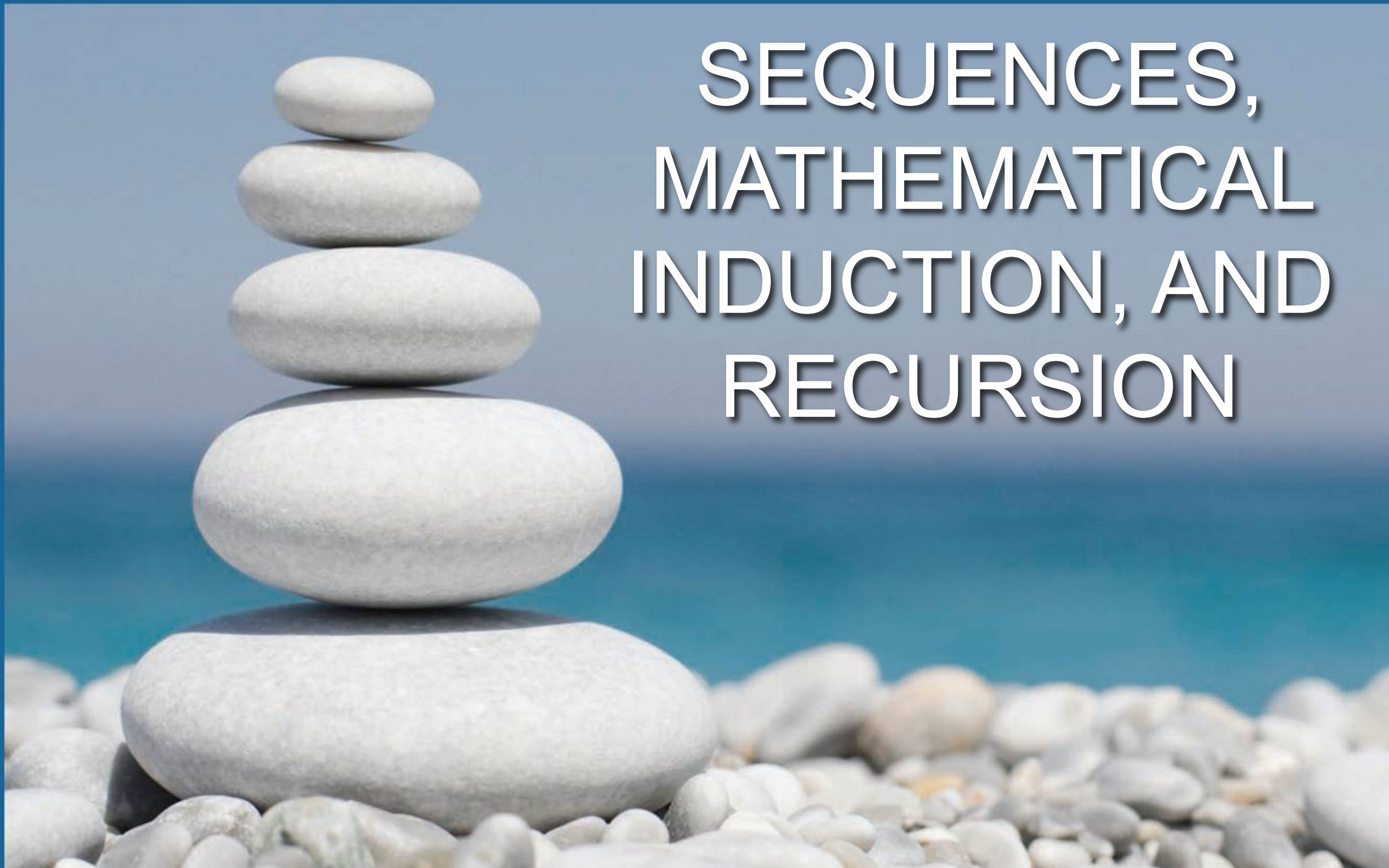
CHAPTER 5



SEQUENCES, MATHEMATICAL INDUCTION, AND RECURSION

Alessandro Artale – UniBZ - http://www.inf.unibz.it/~artale/

SECTION 5.6

Defining Sequences Recursively

Defining Sequences Recursively

A way to define a sequence is to give an explicit formula for its *n*th term.

