CSP2348/CSP5243 Data Structures

Tutorial 02 solutions: Fundamentals of Algorithm Analysis

Task 1: Create a spreadsheet to show the growth rates given in the following table. Observe the differences among these growth rates.

n	1	$\log(n)$	n	$n \times \log(n)$	n^2	n^3	n^{10}	2^n
1	1	0	1	0	1	1	1	2
2	1	1	2	2	4	8	1024	4
4	1	2	4	8	16	64	1048576	16
8	1	3	8	24	64	512	1073741824	256
10	1	3.32	1E1	33.22	1E2	1E3	1E10	1024
20	1	4.32	2E1	86.44	4E2	8E3	1024E10	1048576
40	1	5.32	4E1	212.88	16E2	64E3	1048576E10	1099511627776
80	1	6.32	8E1	505.75	64E2	512E3	1073741824E10	1.2089258196146292E24
100	1	6.64	1E2	664.39	1E4	1E6	1E20	1.2676506002282294E30
200	1	7.64	2E2	1528.77	4E4	8E6	1024E20	1.6069380442589903E60
400	1	8.64	4E2	3457.54	16E4	64E6	1048576E20	2.5822498780869086E120
800	1	9.64	8E2	7715.08	64E4	512E6	1073741824E20	6.668014432879854E240
1000	1	9.97	1E3	9965.78	1E6	1E9	1E30	1.0715086071862673E301
where mEn $\Rightarrow m \times 10^n$								

Task 2: Suppose that the following expressions are the sum of characteristic operations of some algorithms; determine the time complexity of each of these expressions by means of the big-*O* notation.

$n^{10} + 9 \times n^9 + 20 \times n^8$

$\Rightarrow \max\{O(n^{10}), O(9 \times n^9), O(20 \times n^8)\}$ // by Rule1 (see lecture slide 26)

$\Rightarrow \max\{O(n^{10}), O(n^9), O(n^8)\}$ // by normalizing constant factor

$\Rightarrow O(n^{10})$

$(n+1)^4 = n^4 + 4 \times n^3 + 6 \times n^2 + 4 \times n^1 + 1$

$\Rightarrow \max\{O(n^4), O(4 \times n^3), O(6 \times n^2), O(4 \times n^1), O(1)\}$ // by Rule1

$\Rightarrow \max\{O(n^4), O(n^3), O(n^2), O(n^1), O(1)\}$ // by normalizing constant factor

$\Rightarrow O(n^4)$

Or alternatively,

O((n+1)^4) = O((n+1)* (n+1)* (n+1)* (n+1))

=O(n+1) *O(n+1)* O(n+1)* O(n+1) // by Rule2

= O(n) *O(n)* O(n)* O(n)

= O(n* n* n * n) // by Rule2, reversely

=O(n^4)

$$\left(n^2+n\right)^2 = n^4 + 2\times n^3 + n^2$$
$$\Rightarrow \max\left\{O\left(n^4\right), O\left(2\times n^3\right), O\left(n^2\right)\right\}$$
$$\Rightarrow \max\left\{O\left(n^4\right), O\left(n^3\right), O\left(n^2\right)\right\}$$
$$\Rightarrow O\left(n^4\right)$$

Or alternatively,

$O((n^2+n)^2) = O((n^2+n) * (n^2+n))$ // by Rule2

$= O(n^2+n) * O(n^2+n)$

$= O(n^2) * O(n^2)$

$= O(n^2 * n^2) = O(n^4)$

$$n + 0.001\times n^3$$
$$\Rightarrow \max\left\{O(n), O\left(0.001\times n^3\right)\right\}$$
$$\Rightarrow \max\left\{O(n), O\left(n^3\right)\right\}$$
$$\Rightarrow O\left(n^3\right)$$

$$n^3 - 1000\times n^2$$
$$\Rightarrow \max\left\{O\left(n^3\right), O\left(1000\times n^2\right)\right\}$$
$$\Rightarrow \max\left\{O\left(n^3\right), O\left(n^2\right)\right\}$$
$$\Rightarrow O\left(n^3\right)$$

$$n + \log_2(n)$$
$$\Rightarrow \max\left\{O(n), O(\log_2(n))\right\}$$
$$\Rightarrow O(n)$$

$$n^2 + n\times \log_2(n)$$
$$\Rightarrow \max\left\{O\left(n^2\right), O(n\times \log_2(n))\right\}$$
$$\Rightarrow O\left(n^2\right)$$

$$2^n + n^2$$
$$\Rightarrow \max\left\{O\left(2^n\right), O\left(n^2\right)\right\}$$
$$\Rightarrow O\left(2^n\right)$$

$$\left(n^3 + 2\times n\right)/(n+5) = \frac{n^3}{n+5} + \frac{2\times n}{n+5}$$
$$\Rightarrow \max\left\{O\left(\frac{n^3}{n+5}\right), O\left(\frac{lessSignificantPowersOf\ n}{n+5}\right)\right\}$$
$$\Rightarrow O\left(\frac{n^3}{n+5}\right) \Rightarrow O\left(n^2 - \frac{5\times n^2}{n+5}\right)$$
$$\Rightarrow \max\left\{O\left(n^2\right), O(otherLessSignificantPowersOf\ n)\right\}$$
$$\Rightarrow O\left(n^2\right)$$

Or alternatively,

$O((n^3+2\times n) / (n+5))$

$= O(n^3+2\times n) / O(n+5)$ // by Rule3

$= O(n^3) / O(n)$

$= O(n^3 * / n) = O(n^2)$ // by Rule3, reversely

Task 3: Analyse the time complexity of the following methods/algorithms
(Source: exercise 2.4 on page 31 of reference textbook, Java Collections (2001)) ……

MatrixAdd:

Number of additions: $n \times n = n^2$.

In the new matrix, there are n×n elements.

Each element is the result of ONE addition from matrix A and matrix B.

Number of multiplications: 0 **.**

There is no multiplication involved in constructing any new element.

Time complexity: $\max\{O(additions), O(multiplications)\} = \max\{O(n^2), O(1)\} = O(n^2)$.

MatrixMult:

Number of additions: $n \times n \times n = n^3$.

In the new matrix, there are n×n elements.

Each element is the result of n *additions from matrix A and matrix B.*

Number of multiplications: $n \times n \times n = n^3$ **.**

In the new matrix, there are n×n elements.

Each element is the result of n multiplications from matrix A and matrix B.

Time complexity: $\max\{O(additions), O(multiplications)\} = \max\{O(n^3), O(n^3)\} = O(n^3)$.