34. English or French, but not German.

35. English and French, but not German.

36. English, but neither French nor German.

37. Neither English, French, nor German.

A recent survey by the MAD corporation indicates that of the 700 families interviewed, 220 own a television set but no stereo, 200 own a stereo but no camera, 170 own a camera but no television set, 80 own a television set and a stereo but no camera, 80 own a stereo and a camera but no television set, 70 own a camera and a television set but no stereo, and 50 do not have any of these. Find the number of families with:

38. Exactly one of the items.

39. Exactly two of the items.

40. At least one of the items.

41. All of the items.

Using Algorithm 2.3, find the power set of each set. List the elements in the order obtained.

42. $\{a, b\}$

43. $\{a, b, c\}$

A finite set with a elements has b subsets. Find the number of subsets of a finite set with the given cardinality.

44. $a + 1$

45. $a + 2$

46. $a + 5$

47. $2a$

Let A, B, and C be subsets of a finite set U. Derive a formula for each.

48. $|A' \cap B'|$

49. $|A' \cap B' \cap C'|$

***50.** State the inclusion–exclusion principle for four finite sets A_i, $1 \leq i \leq 4$. (The formula contains 15 terms.)

***51.** Prove the formula in Exercise 50.

****52.** State the inclusion–exclusion principle for n finite sets A_i, $1 \leq i \leq n$.

2.5 Recursively Defined Sets

Week 12 ↓

A new way of defining sets is using recursion. (It is a powerful problem-solving technique discussed in detail in Chapter 5.)

Notice that the set of numbers $S = \{2, 2^2, 2^{2^2}, 2^{2^{2^2}}, \ldots\}$ has three interesting characteristics:

(1) $2 \in S$.

(2) If $x \in S$, then $2^x \in S$.

(3) Every element of S is obtained by a finite number of applications of properties 1 and 2 only.

(of an algebraic or geometric expression) from which another is derived, or which is not itself derived from another.

Property 1 identifies explicitly the primitive 原始の element in S and hence ensures that it is nonempty. Property 2 establishes a systematic procedure to construct new elements from known elements. How do we know, for instance, that $2^{2^2} \in S$? By property $1, 2 \in S$; then, by property (2), $2^2 \in S$; now choose $x = 2^2$ and apply property 2 again; so $2^{2^2} \in S$. Property 3 guarantees that in no other way can the elements of S be constructed. Thus the various elements of S can be obtained systematically by applying the above properties.

These three characteristics can be generalized and may be employed to define a set S implicitly. Such a definition is a recursive definition.

Recursively Defined Set

A **recursive definition** of a set S consists of three clauses:

- The **basis clause** explicitly lists at least one primitive element in S, ensuring that S is nonempty.
- The **recursive clause** establishes a systematic recipe to generate new elements from known elements.
- The **terminal clause** guarantees that the first two clauses are the only ways the elements of S can be obtained.

The terminal clause is generally omitted for convenience.

EXAMPLE 2.34 Let S be the set defined recursively as follows.

(1) $2 \in S$. (2) If $x \in S$, then $x^2 \in S$.

Describe the set by the listing method.

SOLUTION:

- $2 \in S$, by the basis clause.
- Choose $x = 2$. Then by the recursive clause, $4 \in S$.
- Now choose $x = 4$ and apply the recursive clause again, so $16 \in$ S. Continuing like this, we get $S = \{2, 4, 16, 256, 65536, \ldots\}$. ∎

The next three examples further 闡明,説明 elucidate make (something) clear; explain the recursive definition.

EXAMPLE 2.35 Notice that the language $L = \{\text{a}, \text{aa}, \text{ba}, \text{aaa}, \text{aba}, \text{baa}, \text{bba}, \ldots\}$ consists of words over the alphabet $\Sigma = \{\text{a}, \text{b}\}$ that end in the letter a. It can be defined recursively as follows.

- $\text{a} \in L$.
- If $x \in L$, then $\text{a}x, \text{b}x \in L$.

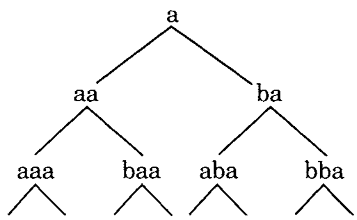
For instance, the word aba can be constructed as follows:

- $\text{a} \in L$. Choosing $x = \text{a}$, $\text{b}x = \text{ba} \in L$.

