Stephen Sallas
02/17/2021
COMP-3270

HW2

1. a. $f(n) = \theta\big(g(n)\big), O\big(g(n)\big), \Omega(g(n))$

   b. $f(n) = \theta\big(g(n)\big), O(g(n)), \Omega(g(n))$

   c. $f(n) = \Omega\big(g(n)\big)$

   d. $f(n) = \Omega\big(g(n)\big)$

   e. $f(n) = O\big(g(n)\big)$

Stephen Sallas
02/17/2021
COMP-3270

2. a. This algorithm finds the minimum integer in an array of integers.

b. $T(n) = 2T(\frac{n}{2})$

Base Case: Assume $n = 2^k$ and n=2 and k=1

$T(2) = 2$ which means the base case is true.

Hypothesis: Assume $T(2^k) = 2^k$

Induction: $T(2^{k-1})$, n=3, k=2

$T(2^{3-1}) = 2T\left(\frac{2^2}{2}\right) = 4$

Induction is proved true.

c.

Level	Level number	Total # of recursive executions at this level	Input size to each recursive execution	Work done by each recursive execution, excluding the recursive calls	Total work done by the algorithm at this level
Root	0	2	$\frac{n}{2}$	c	c
One level below root	1	4	$\frac{n}{4}$	c	2c
Two level below root	2	8	$\frac{n}{8}$	c	4c
The level just above the base case level	log(n-1)	n	1	c	$\frac{11c}{2}$
Base case level	log(n)	0	0	c	n(c)

d. The complexity of the algorithm is T(1) = c, T(n>1) = 2T(n/2) + n.

Stephen Sallas
02/17/2021
COMP-3270

3.

Level	Level number	Total # of recursive executions at this level	Input size to each recursive execution	Work done by each recursive execution, excluding the recursive calls	Total work done by the algorithm at this level
Root	0	n	n(c)	n(c)	n(c)
One level below root	1	7	$\frac{n}{8}$	$\frac{n}{8}$(c)	$n * \frac{7}{8} * c$
Two level below root	2	7^2	$\frac{n}{8^2}$	$\frac{n}{8^2}$(c)	$n * \frac{7^2}{8^2} * c$
The level just above the base case level	$\log_8 n$-1	$\frac{7 \log_8 n}{7}$	8	8(c)	$c * \frac{8}{7} * 7 \log_8 n$
Base case level	$\log_8 n$	$7 \log_8 n$	1	c	$c * 7 \log_8 n$

Stephen Sallas
02/17/2021
COMP-3270

4. Statement of what you have to prove:

$T(n) = 3T\left(\frac{n}{3}\right) + 5; T(1) = 5$

Base Case proof:

T(1) = 5 so T(1) = O(n)

Inductive Hypotheses:

T(k) = O(k) for some fixed c, and $N \ \forall \ n > N$, so T(k) < c(k)

Inductive Step:

$T(K + 1) = O(k + 1)$

$T(K + 1) < 3 * c * \frac{k}{3} + 5$

$T(K + 1) < c * k + 5$

$T(K + 1) < c(k + 1) + 5 - c$

Value of c:

If c = 5, $T(K + 1) < c(k + 1)$

∴ Passed

5. Let $s(n) = n^2 + n$, $f(n) = n^2,\ r(n) = n^2 - n,\ g(n) = n^2$.

Therefore, $f(n) = O\big(s(n)\big),\ g(n) = O(r(n)$

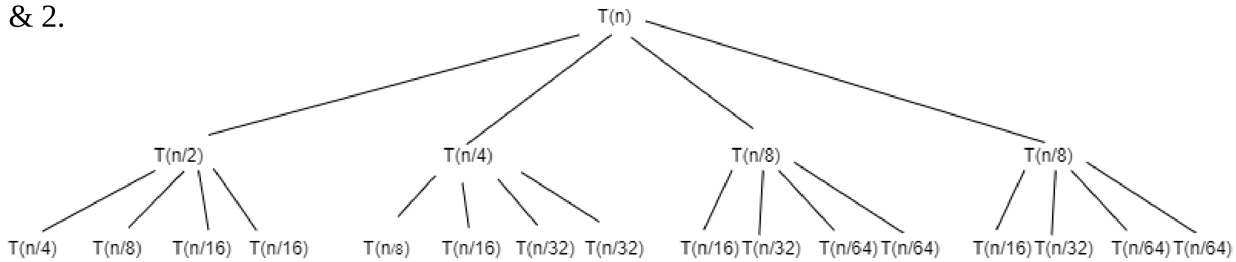
$f(n) - g(n) = n^2 - n^2 = 0 = O(1)$

$O\big(s(n) - r(n)\big) = O\big(n^2 + n - (n^2 - n)\big) = O(n)$

$\therefore f(n) - g(n) \neq O(s(n) - r(n))$

Stephen Sallas
02/17/2021
COMP-3270

6. 1. & 2.



