LAMC Junior Circle January 22, 2012

Oleg Gleizer

The Hanoi Tower

Part 2

Definition 1 *An algorithm is a finite set of clear instructions to solve a problem. An algorithm is called optimal, if the solution it provides is the shortest possible.*

In this class, we shall find an optimal algorithm that solves the Hanoi tower puzzle. Consider the original arrangement with all the disks on the left rod. Let us number the disks 1 through n from the top to the bottom. Let us name the rods A, B, and C from the left to the right. Let us write down the moves in three-symbol words, first the number of the disk, then the name of the rod it is moved from, and finally the name of the rod the disk is moved to. For example, the move that shifts the second disk from rod A to rod C will be denoted as $2AC$.

Problem 1 *Using the above notations, write down the following move: disk five from rod B to rod C.*

Let us first solve the Hanoi tower puzzle with two disks. In this case, it's easy to figure out the algorithm: the first disk goes from rod A to rod B, then the second disk is moved from rod A to rod C, and finally the first disk goes from rod B to rod C. Let us call the algorithm $HT_2(AC)$. In the above notations, it can be written down as follows.

$$HT_2(AC) = 1AB\ 2AC\ 1BC \tag{1}$$

$HT_2(AC)$ not only solves the Hanoi tower puzzle with two disks, but also does it the shortest possible way and thus is optimal. (Why?)

Problem 2 *Write down the shortest algorithm, $HT_2(AB)$, that moves two disks from rod A to rod B.*

Note that the formula you get is nothing but formula 1 with letters B and C switched.

Problem 3 *Rewrite formula 1 with letters A and B switched.*

Write down the optimal algorithm, $HT_2(BC)$, that moves two disks from rod B to rod C.

Do you get the same formula? Why?

Let us consider the three-disk case. To move the third disk from rod A to rod C, we first need to move the upper two disks from rod A to rod B. This is done by the algorithm $HT_2(AB)$ from Problem 2. Further on, the move $3AC$ shifts the third disk from rod A to rod C. Finally, we need to move the first two disks from rod B to rod C. The algorithm doing just that, $HT_2(BC)$, is found in Problem 3. So here comes the algorithm for three disks:

$$HT_3(AC) = HT_2(AB)\ 3AC\ HT_2(BC) \tag{2}$$

or explicitly,

$$HT_3(AC) = 1AC\ 2AB\ 1CB\ 3AC\ 1BA\ 2BC\ 1AC \tag{3}$$

Problem 4 *Use the actual puzzle to check if the above algorithm really works for three disks.*

Problem 5 *Is the algorithm $HT_3(AC)$ optimal? Why or why not?*

Let us call the number of moves needed to solve the puzzle with n disks $N(n)$. Then $N(2) = 3$ and $N(3) = N(2) + 1 + N(2) = 3 + 1 + 3 = 7$.

Problem 6 *Write down the optimal algorithm, $HT_3(BC)$, that moves three disks from rod B to rod C.*

Problem 7 *Write down the optimal algorithm, $HT_3(AB)$, that moves three disks from rod A to rod B.*

Problem 8 *Using Problems 6 and 7 combined, write down the optimal algorithm, $HT_4(AC)$, that moves four disk from rod A to rod C.*

To see whether it really works, apply the algorithm to the actual puzzle.

Problem 9 *Find N(4).*

Problem 10 *Write down the optimal algorithm that solves the puzzle for $n = 5$.*

Apply the algorithm to the actual puzzle. Find $N(5)$.

It's time to generalize: once we know the optimal algorithm $HT_n(AC)$ to move n disks from rod A to rod C, we can construct the algorithm $HT_{n+1}(AC)$ optimal for solving the puzzle with $n+1$ disks. $HT_n(AB)$, the algorithm obtained from $HT_n(AC)$ by switching letters B and C in every three-symbol word, moves the top n disks from rod A to rod B. The next move, $(n+1)AC$, shifts the largest disk from rod A to rod C. Finally, the algorithm $HT_n(BC)$, that is $HT_n(AC)$ with letters A and B replacing each other, moves the upper n disks from rod B to rod C.

$$HT_{n+1}(AC) = HT_n(AB)\ (n+1)AC\ HT_n(BC) \tag{4}$$

$HT_{n+1}(AC)$ is optimal as well. (Why?) Its length

$$N(n+1) = N(n) + 1 + N(n) \tag{5}$$

Problem 11 *Find* $N(10)$*.*

HT_n is an example of a *recursive algorithm.* In order to solve the puzzle with n disks, we need to apply our solution procedure to the problem with $n-1$ disks.

$$HT_n(AC) = HT_{n-1}(AB)\ nAC\ HT_{n-1}(BC)$$

To solve the latter, we run the algorithm for $n-2$ disks,

$$HT_n(AC) = \underbrace{HT_{n-2}(AC)\ (n-1)AB\ HT_{n-2}(CB)}_{HT_{n-1}(AB)}\ nAC$$

$$\underbrace{HT_{n-2}(BA)\ (n-1)BC\ HT_{n-2}(AC)}_{HT_{n-1}(BC)}$$

and so forth.

Problem 12 *Find the optimal algorithm to solve the Hanoi tower puzzle with eight disks having the following initial arrangement: the eighth disk is on rod B, the seventh – on rod A, the first six disks – on rod C. How many moves will it take?*

Consider the Hanoi tower puzzle with four rods (also known as Reve's puzzle). We can easily come with the following winning algorithm for n disks.

- Choose $0 < k < n$.

- Use the optimal algorithm for three rods and k disks to move the first k disks to any rod except for the last one.

- Keeping the first k disks still, move the remaining $n-k$ disks to the last rod by using the optimal algorithm for three rods and $n-k$ disks.

- The problem is reduced to a similar one, but with k disks instead of n.

For any $k = 1, 2, \ldots, n-1$, the algorithm solves the puzzle, but is any of the above $n-1$ algorithms optimal? We don't know! The problem of finding an optimal solution for the Hanoi tower puzzle with more than three rods is still open. Quite often, a path from an elementary school problem to the frontier of the ongoing research is not too long.

Problem 13 *Using a pencil as the forth rod, solve the Hanoi tower puzzle with four rods and five disks.*

Suppose that the monks move a disk every second. Using formula 5, it is not too hard to find $N(64)$. The computation shows that the monks will need more than $500,000,000,000$ years, that is five hundred billion years, to solve the puzzle. Our world will not end in quite some time!