- Now choose x = ba. Then ax = aba $\in L$.

The tree diagram in Figure 2.24 illustrates systematically how to derive the words in L.

Figure 2.24



■

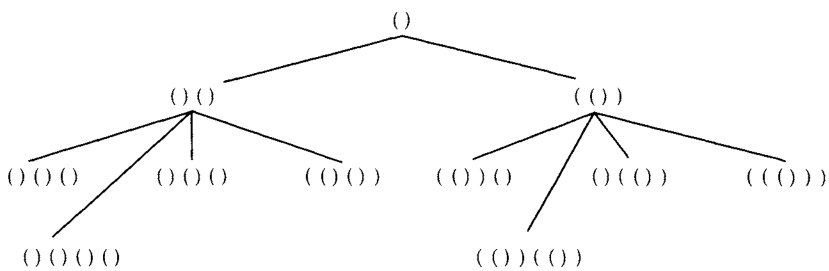
EXAMPLE 2.36

(Legally Paired Parentheses) An important problem in computer science is to determine whether or not a given expression is legally parenthesized. For example, (( )), ( ) ( ), and (( ) ( )) are validly paired sequences of parentheses, but) ( ), ( ) (, and) ( ) (are not. The set S of sequences of legally paired parentheses can be defined recursively as follows:

- ( ) $\in S$.
- If $x, y \in S$, then xy and (x) belong to S.

The tree diagram in Figure 2.25 shows the various ways of constructing the elements in S.

Figure 2.25



■

A simplified recipe to determine if a sequence of parentheses is legally paired is given in Algorithm 2.4.

```
Algorithm Legally Paired Sequence
(* This algorithm determines if a nonempty sequence of parentheses is
   legally paired. Count keeps track of the number of parentheses. It is
   incremented by 1 if the current parenthesis is a left parenthesis.
   and decremented by 1 if it is a right parenthesis. *)
   Begin (* algorithm *)
     count ← 0              (* initialize *)
     read a symbol
     if symbol = left paren then
```

5.1 Recursively Defined Functions

Week 12 (part 2)

Recall that in Section 2.5 we employed recursion to define sets; we invoked the recursive clause to construct new elements from known elements. The same idea can be applied to define functions, and hence sequences as well.

This section illustrates how powerful a problem-solving technique recursion is. We begin with a simple problem:

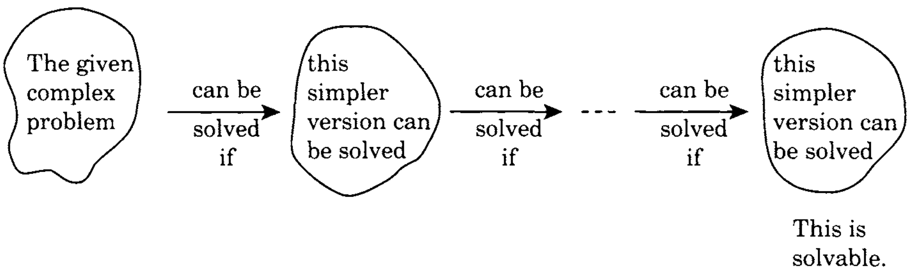
150 year

> **There are n guests at a sesquicentennial ball. Each person shakes hands with everybody else exactly once. How many handshakes are made?**

Suppose you would like to solve a problem such as this. (See Example 5.3.) The solution may not be obvious. However, it may turn out that the problem could be defined in terms of a simpler version of itself. Such a definition is a **recursive definition**. Consequently, the given problem can be solved provided the simpler version can be solved. This idea is pictorially represented in Figure 5.1.

語病?

Figure 5.1



Recursive Definition of a Function

Let $a \in \mathbf{W}$ and $X = \{a, a+1, a+2, \dots\}$. The **recursive definition** of a function f with domain X consists of three parts, where $k \geq 1$:

- **Basis clause** A few initial values of the function $f(a), f(a+1), \dots, f(a+k-1)$ are specified. An equation that specifies such initial values is an **initial condition**.
- **Recursive clause** A formula to compute $f(n)$ from the k preceding functional values $f(n-1), f(n-2), \dots, f(n-k)$ is made. Such a formula is a **recurrence relation** (or **recursion formula**).
- **Terminal clause** Only values thus obtained are valid functional values. (For convenience, we drop this clause from our recursive definition.)

