Algorithms and DataStructure. Huzaifa Hashim

Problemset 5.

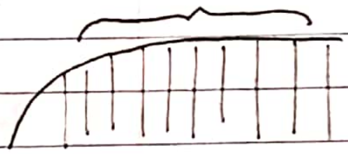
Problem 5.3 DecisionTree

A decision is binary and number of leaves $\geq$ # possible answers = $n!$

$\therefore$ height $\geq$ $(\lg n!)$ which can be proven using Stirling's Formula, Taylor Approx, or a summation

proving $(\lg n!) = \Theta(n \lg n)$ using sums.

same length



$$= \lg n!$$
$$= \lg(n \cdot (n-1) \cdot (n-2) \ldots 1)$$
$$= \lg n + \lg(n-1) + \ldots + \lg 2 + \lg 1$$
$$= \sum_{i=1}^{n} \lg i$$

$\therefore$ since we can ignore the first $n/2$ terms. $i = n/2$

$$= \sum_{i=n/2}^{n} \lg i$$
$$\geq \sum_{i=n/2}^{n} \lg n/2$$

and since each term $\geq \lg n/2$

$$\therefore \sum_{i=n/2}^{n} \lg(n-1)$$
$$\therefore \frac{n}{2} \log n - \frac{n}{2}$$

since the $(-\frac{n}{2})$ is negative and irrelevant.

$(\lg n!) = \Theta n \lg n$ $\therefore$ proven.

*note: this proof is also part of the MIT course 6006 that I took during winter break before Spring 2019 started.

Problem 3.2 Modified Quicksort.

b) Determine & prove the best & worst case for the modified quicksort Algorithm.
Assuming the modified version of QS, the best case time complexity would be achieved if an ~~of~~ input would divide into 3 equal parts and would lead to $O(n\log n)$ time complexity by the following recurrence

$$T(n) = 3T(n/3) + \underbrace{O(n)}_{\text{partition}}$$

Solving this, our best case time complexity will be the most efficient for Quick Sort & its iterations.

The worst case time complexity would be due to an unfortunate pivot selection in the quicksort which would require all possible elements being individually checked.

$$T(n) = T(n-2) + O(n)$$

solving this would result in the worst case time complexity of $O(n^2)$ for quick sort.