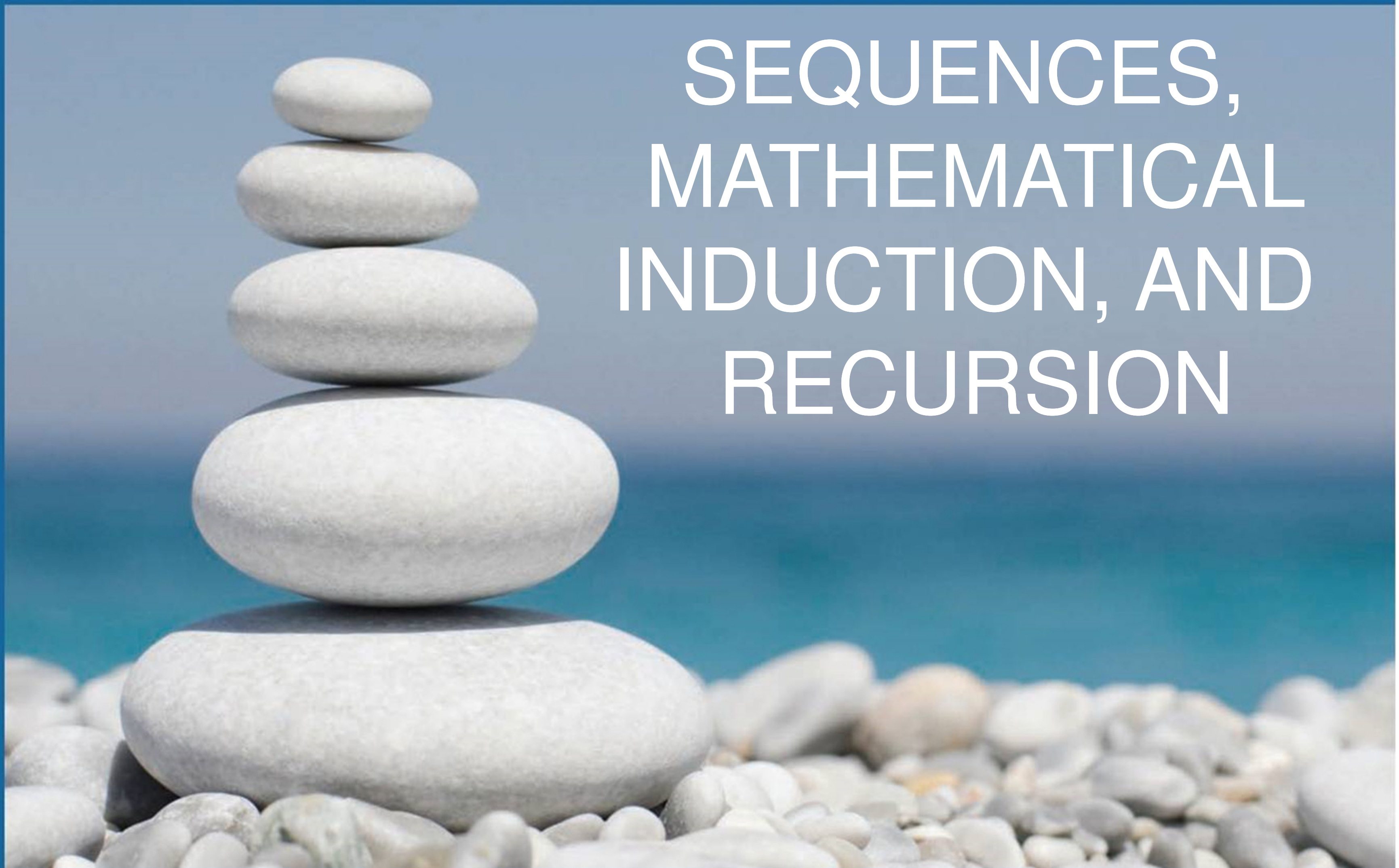
CHAPTER 5



SEQUENCES, MATHEMATICAL INDUCTION, AND RECURSION

SECTION 5.6

Defining Sequences Recursively

Defining Sequences Recursively

A sequence can be defined in a variety of different ways.