Thus the recursive definition of f consists of one or more (a finite number of) initial conditions, and a recurrence relation.

Is the recursive definition of f a valid definition? In other words, if the k initial values $f(a), f(a+1), \ldots, f(a+k-1)$ are known and $f(n)$ is defined in terms of k of its predecessors $f(n-1), f(n-2), \ldots, f(n-k)$, where $n \geq a+k$, is $f(n)$ defined for $n \geq a$? Fortunately, the next theorem comes to our rescue. Its proof uses strong induction and is complicated, so we omit it.

THEOREM 5.1 Let $a \in \mathbf{W}$, $X = \{a, a+1, a+2, \ldots\}$, and $k \in \mathbb{N}$. Let $f : X \to \mathbb{R}$ such that $f(a), f(a+1), \ldots, f(a+k-1)$ are known. Let n be any positive integer $\geq a+k$ such that $f(n)$ is defined in terms of $f(n-1), f(n-2), \ldots$ and $f(n-k)$. Then $f(n)$ is defined for every $n \geq a$. ■

By virtue of this theorem, recursive definitions are also known as **inductive definitions**.

The following examples illustrate the recursive definition of a function.

EXAMPLE 5.1 Define recursively the factorial function f.

SOLUTION:

Recall that the factorial function f is defined by $f(n) = n!$, where $f(0) = 1$. Since $n! = n(n-1)!$, f can be defined recursively as follows:

$f(0) = 1$ ← initial condition

$f(n) = n \cdot f(n-1), \quad n \geq 1$ ← recurrence relation ■

Suppose we would like to compute $f(3)$ using this recursive definition. We then continue to apply the recurrence relation until the initial condition is reached, as shown below:

$$f(3) = 3 \cdot f(2) \tag{5.1}$$

recursive call — return value

$$f(2) = 2 \cdot f(1) \tag{5.2}$$

recursive call — return value

$$f(1) = 1 \cdot f(0) \tag{5.3}$$

recursive call — return value

$$f(0) = 1 \tag{5.4}$$

Since $f(0) = 1$, 1 is substituted for $f(0)$ in Equation (5.3) and $f(1)$ is computed: $f(1) = 1 \cdot f(0) = 1 \cdot 1 = 1$. This value is substituted for $f(1)$ in Equation (5.2) and $f(2)$ is computed: $f(2) = 2 \cdot f(1) = 2 \cdot 1 = 2$. This value is now returned to Equation (5.1) to compute $f(3)$: $f(3) = 3 \cdot f(2) = 3 \cdot 2 = 6$, as expected.

EXAMPLE 5.2 Judy deposits \$1000 in a local savings bank at an annual interest rate of 8% compounded annually. Define recursively the compound amount $A(n)$ she will have in her account at the end of n years.

SOLUTION:
Clearly, $A(0) =$ initial deposit $=$ \$1000. Let $n \geq 1$. Then:

$$\begin{aligned}A(n) &= \begin{pmatrix}\text{compound amount}\\ \text{at the end of the}\\ (n-1)\text{st year}\end{pmatrix} + \begin{pmatrix}\text{interest earned}\\ \text{during the}\\ n\text{th year}\end{pmatrix}\\ &= A(n-1) + (0.08)A(n-1)\\ &= 1.08A(n-1)\end{aligned}$$

Thus $A(n)$ can be defined recursively as follows:

$$\begin{aligned}&A(0) = 1000 && \leftarrow \text{initial condition}\\ &A(n) = 1.08A(n-1), \quad n \geq 1 && \leftarrow \text{recurrence relation}\end{aligned}$$ ■

For instance, the compound amount Judy will have at the end of three years is

$$\begin{aligned}A(3) &= 1.08A(2)\\ &= 1.08[1.08A(1)] = 1.08^2A(1)\\ &= 1.08^2[1.08A(0)] = 1.08^3(1000)\\ &\approx \$1259.71^*\end{aligned}$$

The next two examples illustrate an extremely useful problem-solving technique, used often in discrete mathematics and computer science.

