Assignment2

Question1

Suppose you have algorithms with the five running times listed below. (Assume these are the exact running times.) How much slower do each of these algorithms get when you (a) double the input size, or (b) increase the input size by one?

- (a) n^2
- (b) n^3
- (c) $100n^2$
- (d) $n\ logn$
- (e) 2^n

INPUT SIZE	n^2	n^3	$100n^2$	$n\ logn$	2^n
n	n^2	n^3	$100n^2$	$n\ logn$	2^n
$2n$	$4n^2$	$8n^3$	$400n^2$	$2n\ log(2n)$	4^n
$n+1$	n^2+2n+1	n^3+3n^2+3n+1	$100(n^2+2n+1)$	$(n+1)log(n+1)$	2^{n+1}

Divide the running time by the data in the table to get how much slower each of these algorithms become.

Question2

Assume you have functions f and g such that $f(n)$ is $O(g(n))$. For each of the following statements, decide whether you think it is true or false and give a proof or counterexample.

- (a) $log_2 f(n)$ is $O(log_2 g(n))$
- (b) $2^{f(n)}$ is $O(2^{g(n)})$
- (c) $f(n)^2$ is $O(g(n)^2)$

We have known that $f(n) \leq c \cdot g(n)$ for some $c > 0$ and $n \geq n_0$

(a) False

- If $g(n) = 1$ for all n, $f(n) = 2$ for all n
- Then $log_2 g(n) = log_2 1 = 0$, $log_2 f(n) = log_2 2 = 1$
- For this case, we cannot write $log_2 f(n)$ is $O(log_2 g(n))$

(b) False

- If $f(n) = 2n$, $g(n) = n$ for all n
- Then $2^{f(n)} = 4^n$, $2^{g(n)} = 2^n$
- In $n > 0$, we can never find a case where $4^n \leq c \cdot 2^n$
- For this case, we cannot write $2^{f(n)}$ is $O(2^{g(n)})$

(c) True

- Since $f(n) \leq c \cdot g(n)$ for some $c > 0$ and $n \geq n_0$
- We can easily find that $(f(n))^2 \leq (c \cdot g(n))^2 \rightarrow f(n)^2 \leq c^2 g(n)^2$
- Which prove that $f(n)^2$ is $O(g(n)^2)$