Hunter McMahon

CIS 313 Written assignment 1

1. Use the def. of big-Theta to prove that $n^2 + -n + 5\sqrt{n} = \Theta(n^2)$
   ↳ Big O twice or book def.

- want to show $f(n) = \Theta(g(n))$ iff $f(n) = O(g(n)) \wedge f(n)\Omega(g(n))$ for

1. we know that $f(n) = n^2 - n + 5\sqrt{n}$
   $g(n) = n^2$
2. by def of big-theta
   $0 \le c_1(g(n)) \le f(n) \le c_2(g(n))$
   $=$
   $0 \le c_1 n^2 \le n^2 - n + 5\sqrt{n} \le c_2 n^2$
3. $c_1 n^2 \le n^2 - n + 5\sqrt{n}$ & $n^2 - n + 5\sqrt{n} \le c_2 n^2$
   by definition of big O & big Ω

$c_1 n^2 \le n^2 - n + 5\sqrt{n}$

$c_1 = 1$ $\quad 1(16) \le 16 - 4 + 5\sqrt{4}$

$n_0 = 4$ $\quad 16 \le 12 + 10$

$16 \le 22$

Therefore

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

for $n_0 = 4$ $c_1 = 1$

$n^2 - n + 5\sqrt{n} \le c_2 n^2$

$c_2 = 1$

$n_0 = 4$ $\quad 4^2 - 4 + 5\sqrt{4} \le 1(4^2)$

$12 + 10 \le 16$

$c_2 = 2$ $\quad 4^2 - 4 + 5\sqrt{4} \le 2(16)$

$n_0 = 4$ $\quad 12 + 10 \le 32$

$22 \le 32$

Thus via transpose symmetry $f(n) = O(g(n))$ for

$f(n) = \Omega(g(n))$ $\quad n_0 = 4, c_2 = 2$

4. Therefore, since $f(n)$ is big-O and big Ω of $g(n)$
   By definition of big Omega, $f(n) = n^2 - n + 5\sqrt{n} = \Theta(n^2) = g(n)$

2. Textbook excersize 3.2-1

for asymptotically nonnegative functions $F(n)$ & $g(n)$; Prove that $\max\{F(x), g(n)\} = \Theta(F(n)+g(n))$

Using the def. of big theta

1 by def. of big theta

have: $0 \le C_1(F(n)+g(n)) \le \max\{F(x), g(x)\} \le C_2(F(n)+g(n))$

Want 2 show That their exists constants $C_1, C_2, n_0 > 0$ for all $n \ge n_0$

2. given: functions are asymptotically non-negitive therefore $n_0 > 0$, $F(n) \ge 0$, $g(0) \ge 0$

3. Via def. of function maximum, we know that constant C_2 must be $1 \le$ as otherwise the statement $\max\{F(x), g(x)\} \le C_2(F(n)+g(n))$ would be false

Additionally by the same definition of function maximum, constant C_1 must be $0 \le C_1 \le 1$ as otherwise the statement $0 \le C_1(F(n)+g(n)) \le \max\{F(n), g(n)\}$ would be false

Therefore, There exist positive constants C_1, C_2 & $n_0 > 0$ as n_0 must be > 0, C_1 must be $0 \le C_1 \le 1$, and C_2 must be $1 \le$ meaning that $\max\{F(n), g(n)\} = \Theta(F(n)+g(n))$

TB 3.2-3

3. ~~[illegible]~~ is $2^{n+1} = O(2^n)$? is $2^{2n} = O(2^n)$? is $2^{2^{n+1}} = O(2^{2^n})$.

$2^{n+1} \neq O(2^n)$

$f(n) = 2^{n+1}$ by def $n + 1 > n$ $\quad -n \quad -n$ $\quad 1 > 0$ true

$g(n) = 2^n$

Therefore, the exponent of $f(n)$ will always be a higher degree than that of $g(n)$ thus $f(n)$ is not big-O of $g(n)$ as its exponent and, in turn, Rate of growth will be Larger thus $f(n) = \Omega\, g(n)$

$2^{2n} \neq O(2^n)$

$f(n) = 2^{2n}$ by def of multiplication, $2n = n + n$

$g(n) = 2^n$

$n + n > n$ $\quad -n \quad -n$ $\quad n > 0$

$f(n)$ is not big O of $g(n)$ as $f(n)$ has a higher degree exponent than $g(n)$ as $2n > n$ thus is Rate of growth will be greater therefore $f(n) = \Omega\, g(n)$

$2^{2^{n+1}} \neq O(2^{2^n})$

$f(n) = 2^{2^{n+1}}$

$g(n) = 2^{2^n}$

$2^{n+1} > 2^n$ as $n + 1 > n$ $\quad -n \quad -n$ $\quad 1 > 0$

$f(n)$ is not big O of $g(n)$ as $f(n)$ has a higher degree as $n+1 > n$ therefore it has a higher growth Rate thus $f(n)$ is actually Big omega of $g(n$

TB P 3-2

4. indicate for each pair of expressions whether A is O, o, Ω, ω, or Θ. Assume constants $k \geq 1$, $\epsilon > 0$, $c > 1$. Write answers as yes/no in each box

	A.	B.	O	o	Ω	ω	Θ	
a.	$\log^k n$	n^ϵ	yes	yes	no	no	no	b. is exponential, grows faster than log
b.	n^k	c^n	yes	yes	no	no	no	B. has a changing/growing exponent
c.	$\sqrt{n}$	$n^{\sin n}$	yes	yes	no	no	no	B.
d.	2^n	$2^{n/2}$	no	no	yes	yes	no	
e.	$n^{\log c}$	$c^{\log n}$	yes	yes	no	no	no	A. constant exponent B. growing exponent
f.	$\log(n!)$	$\log(n^n)$	no	no	yes	yes	no	