EXAMPLE 5.3 **(The handshake problem)** There are n guests at a sesquicentennial ball. Each person shakes hands with everybody else exactly once. Define recursively the number of handshakes $h(n)$ that occur.

SOLUTION:
Clearly, $h(1) = 0$, so let $n \geq 2$. Let x be one of the guests. By definition, the number of handshakes made by the remaining $n - 1$ guests among themselves is $h(n-1)$. Now person x shakes hands with each of these

*The symbol $\approx$ means is *approximately equal to*.

$n - 1$ guests, yielding $n - 1$ additional handshakes. So the total number of handshakes made equals $h(n-1) + (n-1)$, where $n \geq 2$.

Thus $h(n)$ can be defined recursively as follows:

$$h(1) = 0 \qquad \leftarrow \text{initial condition}$$

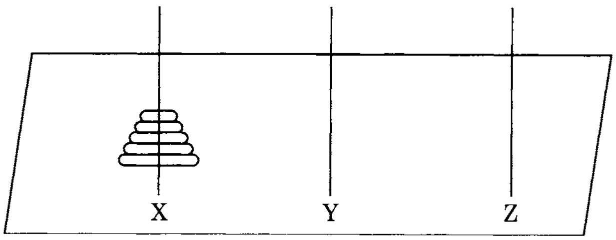
$$h(n) = h(n-1) + (n-1), \quad n \geq 2 \qquad \leftarrow \text{recurrence relation}$$ ■

EXAMPLE 5.4 **(Tower of Brahma*)** According to a legend of India, at the beginning of creation, God stacked 64 golden disks on one of three diamond pegs on a brass platform in the temple of Brahma at Benares† (see Figure 5.2). The priests on duty were asked to move the disks from peg X to peg Z using Y as an auxiliary peg under the following conditions:

- Only one disk can be moved at a time.
- No disk can be placed on the top of a smaller disk.

The priests were told that the world would end when the job was completed.

Figure 5.2



Suppose there are n disks on peg X. Let b_n denote the number of moves needed to move them from peg X to peg Z, using peg Y as an intermediary. Define b_n recursively.

SOLUTION:

Clearly $b_1 = 1$. Assume $n \geq 2$. Consider the top $n - 1$ disks on peg X. By definition, it takes b_{n-1} moves to transfer them from X to Y using Z as an auxiliary. That leaves the largest disk at peg X; it takes one move to transfer it from X to Z. See Figure 5.3.

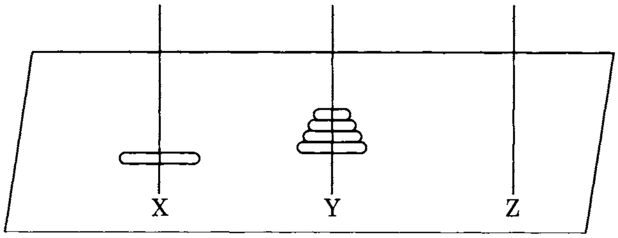
Now the $n - 1$ disks at Y can be moved from Y to Z using X as an intermediary in b_{n-1} moves, so the total number of moves needed is $b_{n-1} + 1 + b_{n-1} = 2b_{n-1} + 1$. Thus b_n can be defined recursively as follows:

$$b_n = \begin{cases} 1 & \text{if } n = 1 \quad \leftarrow \text{initial condition} \\ 2b_{n-1} + 1 & \text{otherwise} \quad \leftarrow \text{recurrence relation} \end{cases}$$ ■

*A puzzle based on the Tower of Brahma was marketed in 1883 under the name **Tower of Hanoi**.

†Benares is now known as Varanasi.

Figure 5.3



For example,

$$
\begin{aligned}
b_4 &= 2b_3 + 1 & &= 2[2b_2 + 1] + 1 \\
&= 4b_2 + 2 + 1 & &= 4[2b_1 + 1] + 2 + 1 \\
&= 8b_1 + 4 + 2 + 1 & &= 8(1) + 4 + 2 + 1 \\
&= 15
\end{aligned}
$$

so it takes 15 moves to transfer 4 disks from X to Z, by this strategy.

The next example also illustrates the same technique. We will take it a step further in Chapter 6.

Week 12

EXAMPLE 5.5

Imagine n lines in a plane such that no two lines are parallel, and no three are concurrent.* Let f_n denote the number of distinct regions into which the plane is divided by them. Define f_n recursively.