One informal way is to write the first few terms with the expectation that the general pattern will be obvious.

We might say, for instance, "consider the sequence 3, 5, 7, ..." Unfortunately, misunderstandings can occur when this approach is used.

The next term of the sequence could be 9 if we mean a sequence of odd integers, or it could be 11 if we mean the sequence of odd prime numbers.

Defining Sequences Recursively

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

For example, a sequence $a_0, a_1, a_2 \ldots$ 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

The third way to define a sequence is to use recursion.

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

• 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:

$$c_2 = c_1 + 2c_0 + 1 \qquad \text{by substituting } k = 2 \text{ into (1)}$$

$$= 2 + 2 \cdot 1 + 1 \qquad \text{since } c_1 = 2 \text{ and } c_0 = 1 \text{ by (2)}$$

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) • $c_4 = 33$

Example 4 – *Showing That a Sequence Given by an Explicit Formula Satisfies a Certain Recurrence Relation*

The sequence of **Catalan numbers**, named after the Belgian mathematician Eugène Catalan (1814–1894), arises in a remarkable variety of different contexts in discrete mathematics. It can be defined as follows: For each integer $n \geq 1$,

$$C_n = \frac{1}{n+1}\binom{2n}{n}.$$

a. Find C_1, C_2, and C_3.

b. Show that this sequence satisfies the recurrence relation $C_k = \frac{4k-2}{k+1}C_{k-1}$ ɔr all integers $k \geq 2$

Example 4 – *Solution*

a. $C_1 = \frac{1}{2}\binom{2}{1} = \frac{1}{2} \cdot 2 = 1,$

$$C_2 = \frac{1}{3}\binom{4}{2} = \frac{1}{3} \cdot 6 = 2,$$

$$C_3 = \frac{1}{4}\binom{6}{3} = \frac{1}{4} \cdot 20 = 5$$

Example 4 – *Solution*

cont'd

b. To obtain the *k*th and $(k - 1)$st terms of the sequence, just substitute *k* and $k - 1$ in place of *n* in the explicit formula for $C_1, C_2, C_3, \ldots$.

$$C_k = \frac{1}{k+1}\binom{2k}{k}$$

$$C_{k-1} = \frac{1}{(k-1)+1}\binom{2(k-1)}{k-1} = \frac{1}{k}\binom{2k-2}{k-1}.$$

Example 4 – *Solution*

cont'd

Then start with the right-hand side of the recurrence relation and transform it into the left-hand side: For each integer $k \geq 2$,

$$\frac{4k-2}{k+1}C_{k-1} = \frac{4k-2}{k+1}\left[\frac{1}{k}\binom{2k-2}{k-1}\right] \quad \text{by substituting}$$

$$= \frac{2(2k-1)}{k+1} \cdot \frac{1}{k} \cdot \frac{(2k-2)!}{(k-1)!(2k-2-(k-1))!} \quad \text{by the formula for } n \text{ choose } r$$

$$= \frac{1}{k+1} \cdot (2(2k-1)) \cdot \frac{(2k-2)!}{(k(k-1)!)(k-1)!} \quad \text{by rearranging the factors}$$

Example 4 – *Solution*

cont'd

$$= \frac{1}{k+1} \cdot (2(2k-1)) \cdot \frac{1}{k!(k-1)!} \cdot (2k-2)! \cdot \frac{1}{2} \cdot \frac{1}{k} \cdot 2k.$$ because $\frac{1}{2} \cdot \frac{1}{k} \cdot 2k = 1$

$$= \frac{1}{k+1} \cdot \frac{2}{2} \cdot \frac{1}{k!} \cdot \frac{1}{(k-1)!} \cdot \frac{1}{k} \cdot (2k) \cdot (2k-1) \cdot (2k-2)!$$ by rearranging the factors

$$= \frac{1}{k+1} \cdot \frac{(2k)!}{k!k!}$$ because $k(k-1)! = k!$, $\frac{2}{2} = 1$, and $2k \cdot (2k-1) \cdot (2k-2)! = (2k)!$

$$= \frac{1}{k+1} \binom{2k}{k}$$ by the formula for n choose r

$$= C_k$$ by defintion of $C_1, C_2, C_3, \ldots$.