Total work:

Level 1 = n

Level 2 = $\frac{n}{2}+\frac{n}{4}+\frac{n}{8}+\frac{n}{8}=n$

Level 3 = $\frac{n}{4}+\frac{n}{8}+\frac{n}{16}+\frac{n}{16}+\frac{n}{8}+\frac{n}{16}+\frac{n}{32}+\frac{n}{32}+2\left(\frac{n}{16}+\frac{n}{32}+\frac{n}{64}+\frac{n}{64}\right)=n$

3. The shortest branch will be the depth at the shallowest point.

$\frac{4}{8^k}=1$

$k=\log_8 n=\log_{2^3} n=\frac{1}{3}\log_2 n$

Depth = $\frac{1}{3}\log_2 n$

4. The longest branch will be the depth at the deepest point.

$\frac{4}{2^k}=1$

$\log_2 k=\log n$

$k\log_2 2=\log n$

$k=\log n$

Depth = $\log n$

5. $T(n)=O(\sum_{j=0}^{\log_2 4-1} n)$

$T(n)=O(n)-1$

$T(n)=\Omega(\sum_{j=0}^{\log_8 4-1} n)$

$T(n)=\Omega(n)-1$

$\therefore$ T(n) = O(n)

Stephen Sallas
02/17/2021
COMP-3270

7. Statement of what you have to prove:

$T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{4}\right) + T\left(\frac{n}{8}\right) + T\left(\frac{n}{8}\right) + n;\ T(1) = c$

Base Case proof:

$T(1) = c$

Inductive Hypotheses:

T(k) = O(k) for some fixed c, and $N\ \forall\ n > N$, so T(k) < c(k)

Inductive Step:

$T(k+1) = T\left(\frac{k+1}{2}\right) + T\left(\frac{k+1}{4}\right) + T\left(\frac{k+1}{8}\right) + T\left(\frac{k+1}{8}\right)$

8. a. $T(n) = 2T\left(\frac{99n}{100}\right) + 100n$

$T(n) = 2T\left(\frac{n}{\left(\frac{100}{99}\right)}\right) + 100n$

$a = 2, b = 100/99$

$n^{\log_{\frac{100}{99}} 2} > f(n) = 100n$

Time complexity = $\theta\left(n^{\log_{\frac{100}{99}} 2}\right)$[

b. $T(n) = 16T\left(\frac{n}{2}\right) + n^3 \log n$

$a = 16, b = 2$

$n^{\log_2 16} > f(n) = n^3 \log n$

Time Complexity = $\theta(n^4)$

c. $T(n) = 16T\left(\frac{n}{4}\right) + n^2$

$a = 16, b = 4$

$n^{\log_4 16} = f(n) = n^2$

Time complexity = $\theta(n^2 \log n)$

Stephen Sallas
02/17/2021
COMP-3270

9. <u>Backwards Substitution</u>

1. $T(n) = 2T(n-1) + 1$

   $T(1) = 2T(1-1) + 1 = 3$

   $T(2) = 2T(2-1) + 1 = 7$

   $T(3) = 2T(3-1) + 1 = 15$

2. $T(n) = 2T(n-1) + 1 = 2(2T(n-2) + 1) + 1$

   $T(n) = 4T(n-2) + (1+2)$

   $T(n) = 4(2T(n-3) + (1+2))$

   $T(n) = 8T(n-3) + (1+2+4)$

   $T(n) = 2^{(n-1)}T(1) + (1+2+4 \ldots + 2^{n-2})$

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

   $T(n) = 2^{n+1} - 1$

3. Base Case: $n = 0, T(0) = 2^{0+1} - 1 = 1$

   Assume: $T(k) = 2^{k+1} - 1$

   So: $T(k+1) = 2^{k+2} - 1$

   $T(k+1) = 2T(k) + 1 => 2(2k^{k+1} - 1) + 1 = 2^{k+2} - 1$

4. Complexity order = 2^n

<u>Forwards Solution</u>

1. $T(n) = 2T(n-1) + 1$

   $T(1) = 2T(1-1) + 1 = 3$

   $T(2) = 2T(2-1) + 1 = 7$

   $T(3) = 2T(3-1) + 1 = 15$

2. $x^k = \frac{x^{n+1}-1}{x-1} = \frac{2^{n+1}-1-1}{2^{n+1}-1-1}$

3. See part 1.

4. T(n) = $2^{n+1} - 1 = O(2^n)$

Stephen Sallas
02/17/2021
COMP-3270

10. $T(n) = T(n-1) + \frac{n}{2},\ T(1) = 1$

$T(n-1) = T(n-2) + \frac{n-1}{2} + \frac{n}{2}$

$T(n-2) = T(n-3) + \frac{n-2}{2} + \frac{n}{2} + \frac{n}{2}$

$T(n-3) = T(n-4) + \frac{n-3}{2} + \frac{n-2}{2} + \frac{n-1}{2} + \frac{n}{2}$

$T(n-k) = T(n-k) + (n-k+1) + (n-k+2) + \cdots +)/2$

So, $T(n) = T(1) + \frac{2+3+\cdots+n}{2}$

$T(n) = \frac{n(n+1)}{8}$

11. Let $n = 2^k$

$T(2^k) = T(k) = 2T(k-1) + k2^k$

$T(2^k) = 2(2T(k-2) + k-1)2^{k-1}) + k2^k$

$T(2^k) = 4(2T(k-3) + (k-2)2^{k-2}) + (k-1)2^k + k2^k$

$T(2^k) = 8T(k-3) + (k-2)2^k + (k-1)2^k + k2^k$

$T(k) = 2^k T(0) + k(k+1)2^{k-1}$

$T(n) = nT(1) + n(\left(\frac{\log_2(n)(1+\log_2(n))}{2}\right) = O(nlog^2 n)$

12. This is pointless because this means the algorithm can having a maximum run-time of O(n^2), but this is true for every algorithm. This algorithm could also have a run-time of anything slower than O(n^2).