Jacob Hreshchyshyn
Assignment 1

1. Use mathematical induction to show that the solution to the recurrence:

$$T(n) = \begin{cases} 2 & \text{if } n=2 \\ 2T(\frac{n}{2})+n & n=2^k, k>1. \end{cases}$$

is $T(n)=n\lg n$

n is an even number greater than 1

Base Case:

When $n=2$: $T(2)=2$

We assume that, for some $n=k$, $T(n)=n\lg n$

Inductive step:

$(\frac{k+1}{2})\lg(\frac{k+1}{2})$

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

$T(k+1) = 2(\frac{k+1}{2})\lg(\frac{k+1}{2})+(k+1)$

$T(k+1) = (k+1)\lg(\frac{k+1}{2})+(k+1)$

$T(k+1) = (k+1)\lg(k+1)-(k+1)\lg(2)+(k+1)$

$T(k+1) = (k+1)\lg(k+1)-(k+1)+(k+1)$

$T(k+1) = (k+1)\lg(k+1)$ ✓

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

2A. Write the following expressions in minimal big-O notation

2a1. n^3+3^n: $\boxed{O(3^n)}$

2a2. $3n\log(5n)$: $\boxed{O(n\log(n))}$

2a3. $100\times2^n+3^n$: $\boxed{O(3^n)}$

2a4. $80n\log n+5n^3+\sqrt{n}$: $\boxed{O(n^3)}$

2a5. $1^3+2^3+\ldots+n^3 = \left(\frac{n(n+1)}{2}\right)^2 = \left(\frac{n^2+n}{2}\right)^2 =$

$\left(\frac{n^2}{2}+\frac{n}{2}\right)\left(\frac{n^2}{2}+\frac{n}{2}\right)$

$\frac{n^4}{4}+\frac{n^3}{4}+\frac{n^3}{4}+\frac{n^2}{4}$

$\therefore \boxed{O(n^4)}$

$\sum_{k=1}^{n} k^3$

Go up to 5

2B. Give a minimal upper bound on $f(n) = 1+2+4+\dots+2^n$. Justify your answers.

$2^0 = 1$

$2^0 + 2^1 = 3$

$2^0+2^1+2^2 = 7$

$2^0+2^1+2^2+2^3 = 15$, $f(n) = 1+2+4+\dots+2^n = 2^{n+1}-1 = 2^n \times 2^1 - 1$

$2^0+2^1+2^2+2^3+2^4 = 31$

$\therefore f(n)$ has upper bound of $\boxed{O(2^n)}$

The crux of the justification lies in proving that $1+2+4+\dots+2^n = 2^{n+1}-1$.
We use induction to do so.

Base Case

$f(0) = 2^0 = 1 = 2^{0+1} - 1$ ✓

We assume that

$f(n) = 1+2+4+\dots+2^n = 2^{n+1}-1$

Inductive Step

$f(n+1) = 1+2+4+\dots+2^n+2^{n+1} = 2^{(n+1)+1} - 1$

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

$= 2^{n+1}+2^{n+1}-1 = 2^{n+2}-1$

$= 4^{n+1}-1 = 2^{n+2}-1$

$= 2^1 \times 2^{n+1} - 1 = 2^{n+2}-1$

$= 2^{n+2}-1 = 2^{n+2}-1$ ✓

$\therefore$ Since $f(n)$ is proven to equal $2^{n+1}-1$, it's upper bound is $\underline{O(2^n)}$

$p = \lg_p m$
$m = 2^p$

3.1 How many bits are needed to write down a positive integer n? Give your answer in big-O notation, as a function of n.

$n = 2^k - 1$ is the function representing the range of positive integers possible using k bits.

The upper bound of this function is $O(2^k)$.

However, n in this case represents the integer to be represented.

To represent 0 – 1, you need 1 bit.
To represent 2 – 3, you need 2 bits
To represent 4 – 7, you need 3 bits
To represent 8 – 15, you need 4 bits
To represent 16 – 31, you need 5 bits

1, 1, 3, 7, 15

$$n = 2^k - 1$$
$$n + 1 = 2^k$$
$$\lg(n+1) = \lg(2^k)$$
$$\lg(n+1) = k$$

$$\boxed{O(\lg(n+1))}$$

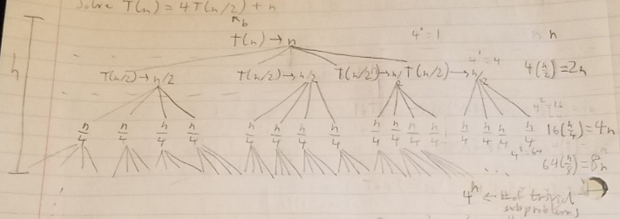
3.2 How many times does the following code print "hello"? Assume n is an integer, and that division rounds down to the nearest integer. Give your answer in big-O notation, as a function of n.

```
while n > 1:
  print "hello"
  n := n/2
```