SOLUTION:

If there is just one line ℓ_1 in the plane, then $f_1 = 2$ (see Figure 5.4). Now consider a second line ℓ_2; it is intersected at exactly one point by ℓ_1. Each half of ℓ_2 divides an original region into two, adding two more regions (see Figure 5.5). Thus $f_2 = f_1 + 2 = 4$. Suppose we add a third line ℓ_3. It is

Figure 5.4

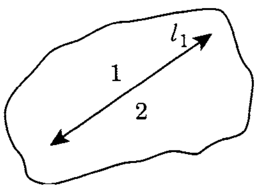
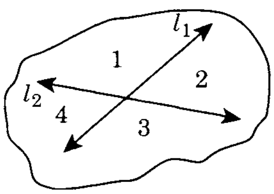


Figure 5.5



*Three or more lines in a plane are **concurrent** if they intersect at a point.

Stirling numbers of the second kind, denoted by $S(n, r)$ and used in combinatorics, are defined recursively as follows, where $n, r \in \mathbb{N}$:

$$S(n,r) = \begin{cases} 1 & \text{if } r = 1 \text{ or } r = n \\ S(n-1, r-1) + rS(n-1, r) & \text{if } 1 < r < n \\ 0 & \text{if } r > n \end{cases}$$

They are named after the English mathematician James Stirling (1692–1770). Compute each Stirling number.

62. $S(2, 2)$

63. $S(5, 2)$

A function of theoretical importance in the study of algorithms is the **Ackermann's function**, named after the German mathematician and logician Wilhelm Ackermann (1896–1962). It is defined recursively as follows, where $m, n \in \mathbf{W}$:

$$A(m,n) = \begin{cases} n+1 & \text{if } m = 0 \\ A(m-1, 1) & \text{if } n = 0 \\ A(m-1,\ A(m, n-1)) & \text{otherwise} \end{cases}$$

Compute each.

64. $A(0, 7)$

65. $A(1, 1)$

66. $A(4, 0)$

67. $A(2, 2)$

Prove each for $n \geq 0$.

68. $A(1, n) = n + 2$

69. $A(2, n) = 2n + 3$

***70.** Predict a formula for $A(3, n)$.

***71.** Prove the formula in Exercise 70, where $n \geq 0$.

5.2 Solving Recurrence Relations

↓ Week 12 (part 3)

The recursive definition of a function f does not provide us with an explicit formula for $f(n)$, but establishes a systematic procedure for finding it. This section illustrates the iterative method of finding a formula for $f(n)$ for a simple class of recurrence relations.

Solving the recurrence relation for a function f means finding an explicit formula for $f(n)$. The **iterative method** of solving it involves two steps:

- Apply the recurrence formula iteratively and look for a pattern to predict an explicit formula.
- Use induction to prove that the formula does indeed hold for every possible value of the integer n.

The next example illustrates this method.

EXAMPLE 5.10 (The handshake problem continued) By Example 5.3, the number of handshakes made by n guests at a dinner party is given by

$$h(1) = 0$$
$$h(n) = h(n-1) + (n-1), n \geq 2$$

Solve this recurrence relation.

SOLUTION:

Step 1 To predict a formula for $h(n)$:

Using iteration,
$$\begin{aligned} h(n) &= h(n-1) + (n-1) \\ &= h(n-2) + (n-2) + (n-1) \\ &= h(n-3) + (n-3) + (n-2) + (n-1) \\ &\vdots \\ &= h(1) + 1 + 2 + 3 + \cdots + (n-2) + (n-1) \\ &= 0 + 1 + 2 + 3 + \cdots + (n-1) \\ &= \frac{n(n-1)}{2} \end{aligned}$$

Step 2 To prove, by induction, that $h(n) = \frac{n(n-1)}{2}$, where $n \geq 1$:

Basis step When $n = 1, h(1) = \frac{1 \cdot 0}{2} = 0$, which agrees with the initial condition. So the formula holds when $n = 1$.

