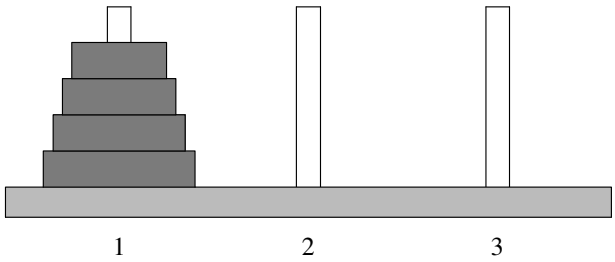
The naive linear search algorithm takes 8,675,309 comparisons, whereas the binary search took just 22 comparisons. In the worst case, that is, determining that a value is not in the vector, the naive linear search algorithm would take 10,000,000 comparisons, whereas binary search would take only $\log_2 10,000,000 = 24$ comparisons.

This concludes the discussion of the binary search algorithm and its associated implementation in R. Using recursion to calculate factorials in the previous section and to conduct a binary search in this section was not necessary. These same algorithms could have been conducted without recursion. The advanced application presented in the next section does not have an easy solution using loops, and in fact, is best solved using recursion.

24.3 Tower of Hanoi

The *Tower of Hanoi* is a puzzle consisting of three rods and n disks of different sizes (that is, diameters) that can slide onto any of the rods. The initial state of the puzzle has the n disks on one rod sorted in order by diameter, with the largest at the bottom and the smallest at the top. A side view of the puzzle (plotted in R!) in the initial state of $n = 4$ disks placed on rod 1 is shown below.



The goal of the puzzle is to move the entire stack of disks from the initial source rod s to a destination rod d, preferably in a minimum number of moves, following these rules:

- each move consists of taking the top disk from one of the stacks and placing it at the top of another stack,
- a larger disk may not be placed on top of a smaller disk.

The purpose of this programming exercise is to give a list of moves that solves the puzzle using a minimum number of moves.

To think of this problem in terms of recursion, consider the problem of moving n disks from source rod s to destination rod d via the following steps:

- move the top $n-1$ disks from the source rod s to the other non-destination rod,
- move the bottom disk from the source rod s to the destination rod d,
- move the $n-1$ disks from the non-destination rod to the destination rod d.

Moving the top $n-1$ disks from one rod to another means that the $(n-1)$-tower is moved in some manner, and may involve many individual moves.

Pseudo-code that is associated with a recursive function named `toh` (for Tower of Hanoi) with these steps for moving `n` disks from source rod `s` to destination rod `d` (indentation denotes nesting) is

```
toh(n, s, d)
  if (n == 1)
    output: move the top disk from rod s to rod d
  else
    toh(n - 1, s, 6 - s - d)
    output: move the top disk from rod s to rod d
    toh(n - 1, 6 - s - d, d)
```

If there is only one disk to be moved, then it is moved from rod `s` to rod `d`. Otherwise, `toh` is called to move the top `n - 1` disks from the source rod `s` to the non-destination rod (which is the rod numbered `6 - s - d`), then the bottom disk is moved from rod `s` to rod `d`, then `toh` is again called to move the top `n - 1` disks from the non-destination rod to the destination rod `d`.

This pseudo-code is translated to the R function `toh` as

```
> toh = function(n, s, d) {
+   if (n == 1) {
+     cat("move the top disk from rod", s, "to rod", d, "\n")
+   }
+   else {
+     toh(n - 1, s, 6 - s - d)
+     cat("move the top disk from rod", s, "to rod", d, "\n")
+     toh(n - 1, 6 - s - d, d)
+   }
+ }
```

When this code is called with

```
> toh(4, 1, 3)
```

which corresponds to moving four disks initially on the source rod 1 to the destination rod 3, the minimal sequence of moves is displayed.

```
move the top disk from rod 1 to rod 2
move the top disk from rod 1 to rod 3
move the top disk from rod 2 to rod 3
move the top disk from rod 1 to rod 2
move the top disk from rod 3 to rod 1
move the top disk from rod 3 to rod 2
move the top disk from rod 1 to rod 2
move the top disk from rod 1 to rod 3
move the top disk from rod 2 to rod 3
move the top disk from rod 2 to rod 1
move the top disk from rod 3 to rod 1
move the top disk from rod 2 to rod 3
move the top disk from rod 1 to rod 2
move the top disk from rod 1 to rod 3
move the top disk from rod 2 to rod 3
```

This particular version of the problem corresponds to the one depicted in the figure at the beginning of this section. The solution took 15 moves. Notice that the initial goal is to somehow move the top three disks on rod 1 to another rod, a goal that is accomplished in the first seven moves. At this