Week12: Tutorial # 10

This week we covered searching techniques emphasizing on time complexity. I think the majority of students are clear about searching techniques but still not sure about how to estimate the running time of an algorithm.

The goal of this week tutorial is to practice more recursions and estimating the running time of a recursive algorithm.

The first problem is to find if two strings are the same. I have attached the code, which explains how to solve the problem.

To estimate the running time, the first question that I would ask students, after they solved the problem, is what is the size of the input? Remind them that the running time of an algorithm is a function of its input size (i.e `O(n)`, `O(log n)`, etc. n: input size). Hopefully they get to this answer that `n=len(input str)`. Now you can start counting the number of the operations:

- based on the given code for `same_string` function, at most `c` (constant) number of operation is executed each time the function is called. To be exact, based on the model that we introduced in the class, for this function `c = 8`. In case you are not sure how I get this number, here is how it works: every indexing into an array, each comparison, each function call, each assignment and each return statement is regarded as 1 operation. Since we have 3 comparisons and 2 indexing into the list, 1 function call, 1 assignment operation and 1 return statement therefore c = 8.
- For each character in the string, the function is called. Since the function is called n times (len(str)), therefore running time would be 8n. Now you are looking for a function for which this inequation works: 8n <= c.n ➔ if c = 8 ➔ running time is O(n)

Do the same for the rest of the functions. All of them run in O(n).

If you had time, ask students to find the running time of the recursive functions that they've seen before (i.e. binary Fibonacci recursion, Htree, etc.).