Induction step Assume $h(k) = \frac{k(k-1)}{2}$ for any $k \geq 1$. Then:

$$h(k+1) = h(k) + k, \qquad \text{by the recurrence relation}$$

$$= \frac{k(k-1)}{2} + k, \qquad \text{by the induction hypothesis}$$
$$= \frac{k(k+1)}{2}$$

Therefore, if the formula holds for $n = k$, it also holds for $n = k + 1$. Thus, by PMI, the result holds for $n \geq 1$. ■

More generally, using iteration we can solve the recurrence relation

$$a_n = a_{n-1} + f(n) \tag{5.5}$$

as follows:

$$\begin{aligned}
a_n &= a_{n-1} + f(n) \\
&= [a_{n-2} + f(n-1)] + f(n) = a_{n-2} + f(n-1) + f(n) \\
&= [a_{n-3} + f(n-2)] + f(n-1) + f(n) \\
&= a_{n-3} + f(n-2) + f(n-1) + f(n) \\
&\vdots \\
&= a_0 + \sum_{i=1}^{n} f(i)
\end{aligned} \tag{5.6}$$

You can verify that this is the actual solution of the recurrence relation (5.5).

For example, in the handshake problem $f(n) = n - 1$ and $h(0) = 0$, so the solution of the recurrence relation is

$$\begin{aligned}
h(n) &= h(0) + \sum_{i=1}^{n} f(i) = 0 + \sum_{i=1}^{n} (i-1) \\
&= \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}, \quad n \geq 1
\end{aligned}$$

which is exactly the solution obtained in the example.

EXAMPLE 5.11 Solve the recurrence relation in Example 5.6.

SOLUTION:

Notice that a_n can be redefined as

$$a_n = a_{n-1} + \frac{n(n+1)}{2}, \quad n \geq 1$$

where $a_0 = 0$. Comparing this with recurrence relation (5.5), we have $f(n) = \dfrac{n(n+1)}{2}$. Therefore, by Equation (5.6),

$$
\begin{aligned}
a_n &= a_0 + \sum_{i=1}^{n} f(i) \\
&= a_0 + \sum_{i=1}^{n} \frac{i(i+1)}{2} = 0 + \frac{1}{2}\sum_{i=1}^{n}(i^2 + i) \\
&= \frac{1}{2}\left(\sum_{i=1}^{n} i^2 + \sum_{i=1}^{n} i\right) \\
&= \frac{1}{2}\left[\frac{n(n+1)(2n+1)}{6} + \frac{n(n+1)}{2}\right] \\
&= \frac{n(n+1)}{2}\left(\frac{2n+1}{6} + \frac{1}{2}\right) = \frac{n(n+1)}{2} \cdot \frac{2n+4}{6} \\
&= \frac{n(n+1)(n+2)}{6}, \quad n \geq 0
\end{aligned}
$$

■

The following illustration of the iterative method brings us again to the Tower of Brahma puzzle.

EXAMPLE 5.12 Recall from Example 5.4 that the number of moves needed to transfer n disks from peg X to peg Z is given by

$$
\begin{aligned}
b_1 &= 1 \\
b_n &= 2b_{n-1} + 1, n \geq 2
\end{aligned}
$$

Solve this recurrence relation.

SOLUTION:

Step 1 To predict a formula for b_n:
Using iteration,

$$
\begin{aligned}
b_n &= 2b_{n-1} + 1 \\
&= 2[2b_{n-2} + 1] + 1 = 2^2 b_{n-2} + 2 + 1 \\
&= 2^2[2b_{n-3} + 1] + 2 + 1 = 2^3 b_{n-3} + 2^2 + 2 + 1 \\
&\vdots \\
&= 2^{n-1} b_1 + 2^{n-2} + \cdots + 2^2 + 2 + 1 \\
&= 2^{n-1} + 2^{n-2} + \cdots + 2 + 1 \\
&= 2^n - 1, \quad \text{by Exercise 8 in Section 4.4.}
\end{aligned}
$$

Step 2 You may prove by induction that $b_n = 2^n - 1$, where $n \geq 1$. ■

More generally, you may verify that the solution of the recurrence relation $a_n = ca_{n-1} + 1$, where c is a constant ($\neq 1$), is

$$a_n = c^n a_0 + \frac{c^n - 1}{c - 1}$$

For instance, in Example 5.12, $b_0 = 0$ and $c = 2$, so

$$b_n = 2^n \cdot 0 + \frac{2^n - 1}{2 - 1} = 2^n - 1$$

as expected.

