German University in Cairo
Department of Computer Science
Dr. Nourhan Ehab

CSEN 703 Analysis and Design of Algorithms, Winter Term 2022
Practice Assignment 3

Exercise 3-1 From CLRS (©MIT Press 2001)

Insertion sort can be expressed as a recursive procedure as follows:
In order to sort $A[1..n]$, we recursively sort $A[1..n-1]$ then insert $A[n]$ into the sorted array $A[1..n-1]$. Write a recurrence for the running time of this recursive version of insertion sort.

Exercise 3-2 From CLRS (©MIT Press 2001)

Use a recursion tree to determine a good asymptotic upper bound on the recurrence
$T(n) = 3T(\lfloor n/2 \rfloor) + n$.

Exercise 3-3 From CLRS (©MIT Press 2001)

Solve the following recurrence using the recursion tree method. $T(n) = 7T(n/2) + n^2$

Exercise 3-4 From CLRS (©MIT Press 2001)

Use the divide-and-conquer approach to write an algorithm that finds the largest item in a list of n items. Analyze your algorithm and get it's worst-case time complexity.

Exercise 3-5

Write a divide-and-conquer algorithm for the **Towers of Hanoi** problem. The Towers of Hanoi problem consists of three pegs and n disks of different sizes. The objective is to move the disks that are stacked on one of the three pegs (in decreasing order of their size) to a new peg using the third one as a temporary peg. The problem should be solved according to the following rules:

i when a disk is moved, it must be placed on one of the three pegs;

ii only one disk may be moved at a time, and it must be the top disk on one of the pegs; and

iii a larger disk may never be placed on top of a smaller disk.

What is the worst-case time complexity of your algorithm?

```
public class Hanoi {
    enum Tower { A, B, C };

    static void moveDisk( Tower from, Tower to ) {
        System.out.println(
            "Move disk from " + from + " to " + to
        );
    }

    public static void towers( int n, Tower x, Tower y, Tower z ) {
        if ( n > 0 ) {
            towers( n-1, x, z, y );
            moveDisk( x, y );
            towers( n-1, z, y, x );
        }
    }
}
```