Q2.3) Let T(n) be the maximum number of guesses required to correctly identify a secret word that is randomly chosen from a dictionary with exactly n words. Determine a recurrence relation for T(n), explain why the recurrence relation is true, and then apply the Master Theorem to show that T(n)=Θ(logn).

Here, the algorithm is recursively calling the function by dividing the array in to two halves at each iteration in n/2 and checking the middle word. If a match occurs, then the word is guessed. If the middle word comes after the word that needs to be guessed, then the word is searched in the sub-array to the left of the middle word. Otherwise, the word is searched for in the sub-array to the right of the middle word. This process continues on the sub-array as well until the size of the subarray reduces to 1 and the word is found.

The recurrence relation for WordGuessing game algorithm would be the height of the balanced binary tree, where n is the number of nodes

$T(n) = T(n/2) + c$
Where, c = constant = O(1)

Because, after each iteration the problem size (array size n) reduces by half and we are only checking one of the subarray in each iteration depending upon the condition and not both.

$n \rightarrow n/2 \rightarrow n/4 \rightarrow n/8$ .....

Calculating recursion at each iteration we get,

$T(n) = T(n/2) + 1$
$T(n/2) = T(n/4) + 1$

Substituting in T(n),

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

Similarly,

$T(n) = T(n/8) + 3$
$T(n) = T(n/16) + 4$
.
.
.
$T(n/n) = T(n/2^k) + k$
Where, k is the number of iterations.

For simplicity, assume n is a power of 2, then T(1) will be 0. because with array size 1, we can find the right one without a comparison. Similarly, T(2) = 1 because we have just to check one comparison.

Solving for initial condition, $n/2^k = 1$

$2^k = n$
So, $k = \log(n)$

The height of the balanced binary tree, where n is the number of nodes is log(n).

So, $T(n) = \log(n) + 1$

Using Master Theorem,

$T(n) = T(n/2) + 1$

Here, a = 1 and b = 2, $f(n) = 1 = n^0$

$g(n) = n^{\log_b a} = n^{\log_2 1} = n^0 = 1$

Here, $f(n) = \Theta(n^{\log_b a}) = \Theta(n^0) = \Theta(1)$

So, using **Master's theorem case 2, as stated in the book CLRS**, it states that if $f(n) = \Theta(n^{\log_b a})$, then

$T(n) = \Theta(n^{\log_b a} \log n)$

So, here $T(n) = \Theta(n^0 \log n) = \Theta(\log n)$

Hence, $T(n) = \Theta(\log n)$ is proved.