Algorithms Data Structures 2018/19 Coursework

rwcj49

Q4

(a)

$2x^4$ is $\mathcal{O}(x^3 + 3x + 2) \rightarrow$ *False*

PROOF:

Simplify $\mathcal{O}(x^3 + 3x + 2)$ to $\mathcal{O}(x^3)$, ignoring low order terms

For $x > 0, \quad x^4 \geq x^3$

There exists no $k > 0$ & $c > 0$ such that $x^4 \leq c \cdot x^3$, when $x \geq k$
Therefore the statement is false.

(b)

$4x^3 + 2x^2 \cdot logx + 1$ is $\mathcal{O}(x^3) \rightarrow$ *True*

PROOF:

Can rewrite as:

$$f_1(x) + f_2(x) + f_3(x)$$

Where:
$f_1(x) = 4x^3$ is $\mathcal{O}(x^3)$, taking $c = 4$ and any $k > 0$
$f_2(x) = 2x^2 \cdot \log x$ is $\mathcal{O}(x^2 \cdot \log x)$, taking $c = 2$ and any $k > 0$
$f_3(x) = 1$ is $\mathcal{O}(1)$, taking $c = 1$ and any k
Therefore, by sum rule, $4x^3 + 2x^2 \cdot logx + 1$ is $\mathcal{O}(x^3)$ and so the statement is true

(c)

$3x^2 + 7x + 1$ is $\omega(x \cdot logx) \rightarrow$ ***True***

PROOF:

$$3x^2 + 7x + 1 = \omega(x \cdot logx) \implies x \cdot logx = o(3x^2 + 7x + 1)$$

$$\lim_{x\to\infty} \frac{x \cdot \log x}{3x^2 + 7x + 1} = \lim_{x\to\infty} \frac{\frac{\log x}{x}}{3 + \frac{7}{x} + \frac{1}{x^2}} = \frac{0}{3} = 0$$

Therefore, $x \cdot logx = o(3x^2 + 7x + 1)$ is true, and so too is the original statement of $3x^2 + 7x + 1 = \omega(x \cdot logx)$

(d)

$x^2 + 4x$ is $\Omega(x \cdot logx) \rightarrow$ ***True***

PROOF:

For $k = 2, c = 1$:

$x^2 + 4x \geq c \cdot x \cdot \log x$
$= 2^2 + 4(2) \geq 1 \cdot 2 \cdot 1$
$12 \geq 2$
This is true and therefore so too is the original statement.

(e)

$f(x) + g(x)$ is $\Theta(f(x) \cdot g(x)) \rightarrow$ ***True***

PROOF:
$deg(f(x) + g(x)) \leq deg(f(x)g(x))$

Q5

(a)

$T(n) = 9T(n/3) + n^2$
$\implies T(n) = \Theta(n^2 \log n)$ (Case 2)

(b)

$T(n) = 4T(n/2) + 100n$
$\implies T(n) = \Theta(n^2)$ (Case 1)

(c)

$T(n) = 2^n T(n/2) + n^3$
Cannot use Master Theorem:
To use Master Theorem, recurrence must be of the form:

$$T(n) = aT(n/a) + f(n)$$

Where $a \geq 1$ and $b \geq 1$
Here a is 2^n, not a constant .

(d)

$T(n) = 3T(n/3) + c \cdot n$
$\implies T(n) = \Theta(n^2 \log n)$ (Case 2)

(e)

$T(n) = 0.99T(n/7) + 1/(n^2)$
Cannot use Master Theorem:
To use Master Theorem, recurrence must be of the form:

$$T(n) = aT(n/a) + f(n)$$

Where $a \geq 1$ and $b \geq 1$
$a < 1$ therefore not compatible with Master Theorem.

Q6

(a)

See rwcj49_q6a.py

(b)

MergeSort has worst complexity of $\mathcal{O}(n \cdot \log n)$
SelectionSort has worst complexity $\mathcal{O}(n^2)$

While Selectionsort has worse complexity than MergeSort for large arrays, in this algorithm, SelectionSort only ever acts on small arrays (length ≤ 4), and so the overall worst case input will be that which gives highest complexity for MergeSort.
This worst case MergeSort is that which involves most comparisons
e.g.

$$A = [8, 16, 4, 12, 6, 14, 2, 10, 7, 15, 3, 11, 5, 13, 1, 9]$$

This array will require swaps at every possible stage, giving the highest possible complexity for an array of this length.

(c)

See rwcj49_q6c.py