For example, a sequence a_0, a_1, a_2 . . . can be specified by writing

$$a_n = \frac{(-1)^n}{n+1} \quad \text{for all integers } n \geq 0.$$

The advantage of defining a sequence by such an explicit formula is that each term of the sequence is uniquely determined and can be computed in a fixed, finite number of steps, by substitution.

Defining Sequences Recursively

Another way to define a sequence is to use recursion.
It is similar to use the Induction Principle.

This requires giving both an equation, called a *recurrence relation*, that defines each later term in the sequence by reference to earlier terms (induction step) and also one or more initial values for the sequence (basis step).

• Definition

A **recurrence relation** for a sequence $a_0, a_1, a_2, \ldots$ is a formula that relates each term a_k to certain of its predecessors $a_{k-1}, a_{k-2}, \ldots, a_{k-i}$, where i is an integer with $k - i \geq 0$. The **initial conditions** for such a recurrence relation specify the values of $a_0, a_1, a_2, \ldots, a_{i-1}$, if i is a fixed integer, or $a_0, a_1, \ldots, a_m$, where m is an integer with $m \geq 0$, if i depends on k.

Example 1 – *Computing Terms of a Recursively Defined Sequence*

Define a sequence $c_0, c_1, c_2, \ldots$ recursively as follows: For all integers $k \geq 2$,

$$(1) \quad c_k = c_{k-1} + kc_{k-2} + 1 \quad \text{recurrence relation}$$

$$(2) \quad c_0 = 1 \quad \text{and} \quad c_1 = 2 \quad \text{initial conditions.}$$

Find c_2, c_3, and c_4.

Solution:

$$\begin{aligned} c_2 &= c_1 + 2c_0 + 1 && \text{by substituting } k = 2 \text{ into (1)} \\ &= 2 + 2 \cdot 1 + 1 && \text{since } c_1 = 2 \text{ and } c_0 = 1 \text{ by (2)} \end{aligned}$$

Example 1 – *Solution*

cont'd

(3) • $c_2 = 5$

$c_3 = c_2 + 3c_1 + 1$ by substituting $k = 3$ into (1)

$= 5 + 3 \cdot 2 + 1$ since $c_2 = 5$ by (3) and $c_1 = 2$ by (2)

(4) • $c_3 = 12$

$c_4 = c_3 + 4c_2 + 1$ by substituting $k = 4$ into (1)

$= 12 + 4 \cdot 5 + 1$ since $c_3 = 12$ by (4) and $c_2 = 5$ by (3)

(5) • $c_4 = 33$

Examples of Recursively Defined Sequences

Examples of Recursively Defined Sequences

Recursion is one of the central ideas of computer science.

To solve a problem recursively means to find a way to break it down into smaller subproblems each having the same form as the original problem, and

- the subproblems are small and easy to solve, and
- the solutions of the subproblems can be woven together to form a solution to the original problem.

Example 2 – *The Tower of Hanoi*

In 1883 a French mathematician, Édouard Lucas, invented a puzzle that he called The Tower of Hanoi (La Tour D’ Hanoï).

The puzzle consisted of eight disks of wood with holes in their centers, which were piled in order of decreasing size on one pole in a row of three. Those who played the game were supposed to move all the disks one by one from one pole to another, never placing a larger disk on top of a smaller one.

Example 2 – *The Tower of Hanoi* cont'd

The puzzle offered a prize of ten thousand francs (about $34,000 US today) to anyone who could move a tower of 64 disks by hand while following the rules of the game. (See Figure 5.6.2) Assuming that you transferred the disks as efficiently as possible, how many moves would be required to win the prize?

