算法 hw 2. 11912039 郑鑫颖.

Chapter 2 : Exercise 1 and 5

1. 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) $n \log n$

(e) 2^n

(a) double the size : $(2n)^2 = 4n^2$ slow down 4 times

increase by 1: $(n+1)^2 = n^2+2n+1$ time increase $(2n+1)$

(b) $(2n)^3 = 8n^3$ slow down by 8 times.

$(n+1)^3 = n^3+3n^2+3n+1$ time increases $(3n^2+3n+1)$

(c) $100(2n)^2 = 400n^2$ slow down 4 times.

$100(n+1)^2 = 100(n^2+2n+1) = 100n^2+200n+10$ time increase $(200n+10)$

(d) $2n \log 2n = 2n(\log 2 + \log n) = 2n\log 2 + 2n\log n$

time increases by $2n\log 2 + n\log n$

time increases

$(n+1)\log(n+1) - n\log n = \log(n+1) + n[\log(n+1) - \log n]$

(e) $2^{2n} = (2^n)^2$ time becomes square of the original time

$2^{n+1} = 2 \cdot 2^n$ time doubles.

5. 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 $O(g(n)^2)$.

$f(n) = O(g(n)) \Rightarrow$ there exist C_1、C_2.

for $n \geq C_2$ $f(n) \leq C_1 g(n)$

(a) False assume $g(n)=1$ $f(n)=2$ obviously 2 is $O(1)$

but $\log_2 f(n) = 1$ $\log_2 g(n) = 0$. So we cannot find

C_1, C_2, such that for $n \geq C_2$ $1 \leq 0 \cdot C_1$

(b) False suppose $f(n) = kn$ $k \geq 2$ $g(n) = n$. not constant

2^{kn} compare with 2^n $2^{kn} = 2^n \cdot 2^{(k-1)n}$

there's no C_1, C_2 such that for $n \geq C_2$ $2^{kn} \leq 2^n$

(c) True since $f(n) \leq C_1 g(n)$ for all $n \geq C_2$

so $f^2(n) \leq C_1^2 g^2(n)$ for all $n \geq C_2$.

So $f(n)^2 = O(g(n)^2)$