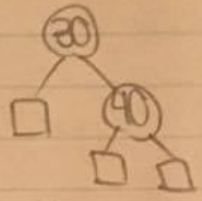
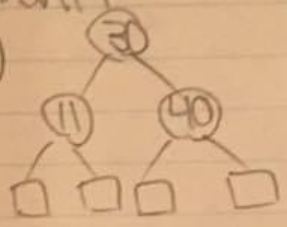
Jordan (Yu-Lin) Wang
V00786970

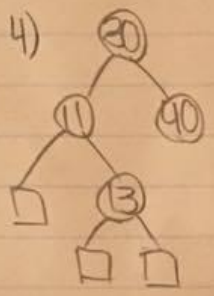
May 26th 2017 CSC226 Assignment 1

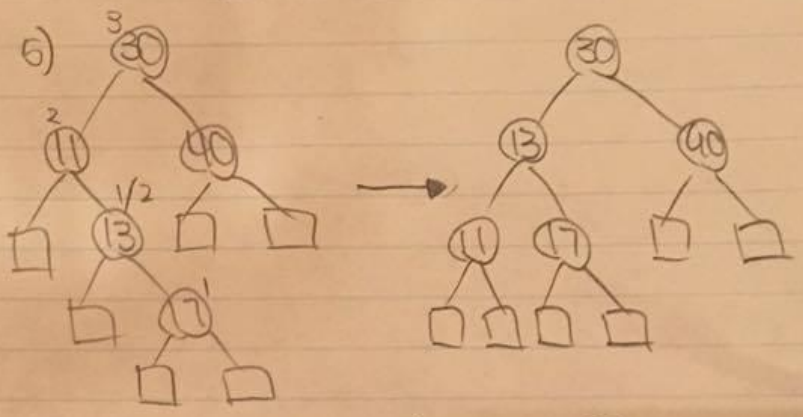
#1)

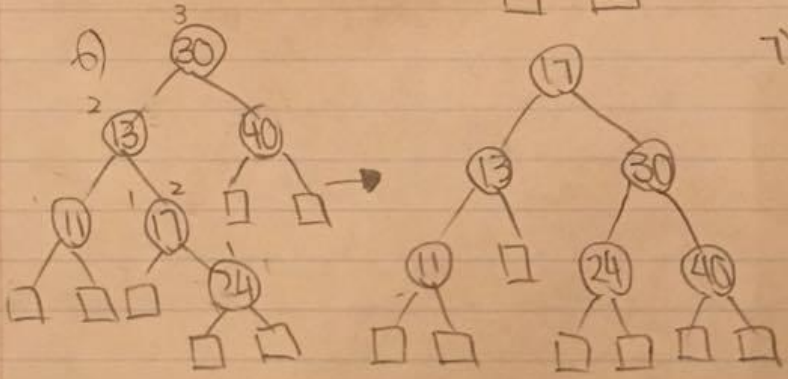
1) 

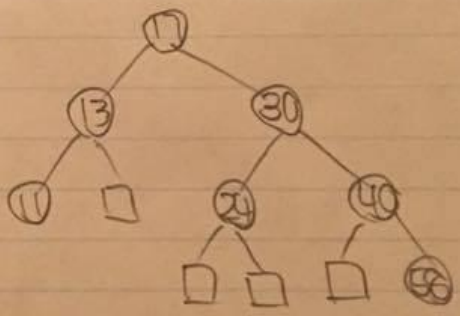
2) 

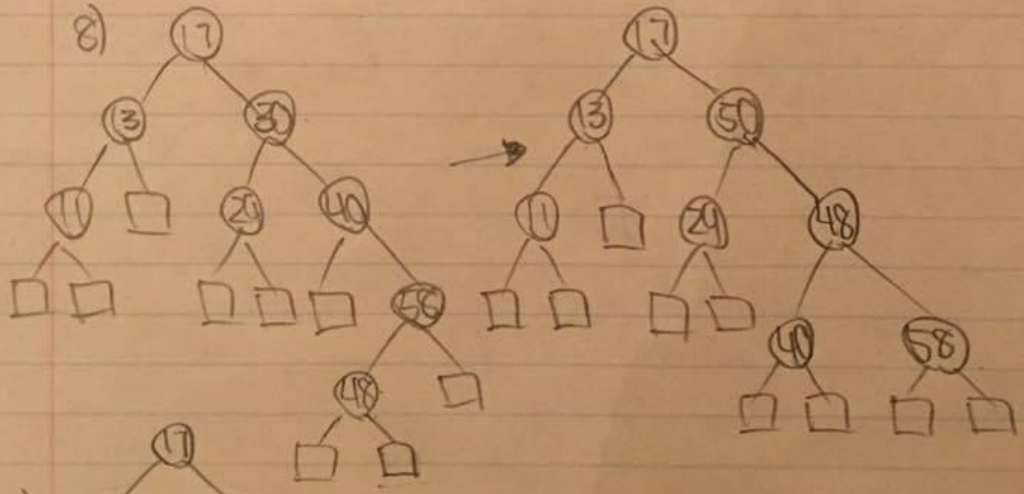
3) 

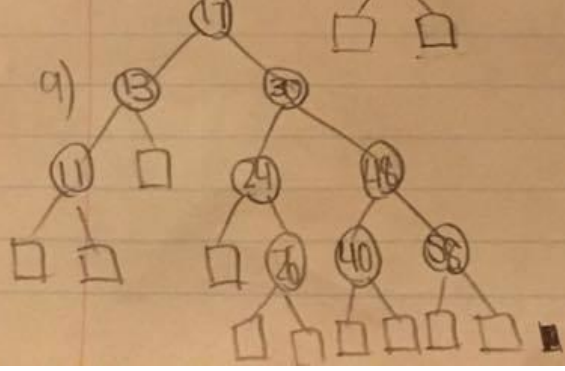
4) 