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 to do this in such a way that when the process is repeated many times, the last of 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 5 – *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 5 – *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?

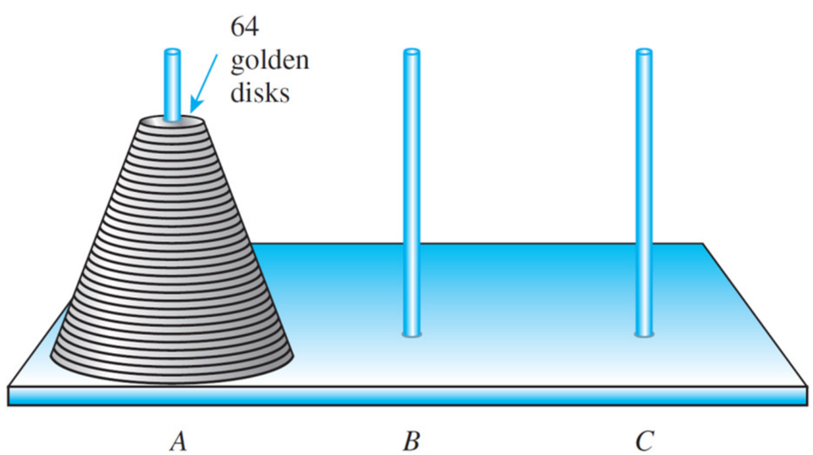


Figure 5.6.2

Example 5 – *Solution*

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

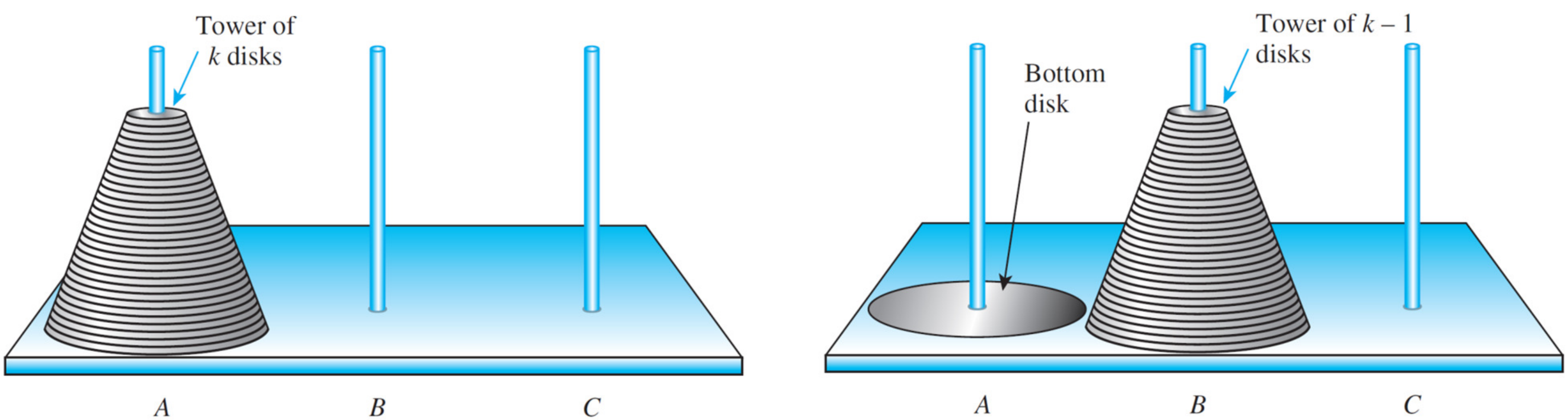
Suppose that you, somehow or other, have found the most efficient way possible to transfer a tower of $k - 1$ disks one by one from one pole to another, obeying the restriction that you never place a larger disk on top of a smaller one.

