Albert Owusu-Asare

Algorithms Homework

Problems 2-2, 2-8 e-h, 2-14, 2-15, 2-19, 2-37

2-2) We can translate the function's "for loops" into nested summations we obtain,

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

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

recall trick from last time and observe that

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

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

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

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

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

$$= \sum_{i=1}^{n} (i^2 + i)$$

$$= \sum_{i=1}^{n} i^2 + \sum_{i=1}^{n} i$$

$= \frac{n(n+1)(n+2)}{3}$ using wolphram

we skip Algebraic manipulation and obtain the expression to the left.

big $O((n^3)/3)$

2-8)

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

$\lim_{n\to\infty} \frac{g(n)}{f(n)} = \frac{\log n}{n\log n + n} = 0$ hence $f(n)$ dominates

$f(n) = \Omega(g(n))$

f) $f(n) = 10$; $g(n) = \log 10$

$f(n) = \Theta(g(n))$ you can find a constant that bounds both ways

g) $f(n) = 2^n$; $g(n) = 10n^2$

$\lim_{n\to\infty} \frac{g(n)}{f(n)} = \frac{10n^2}{2^n} = 0$ hence $f(n)$ dominate

$f(n) = \Omega(g(n))$

(h) $f(n) = 2^n$; $g(n) = 3^n$

$\lim_{n\to\infty} \frac{g(n)}{f(n)} = \lim_{n\to\infty} \frac{3^n}{2^n} = \infty$ hence $g(n)$ dominates

$f(n) = O(g(n))$

2-14

$f_1(N) = \Omega(g_1(n))$ means that there exist some c_1 such that

$$f_1(n) \geq c_1 \cdot g_1(n)$$

Similarly

$f_2(n) = \Omega(g_2(n))$ means that there exists some c_2 such that

$$f_2(n) \geq c_2 \cdot g_1(n)$$

$$\Rightarrow \quad f_1(n) + f_2(n) \geq c_1 \cdot g_1(n) + c_2 \cdot g_2(n)$$
$$\geq (\max c_1 c_2)\, g_1(n) + (\max c_1 c_2)\, g_2(n)$$
$$\geq [(\max c_1 c_2)]\, g_1(n) + g_2(n)$$

let $(\max c_1 c_2) = c_3$

$$= c_3 [g_1(n) + g_2(n)]$$

By definition we have shown that $f_1(n) + f_2(n) = \Omega(g_1(n) + g_2(n))$

2-15

$f_1(n) = O(g_1(n))$ by definition, means there exists some c_1 such that

$$f_1(n) \leq c_1 \cdot g_1(n)$$

we also know that $f_2(n) = O(g_2(n))$ by definition means that there exist some c_2 such that

$$f_2(n) \leq c_2 \cdot g_2(n)$$

$$f_1(n) \cdot f_2(n) \leq c_1 \cdot g_1(n) \cdot c_2 g_2(n)$$
$$\leq c_1 c_2\, g_1(n) \cdot g_2(n)$$
let $c_3 = c_1 c_2$
$$= c_3\, g_1(n) \cdot g_2(n)$$

hence by definition $f_2(n) \cdot f_1(n) = O(g_1(n) \cdot g_2(n))$

6

$\log(\log n)$	n	$(1/3)^n$
$\log n$	$n \log n$	$(3/2)^n$
$\ln n$	$(\log n)^2$	2^n
$n^{1/3} + \log n$	n^2	
$\sqrt{n}$	$n^2 + \lg n$	
$\frac{1}{\log n} n$	n^3	
	$7n^5 - n^3 + n$	

2-37

For multiply two numbers x ~~and~~ and y if say we are working in base 10 the biggest possible number for y is about 10^n for an n-digit number. So we have worst case $O(b^n)$ where b is the base.

Thus we are adding b^n times but ~~to~~ 2 add any two numbers we must go digit by digit. Yes the digit by digit addition is constant time but we go through all n digits ~~2n~~

We do so 2 times because we sum the ~~we~~ two numbers and also sum the result with the next number in the repeated addition. So $2n \times y - 1$. Thus all the way to the last addition

$$2n \times b^n$$

So

$$O(n \cdot b^n)$$