Jonathan Rittmayer
Mat # 30001299

Algorithms & Data Structures
Assignment # 2.2

A) $36\,T(\frac{n}{6}) + 2n$ can be solved using the Master Theorem.

$a = 36$
$b = 6$ $\quad T(n) = aT(\frac{n}{6}) + f(n)$
$F(n) = 2n$

$$n^{\log_b a} = n^{\log_6 36} = n^2$$

$$f(n) = O(n^{\log_b a - ?}) \text{ For } ? = 1$$

$$\Rightarrow T(n) = \Theta(n^{\log_b a}) = \Theta(n^{\log_6 36}) = \Theta(n^2)$$

Upper Bound: $O(n^2)$
Lower Bound: $\Omega(F(n)) = \Omega(n)$

B) $5T(\frac{n}{3}) + 17n^{1.2}$ can be solved using Master Theorem.

$a = 5$
$b = 3$ $\quad T(n) = aT(\frac{n}{b}) + f(n)$
$f(n) = 17n^{1.2}$

$$n^{\log_b a} = n^{\log_3 5} = n^{1.46}$$

$$F(n) = O(n^{\log_b a - ?}) \text{ For } ? = 0.26$$

$$\Rightarrow T(n) = \Theta(n^{1.46})$$

Upper Bound: $O(n^{1.46})$
Lower Bound: $\Omega(n)$

VORWEG GEHEN

Jonathan Rittmayer
Matriculation # 30001299

Algorithms & Data Structures
Assignment 2.2

c) $12T(\frac{n}{2}) + n^2 \log n$ can be solved using the substitution Method

Induction:

Assumption: $T(n) = \Theta(n^3)$

Base case:

Because $T(n) = \Theta(1^3)$ for all n in the interval $1 \leq n \leq n_0$, where $n_0 = 2$. As such Base case is proven $\square$

Induction step:

- Assume that $T(n) \leq c_1 k^3 - c_2 k^2$ for $k < n$
- Prove that $T(n) \leq c_1 n^3 - c_2 n^2$

$T(n) = 12T(\frac{n}{2}) + n^2 \log_n$

$\Rightarrow 12\left(c_1(\frac{n}{2})^3 - c_2(\frac{n}{2})^2\right) + n^2 \log_n$

$\Rightarrow \frac{3}{2} c_1 n^3 - 6c_2 n^2 + n^2 \log n$

$\Rightarrow \frac{3}{2} c_1 n^3 - (6c_2 n^2 - n^2 \log_n)$

$\Rightarrow \frac{3}{2} c_1 n^3 - (6c_2 n^2 - n^2 \log_n) \leq c_1 n^3 - c_2 n^2$