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

Example 5 – *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)

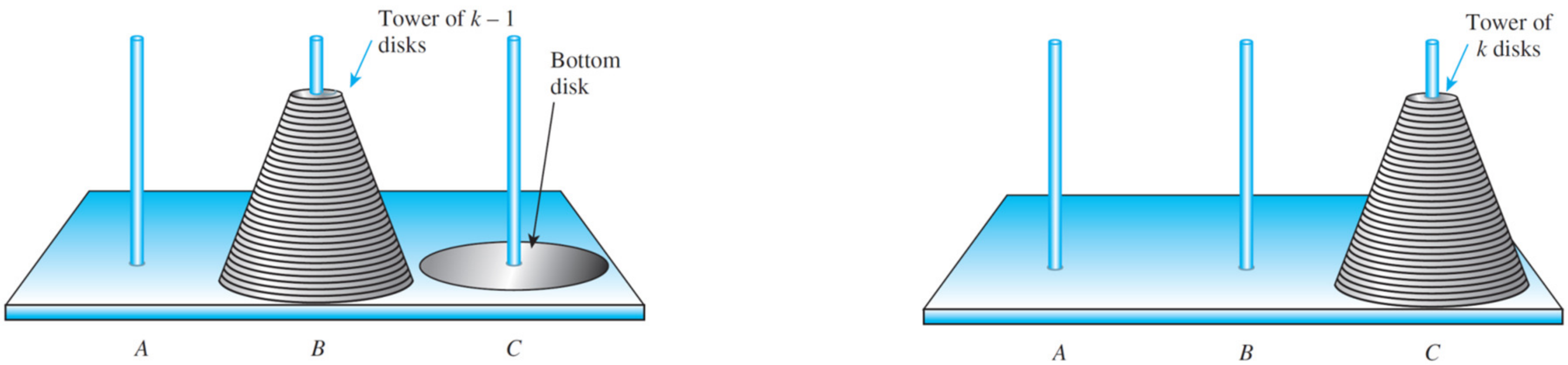
Position after Transferring $k – 1$ Disks from *A* to *B*
(b)

Moves for the Tower of Hanoi

Figure 5.6.3

Example 5 – *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. But the point of thinking recursively is not to get caught up in imagining the details of how those moves will occur.

Example 5 – *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 5 – *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 5 – *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 5 – *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

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

For each integer $n \geq 1$, let

$$m_n = \left[\begin{array}{l}\text{the minimum number of moves needed to transfer}\\ \text{a tower of } n \text{ disks from one pole to another}\end{array}\right]$$

Example 5 – *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 5 – *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 5 – *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 $k \geq 2$,

(1) $m_k = 2m_{k-1} + 1$ recurrence relation

(2) $m_1 = 1$ initial conditions

Example 5 – *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 5 – *Solution*

cont'd

We can compute m_{64} on a calculator.

The approximate result is

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

which is obtained by the estimate of

$$60 \cdot 60 \cdot 24 \cdot (365.25) = 31,557,600$$

60	60	24	(365.25)	$31,557,600$
seconds per minute	minutes per hour	hours per day	days per year	seconds per year

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!

Recursive Definitions of Sum and Product

Recursive Definitions of Sum and Product

Addition and multiplication are called *binary* operations because only two numbers can be added or multiplied at a time. Careful definitions of sums and products of more than two numbers use recursion.

