293 (Towers of Hanoi) There are 3 towers and n disks. The disks are graduated in size; disk 0 is the smallest and disk n–1 is the largest. Initially tower A holds all n disks, with the largest disk on the bottom, proceeding upwards in order of size to the smallest disk on top. The task is to move all the disks from tower A to tower B, but you can move only one disk at a time, and you must never put a larger disk on top of a smaller one. In the process, you can make use of tower C as intermediate storage.

(a)√ Using the command *MoveDisk from to* to cause a robot arm to move the top disk from tower *from* to tower *to* , write a program to move all the disks from tower A to tower B.

§ see book Section 4.3

(b)√ Find the execution time, counting *MoveDisk* as time 1 , and all else free.

§ see book Section 4.3

(c) Suppose that the posts where the disks are placed are arranged in an equilateral triangle, so that the distance the arm moves each time is constant (one side of the triangle to get into position plus one side to move the disk), and not dependent on which disk is being moved. Suppose the time to move a disk varies with the weight of the disk being moved, which varies with its area, which varies with the square of its radius, which varies with the disk number. Find the execution time.

§ We prove

$$\begin{array}{ll} & \mathit{MovePile} \\ \Leftarrow & \textbf{if } n{=}0 \textbf{ then } ok \textbf{ else } n{:=}\ n{-}1.\ \mathit{MovePile}.\ \mathit{MoveDisk}.\ \mathit{MovePile}.\ n{:=}\ n{+}1 \textbf{ fi} \end{array}$$

where

$$\begin{array}{l} \mathit{MoveDisk} \;=\; t{:=}\ t + a{\times}n^2 + b{\times}n + c \\ \mathit{MovePile} \;=\; t{:=}\ t + (3{\times}a + b + c){\times}(2^n - 1) - a{\times}n^2 - (2{\times}a + b){\times}n \end{array}$$

for some constants a , b , and c . The proof is by cases. First,

$$\begin{array}{ll} & n{=}0 \wedge ok \\ = & n{=}0 \wedge (t{:=}\ t + (3{\times}a + b + c){\times}(2^n - 1) - a{\times}n^2 - (2{\times}a + b){\times}n) \\ \Rightarrow & \mathit{MovePile} \end{array}$$

Last,

$$\begin{array}{ll} & n{\ne}0 \wedge (n{:=}\ n{-}1.\ \mathit{MovePile}.\ \mathit{MoveDisk}.\ \mathit{MovePile}.\ n{:=}\ n{+}1) \\ = & n{\ne}0 \wedge (n{:=}\ n{-}1.\ \mathit{MovePile}.\ \mathit{MoveDisk}.\ \mathit{MovePile}.\ n'{=}n{+}1 \wedge t'{=}t) \\ & \hfill \text{Substitution Law 4 times} \\ = & \quad n{\ne}0 \;\wedge\; n'{=}n{-}1{+}1 \\ & \wedge\;\; t' = t + a{\times}(n{-}1)^2 + b{\times}(n{-}1) + c \\ & \qquad + 2{\times}(3{\times}a + b + c){\times}(2^{n-1} - 1) - 2{\times}a{\times}(n{-}1)^2 - 2{\times}(2{\times}a + b){\times}(n{-}1) \\ & \hfill \text{simplify} \\ = & n{\ne}0 \;\wedge\; n'{=}n \;\wedge\; t' = t + (3{\times}a + b + c){\times}(2^n - 1) - a{\times}n^2 - (2{\times}a + b){\times}n \\ \Rightarrow & \mathit{MovePile} \end{array}$$

(d)√ Find the maximum memory space required by the program, counting a recursive call as 1 location (for the return address) and all else free.

§ see book Section 4.3

(e)√ Find the average memory space required by the program, counting a recursive call as 1 location (for the return address) and all else free.

§ see book Section 4.3

(f) Find a simple upper bound on the average memory space required by the program, counting a recursive call as 1 location (for the return address) and all else free.

§ Prove the space-time product is at most $(s+n)\times(2^n-1)$ and then prove the average space is at most n .