-2 in CLRS Text book.

A	B	O	o	Ω	ω	Θ
$\lg^k n$	n^ϵ	yes	yes	no	no	no
n^k	c^n	yes	yes	no	no	no
$\sqrt{n}$	$n^{\sin n}$	no	no	no	no	no
2^n	$2^{n/2}$	no	no	yes	yes	no
$n^{\log c}$	$c^{\log n}$	yes	no	yes	no	yes
$\log(n!)$	$\log(n^n)$	yes	no	yes	no	yes

3. You have three algorithms to a problem and you do not know their efficiency, but fortunately, you find the recurrence formulas for each solution, which are shown as follows:

A: $T(n) = 5T\left(\frac{n}{2}\right) + \Theta(n)$

B: $T(n) = 2T\left(\frac{9n}{10}\right) + \Theta(n)$

C: $T(n) = T\left(\frac{n}{3}\right) + \Theta(n^2)$

Please give the running time of each algorithm (in Θ notation), and which of your algorithms is the fastest (You probably can do this without a calculator)?

Solution:

For algorithm A: $T(n) = 5T\left(\frac{n}{2}\right) + \Theta(n)$,

$a = 5,\ b = 2,\ f(n) = \Theta(n)$, $\therefore d = 1,\ \log_b a = \log_2 5 > 1 = d$, so $T(n) = \Theta\left(n^{\log_b a}\right) =$

$\Theta(n^{\log_2 5})$.

For algorithm B: $T(n) = 2T\left(\frac{9n}{10}\right) + \Theta(n)$,

$a = 2,\ b = \frac{10}{9},\ f(n) = \Theta(n)\ ,\ \therefore d = 1,\ \log_b a = \log_{\frac{10}{9}} 2 > 1 = d,$ so $T(n) = \Theta\left(n^{\log_b a}\right) =$

$\Theta(n^{\log_{\frac{10}{9}} 2})$.

For algorithm C: $T(n) = T\left(\frac{n}{3}\right) + \Theta(n^2)$,

$a = 1,\ b = 3,\ f(n) = \Theta(n^2)\ ,\ \therefore d = 2,\ \log_b a = \log_3 1 < 2 = d,$

so $T(n) = \Theta(f(n)) = \Theta(n^d) = \Theta(n^2)$.

$\log_2 5 > \log_2 4 = 2,\ \log_2 5 < \log_2 8 < 3 = \log_{\frac{10}{9}}(\frac{10}{9})^3 = \log_{\frac{10}{9}} \frac{1000}{729} < \log_{\frac{10}{9}} 2$. So, we have

$2 < \log_2 5 < \log_{\frac{10}{9}} 2$. The 3rd solution C is the fastest.

4. Use the substitution method to prove that $T(n) = 2T\left(\frac{n}{2}\right) + cn\log_2 n$ is $(n\,(\log_2 n)^2)$.

Solution:

$T(1) = 1 > 0 = d \times 1 \times \log_2 1$, so, we set the boundary as 2. Our base is $T(2) \leq d * 2\,(\log_2 2)^2$.

Induction Hypothesis: If for all k < n we have $T(k) \leq dk\,(\log_2 k)^2$.

$T(n) = 2T\left(\frac{n}{2}\right) + cn\log_2 n \leq 2d\frac{n}{2}(\log_2 \frac{n}{2})^2 + cn\log_2 n = dn(\log_2 n - 1)^2 + cn\log_2 n$

$= dn(\log_2 n)^2 - 2dn\log_2 n + dn + cn\log_2 n.$

If we set $-2dn\log_2 n + dn + cn\log_2 n \leq 0$, formula above is smaller than $dn\,(\log_2 n)^2$,

$\leftrightarrow\ (2\log_2 n - 1)nd \geq cn\log_2 n$

$\leftrightarrow\ (2\log_2 n - 1)d \geq c\log_2 n$

$\leftrightarrow\ d \geq \frac{c\log_2 n}{2\log_2 n - 1} = c\frac{1}{2 - \frac{1}{\log_2 n}},\ c\frac{1}{2 - \frac{1}{\log_2 n}}$ monotone decrease from 2 to $+\infty$,

$c\frac{1}{2 - \frac{1}{\log_2 n}} \leq c\frac{1}{2 - \frac{1}{\log_2 2}} = c$. That is, if we set $d \geq c,\ T(n) \leq dn\,(\log_2 n)^2$

Therefore, $T(n) = O(n\,(\log_2 n)^2)$.