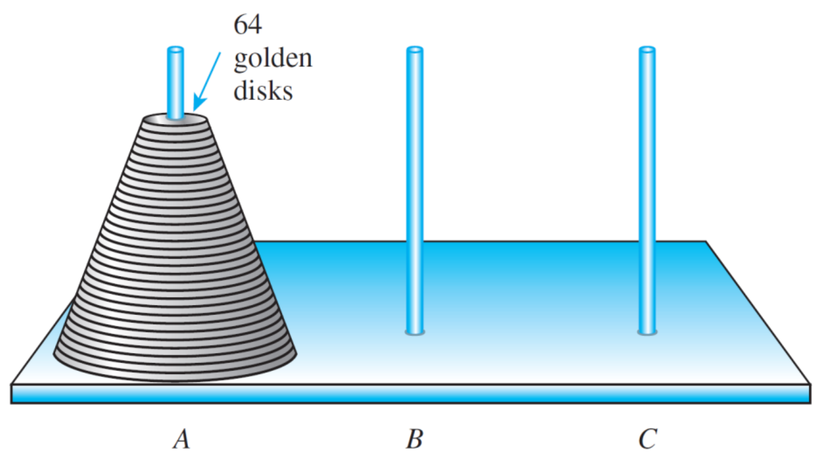


Figure 5.6.2

Example 2 – *Solution*

An elegant and efficient way to solve this problem is to think recursively.

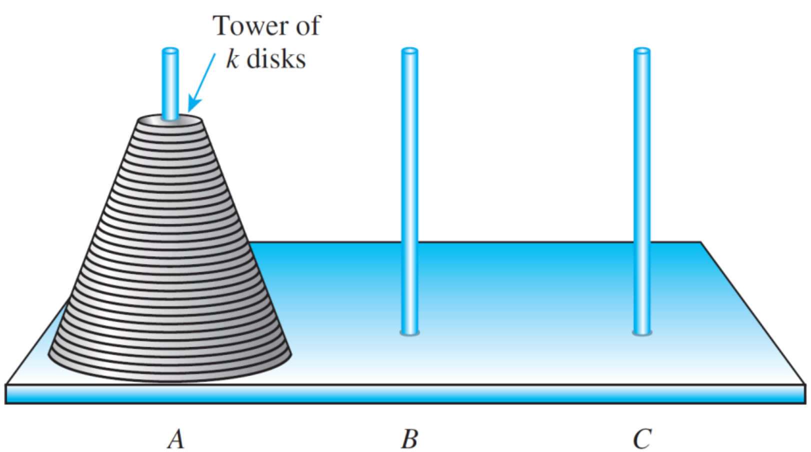
Suppose that you have found the most efficient way possible to transfer a tower of $k - 1$ disks one by one from one pole to another, obeying the restriction that you never place a larger disk on top of a smaller one.

What is the most efficient way to transfer a tower of k disks from one pole to another?

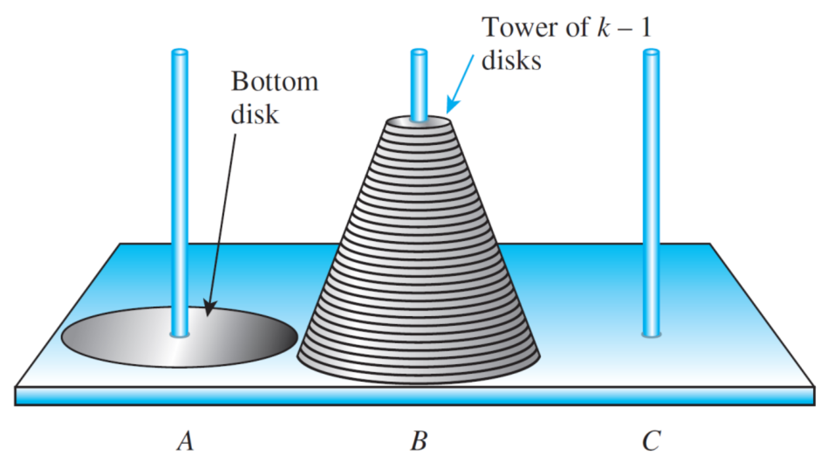
Example 2 – *Solution*

cont'd

The answer is sketched in Figure 5.6.3, where pole A is the initial pole and pole C is the target pole, and is described as follows:



