Towers of Hanoi

- Three pegs, one with n disks of decreasing diameter; two other pegs are empty
- Task: move all disks to the third peg under the following constraints
  - Can move only the topmost disk from one peg to another in one step
  - Cannot place a smaller disk below a larger one
- An example where recursion is much easier to formulate than a loop-based solution

Towers of Hanoi

- We want to write a recursive method shift (n, source, target, using) which moves n disks from peg 'source' to 'target' with the help of peg 'using' for intermediate transfers
- The first step is to formulate the algorithm
  - Observation: shift (n, source, target, using) Ξ shift (n-1, source, using, target) followed by transferring the largest disk from peg 'source' to peg 'target' and then calling shift (n-1, using, target, source)
  - Stopping condition: n = 1

```
class hanoi{

static int counter = 0;

   public static void shift(int n, char source, char
    target, char using){
     counter = counter + 1;

     if (n==1) System.out.println(source+" -> "+target);
       else if (n > 1) {
          shift(n-1,source,using,target);
          System.out.println(source+" -> "+target);
          shift(n-1,using,target,source);
       }
    } // How many moves needed? 2^n-1
```

```
    public static void main (String args[])
    {
      int n = 3;
      shift(n,'a','c','b');

      System.out.println(counter);

    }
}
```

Towers of Hanoi

- Total number of method calls

  Let T_n be the number of method calls to solve for n disks

  $T_n = 2T_{n-1} + 1$ for $n > 1$; $T_1 = 1$