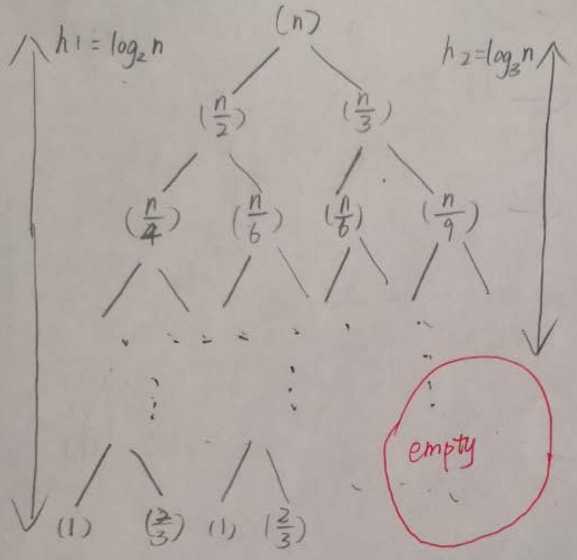
5. First use the iteration method to solve the recurrence

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

Then use the substitution method to verify your solution.

Solution:

Basicly, we can use recursion tree to analyze running time.



$\Rightarrow n$

$\Rightarrow \frac{n}{2}+\frac{n}{3}=\frac{5}{6}n=n\times\frac{5}{6}$

$\Rightarrow \frac{n}{2}\times\frac{5}{6}+\frac{n}{3}\times\frac{5}{6}=n\times(\frac{5}{6})^2$

$\vdots$

As you can see, the right child of a node in this tree has smaller problem size, which means there are some empty in this tree, which is marked by red cycle. we can compute the left height h_1 and right height h_2 of this tree to get the upper bound and lower bound.

$\therefore$ We have

$$C_1 = n+\frac{5}{6}n+\cdots+(\frac{5}{6})^{\log_3 n}\cdot n \leq T(n) \leq n+\frac{5}{6}n+\cdots+(\frac{5}{6})^{\log_2 n}\cdot n = C_2$$

Note. $a^{\log_b n} = n^{\log_b a}$. Since $a^{\log_b n} = a^{\log_b a^{\log_a n}} = a^{\log_a n \log_b a} = n^{\log_b a}$

$$C_1 = n(1+\frac{5}{6}+\cdots+(\frac{5}{6})^{(\log_3 n)}) = n\frac{1-(\frac{5}{6})^{\log_3 n+1}}{1-\frac{5}{6}} = 6n(1-\frac{5}{6}n^{\log_3\frac{5}{6}}), \text{ which is } \Theta(n)$$

$$C_2 = n(1+\frac{5}{6}+\cdots+(\frac{5}{6})^{\log_2 n}) = n\frac{1-(\frac{5}{6})^{\log_2 n+1}}{1-\frac{5}{6}} = 6n(1-\frac{5}{6}n^{\log_2\frac{5}{6}}). \text{ which is } \Theta(n)$$

So, we can get $T(n)$ is $\Theta(n)$.

The upper bound:

IH: $T(k) \le dk$ for all $k<n$

$T(n) = T(\frac{n}{2}) + T(\frac{n}{3}) + n \le d\frac{n}{2} + d\frac{n}{3} + n = d\frac{5}{6}n + n$

In order to make $T(n) \le dn$, we can set

$\frac{5}{6}dn + n \le dn$

$\Leftrightarrow \quad n \le \frac{d}{6}n \Rightarrow d \ge 6$

If we set $d \ge 6$, $T(n) \le dn$, which means $T(n)$ is $O(n)$.

The lower bound:

IH: $T(k) \ge ck$ for all $k<n$

$T(n) = T(\frac{n}{2}) + T(\frac{n}{3}) + n \ge c\frac{n}{2} + c\frac{n}{3} + n = c\frac{5}{6}n + n$

In order to make $T(n) \ge cn$, we can set

$c\frac{5}{6}n + n \ge cn$

$\Leftrightarrow \quad n \ge \frac{c}{6}n \Rightarrow c \le 6$

If we set $c \le 6$, $T(n) \ge cn$, which means $T(n)$ is $\Omega(n)$.

So, $T(n)$ is $\Theta(n)$, the constant coefficient is 6.