Initial Position
(a)



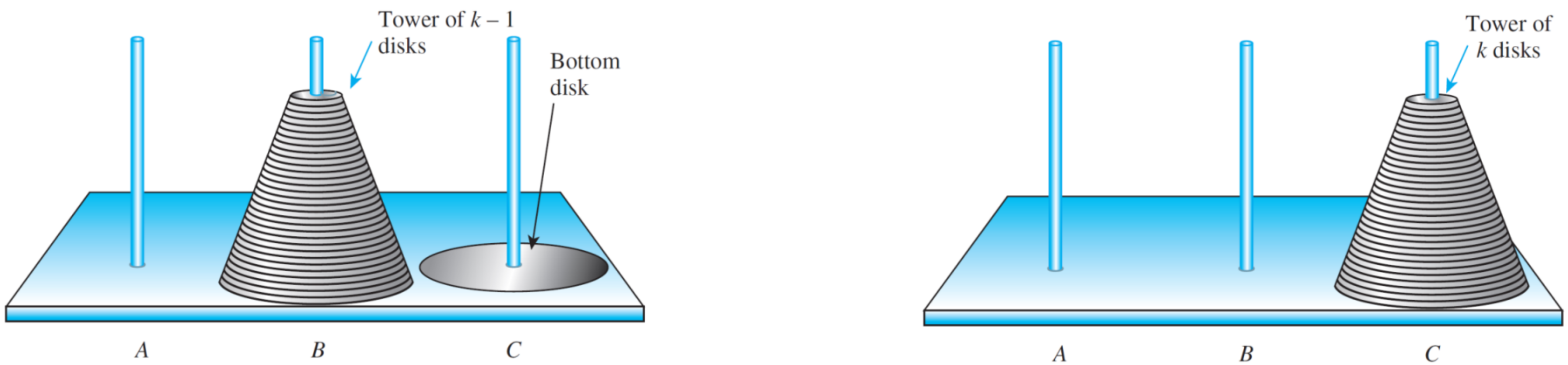
Position after Transferring $k - 1$ Disks from A to B
(b)

Moves for the Tower of Hanoi

Figure 5.6.3

Example 2 – *Solution*

cont' d



Position after Moving the Bottom Disk from *A* to *C*
(c)

Position after Transferring $k-1$ Disks from *B* to *C*
(d)

Moves for the Tower of Hanoi

Figure 5.6.3

Step 1: Transfer the top $k-1$ disks from pole *A* to pole *B*. If $k > 2$, execution of this step will require a number of moves of individual disks among the three poles (the point of thinking recursively is not to detail of how those moves will occur).

Example 2 – *Solution*

cont'd

Step 2: Move the bottom disk from pole *A* to pole *C*.

Step 3: Transfer the top $k – 1$ disks from pole *B* to pole *C*. (Again, if $k > 2$, execution of this step will require more than one move.)

To see that this sequence of moves is most efficient, observe that to move the bottom disk of a stack of k disks from one pole to another, you must first transfer the top $k – 1$ disks to a third pole to get them out of the way.

Example 2 – *Solution*

cont'd

Thus transferring the stack of k disks from pole A to pole C requires at least two transfers of the top $k - 1$ disks:

- one to transfer them off the bottom disk to free the bottom disk so that it can be moved, and
- another to transfer them back on top of the bottom disk after the bottom disk has been moved to pole C.