Let us pursue Example 5.12 a bit further. Suppose there are 64 disks at peg X, as in the original puzzle, and it takes 1 second to move a disk from one peg to another. Then it takes a total of $2^{64} - 1$ seconds to solve the puzzle.

To get an idea how incredibly large this total is, notice that there are about $365 \cdot 24 \cdot 60 \cdot 60 = 31{,}536{,}000$ seconds in a year. Therefore,

$$\begin{aligned} \text{Total time taken} &= 2^{64} - 1 \text{ seconds} \\ &\approx 1.844674407 \times 10^{19} \text{ seconds} \\ &\approx 5.84942417 \times 10^{11} \text{ years} \\ &\approx 600 \text{ billion years!} \end{aligned}$$

Ch 2.5 + 5.1 + 5.2
6:21 | 8:47 read (2.5h)
↑ Week 12

Intriguingly, according to some estimates, the universe is only about 18 billion years old.

an 1, 3, 6, 10, 15, 21, 28
n 1 2 3 4 5 6 7
(1,1) (2,3) ~~(3,6)~~ (4,10) (5,15) (6,21) (7,28)
n·x
n·x+n

Exercises 5.2

12/8 7:07 | do 8:11 | correct 8:16

Using the iterative method, predict a solution to each recurrence relation satisfying the given initial condition.

√ **1.** $s_0 = 1$, $s_n = 2s_{n-1}, n \geq 1$

$s_1 = 2$, $s_2 = 4$, $s_3 = 8$ $\quad s_n = 2^n, n \geq 0$ ✓

× **2.** $a_1 = 1$, $a_n = a_{n-1} + n, n \geq 2$

$a_2 = 3$, $a_3 = 6$, $a_4 = 10$

$$\begin{aligned} a_n &= a_{n-1} + n \\ &= a_{n-2} + n + n \\ &= a_{n-3} + n + n + n \\ &\vdots \\ &= a_1 + n(n-1) \\ &= n^2 - n + 1 \end{aligned}$$

(but n=3 is wrong

× **3.** $a_0 = 1$, $a_n = a_{n-1} + n, n \geq 1$

$a_1 = 2$, $a_2 = 4$, $a_3 = 7$ $\quad a_n = \frac{n(n+1)}{2} + 1$, $n \geq 1$

$$\begin{aligned} a_n &= a_{n-1} + n \\ &= a_{n-2} + n + n \\ &= a_{n-3} + n + n + n \\ &\vdots \\ &= a_0 + n^2 \\ &= n^2 + 1 \end{aligned}$$

(but n=2 is wrong!!)

× **4.** $a_1 = 1$, $a_n = a_{n-1} + (2n - 1), n \geq 2$

$a_2 = 4$, $a_3 = 9$, $a_4 = 16$ $\quad a_n = n^2, n \geq 1$

× **5.** $a_0 = 0$, $a_n = a_{n-1} + 4n, n \geq 1$

$a_1 = 4$, $a_2 = 12$, $a_3 = 24$ $\quad a_n = 2n(n+1) = 2n^2 + 2n$, $n \geq 0$

$$\begin{aligned} a_n &= a_{n-1} + 4n \\ &= a_{n-2} + 4n + 4n \\ &= a_{n-3} + 4n + 4n + 4n \\ &\vdots \\ &= a_0 + 4n \cdot n \\ &= 4n^2 \end{aligned}$$

(but n=2 is wrong!!)

× **6.** $s_1 = 1$, $s_n = s_{n-1} + n^3, n \geq 2$

$s_2 = 9$ 3^2, $s_3 = 36$ 6^2, $s_4 = 100$ 10^2

$(\sqrt{s_{n-1}} + n)^2$

$$\begin{aligned} s_n &= s_{n-1} + n^3 \\ &= s_{n-2} + n^3 + n^3 \\ &= s_{n-3} + n^3 + n^3 + n^3 \\ &\vdots \\ &= s_1 + n^3 \cdot (n-1) \\ &= n^4 - n^3 + 1 \end{aligned}$$

(n=3 is wrong!!)

1. $s_n = 2^n, n \geq 1$

3. $a_n = n(n+1)/2 + 1, n \geq 1$

5. $a_n = 2n(n+1), n \geq 0$