$\Rightarrow \frac{3}{2} c_1 n^3 - n^2(6c_2 - \log_n) \leq c_1 n^3 - c_2 n^2$

$\Theta(1) \leq c_1 n^3 - c_2 n^2$ for $c_2 \geq 1$

$T(n) = \Theta(n^3)$ for $c_1 > 1$

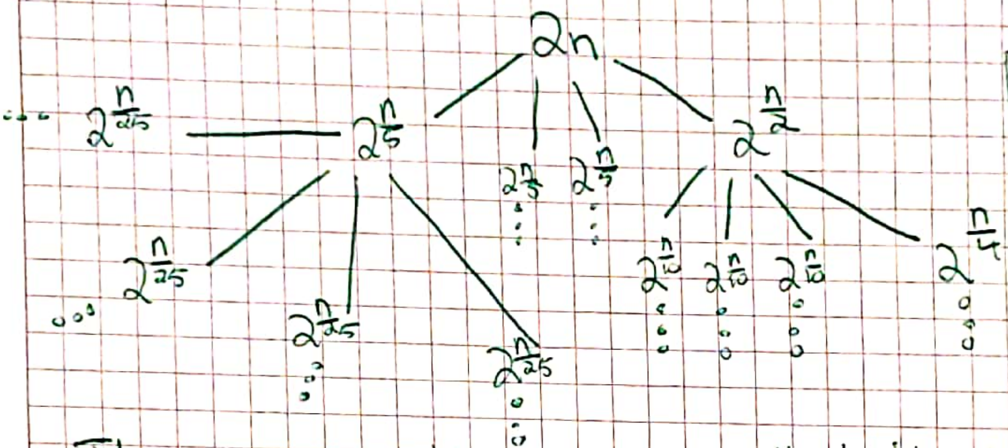
Upper Bound: $T(n) = O(n^3)$

Lower Bound: $T(n) = \Omega(n^2)$

Jonathan Rittmayer
Mat # 300012994

Algorithms & Data Structures
Assignment 2.2

D) $3T(\frac{n}{5}) + T(\frac{n}{2}) + 2^n$ can be solved using Recursion tree theorem



Cost: 2^n

$3 \cdot 2^{\frac{n}{5}} + 2^{\frac{n}{2}}$

$9 \cdot 2^{\frac{n}{25}} + 6 \cdot 2^{\frac{n}{10}} + 2^{\frac{n}{4}}$

The recursive tree grows so that the leftmost branch is the shortest and the right most is the longest. Branches in-between vary in size because they are multiples of 2 & 5.

Left most Height: $h_1 = \log_5 n$

Right most Height: $h_2 = \log_2 n$

$$\sum_{k=0}^{h_1} 3^k \cdot 2^{\frac{n}{5^k}} + \sum_{k=0}^{h_2} 2^{\frac{n}{2^k}} \Rightarrow$$

$$\Rightarrow \left(2^n + 3 \cdot 2^{\frac{n}{5}} + 9 \cdot 2^{\frac{n}{25}} + \ldots + 3^{h_1} \cdot 2^{\frac{n}{25h_1}}\right) + \left(2^n + 2^{\frac{n}{2}} + 2^{\frac{n}{4}} + \ldots 2^{\frac{n}{h_2}}\right)$$

$\Rightarrow 2^n$ (all terms)

$\Rightarrow T(n) = \Theta(2^n)$

Upper Bound: $O(2^n)$

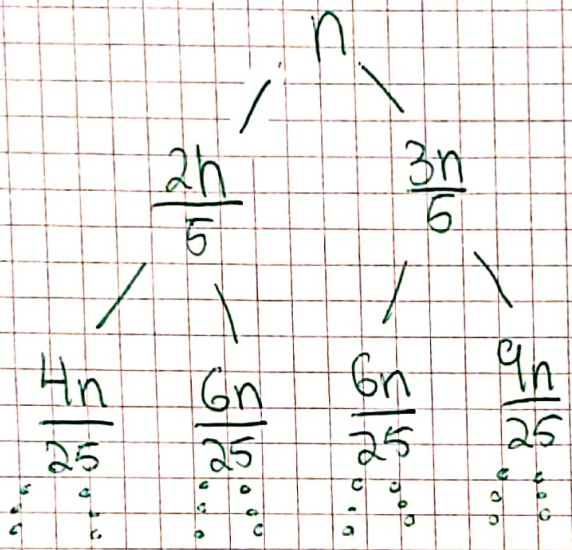
Lower Bound: $\Omega(2^n)$

Jonathan Rittmayer
Mat # 30001299

Algorithms & Data Structures

Assignment 2.2

E) $T\left(\frac{2n}{5}\right) + T\left(\frac{3n}{5}\right) + \Theta(n)$ can be solved using the recursion ~~tree~~ tree theorem



The height of the tree is $h = \log_5 3n = \log_5 3 + \log_5 n = k + \log_5 n$ where k is a constant.

$$\sum_{x=0}^{h} n = \sum_{x=0}^{h} 1 = n\frac{h(h+1)}{2} = n\frac{h^2+h}{2} = \frac{n(\log_5 n)^2 + n(\log_5 n)}{2}$$

As n approaches 0, the term $\frac{n(\log_5 n)^2}{2}$ grows faster

As such $T(n) = \Theta(n(\log_5 n)^2)$

UpperBound: $O(n\log^2_5 n)$

LowerBound: $\Omega(n\log^2_5 n)$