Towers of Hanoi pseudocode makes it clear that the number of moves for a problem with n disks is:

$$f(n) = \begin{cases} 1 & n = 1 \\ 2f(n-1) + 1 & n > 1 \end{cases}$$

It is a matter of great urgency to determine an analytical expression for this quantity — recall that the end of the world depends on this.

The most obvious thing about the recurrence is that the number of moves doubles if we add a disk. Thus, while increasing the disks from 1 to n, the computation (moves) will double at each stage. This suggests a complexity of 2^n.

If we run Hanoi with $n = 1, 2, 3$, it's easy to see that the number of moves is $1, 3, 7$ respectively. This suggests a complexity of $2^n - 1$. We can check this with induction.

⋆ **Claim:** $f(n) = 2^n - 1$

Proof:

Basis: First we prove that the claim is true for some n, say $n = 1$:

Induction Step: Now we assume that the claim is true for n, and prove that it is true for $n + 1$:

Therefore, $f(n) = 2^n - 1$ for all n. ■

⋆ **Example: Merge Sort**

Given n elements, divide them into 2 groups of size $n/2$. Recursively sort the two subproblems using Merge Sort. Then merge the sorted sublists together to form 1 sorted list of size n. In the merging phase, the smallest element in each sublist are compared and the smaller of these two elements is added next to the sorted list and removed from its sublist. It turns out that the merge step can be completed with $n - 1$ comparisons in the worst case.

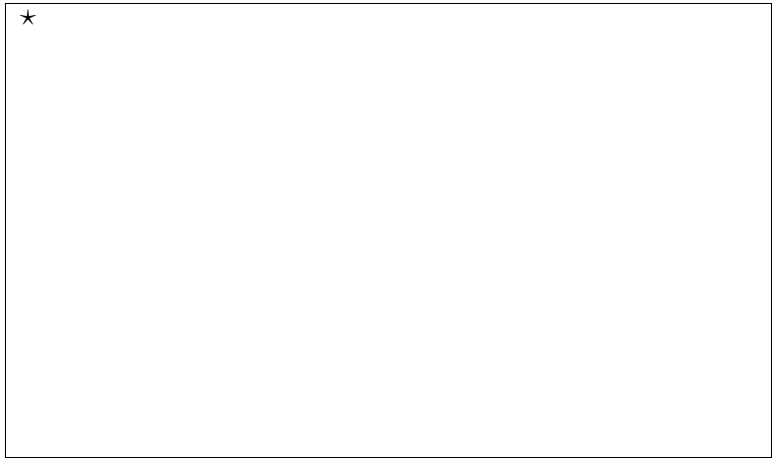
To get an upper bound on the number of comparisons done by merge sort, we'll just use the fact that the number of comparisons in the merge phase is at most n. Let $C(n)$ denote the number of comparisons done on a list of n elements. If n is a power of 2, we have the recurrence equation:

$$C(n) = 0 \qquad n = 1$$
$$C(n) < C(\lfloor n/2 \rfloor) + C(\lceil n/2 \rceil) + n \quad n > 1$$

Solve this recurrence using the **iteration method**:

⋆

$C(n) < C(n/2) + n$



Example 3: Celebrity Problem

Among n people, a celebrity is someone who is known by everyone but does not know anyone. Suppose you want to identify a celebrity among n people. You are allowed to ask questions to of the form "does person x know person y?" Your goal is to minimize the number of questions.


```
function Celebrity1({1,2,...,n})
   if n = 1 then
        return n
   else x <-- Celebrity1({1,2,...,n-1})
        if x not= ‘‘none’’ and (n knows x) and (x does not know n) then
           return x
        else {determine if the nth person is a celebrity}
           for i <-- 1 to n-1 do
               if (n knows i) and (i does not know n) then
                   return ‘‘none’’
           return n
```

```
function Celebrity2({1,2,...,n})
   if n = 1 then return n
   if n knows n-1 then {n cannot be the celebrity}
         x <-- Celebrity2({1,2, ..., n-1})
         if x = ‘‘none’’ then return ‘‘none’’
         else if (n knows x) and (x does not know n) then
                 return x
              else return ‘‘none’’
   else {n-1 cannot be the celebrity}
         x <-- Celebrity2({1,2, ..., n-2,n})
         if x = ‘‘none’’ then return ‘‘none’’
         else if (n-1 knows x) and (x does not know n-1) then
                 return x
              else return ‘‘none’’
```

The two functions output either the number of the celebrity, or "none" if no celebrity exists in the set. There can be at most one celebrity in the set, and in a set of size 1, the single person is a celebrity.

Which of the Celebrity functions uses the fewest questions? To answer this, we first find recurrence relations that describe the number of questions used by each function. Then, we solve these recurrences to find closed form expressions.

Let $C_1(n)$ and $C_2(n)$ be the "worst case" number of comparisons done by the Celebrity1 and Celebrity2 algorithms. Examination of those algorithms shows that

$\star$

$$C_1(n) = \begin{cases} & n = 1 \\ & n > 1 \end{cases}$$

$$C_2(n) = \begin{cases} & n = 1 \\ & n > 1 \end{cases}$$

The base cases are obtained by noting that, in an input set containing only one person, our convention is that this person is a celebrity and no questions need to be asked. We can solve these recurrences using the iteration method, to learn which algorithm is better. We start with $C_1(n)$.

$\star$

$C_1(n) =$

⋆

$C_1(n) =$

We conclude that the worst-case number of questions asked by the function Celebrity1 is $n(n+1)-2$. (This does not look good, since the naive algorithm which asks each person whether s/he knows each other person uses $n(n-1)$ comparisons; clearly if $n \geq 2$ then $n(n+1)-2 > n(n-1)$. The reason for this is that, as written, the Celebrity1 algorithm repeats some questions and also sometimes asks redundant questions such as whether a person knows him/her-self.)

We next solve the recurrence for $C_2(n)$.

⋆

$C_2(n) =$

We see that Celebrity2 uses fewer questions than the "naive" algorithm that does $n(n-1)$ comparisons, if $n > 3$. For example, if $n = 100$ Celebrity2 uses 297 questions whereas the naive algorithm uses $9,900$ questions.