Practice M2

Tuesday, October 5, 2021 9:25 PM

2-1, 2-8, 2-23

2-1 What value is returned by the following function? Express your answer as a function of n. Give the worst-case running time using the Big-O notation.

```
function mystery (n)
    r := 0
    for i := 1 to n-1 do
        for j := i+1 to n do
            for k := 1 to j do
                r := r+1
    return (r)
```

$$\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\sum_{k=1}^{j} 1 = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} j$$

$$= \sum_{i=1}^{n-1}\left(\sum_{j=1}^{n} j - \sum_{j=1}^{i} j\right)$$

$$= \sum_{i=1}^{n-1}\left[\frac{n(n+1)}{2} - \frac{i(i+1)}{2}\right]$$

$$= \frac{1}{2}\sum_{i=1}^{n-1}\left(n^2+n-i^2-i\right)$$

$$= \frac{1}{2}\left[(n-1)n^2 + (n-1)n - \left[\frac{n(n+1)(2n+1)}{6} - n^2\right] - \left(\frac{n(n+1)}{2} - n\right)\right]$$

$$\boxed{= \frac{n(n(n+1))}{2} - \frac{n(n+1)(2n+1)}{12} - \frac{n(n+1)}{4}}$$

$= 0$ when $n = 1$

$\Rightarrow$ Big-O notation is $\boxed{O(n^3)}$

2-8 For each of the following pairs of functions, either $f(n)$ is in $O(g(n))$, $f(n)$ is in $\Omega(g(n))$, or $f(n) = \Theta(g(n))$. Determine which relationship is correct & explain why.

a) $f(n) = \log n^2$; $g(n) = \log n + 5$

- $\log n^2 = 2\log n$

$2\log n \leq 2\log n + 10$

$2\log n \leq 2(\log n + 5)$

$\Rightarrow \boxed{\log n^2 = O(\log n + 5)}$ (1)

- $\log n + 5 \leq \log n + 5\log n$

$\log n + 5 \leq 6\log n$

$\log n + 5 \leq 3 \cdot 2\log n$

$\log n + 5 \leq 3 \cdot \log n^2$

$\log n^2 \geq \frac{\log n + 5}{3}$

$\log n^2 \geq c(\log n + 5)$ $[c = \frac{1}{3}]$

$\Rightarrow \boxed{\log n^2 = \Omega(\log n + 5)}$ (2)

- From (1) & (2) $\Rightarrow \boxed{\log n^2 = \Theta(\log n + 5)}$

b) $f(n) = \sqrt{n}$; $g(n) = \log n^2$

$g(n) = \log n^2 = 2\log n$

$$\lim_{n\to\infty} \frac{\sqrt{n}}{2 \cdot \log(n)} = 2 \cdot \lim_{n\to\infty} \frac{\sqrt{n}}{\log(n)} = \infty$$

$\Rightarrow \boxed{f(n) = \Omega(g(n))}$

c) $f(n) = \log^2 n$

$g(n) = \log n$

$$\lim_{n\to\infty} \frac{\log^2(n)}{\log(n)} = \lim_{n\to\infty} \log(n) = \infty$$

$\boxed{f(n) = \Omega(g(n))}$

d) $f(n) = n$

$g(n) = \log^2 n$

$$\lim_{n\to\infty} \frac{n}{\log^2(n)} = \lim_{n\to\infty}\left[\left(\frac{\sqrt{n}}{\log(n)}\right)^2\right]$$

$$= \left(\lim_{n\to\infty} \frac{\sqrt{n}}{\log(n)}\right)^2 = \infty$$

$\Rightarrow \boxed{f(n) = \Omega(g(n))}$

e) $f(n) = n\log n + n$

$g(n) = \log n$

$$\lim_{n\to\infty} \frac{n * \log(n) + n}{\log(n)} = \lim_{n\to\infty}\left[\frac{n \cdot \log(n)}{\log(n)} + \frac{n}{\log(n)}\right]$$

$$= \lim_{n\to\infty}\left(n + \frac{n}{\log(n)}\right) = \infty$$

$\Rightarrow \boxed{f(n) = \Omega(g(n))}$

f) $f(n) = 10$

$g(n) = \log 10$

$f(n) = 10$ is a constant

$g(n) = \log 10$ is a constant

$\rightarrow f(n) = c \cdot g(n)$ or $g(n) = c \cdot f(n)$

with c being a constant

$\Rightarrow \boxed{f(n) = \Theta(g(n))}$

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

$g(n) = 10n^2$

$$\lim_{n\to\infty} \frac{2^n}{10n^2} = \frac{1}{10}\left(\lim_{n\to\infty} \frac{2^n}{n^2}\right)$$

$$= \frac{1}{10}\left(\lim_{n\to\infty} \frac{\ln(2) \cdot 2^n}{2 \cdot n}\right) \text{ L' Hopital's Rule}$$

$$= \frac{1}{10}\left(\lim_{n\to\infty} \frac{2\ln(2) \cdot 2^n}{2 \cdot n}\right) \text{ L' Hopital's Rule}$$