Example 2 – *Solution*

cont'd

If the bottom disk were not moved directly from pole *A* to pole *C* but were moved to pole *B* first, at least two additional transfers of the top $k - 1$ disks would be necessary:

- one to move them from pole *A* to pole *C* so that the bottom disk could be moved from pole *A* to pole *B*, and
- another to move them off pole *C* so that the bottom disk could be moved onto pole *C*.

This would increase the total number of moves and result in a less efficient transfer.

Example 2 – *Solution*

cont'd

Thus the minimum sequence of moves must include going from the initial position (a) to position (b) to position (c) to position (d).

It follows that

$$\begin{bmatrix} \text{the minimum} \\ \text{number of moves} \\ \text{needed to transfer} \\ \text{a tower of } k \text{ disks} \\ \text{from pole } A \text{ to} \\ \text{pole } C \end{bmatrix} = \begin{bmatrix} \text{the minimum} \\ \text{number of} \\ \text{moves needed} \\ \text{to go from} \\ \text{position (a)} \\ \text{to position (b)} \end{bmatrix} + \begin{bmatrix} \text{The minimum} \\ \text{number of} \\ \text{moves needed} \\ \text{to go from} \\ \text{position (b)} \\ \text{to position (c)} \end{bmatrix} + \begin{bmatrix} \text{the minimum} \\ \text{number of} \\ \text{moves needed} \\ \text{to go from} \\ \text{position (c)} \\ \text{to position (d)} \end{bmatrix} \qquad 5.6.1$$

For each integer $n \geq 1$, let

$$m_n = \begin{bmatrix} \text{the minimum number of moves needed to transfer} \\ \text{a tower of } n \text{ disks from one pole to another} \end{bmatrix}$$

Example 2 – *Solution*

cont'd

Note that the numbers m_n are independent of the labeling of the poles; it takes the same minimum number of moves to transfer n disks from pole A to pole C as to transfer n disks from pole A to pole B, for example.

Also the values of m_n are independent of the number of larger disks that may lie below the top n, provided these remain stationary while the top n are moved:

> Because the disks on the bottom are all larger than the ones on the top, the top disks can be moved from pole to pole as though the bottom disks were not present.

Example 2 – *Solution*

cont'd

Going from position (a) to position (b) requires m_{k-1} moves, going from position (b) to position (c) requires just one move, and going from position (c) to position (d) requires m_{k-1} moves.

By substitution into equation (5.6.1), therefore,

$$
\begin{aligned}
m_k &= m_{k-1} + 1 + m_{k-1} \\
&= 2m_{k-1} + 1 \qquad \text{for all integers } k \geq 2.
\end{aligned}
$$

The initial condition, or base, of this recursion is found by using the definition of the sequence.

Example 2 – *Solution*

cont'd

Because just one move is needed to move one disk from one pole to another,

$$m_1 = \left[\begin{array}{l}\text{the minimum number of moves needed to move} \\ \text{a tower of one disk from one pole to another}\end{array}\right] = 1.$$

Hence the complete recursive specification of the sequence $m_1, m_2, m_3, \ldots$ is as follows:
For all integers $k \geq 2$,

$$\begin{array}{lll}(1) & m_k = 2m_{k-1} + 1 & \text{recurrence relation} \\ (2) & m_1 = 1 & \text{initial conditions}\end{array}$$

Example 2 – *Solution*

cont'd

Here is a computation of the next five terms of the sequence:

$$(3) \quad m_2 = 2m_1 + 1 = 2 \cdot 1 + 1 = 3 \qquad \text{by (1) and (2)}$$

$$(4) \quad m_3 = 2m_2 + 1 = 2 \cdot 3 + 1 = 7 \qquad \text{by (1) and (3)}$$

$$(5) \quad m_4 = 2m_3 + 1 = 2 \cdot 7 + 1 = 15 \qquad \text{by (1) and (4)}$$

$$(6) \quad m_5 = 2m_4 + 1 = 2 \cdot 15 + 1 = 31 \qquad \text{by (1) and (5)}$$

$$(7) \quad m_6 = 2m_5 + 1 = 2 \cdot 31 + 1 = 63 \qquad \text{by (1) and (6)}$$

Going back to the legend, suppose the priests work rapidly and move one disk every second.