$\rightarrow \boxed{O(\lg(n))}$

4. Use a recursion tree to determine a good asymptotic upper bound on the recurrence $T(n) = 4T(n/2) + n$. Use the substitution method to verify your answer.

Solve $T(n) = 4T(n/2) + n$



$4^h \leftarrow$ # of trivial subproblems

$\Theta(1)$

$n + 2n + 2^2 n + 2^3 n + 2^4 n$

$h = \log_b n$ *

$\therefore T(n) = 4^h \Theta(1) + \sum_{k=0}^{h-1} 2^k n$

$h = \lg n \longrightarrow h = \lg n$

$T(n) = 4^{\lg n} \Theta(1) + \sum_{k=0}^{h-1} 2^k n$

$T(n) = n^{\lg 4} \Theta(1) + \sum_{k=0}^{h-1} 2^k n$

$n^{\lg 4} = n^2$

$T(n) = n^2 \Theta(1) + n \sum_{k=0}^{h-1} 2^k$

$T(n) = n^2 \Theta(1) + n \left[\underbrace{1 + 2 + 2^2 + \dots + 2^{h-1}}_{\text{Geometric Series}} \right]$

$T(n) = n^2 \Theta(1) + n \left(\frac{1-2^h}{1-2} \right) \longrightarrow \left(\frac{1-2^h}{-1} \right) = 2^h - 1$

* $2^{\lg n} - 1$

$n - 1$

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

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

4 cont. Substitution Justification

$$T(n) = 4T(n/2) + n$$

Guessed from previous answer $O(n^2)$

Inductive hypothesis: $T(k) \le c_1 k^2 - c_2 k$ for $k < n$.

$$\begin{aligned} T(n) &= 4T(n/2) + n \\ &\le 4(c_1 (n/2)^2 - c_2 (n/2)) + n \\ &= c_1 n^2 - 2c_2 n + n \\ &= c_1 n^2 - c_2 n - (c_2 n - n) \end{aligned}$$

$$T(n) \le c_1 n^2 - c_2 n \quad \text{if } c_2 > 1$$

This is true, for example, when $c_1 = 100$.

Base case: $T(n) = O(1)$ for all $n < n_0$ where n_0 is appropriate.

For $n \le n_0$ we have $O(1) \le cn^2$, if c is large enough.

$\therefore \boxed{T(n) = O(n^2)}$

5. Write a recurrence for the running time of this recursive version of insertion sort and also solve it.

only 1 shifted key / 1 sub problem that is 1 element smaller than full problem — must traverse list in linear time before finding appropriate place to put value

Problem: Sort $A[1..n]$ = Sort $A[1..n-1]$ + insert $A[n]$ into sorted list.

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

Also, when A has only 1 element $(n=1)$, sorting is done in constant time.

$h = n$

$$n \downarrow$$
$$T(n-1) = n-1 \downarrow$$
$$T((n-1)-1) = n-2 \downarrow$$
$$n-3$$
$$\vdots$$
$$\Theta(1)$$

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

So

$$\begin{aligned} & T(1) - T(0) = 1 \\ + \; & T(2) - T(1) = 2 \\ + \; & T(3) - T(2) = 3 \\ + \; & \vdots \\ + \; & \vdots \\ + \; & T(n) - T(n-1) = n \end{aligned}$$

$$(T(1) - T(0)) + (T(2) - T(1)) + \dots + (T(n) - T(n-1)) = \frac{n(n+1)}{2}$$

$$-T(0) + T(n) = \frac{n(n+1)}{2}$$

$$T(n) = \frac{n(n+1)}{2} + \underbrace{T(0)}_{\text{let this be 1}}$$

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

Master Method applies to recurrences of $aT(n/b) + f(n)$
where $a \geq 1$, $b > 1$, and $f(n)$ is asymptotically positive (positive always)

6. For each of the following recurrences, give an expression for the runtime $T(n)$ if the recurrence can be solved with the Master Theorem. Otherwise, indicate that the Master Theorem does not apply. Justify your answer with reasoning.