$$= \frac{\ln(2)}{10}\lim_{n\to\infty} 2^n = \infty$$

$\Rightarrow \boxed{f(n) = \Omega(g(n))}$

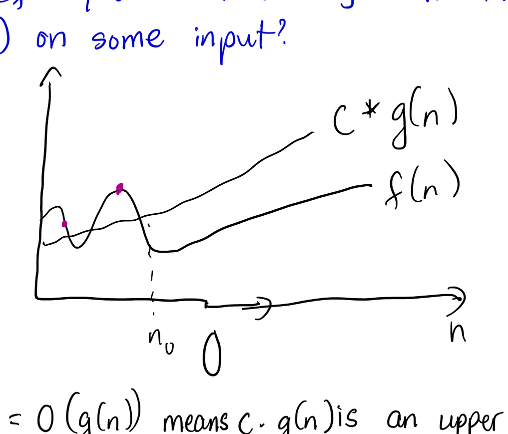
h) $f(n) = 2^n$

$g(n) = 3^n$

$$\lim_{n\to\infty} \frac{2^n}{3^n} = \lim_{n\to\infty}\left(\frac{2}{3}\right)^n = 0$$

$\boxed{f(n) = O(g(n))}$

2-23 a) If I prove that an algorithm takes $O(n^2)$ worst case time, is it possible that it takes $O(n)$ on some input?



$f(n) = O(g(n))$ means $c \cdot g(n)$ is an upper bound on $f(n)$. Thus there exists some constant c such that $f(n)$ is always $\leq c \cdot g(n)$, for large enough n (i.e., $n \geq n_0$ for some constant n_0)

$O(n^2)$ worst case means that the upper bound of the worst case is $O(n^2)$, it does not mean that all cases have the same upper bound.

$\Rightarrow$ **Yes** it is possible that it takes $O(n)$ on some input.

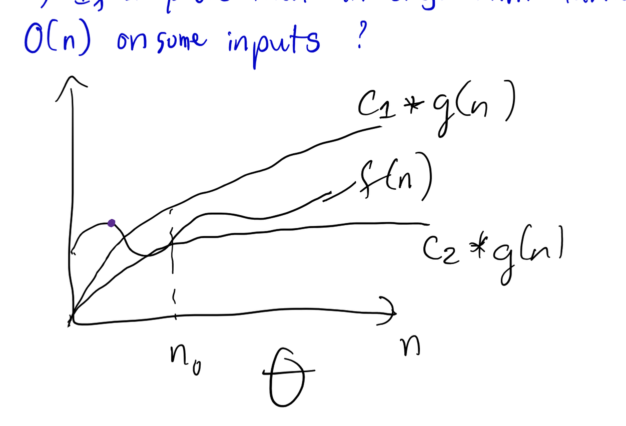
b) a) If I prove that an algorithm takes $O(n^2)$ worst case time, is it possible that it takes $O(n)$ on all inputs?

Again, $f(n) = O(g(n))$ means $c \cdot g(n)$ is an upper bound on $f(n)$. Thus there exists some constant c such that $f(n)$ is always $\leq c \cdot g(n)$, for large enough n (i.e., $n \geq n_0$ for some constant n_0)

$O(n^2)$ worst case means this is the upper bound of the worst case. This could be the upper bound of the worst case only.

Yes It is possible that it takes $O(n)$ on all inputs since $O(n)$ still follows the upper bound of $O(n^2)$.

c) If I prove that an algorithm takes $\Theta(n^2)$ worst-case time, is it possible that it takes $O(n)$ on some inputs?



Yes It is possible because again $\Theta(n^2)$ is at the worst-case. Other cases still can take $O(n)$.

d) If I prove that an algorithm takes t-case time, is it possible that it takes $O(n)$ on all inputs?

Since Θ has both the lower & upper bound, the worst-case has to follow $\Theta(n^2)$.

$\Rightarrow$ **No** all input cannot be $O(n)$.

e) Is the function $f(n) = \Theta(n^2)$, where $f(n) = 100n^2$ for even n

$f(n) = 20n^2 - n\log_2 n$ for odd n ?

Definition: $f(n)$ is $\Theta(g(n))$ if

$f(n) \leq c_1 * g(n)$ whenever $n > k$ — Big-O

and where c_1, c_2, k are possitive

$f(n) \geq c_2 * g(n)$ whenever $n > k$ — Big Ω

Even function:

$f(n) = 100n^2$ for even n

$g(n) = n^2$

$c_1 = 100$, $k = 0$:

$100n^2 \leq c_1 \cdot n^2$ whenever $n > k$

$100n^2 \leq 100 \cdot n^2$ for even n

$100 \leq 100$ whenever n is even ✓

$c_2 = 100$, $k =$ even #:

$100n^2 \geq c_2 n^2$ when n is even

$100n^2 \geq 100 \cdot n^2$ when n is even

$100 \geq 100$ ✓

$\Rightarrow f(n) = \Theta(n^2)$

Odd function:

$f(n) = 20n^2 - n\log_2 n$

$g(n) = n^2$

$c_1 = -1$, $k = 0$

$20n^2 - n\log_2 n \leq c_1 \cdot n^2$ when $n > 0$ & n is odd

$20n^2 - n\log_2 n \leq -n^2$ when $n > 0$ & n is odd ✓

$c_2 = 1$, $k = 0$

$20n^2 - n\log_2 n \leq c_2 \cdot n^2$ when $n > 0$ & n is odd

$20n^2 - n\log_2 n \leq 1 \cdot n^2$ when $n > 0$ & n is odd ✓

$\Rightarrow f(n) = \Theta(n^2)$

$\Rightarrow$ **Yes** Both even & odd functions are $\Theta(n^2)$