1. for (int i = 1; i < n^4; i *= n) {

for (int j = 0; j < 4 * n; j += 2) {

for (int k = 1; k < n; k *= 3) {

Constant number of C of specified operations

}

}

}

$T(n) = 4 * 2n * \log_3 n = 8cn\log_3 n$.

$\therefore T(n) \in O(n\log n)$.

Reference.

Textbook pg 19-20

2. $T_A(n) = cn^2$, for an array of n=100 objects

$T_A(100) = 10000c$

$2 = 10000c$

$c = 0.0002$

To process an array of n=1000000 objects,

$T_A(1000000) = c * 1000000^2$

$= 0.0002 * 1000000^2$

$= 2.0 * 10^8$

Method A will process for $2.0 * 10^8$ milliseconds.

$T_B(n) = cn\sqrt{n}$, for an array of n=100 objects

$T_B(100) = 100c\sqrt{100}$

$10 = 1000c$

$c = 0.01$

To process an array of n=1000000 objects,

$T_B(1000000) = c * 1000000\sqrt{1000000}$

$= 0.01 * 1000000000$

$= 1.0 * 10^7$

Method B will process for $1.0 * 10^7$ milliseconds.

Reference.

Textbook Exercises 1.1.1 and 1.1.2

3.

m	Equations
1	$T(n) = 5T(n/5) + 5$ with base condition $T(1) = 0$.
	solve for $T(n/5) = 5T(n/25) + 5$
2	$T(n) = 5(5T(n/25) + 5) + 5 = 25T(n/25) + 30$
	solve for $T(n/25) = 5T(n/125) + 5$
3	$T(n) = 5(5(5T(n/125) + 5) + 5) + 5 = 125T(n/25) + 155$
. . .	. . .

There is a pattern in the equations and now you can create a generalised solution using the m iterations.

General solution: $T(m) = 5^m * T(n/5^m) + \sum_{i=1}^{m} 5^i$

In this question, we're assuming $n = 5^m$ with the integer $m = \log_5 n$, (let $\log_5 n = \log n$ for simplicity)

$T(m) = 5^m * T(n/5^m) + \sum_{i=1}^{m} 5^i$

$T(\log n) = n * T(n/n) + \sum_{i=1}^{\log(n)} 5^i$

$= n * T(1) + \sum_{i=1}^{\log(n)} 5^i$ substitute $T(1) = 0$,

$= \sum_{i=1}^{\log(n)} 5^i$

$\therefore$ **$T(\log n) = \sum_{i=1}^{\log(n)} 5^i$**

Reference.

Textbook pg 31-35

https://www.youtube.com/watch?v=HzV6tY6OuBk

4. Prove that f(n) is Θ(g(n)) iff both f(n) is O(g(n)) and f(n) is Ω(g(n)).

Because it is an 'iff', We must prove for both 'if f(n) is Θ(g(n)) then f(n) is O(g(n)) and f(n) is Ω(g(n))' and 'if f(n) is O(g(n)) and f(n) is Ω(g(n)) then f(n) is Θ(g(n))'.

If f(n) is Θ(g(n)) then f(n) is O(g(n)) and f(n) is Ω(g(n)).

Let f and g be functions, such that f(n) = Θ(g(n)).

By definition of Θ, there exist two positive constants c_1 and c_2 such that $c_1g(n) \leq f(n) \leq c_2g(n)$.

We can separate this into two parts:

$f(n) \leq cg(n)$. Therefore f(n) = O(g(n)) by definition.

$cg(n) \leq f(n)$. Therefore f(n) = Ω(g(n)) by definition.

If f(n) is O(g(n)) and f(n) is Ω(g(n)) then f(n) is Θ(g(n)).

Let f and g be functions, such that f(n) = O(g(n)).

By definition of O, there exists a positive real constant, c and a positive integer n_0 such that, $cg(n) \geq f(n)$ for all $n > n_0$.

Let f and g be functions, such that f(n) = Ω(g(n)).

By definition of Ω, there exists a positive real constant, c and a positive integer n_0 such that, $cg(n) \leq f(n)$ for all $n > n_0$.

We can combine these two and get:

$c_1g(n) \leq f(n) \leq c_2g(n)$. Therefore f(n) = Θ(g(n)) by definition.