6.a. $T(n) = 2T(n/2) + n^4$

$a = 2,\ b = 2,\ f(n) = n^4$

$n^{\log_b a} = n^{\log_2 2} = n$

Case 3: $f(n) = \Omega(n^{1+\varepsilon})$ for $\varepsilon = 3$

and $2(cn/2)^4 \leq cn^4$ for $c = \frac{1}{2}$

$\therefore \boxed{T(n) = \Theta(n^4)}$

6.b. $T(n) = T(7n/10) + n \longrightarrow$ Master Method does not apply because the relation does not match the required form. A factor of 7 exists in the numerator when it should not.

$a = 1,\ b = 10 \qquad f(n) = n$

$n^{\log_b a} = n^{\log_{10} 1} = 1$

6.c. $T(n) = 16T(n/4) + n^2$

$a = 16,\ b = 4,\ f(n) = n^2$

$n^{\log_b a} = n^{\log_4 16} = n^2$

Case 2: $f(n) = \Theta(n^2)$

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

6.d. $T(n) = 2T(n/4) + n^{\frac{1}{2}}$

$a = 2,\ b = 4 \qquad f(n) = n^{\frac{1}{2}}$

$n^{\log_b a} = n^{\log_4 2} = n^{\frac{1}{2}}$

Case 2: $f(n) = \Theta(n^{\frac{1}{2}})$

$\therefore \boxed{T(n) = \Theta(\sqrt{n} \lg n)}$

$4^2 = 16$

$2^{\frac{1}{2}} = \sqrt{2}$

$n^{\frac{1}{2}}$ faster growth the log n

6.e. $T(n) = \sqrt{2}\, T(n/2) + \log n$

$a = \sqrt{2}$, $b = 2$ $\quad f(n) = \log n$

$n^{\log_b a} = n^{\log_2 \sqrt{2}} = n^{\frac{1}{2}}$

Case 1: $f(n) = O(n^{\frac{1}{2} - \varepsilon})$ for $\varepsilon = \frac{1}{3}$

$\therefore \boxed{T(n) = \Theta(\sqrt{n})}$

$\left(c\frac{n}{8}\right)^2 \log\left(c\frac{n}{8}\right)$

6.f. $T(n) = 64T(n/8) - n^2\log(n)$

$a = 64$, $b = 8$ $\quad f(n) = n^2\log(n)$

$n^{\log_b a} = n^{\log_8 64} = n^2$

$f(n) = \Omega(n^{2+\varepsilon})$ when $\varepsilon = 0.1$

and $64\left(c\frac{n}{8}\right)^2 \log\left(c\frac{n}{8}\right) \leq cn^2\log(n)$ when $c = \frac{1}{2}$

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

6.g. $T(n) = 2T(n/4) + n^{0.51}$

$a = 2$, $b = 4$ $\quad f(n) = n^{0.51}$

$n^{\log_b a} = n^{\log_4 2} = n^{0.5}$

$f(n) = \Omega(n^{0.5+\varepsilon})$ when $\varepsilon = 0.01$

and $2\left(c\frac{n}{4}\right)^{0.51} \leq cn^{0.51}$ when $c = \frac{1}{2}$

$\therefore \boxed{T(n) = \Theta(n^{0.51})}$

6.h. $T(n) = 16T(n/4) + n!$

$a = 16$, $b = 4$ $\quad f(n) = n!$

$n^{\log_b a} = n^{\log_4 16} = n^2$

$f(n) = \Omega(n^{2+\varepsilon})$ when $\varepsilon = 3$

and $16\left(c\frac{n}{4}\right)! \leq cn!$ when $c = \frac{1}{2}$

$\therefore T(n) = \Theta(n!)$

6.i. $T(n) = 0.5\,T(n/2) + 1/n$

$a = 0.5,\ b = 2$ $\qquad f(n) = \frac{1}{n}$

$n^{\log_b a} = n^{\log_2 0.5} = \frac{1}{n}$

$\therefore T(n) = \Theta(\frac{1}{n} \log(n))$

However, $\frac{1}{n}$ represents negative work.

$\therefore$ Master Theorem does not apply here.

6.j. $T(n) = 2^n + (n/2) + n^n$

$a = 2^n,\ b = 2$ $\qquad f(n) = n^n$

$n^{\log_b a} = n^{\log_2 2^n} = n^n$

$\therefore \boxed{T(n) = \Theta(n^n \log(n))}$