Let $T(n)$ = running time of `MATRIX_MULTIPLY`.

$$
\begin{aligned}
T(n) &= C_1(n+1) + C_2\sum_{i=1}^{n}(n-i+2) + C_3\sum_{i=1}^{n}(n-i+1) + C_4\sum_{i=1}^{n}\sum_{j=i}^{n}(j-i+2) + C_5\sum_{i=1}^{n}\sum_{j=i}^{n}(j-i+1) + C_6 \\
&= C_1(n+1) + C_2\frac{n(n+3)}{2} + C_3\frac{n(n+1)}{2} + C_4\left(\frac{n(n+1)(2n+1)}{12} + 3\frac{n(n+1)}{4}\right) \\
&\quad + C_5\left(\frac{n(n+1)(2n+1)}{12} + \frac{n(n+1)}{4}\right) + C_6 \\
&= (C_4+C_5)\frac{n^3}{6} + \left(\frac{C_2}{2} + \frac{C_3}{2} + C_4 + \frac{C_5}{2}\right)n^2 + \left(C_1 + \frac{3C_2}{2} + \frac{C_3}{2} + \frac{5C_4}{6} + \frac{C_5}{3}\right)n + (C_1 + C_6)
\end{aligned}
$$

$T(n) = O(n^3)$. (In fact, $T(n) = \Theta(n^3)$.)

Question 1 [redacted] For each function $f(n)$ and time t in the following table, determine the largest size n of a problem that can be solved in time t, assuming that the algorithm to solve the problem takes $f(n)$ microseconds.

$f(n)$	t						
	1 second	1 minute	1 hour	1 day	1 month	1 year	1 century
$\log_2 n$	2^{10^6}	$2^{6\times10^7}$	$2^{3.6\times10^9}$	$2^{8.64\times10^{10}}$	$2^{2.59\times10^{12}}$	$2^{3.11\times10^{13}}$	$2^{3.11\times10^{15}}$
$\sqrt{n}$	10^{12}	3.6×10^{15}	1.29×10^{19}	$7.46*10^{21}$	6.71×10^{24}	9.67×10^{26}	9.67×10^{30}
n	10^6	6×10^7	3.6×10^9	8.64×10^{10}	2.59×10^{12}	3.11×10^{13}	3.11×10^{15}
$n\log_2 n$	6.27×10^4	2.80×10^6	1.33×10^8	2.75×10^9	7.18×10^{10}	7.87×10^{11}	6.76×10^{13}
n^2	1.00×10^3	7.74×10^3	6×10^4	2.93×10^5	1.60×10^6	5.57×10^6	5.57×10^7
n^3	1.00×10^2	3.91×10^2	1.53×10^3	4.42×10^3	1.37×10^4	3.14×10^4	1.45×10^5
2^n	19	25	31	36	41	44	51
$n!$	9	11	12	13	15	16	17

Assumptions: 1 month has 30 days, and 1 year has 12 months. The answers would be slightly different if you make different assumptions.

I also truncated the results so that they fit in the table.

Question 2 [redacted] Rank the following functions by order of growth, i.e., $g_i = \Omega(g_{i+1})$:

- n^2
- $n!$
- n^3
- $(\log_2 n)^2$

- $\log_2(n!)$
- $n2^n$
- $2^{\log_2 n}$
- $n^{1/\log_2 n}$

Justify your answer.

The ranked list (with justification):

1. $n!$ $(= n \times (n-1) \times ... \times 2 \times 1)$
2. $n2^n$ $(= n \underbrace{\times 2 \times 2 \times ... \times 2 \times 2}_{n} < n!$ when $n \geq 6)$
3. n^3
4. n^2
5. $\log_2(n!)$ $(= \log_2 n + \log_2(n-1) + ... \log_2(1) = O(n \log n))$
6. $2^{\log_2 n}$ $(= n)$
7. $(\log_2 n)^2$ (see lecture notes)
8. $n^{1/\log_2 n}$ $(= 2 = O(1))$

Since $b > 0$, the inequality still holds when all parts are raised to the power b:

$$0 \le \left(\frac{1}{2}n\right)^b \le (n+a)^b \le (2n)^b ,$$

$$0 \le \left(\frac{1}{2}\right)^b n^b \le (n+a)^b \le 2^b n^b .$$

Thus, $c_1 = (1/2)^b$, $c_2 = 2^b$, and $n_0 = 2\,|a|$ satisfy the definition.

Solution to Exercise 3.1-3

Let the running time be $T(n)$. $T(n) \ge O(n^2)$ means that $T(n) \ge f(n)$ for some function $f(n)$ in the set $O(n^2)$. This statement holds for any running time $T(n)$, since the function $g(n) = 0$ for all n is in $O(n^2)$, and running times are always nonnegative. Thus, the statement tells us nothing about the running time.

[redacted] Question 3

$2^{n+1} = O(2^n)$, but $2^{2n} \ne O(2^n)$.

To show that $2^{n+1} = O(2^n)$, we must find constants $c, n_0 > 0$ such that

$0 \le 2^{n+1} \le c \cdot 2^n$ for all $n \ge n_0$.

Since $2^{n+1} = 2 \cdot 2^n$ for all n, we can satisfy the definition with $c = 2$ and $n_0 = 1$.

To show that $2^{2n} \ne O(2^n)$, assume there exist constants $c, n_0 > 0$ such that

$0 \le 2^{2n} \le c \cdot 2^n$ for all $n \ge n_0$.

Then $2^{2n} = 2^n \cdot 2^n \le c \cdot 2^n \Rightarrow 2^n \le c$. But no constant is greater than all 2^n, and so the assumption leads to a contradiction.

Solution to Exercise 3.1-8

$\Omega(g(n,m)) = \{f(n,m) :$ there exist positive constants c, n_0, and m_0 such that $0 \le cg(n,m) \le f(n,m)$ for all $n \ge n_0$ and $m \ge m_0\}$.

$\Theta(g(n,m)) = \{f(n,m) :$ there exist positive constants c_1, c_2, n_0, and m_0 such that $0 \le c_1 g(n,m) \le f(n,m) \le c_2 g(n,m)$ for all $n \ge n_0$ and $m \ge m_0\}$.

Question 4.

From $f(n) = \Theta(g(n))$, we have that:

$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n), \forall n > n_0$

We can choose the constants c_1 and c_2 from here and use them in the definitions of O and Ω to show that both hold.

From $f(n) = \Omega(g(n))$ and $f(n) = O(g(n))$:

$0 \leq c_3 g(n) \leq f(n), \forall n > n_1;$

$0 \leq f(n) \leq c_4 g(n), \forall n > n_2$

If we let $n_3 = max(n_1, n_2)$ and merge the two inequalities, we get:

$0 \leq c_3 g(n) \leq f(n) \leq c_4 g(n), \forall n > n_3$, which is the definition of Θ.