- **Definition**

Given numbers $a_1, a_2, \ldots, a_n$, where n is a positive integer, the **summation from $i = 1$ to n of the a_i**, denoted $\sum_{i=1}^{n} a_i$, is defined as follows:

$$\sum_{i=1}^{1} a_i = a_1 \quad \text{and} \quad \sum_{i=1}^{n} a_i = \left(\sum_{i=1}^{n-1} a_i\right) + a_n, \quad \text{if } n > 1.$$

The **product from $i = 1$ to n of the a_i**, denoted $\prod_{i=1}^{n} a_i$, is defined by

$$\prod_{i=1}^{1} a_i = a_1 \quad \text{and} \quad \prod_{i=1}^{n} a_i = \left(\prod_{i=1}^{n-1} a_i\right) \cdot a_n, \quad \text{if } n > 1.$$

Recursive Definitions of Sum and Product

The effect of these definitions is to specify an *order* in which sums and products of more than two numbers are computed. For example,

$$\sum_{i=1}^{4} a_i = \left(\sum_{i=1}^{3} a_i\right) + a_4 = \left(\left(\sum_{i=1}^{2} a_i\right) + a_3\right) + a_4 = ((a_1 + a_2) + a_3) + a_4.$$

The recursive definitions are used with mathematical induction to establish various properties of general finite sums and products.

Example 9 – *A Sum of Sums*

Prove that for any positive integer n, if $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_n$ are real numbers, then

$$\sum_{i=1}^{n}(a_i + b_i) = \sum_{i=1}^{n} a_i + \sum_{i=1}^{n} b_i.$$

Solution:

The proof is by mathematical induction. Let the property $P(n)$ be the equation

$$\sum_{i=1}^{n}(a_i + b_i) = \sum_{i=1}^{n} a_i + \sum_{i=1}^{n} b_i. \qquad \leftarrow P(n)$$

Example 9 – *Solution*

cont'd

We must show that $P(n)$ is true for all integers $n \geq 0$.We do this by mathematical induction on n.

Show that $P(1)$ is true: To establish $P(1)$, we must show that

$$\sum_{i=1}^{1}(a_i + b_i) = \sum_{i=1}^{1} a_i + \sum_{i=1}^{1} b_i. \qquad \leftarrow P(1)$$

But

$$\sum_{i=1}^{1}(a_i + b_i) = a_1 + b_1 \qquad \text{by definition of } \Sigma$$

$$= \sum_{i=1}^{1} a_i + \sum_{i=1}^{1} b_i \qquad \text{also by definition of } \Sigma.$$

Hence $P(1)$ is true.

Example 9 – *Solution*

cont'd

Show that for all integers $k \geq 1$, if $P(k)$ is true then $P(k + 1)$ is also true: Suppose $a_1, a_2, \ldots, a_k, a_{k+1}$ and $b_1, b_2, \ldots, b_k, b_{k+1}$ are real numbers and that for some $k \geq 1$

$$\sum_{i=1}^{k}(a_i + b_i) = \sum_{i=1}^{k} a_i + \sum_{i=1}^{k} b_i. \qquad \leftarrow P(k) \text{ inductive hypothesis}$$

We must show that

$$\sum_{i=1}^{k+1}(a_i + b_i) = \sum_{i=1}^{k+1} a_i + \sum_{i=1}^{k+1} b_i. \qquad \leftarrow P(k+1)$$

[We will show that the left-hand side of this equation equals the right-hand side.]

Example 9 – *Solution*

cont'd

But the left-hand side of the equation is

$$\sum_{i=1}^{k+1}(a_i + b_i) = \sum_{i=1}^{k}(a_i + b_i) + (a_{k+1} + b_{k+1})$$ by definition of Σ

$$= \left(\sum_{i=1}^{k} a_i + \sum_{i=1}^{k} b_i\right) + (a_{k+1} + b_{k+1})$$ by inductive hypothesis

$$= \left(\sum_{i=1}^{k} a_i + a_{k+1}\right) + \left(\sum_{i=1}^{k} b_i + b_{k+1}\right)$$ by the associative and cummutative laws of algebra

Example 9 – *Solution*

cont'd

$$= \sum_{i=1}^{k+1} a_i + \sum_{i=1}^{k+1} b_i \qquad \text{by definition of } \Sigma$$

which equals the right-hand side of the equation. *[This is what was to be shown.]*