ICCS313: Assignment 2

Natthakan Euaumpon
natthakaneuaumpon@gmail.com

October 2019

1: Part1

(a)
$2T(\frac{n}{3}) + 1$
$a = 2, b = 3, d = 0$
$log_b a = log_3 9$
$log_3 9 > 0$
$T(n) = O(n^{log_3 2})$

(b)
$5T(\frac{n}{4}) + n$
$a = 5, b = 4, d = 1$
$logba = log45$
$log45 > 1$
$T(n) = O(n^{log_4 5})$

(c)
$7T(\frac{n}{7}) + n$
$a = 7, b = 7, d = 1$
$log_b a = log_7 7$
$1 = 1$
$T(n) = O(nlogn)$

(d)
$9T(\frac{n}{3}) + n^2$
$a = 9, b = 3, d = 2$
$log_b a = log_3 9$
$2 = 2$
$T(n) = O(n^2 logn)$

(e)
$8T(\frac{n}{2}) + n^3$
$a = 8, b = 2, d = 3$
$log_b a = log_2 8$
$3 = 3$
$T(n) = O(n^3 logn)$

(f)
$T(n - 1) + 2$
$(T(n - 2) + 2) + 2$
$(T(n - 3) + 2) + 2 + 2$
$(T(n - 4) + 2) + 2 + 2 + 2$
.
.
.

Natthakan Euaumpon
natthakaneuaumpon@gmail.com

$(T(0) + 2) + 2(n - 1)$
T(n-1) run n times so it is $2n$
$T(n) = O(n)$

(g)
$T(n - 1) + n^c$
$= (T(n - 2) + (n - 1)^c) + n^c$
$= (T(n - 3) + (n - 2)^c) + (n - 1)^c + n^c$
.
.
.
$= (T(0) + (n - (n + 1))^c) + 2^c + ... + n^c$
$= T(0) + 1^c + 2^c + ... + n^c$
$T(n) = 1^c + 2^c + ... + n^c, \forall n \geq 1$
$T(n) = n^{c+1}, \forall n \geq 1$
$T(n) \leq n^{c+1} whenc = 1and \forall n \geq n_0 = 1$
$T(n) = O(n^{c+1})$

(h)
$T(n - 1) + c^n$
$= (T(n - 2) + c^{n-1}) + c^n$
$= (T(n - 3) + c^{n-2}) + c^{n-1} + c^n$
.
.
.
$= (T(0) + c^{n(n-1)}) + c^2 + ... + c^n$
$= T(0) + c^1 + c^2 + ... + c^n$
$T(n) = c^1 + c^2 + ... + c^n$
$cT(n) = c^2 + c^3 + ... + c^{n+1}$
$cT(n) - T(n) = (c^2 + c^3 + ... + c^{n+1}) - (c^1 + c^2 + ... + c^n)$
$T(n) \cdot (c - 1) = c^{n+1} - c$
$T(n) = \frac{c \cdot (c^n - 1)}{c - 1}$
$\lim_{n \to \infty} \frac{\frac{c \cdot (c^n - 1)}{c - 1}}{c^n}$
$\lim_{n \to \infty} \frac{c \cdot (c^n - 1)}{c^n \cdot (c - 1)}$
$\lim_{n \to \infty} \frac{c^{n+1} - c}{c^{n+1} = c^n}$
$= 1$
$T(n) \in \Theta(c^n)$, therefore $T(n) \in O(c^n)$

(i)
$2T(n - 1) + 1$
$= 2(2T(n - 2) + 1) + 1$
$= 2(2(T(n - 3) + 1)) + 2 + 1$
.
.
.
$= T(0) + 1 + 2 + 4 + ... + 2^{n-1}$
$T(n) = 1 + 2 + 4 + ... + 2^{n-1}$
$2T(n) = 2 + 4 + ... + 2^n$
$2T(n) - T(n) = (2 + 4 + ... + 2^n) - (1 + 2 + 4 + ... + 2^{n-1})$
$T(n) = 2^n - 1$

Natthakan Euaumpon
natthakaneuaumpon@gmail.com

$\lim_{n\to\infty} \frac{2^n}{2^n}$
$= 1$
$T(n) \in \Theta(2^n)$, therefore $T(n) \in O(2^n)$

(j)

$T(n^{\frac{1}{2}}) + 1$
$= (T(n^{\frac{1}{4}}) + 1) + 1$
$= (T(n^{\frac{1}{8}}) + 1) + 1 + 1$
.
.
.
$= T(n^{\frac{1}{2^k}}) + k$
n need to be greater than or equal to 2 to get $T(0)$.
So, $2 = n^{\frac{1}{2^k}}$
$\log_n 2 \frac{1}{2^k}$
$2^k = \frac{1}{\log_n 2}$
$k = \log_2 \frac{1}{log_n 2}$
$k = \log_2 \log_2 n$
$T(n) = \log_2 \log_2 n$
$\lim_{n\to\infty} \frac{\log_2 \log_2 n}{\log_2 \log_2 n}$
$= 1$
$T(n) \in \Theta(\log_2^2 n)$, therefore $T(n) \in O(\log_2^2 n)$

2: Part2

Algorithm A:
$T(n) = 3T(\frac{b}{3}) + O(n)$
By using master theorem:
$a = 3, b = 3, d = 1$
$\log_b a = log_3 3 = 1$
$1 = 1$
$= O(n \log n)$ Algorithm B:
$T(n) = T(n-1) + O(n)$
$T(n-1) + O(n)$
$= (T(n-2) + O(n)) + O(n)$
.
.
.
$T(n) = O(n^2)$
Algorithm C:
$T(n) = 2T(\frac{n}{3}) + O(n^2)$
By using master theorem:
$\log_b a = \log_3 2$
$\log_3 2 < 2$
$= O(n^2)$
Algorithm D:
$T(n) = 5T(\frac{n}{4}) + O(n)$
By using master theorem:
$\log_b a = \log_4 5$

Natthakan Euaumpon
natthakaneuaumpon@gmail.com

$\log_4 5 > 1$
$= O(n^{\log_4 5})$
We should use algorithm d as it has lowest upper bound of time complexity.