Algorithms 4

Divij Singh

15/02/18

1

(a) $2n^2 + 6 = O(n^2)$
As there exist positive constants, c and n_0
(b) $n^2 + 1 \neq O(n^3)$
As the worst case complexity cannot be of a higher power than the highest power in the equation.
(c) $3n^2 + 5n + 6 \neq O(n)$
As the worst case power cannot be lower than the highest power in the equation.
(d) $2n + 5 = \Theta(n)$
As positive constants c_1, c_2 and n_0 exist.
(e) $2n + 5 \neq \Theta(n^2)$
As the power of $\Theta(x)$ cannot be higher than that in the equation.
(f) $2n + 5 \neq \Theta(1)$
As there is no n_0 such that $0 \leq c_1 f(n) \leq g(n) \leq c_2 f(n)$
(g) $4n^2 + 3 = \Omega(n^2)$
As there is a point such that for all $n \geq n_0$, we have $0 \leq cf(n) \leq g(n)$
(h) $3n^2 + 4 = \Omega(n)$
As there is a point such that for all $n \geq n_0$, we have $0 \leq cf(n) \leq g(n)$
(i) $5n^2 + 1 \neq \Omega(n^3)$
As there is no point such that for all $n \geq n_0$, we have $0 \leq cf(n) \leq g(n)$

2

In order for $g(n) = \Theta(f(n))$ to be true, it must be true that $f(n) = g(n)$
However, in order for $g(n) = \Omega(f(n))$ to be true, $f(n) \leq g(n)$ must be true, and in order for $g(n) = O(f(n))$ to be true, $g(n) \leq f(n)$ must be true.
Therefore it is not enough that just $g(n) = \Omega(f(n))$ and $g(n) = O(f(n))$ are true, but also that $g(n) = f(n)$

3

As b is positive, we know that $0 \leq (1/2)^b n^b \leq (n+a)^b \leq 2^b n^b$
Thus, with $c_1 = (1/2)^b$, $c_2 = 2^b$, and $n_0 = 2(a)$, we match the equation.

4

The statement is meaningless, as Big-O notation is meant to give the worst-case runtime. Therefore, the statement should be that the running time of the algorithm is *at most* $O(n^2)$

5

This statement is accurate, as we saw in the proof of question number 2.