Jordan Giebas

CSE 331

HW4

CSE 331 HW#4

#1) a) $n(n+1)$, $2000n^2$

Same

b) $100n^2$ & $0.01n^3$

smaller

c) $\log_2 n$ & $\log_e n$

bigger

d) $(\log_2 n)^2$ & $\log_2 n^2$

bigger

e) 2^{n-1} & 2^n

Same

CSE 331 HW#4

#3) put in order of growth Θ

$N, \sqrt{N}, n^{1.5}, n^2, n\lg n, n\lg\lg n, n\lg^2 n, n\lg(n^2), \frac{2}{n}, 2^n, 2^{\frac{n}{2}},$
$37, n^2\lg n, n^3$

when elim at

$\frac{2}{N} < 37 < \sqrt{n} < n < n\lg\lg n < n\lg n \leq N\lg(N^2) < n\lg^2 n < n^{1.5} < n^2 < n^3$
$< 2^{\frac{N}{2}} < 2^N$

something very close

#2 mystery(n) {
//n > 0 (input)
(sum) S ← 0
for i ← 1 to n do:
S ← S + i*i
return S

(a) This algo computes the sum of squares
$$\sum i^2 = \frac{n(2n+1)(n+1)}{6}$$

(b) Cost-Time Analysis?

Pseudo-Code	Cost	# of times
S ← 0	c_1	$\Theta(1)$
for i ← 1 to n do:	c_2	n
S ← S + i*i	c_3	n
return S	c_4	$\Theta(1)$

$\Rightarrow T(n) = c_1\Theta(1) + c_2 n + c_3 n + c_4\Theta(1)$

$\Rightarrow T(n) = \Theta(n)$

CSE 331 HW#4

#1) a) $n(n+1)$, $2000n^2$

Same

b) $100n^2$ & $0.01n^3$

smaller

c) $\log_2 n$ & $\log_e n$

bigger

d) $(\log_2 n)^2$ & $\log_2 n^2$

bigger

e) 2^{n-1} & 2^n

Same

CSE 331 HW#4

#3) put in order of growth Θ

$N, \sqrt{N}, n^{1.5}, n^2, n\lg n, n\lg\lg n, n\lg^2 n, n\lg(n^2), \frac{2}{n}, 2^n, 2^{\frac{n}{2}},$
$37, n^2\lg n, n^3$

when elim out

$\frac{2}{N} < 37 < \sqrt{n} < n < n\lg\lg n < n\lg n \leq N\lg(N^2) < n\lg^2 n < n^{1.5} < n^2 < n^3$
$< 2^{\frac{N}{2}} < 2^N$

something very close

#2 (sum)

```
Mystery(n) {
  //n >= 0 (input)
  S <- 0
  for i <- 1 to n do:
     S <- S + i*i
  return S
```

(a) This algo computes the sum of squares

$$\sum i^2 = \frac{n(2n+1)(n+1)}{6}$$

(b) Cost-Time Analysis?

Pseudo-Code	Cost	# of times
$S \leftarrow 0$	c_1	$\Theta(1)$
for $i \leftarrow 1$ to n do:	c_2	n
$S \leftarrow S + i*i$	c_3	n
return S	c_4	$\Theta(1)$

$\Rightarrow T(n) = c_1\Theta(1) + c_2 n + c_3 n + c_4\Theta(1)$

$\Rightarrow T(n) = \Theta(n)$

CSE 331 HW4

#4) $T(n) = \begin{cases} 1 & \text{if } n=1 \\ 4T(\frac{n}{2}) + n & \text{if } n \geq 2 \end{cases}$

$T(n) = 4T(\frac{n}{2}) + n$ 0 sub

$\Rightarrow T(n) = 4\left[4T(\frac{n}{4}) + \frac{n}{2}\right] + n$ 1 sub

$T(n) = 16T(\frac{n}{4}) + 3n$

$\Rightarrow T(n) = 16\left[4T(\frac{n}{8}) + \frac{n}{8}\right] + 3n$ 2 sub

$T(n) = 64T(\frac{n}{8}) + 5n$

pattern: powers of 2, odd #s

Let k = # of substitutions then $\exists$ a guessed formula

$T(n) = 4^{k+1} T\left(\frac{n}{2^{k+1}}\right) + (2k+1)n$

Want basecase $T(1) = 1$

so $\frac{n}{2^{k+1}} = 1 \Rightarrow \boxed{k = \lg n - 1}$

so $T(n=k) = 4^{(\lg_2 n - 1)+1} T\left(\frac{n}{2^{(\lg_2 n - 1)+1}}\right) + (2(\lg_2 n - 1) + 1)n$

$= n^2 T(1) + 2n\log_2 n - n$

$= n^2 + 2n\log_2 n - n$ $(T(1) = 1$ base case$)$

$\Rightarrow \boxed{T(n) = \Theta(n^2)}$

w/ Master Theorem

$T(n) = 4T(\frac{n}{2}) + n$

$a = 4$ $\log_2 4 = 2$ $f(n) = O(n^2)$ double ✓

$b = 2$ $\Rightarrow \boxed{T(n) = \Theta(n^{\log_b a}) = \Theta(n^2)}$ ✓

$f(n) = n$