Recursion

solveTowers(N, Src, Aux, Dst)
    if N is 0 exit
    solveTowers(N-1, Src, Dst, Aux)
    move from Src to Dst
    solveTowers(N-1, Aux, Src, Dst)

\[ T(n) \text{ is \# moves (total)} \]

\[
T(n) = T(n-1) + 1 + T(n-1) \\
= 2T(n-1) + 1
\]

recurrence relation
When analyzing recursive code snippets:

1) Find the recurrence relation
2) Solve the recurrence (substitution, tree)
3) Prove the solution by induction

Hanoi Towers:

1) \( T(n) = 2T(n-1) + 1 \)
2) \( T(1) = 1 \)

Solve it!
Substitution

\[
T(n) = 2T(n-1) + 1
\]

\[
2^1 T(n-1) = 2^2 T(n-2) + 1 \cdot 2^1
\]

\[
2^2 T(n-2) = 2^3 T(n-3) + 1 \cdot 2^2
\]

\[
\vdots
\]

\[
2^{n-1} T(1) = 2^n T(0) + 1 \cdot 2^{n-1}
\]

\[
T(n) = 1 + 2 + 2^2 + \ldots + 2^{n-1}
\]

\[
T(n) = 2^n - 1 
\]

is \( \Theta(2^n) \)

\[
\begin{align*}
\text{when } n = 2 & \quad T(2) = 3 \\
\end{align*}
\]

3) Prove \( T(n) = 2^n - 1 \) by induction (necessary because of the substitution)
\[ n=2 \]

**Base:** \[ T(2) = 3 \] is true.

**Hypothesis:** \[ n = k \] \[ T(k) = 2^k - 1 \]

**Inductive Step:** \[ n = k+1 \]

\[
T(k+1) = 2T(k) + 1 \quad \text{(recurrence)}
\]
\[
= 2(2^k - 1) + 1 \quad \text{(because of hypothesis)}
\]
\[
= 2^{k+1} - 2 + 1
\]
\[
= 2^{k+1} - 1
\]
Use of a recursion tree to solve the recurrence relation.

Hanoi called for $N = 3$

Number of non-recursive operations for each node is 1.

Sum of all nodes = $2^N - 1$
factorial(n)
  if (n ≤ 1)
    return 1
  else
    return n * factorial(n-1);

1. \( T(n) = T(n-1) + C \)
   \( T(1) = 1 \)

   \( T(3) = T(2) + C \)
   \( T(2) = T(1) + C \)
   \( T(1) = 1 \)

2. \( T(n) = C(n-1) + 1 \)
3. In case we ask you you have to prove it by induction

\[ T(n) \text{ is } \Theta(n) \]

Should I prove by induction

\[ T(n) = c(n-1)+1 \text{ or directly } T(n) \text{ is } \Theta(n)? \]

(For our where you can only find an inequality \( 1 \leq T(n) \leq \).)