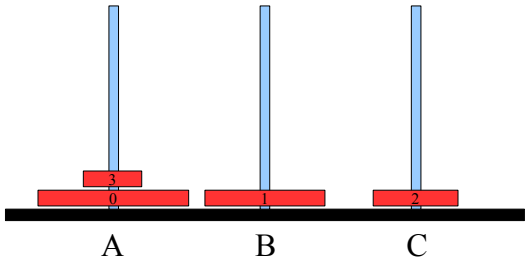
Generalized Tower of Hanoi

In the traditional Towers of Hanoi problem, we start with an initial configuration with all the disks on one tower and want to transform it to a final configuration with all the disks on a different tower. In this generalization, we will solve the problem of transforming any legal configuration into any other legal configuration.

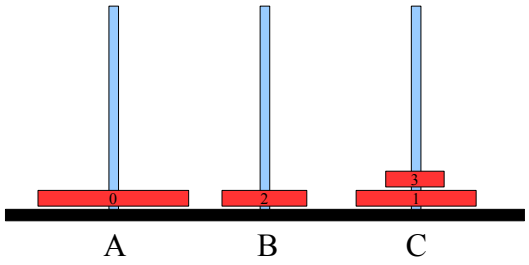
One issue is the problem representation. The traditional problem can be described by three attributes, the number of disks, the from tower and the to tower, so (3, A, B) means start with 3 disks on tower A and move them all to tower B. This description is inadequate to represent the more general problem. We can represent the generalized problem by considering the disks in size order, large to small or small to large. I will use large to small. If we list the names of the towers the disks reside on, biggest disk first, smallest last, then any legal configuration can be represented as a string or array of tower names. If we use the large to small order, we can consider the string as a set of instructions on how to construct the configuration: ABCB means put the largest disk on tower A, the next largest on tower B etc.

We should note that in the discussion that follows, the recursive method of solution can be expressed without reference to the representation presented. The solution could have been presented entirely in tower diagrams without using the string or array description at all. The solution method is at higher level of abstraction that could be applied to any problem representation. I presented the solution method using both tower diagrams and the string representation only to avoid repeating the solution twice. It is important generally when solving problems to recognize that the problem can be addressed at different levels of abstraction, and it is best to express the problem solution at the highest level of abstraction so that different problem representations can be considered, and the most appropriate representation selected. In this case, we could choose the most obvious representation, using three objects, one for each tower, and each object could contain a description of the disks it contains. The string/array representation is simpler, and lends itself to an efficient implementation of the recursive solution.

Initial Configuration: ABCA

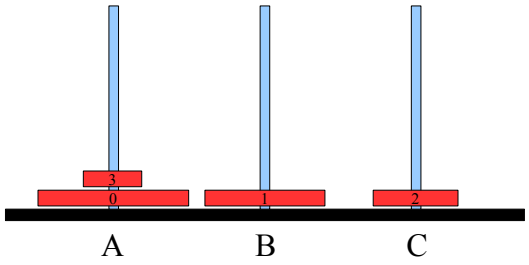


Final Configuration: ACBC

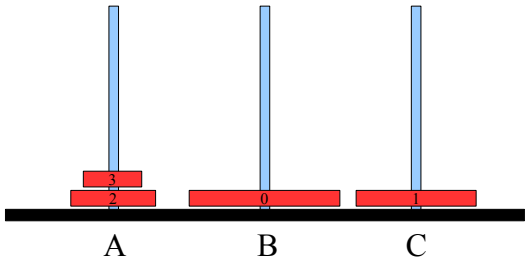


The problem is to transform ABCA → ACBC
Because the big disk is on the same tower in both configurations, we can just ignore the big disk and the solution to the 4 disk problem immediately reduces to the 3 disk problem: transform BCA → CBC

Initial Configuration: ABCA



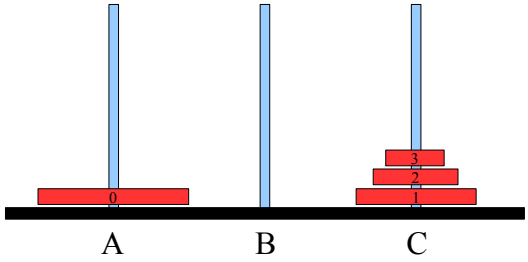
Final Configuration: BCAA



The problem is to transform ABCA → BCAA
Because the big disk is not on the same tower, we will have to move it from A to B. Before we can do this, we have to clear off any disks on top of the big disk, and get tower B empty, so the problem can be done in three stages:

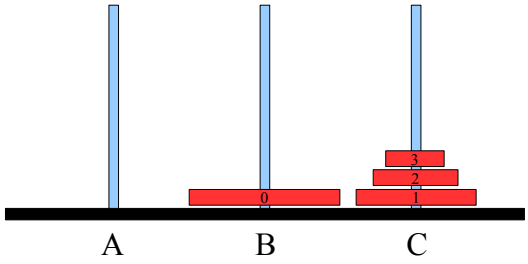
Stage 1:
From the Initial Configuration, get all but the big disk on tower C:

ABCA → ACCC



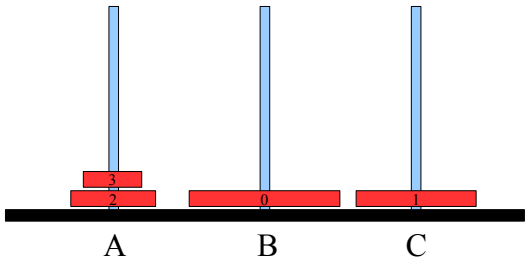
Stage 2:
Move the big disk from A to B:

ACCC → BCCC



Stage 3:
Get the disks off tower C and onto the towers where they belong in the final configuration:

BCCC → BCAA



Stage 2 is trivial, just move A to B, stages 1 and 3 don't move the big disk, so they reduce to 3 disk problems.

Summary:

To change an initial configuration involving n disks to a final configuration of n disks:

if n = 0, do nothing.

For n > 0:

if the big disk is on the same tower in both initial and final configurations, ignore the big disk and solve the problem for the n - 1 remaining disks.

Example: to solve ABBCAB → ACBAAC (5 disk problem)
just solve BBCAB → CBAAC (4 disk problem)

If the big disk is on different towers in the initial and final configurations, then do this:

let other = the tower that does not have the big disk in either the initial or final configurations.

Ignoring the biggest disk, move all the other disks onto the other tower.

Move the big disk to the tower it belongs on in the final configuration.

Ignoring the biggest disk, move all the other disks onto the towers they belong on.

Example: to solve ABBCAB → BCBAAC (5 disk problem)
(big disk on A, going to B, other = C)

solve BBCAB → CCCCC (4 disk problem)

move big disk from A to B

solve CCCCC → CBAAC (4 disk problem)