Then the time from the beginning of creation to the end of the world would be m_{64} seconds.

Example 2 – *Solution*

cont' d

We can compute m_{64} on a calculator.

The approximate result is

$$\begin{aligned} 1.844674 \times 10^{19} \text{ seconds} &\cong 5.84542 \times 10^{11} \text{ years} \\ &\cong 584.5 \text{ billion years,} \end{aligned}$$

which is obtained by the estimate of

$$\underset{\substack{\uparrow \\ \text{seconds per} \\ \text{minute}}}{60} \cdot \underset{\substack{\uparrow \\ \text{minutes} \\ \text{per} \\ \text{hour}}}{60} \cdot \underset{\substack{\nwarrow \\ \text{hours} \\ \text{per} \\ \text{day}}}{24} \cdot \underset{\substack{\nwarrow \\ \text{days} \\ \text{per} \\ \text{year}}}{(365.25)} = \underset{\substack{\uparrow \\ \text{seconds} \\ \text{per} \\ \text{year}}}{31,557,600}$$

seconds in a year (figuring 365.25 days in a year to take leap years into account). Surprisingly, this figure is close to some scientific estimates of the life of the universe!

SECTION 5.7

Solving Recurrence Relations by Iteration

The Method of Iteration

Solving Recurrence Relations by Iteration

Suppose you have a sequence that satisfies a certain recurrence relation and initial conditions.

It is often helpful to know an explicit formula for the sequence, especially if you need to compute terms with very large subscripts.

Such an explicit formula is also called a **solution** to the recurrence relation.

The Method of Iteration

The most basic method for finding an explicit formula for a recursively defined sequence is by using the **method of iteration**.

Iteration works as follows: Given a sequence $a_0, a_1, a_2, \ldots$ defined by a recurrence relation and initial conditions, you start from the initial conditions and calculate successive terms of the sequence until you see a pattern developing.

At that point you guess an explicit formula.

Example 5 – *An Explicit Formula for the Tower of Hanoi Sequence*

The Tower of Hanoi sequence $m_1, m_2, m_3, \ldots$ satisfies the recurrence relation

$$m_k = 2m_{k-1} + 1 \quad \text{for all integers } k \geq 2$$

and has the initial condition

$$m_1 = 1.$$

Use iteration to guess an explicit formula for this sequence, to simplify the answer.

Example 5 – *Solution*

By iteration

$$m_1 = 1$$

$$m_2 = 2m_1 + 1 = 2 \cdot 1 + 1 \qquad = 2^1 + 1,$$

$$m_3 = 2m_2 + 1 = 2(2 + 1) + 1 \qquad = 2^2 + 2 + 1,$$

$$m_4 = 2m_3 + 1 = 2(2^2 + 2 + 1) + 1 \qquad = 2^3 + 2^2 + 2 + 1,$$

$$m_5 = 2m_4 + 1 = 2(2^3 + 2^2 + 2 + 1) + 1 = 2^4 + 2^3 + 2^2 + 2 + 1.$$

Example 5 – *Solution*

cont'd

These calculations show that each term up to m_5 is a sum of successive powers of 2, starting with $2^0 = 1$ and going up to 2^k, where k is 1 less than the subscript of the term.