5) 

6) 

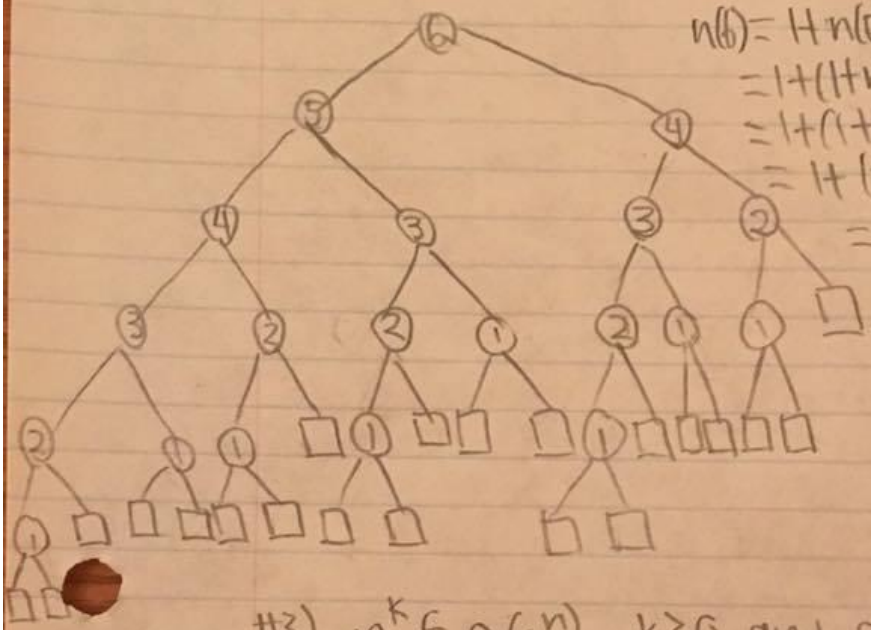
7) 

8) 

9) 

#2) minimal interval node in AVL

$$= n(h) \begin{cases} 1 & h=1 \\ 2 & h=2 \\ 1 + n(h-1) + n(h-2) & h \geq 3 \end{cases}$$



$n(6) = 1 + n(5) + n(4)$

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

$= 1 + (1 + (1 + n(3) + 2) + (1 + 2 + 1)) + (1 + (1 + 2 + 1) + 2)$

$= 1 + (3 + 1 + 2 + 1) + 5 + 7$

$= 1 + 7 + 12 = 20$

#3) $n^k \in O(a^n)$ $k > 0$ and $a > 1$

Let $f(n) = n^k$ and $g(n) = a^n$.

$f(n)$ is $o(g(n))$ if & only if $\lim_{n \to \infty} \frac{f(n)}{g(n)} = \emptyset$

L Hopital's Rule:

$$\lim_{n \to \infty} \frac{n^k}{a^n} = \lim_{n \to \infty} \frac{k \cdot n^{k-1}}{a^n \ln(a)} = \lim_{n \to \infty} \frac{k\, x^{k-2}(k-1)}{a^n \ln^2(a)} = \cdots$$

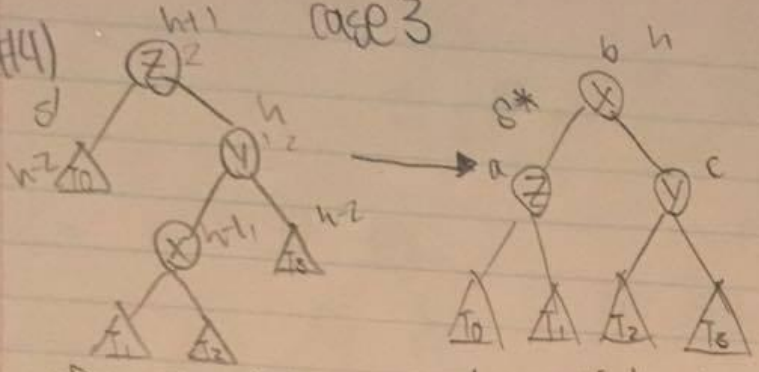
after k times $\longrightarrow \lim_{n \to \infty} \frac{k!}{a^n \ln^k(a)} = \emptyset$ ✓

Thus, n^k is $O(a^n)$

n^k is $O(a^n)$ if & only if $\exists$ a constant $c > \emptyset$ and an int $n_0 > 0$ such that $f(n) \leq c \cdot g(n)$ for all $n \geq n_0$

Therefore, n^k is $o(a^n)$

#4) case 3



- after insertion
- Z is first imbalanced
- tree rooted at node x after rebalancing

The T_0 is same tree for both of S & S', which height is $(t_3, S') = h-2$
this tells the height $(y, S') = h$ and height $(Z, S) = h+1$
since the height $(y, s) = h-1$, both height (x, s) < height $(t_3, S) \leq h-2$
since height $(y, s') = h$ and y is balanced for S'
since both of T_0 & T_3 have the height of $h-2$ in S^*
then height $(z, s^*) = h-1$ and height $(y, s^*) = h-1$.
since both Z & y have the height of $h-1$
height $(x, S^*) = h$ = height (S^*)

Thus, h = height (S^*) = height (S) ∎

#6) The mergesort is the most efficient algorithm to be used by Rustbucket to sort the music files.
Mergesort has the worst case running time of $O(n \log n)$, which is the fastest comparison based sorting
The condition of it returns -1 can be added.
By doing this, it will recomplete the comparison until it returns the value of 0 or 1.
The possibility of returning -1 is 50/50, this will increase twice the time; $2 \cdot n \log n$ to complete the sort $O(n \log n)$

↓ Constant