TBP 3-3 a

5. Rank the functions by order of growth such that $[g_1, \ldots g_{30}]$ satisfies $g_1 = \Omega(g_2)$ $g_2 = \Omega(g_3)$ etc. - Partition the list into $\equiv$ classes such that functions that are $f(n) = \Theta g(n)$ are in the same class

$\lg(\lg^* n)$ $\quad 2^{\lg^* n}$ $\quad (\sqrt{2})^{\lg n}$ $\quad n^2$ $\quad n!$ $\quad (\lg(n))!$ $\quad (\frac{3}{2})^n$ $\quad n^3$ $\quad \lg^2 n$ $\quad \lg(n!)$

2^{2^n} $\quad n^{1/\lg(n)}$ $\quad \ln(\ln(n))$ $\quad \lg^* n$ $\quad n \cdot 2^n$ $\quad n^{\lg(\lg(n))}$ $\quad \ln(n)$ $\quad 1$ $\quad 2^{\lg(n)}$ $\quad (\lg(n))^{\lg(n)}$

e^n $\quad 4^{\lg(n)}$ $\quad (n+1)!$ $\quad \sqrt{\lg(n)}$ $\quad \lg^*(\lg(n))$ $\quad 2^{\sqrt{2\lg(n)}}$ $\quad n$ $\quad 2^n$ $\quad n(\lg(n))$ $\quad 2^{2^{n+1}}$

Tendancies
exponents
>
factorials
>
polynomials
>
logs
>
constants

1. $2^{2^{(n+1)}}$
2. 2^{2^n}
3. $(n+1)!$
4. $n!$
5. e^n
6. $n \cdot 2^n$
7. n^3
8. 2^n
9. $(3/2)^n$
10. n^2
11. $n \log(n)$
12. $\log(n!)$
13. n
14. $4^{\log(n)}$
15. $\log(n)^{\log(n)}$
16. $n^{\log(\log(n))}$
17. $\log(x)!$
18. $\log^2(x)$
19. $2^{\log(x)}$
20. $\ln(n)$
21. $2^{\sqrt{\log(n)}}$
22. $\sqrt{2}^{\log(n)}$
23. $2^{\log^*(n)}$
24. $\ln(\ln(n))$
25. $\sqrt{\log(n)}$
26. $\log^*(n)$
27. $\log^*(\log(n))$
28. $\log(\log^*(n))$
29. $n^{1/\log(n)}$
30. 1

$\lg^*(n)$

$M/S = \frac{1}{1,000}$ a second

3600 seconds = 1 hour

$3600 / \frac{1}{1,000} = \frac{36,000}{1} \cdot \frac{1,000}{1} = 3,600,000$

6. An algorithm takes .4 ms for input size 50 (this allows us to determine constant c, which will be different in each case). how large of an input can be solved in one hour if the run time is...?

A. $C * n$

4.5×10^8 is the input size that can be handled in

$\frac{C * 50}{50} = \frac{.4}{50}$

$C = .008$

$\frac{.008 * n}{.008} = \frac{36,000,000}{.008} = \boxed{4.5 \times 10^8}$

B. $C \cdot n \log n$

$C \cdot 50 \log(50) = .4$

$\frac{C \cdot 84.94}{84.94} = \frac{.4}{84.94}$ $C = .0047$

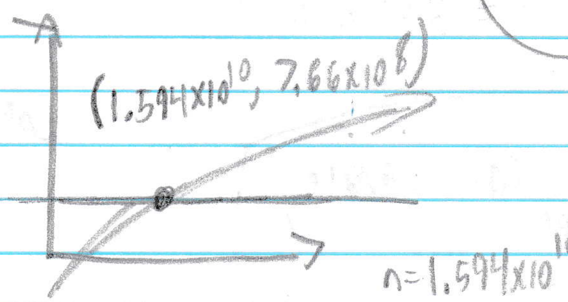
$\frac{.0047 \cdot (n \log(n))}{.0047} = \frac{3,600,000}{.0047}$

$n \log(n) = 765957446.8$

$\log(n^n) = 765957446.8$

$n^n = 10^{765957446.8}$

The resulting graph was along the lines of



got stuck so I solved by using desmos to plot the lines

$y = .0047 \cdot n \log(n)$

$y = 765957446.8$

$\boxed{\text{Thus the input size that can be solved in an hour is around } 1.594 \times 10^{10} \text{ for Runtime } C \cdot n \log n}$

c. $C \cdot n^3$

$C \cdot 50^3 = .4$

$=$

$\frac{C\,125000}{125000} = \frac{.4}{125000}$ $= C = 3.2\times10^{-6}$

$\frac{3.2\times10^{-6}\, n^3}{3.2\times10^{-6}} = \frac{3,600,000}{3.2\times10^{-6}}$

$\sqrt{n^3} = \sqrt{1.125\times10^{12}}$

$n = 10,400,410$

the Algorithm can handle 10,400 inputs per hour w/ a Run time of $C \cdot n^3$

d. $C \cdot 2^n$

$C \cdot 2^{50} = .4$

$=$

$\frac{C(1.12\times10^{15})}{\div 1.12\times10^{15}} = \frac{.4}{1.12\cdot10^{15}}$

$C = 3.55\times10^{-16}$

$\frac{3.55\times10^{-16} \cdot 2^n}{3.55\times10^{-16}} = \frac{3,600,000}{\div 3.55\times10^{-16}}$

$2^n = 1.01\times10^{22}$

$=$

$\log_2(1.01\times10^{22}) = n$

$73.10 = n$

The Algorithm can handle 73 inputs per hour w/ Run time $C \cdot n^3$