Reference.

Textbook pg 21-23

Textbook Exercise 1.3.4

http://math.stackexchange.com/questions/301196/proving-big-theta-if-and-only-if-big-o-and-big-omega

5. **Yes, it is correct.**

As you can see in figure1 below, there are three functions representing $f(n)=n$, $g(n)=n^2$ and $h(n)=n^3$. By definition of Θ, there exist two positive constants c_1 and c_2 (in this case they are both 1) such that $c_1f(n) \le g(n) \le c_2h(n)$. Therefore $T(n)$ is $\Theta(n^2)$.

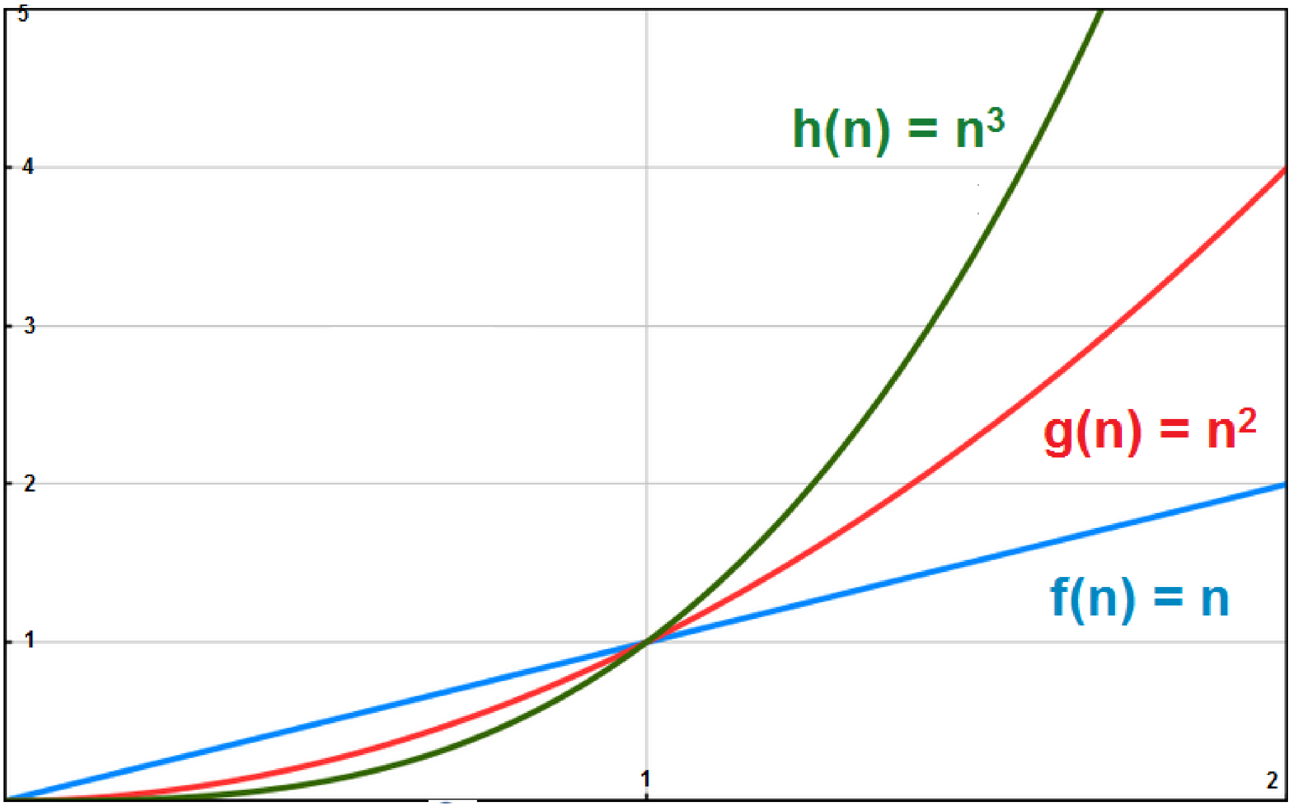

Figure1. A screenshot of a graph that shows $f(x) = x$, $f(x) = x^2$ and $f(x) = x^3$.

Reference.

Textbook pg 21-23

http://graphsketch.com/