6. Solving the recurrence: (Base is 2)

$$T(n) = 2T(\sqrt{n}) + \log n$$

(*Hint: Making change of variable*)

Solution:

If we set $m = \log n$, we will have

$$T(2^m) = 2T\left(2^{\frac{m}{2}}\right) + m$$

And we set $S(m) = T(2^m)$, we will get

$$S(m) = 2S\left(\frac{m}{2}\right) + m$$

We can easily use master method to get the solution of it: $S(m) = \Theta(m \log m)$. And we substitute n back to this solution, $T(n) = T(2^m) = S(m) = \Theta(m \log m) = \Theta(\log n \log(\log n))$.

7. You have 5 algorithms, A1 took $O(n)$ steps, A2 took $\Theta(n \log n)$ steps, and A3 took $\Omega(n)$ steps, A4 took $O(n^3)$ steps, A5 took $o(n^2)$ steps. You had been given the exact running time of

each algorithm, but unfortunately you lost the record. In your messy desk you found the following formulas:

(a) $7n \log_2 n + 12 \log_2 \log_2 n$

(b) $7\left(2^{2\log_2 n}\right) + \frac{n}{2} + 957$

(c) $\frac{2^{\log_4 n}}{3} + 120n + 7$

(d) $(\log_2 n)^2 + 75$

(e) $7n!$

(f) $2^{3\log_2 n}$

(g) $2^{2\log_2 n}$

For each algorithm write down all the possible formulas that could be associated with it.

Solution:

Simplify and rewrite:

(a) $7n\log_2 n + 12\log_2\log_2 n$

(b) $7\cdot(2^{2(\log_2 n)}) + \frac{n}{2} + 7 = 7n^2 + \frac{n}{2} + 7$

(c) $\frac{2^{\log_4 n}}{3} + 120n + 7 = 120n + \frac{\sqrt{n}}{3} + 7$

(d) $(\log_2 n)^2 + 75$

(e) $7n!$

(f) $2^{3\log_2 n} = n^3$

(g) $2^{2\log_2 n} = n^2$

A1: (c) (d)

A2: (a)

A3: (a) (b) (c) (e) (f) (g)

A4: (a) (b) (c) (d) (f) (g)

A5: (a) (c) (d)

8. Determine the time complexity of following code pieces (Figure 1), using big-O notation (every single statement has O(1) time complexity).

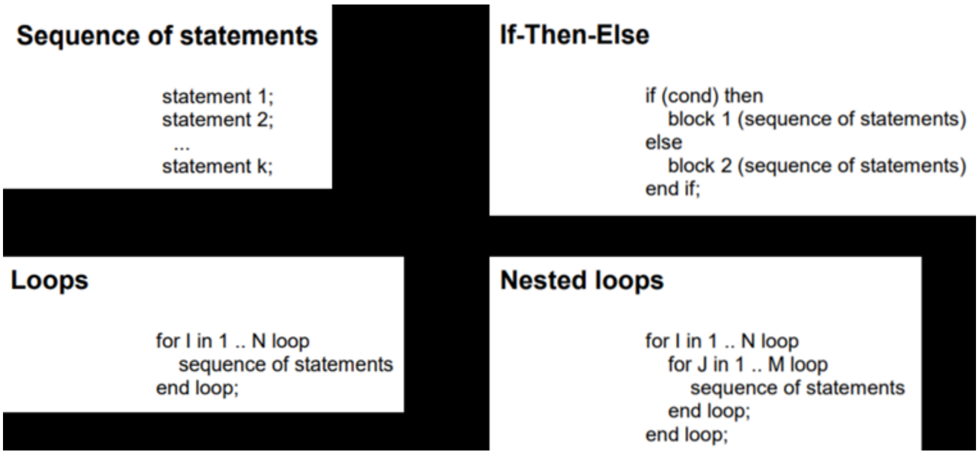


Figure 1

Solution:

Sequence of statements: O(1) or O(k)

If-else: O(1) or O(k)

Loops: O(N) or O(Nk)

Nested loops: O(NM) or O(NMk)

9. Analyze the time complexity of following code pieces (Figure 2), using big-O notation (assume ‘A’ in the following code is an integer array).

```
low = 0
high = N - 1
while (low <= high) {
    // invariants: value > A[i] for all i < low
                   value < A[i] for all i > high
    mid = (low + high) / 2
    if (A[mid] > value)
        high = mid - 1
    else if (A[mid] < value)
        low = mid + 1
    else
        return mid
```

Figure2: Code

Solution: O(logN)