For instance, for $n = 6$,

$$m_6 = 2m_5 + 1 = 2(2^4 + 2^3 + 2^2 + 2 + 1) + 1 = 2^5 + 2^4 + 2^3 + 2^2 + 2 + 1.$$

Example 5 – *Solution*

cont'd

Thus it seems that, in general,

$$m_n = 2^{n-1} + 2^{n-2} + \cdots + 2^2 + 2 + 1.$$

By the formula for the sum of a geometric sequence (Theorem 5.2.3),

Theorem 5.2.3 Sum of a Geometric Sequence

For any real number r except 1, and any integer $n \geq 0$,

$$\sum_{i=0}^{n} r^i = \frac{r^{n+1} - 1}{r - 1}.$$

$$2^{n-1} + 2^{n-2} + \cdots + 2^2 + 2 + 1 = \frac{2^n - 1}{2 - 1} = 2^n - 1.$$

Example 5 – *Solution*

cont'd

Hence the explicit formula seems to be

$$m_n = 2^n - 1 \quad \text{for all integers } n \geq 1.$$

Checking the Correctness of a Formula by Mathematical Induction

Checking the Correctness of a Formula by Mathematical Induction

It is all too easy to make a mistake and come up with the wrong formula.

That is why it is important to confirm your calculations by checking the correctness of your formula.

The most common way to do this is to use mathematical induction.

Example 7 – *Using Mathematical Induction to Verify the Correctness of a Solution to a Recurrence Relation*

cont' d

Let m_k be the minimum number of moves needed to transfer a tower of *k* disks from one pole to another. Then,

If $m_1, m_2, m_3, \ldots$ is the sequence defined by

$$m_k = 2m_{k-1} + 1 \quad \text{for all integers } k \geq 2, \text{ and}$$

$$m_1 = 1,$$

then $m_n = 2^n - 1$ for all integers $n \geq 1$.

Use mathematical induction to show that this formula is correct.

Example 7 – *Solution*

cont' d

Proof of Correctness:

Let m_1, m_2, m_3, . . . be the sequence defined by specifying that $m_1 = 1$ and $m_k = 2m_{k+1} + 1$ for all integers $k \geq 2$, and let the property $P(n)$ be the equation

$$m_n = 2^n - 1 \qquad \leftarrow P(n)$$

We will use mathematical induction to prove that for all integers $n \geq 1$, $P(n)$ is true.

Show that *P*(1) is true:

To establish $P(1)$, we must show that

$$m_1 = 2^1 - 1. \qquad \leftarrow P(1)$$

Example 7 – *Solution*

cont' d

But the left-hand side of $P(1)$ is

$$m_1 = 1 \qquad \text{by definition of } m_1, m_2, m_3, \ldots,$$

and the right-hand side of $P(1)$ is

$$2^1 - 1 = 2 - 1 = 1.$$

Thus the two sides of $P(1)$ equal the same quantity, and hence $P(1)$ is true.

Example 7 – *Solution*

cont'd

Show that for all integers *k* ≥ 1, if *P*(*k*) is true then *P*(*k* + 1) is also true:

*[Suppose that P(*k*) is true for a particular but arbitrarily chosen integer k ≥* 1. *That is:]*

Suppose that *k* is any integer with *k* ≥ 1 such that

$$m_k = 2^k - 1. \qquad \leftarrow P(k) \text{ inductive hypothesis}$$

[We must show that P(*k* + 1) *is true. That is:]*

We must show that

$$m_{k+1} = 2^{k+1} - 1. \qquad \leftarrow P(k+1)$$

Example 7 – *Solution*

But the left-hand side of *P*(*k* + 1) is

$$
\begin{aligned}
m_{k+1} &= 2m_{(k+1)-1} + 1 && \text{by definition of } m_1,\ m_2,\ m_3, \ldots \\
&= 2m_k + 1 \\
&= 2(2^k - 1) + 1 && \text{by substitution from the inductive hypothesis} \\
&= 2^{k+1} - 2 + 1 && \text{by the distributive law and the fact that } 2 \cdot 2^k = 2^{k-1} \\
&= 2^{k+1} - 1 && \text{by basic algebra}
\end{aligned}
$$

which equals the right-hand side of *P*(*k* + 1). *[Since the basis and inductive steps have been proved, it follows by mathematical induction that the given formula holds for all integers n ≥ 1.]*