

7.1 Sequences

Survey

The following questions are part of a zyBooks survey to help us improve our content so we can offer the best experience for students. The survey can be taken anonymously and takes just 3-5 minutes. Please take a short moment to answer by clicking the following link.

Link: [Student survey](#)

A **sequence** is a special type of function in which the domain is a set of consecutive integers. For example, a sequence can be defined to denote a student's GPA for each of the four years the student attended college. The domain of the function is $\{1, 2, 3, 4\}$ for each of the four years. The function g is specified by the numerical values for the GPA in each year:

$$g(1) = 3.67, \quad g(2) = 2.88, \quad g(3) = 3.25, \quad g(4) = 3.75$$

When a function is specified as a sequence, using subscripts to denote the input to the function is more common, so g_k is used instead of $g(k)$. A value g_k is called a **term** of a sequence, and k is the **index** of g_k . The sequence is written:

$$g_1 = 3.67, \quad g_2 = 2.88, \quad g_3 = 3.25, \quad g_4 = 3.75$$

When the function name, as in g , and the indices are understood (or not important), the sequence can be specified by listing just the terms: **3.67, 2.88, 3.25, 3.75**.

The entire sequence is denoted by $\{g_k\}$, whereas g_k (without the curly braces) is used to denote a single term in the sequence. While the first index is commonly 0 or 1, a sequence may start with any integer, as in the following example:

$$a_{-2} = 0, \quad a_{-1} = 1, \quad a_0 = 1, \quad a_1 = 0$$

A sequence with a finite domain is called a **finite sequence**. In a finite sequence, there is an **initial index** m and a **final index** n , where $n \geq m$. Then a_m is the **initial term** and a_n is the **final term**.

$$a_m, \quad a_{m+1}, \quad \dots, \quad a_n$$

A sequence with an infinite domain is called an **infinite sequence**. In an infinite sequence, the indices go to infinity in the positive direction. There is an initial index m , and the sequence is

defined for all indices k such that $k \geq m$:

$$a_m, a_{m+1}, a_{m+2}, \dots$$

A sequence can be specified by an **explicit formula** showing how the value of term a_k depends on k . For example, $d_k = 2^k$ for $k \geq 1$. The infinite sequence $\{d_k\}$ starts with: 2, 4, 8, 16,

**PARTICIPATION
ACTIVITY**

7.1.1: Sequence basics.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Animation captions:

1. Finite sequence example: $S_1 = 2, S_2 = 5, S_3 = -1, S_4 = 7, S_5 = 8$. The sequence $\{S_k\}$ is finite. The indices are 1, 2, 3, 4, 5.
2. The terms are 2, 5, -1, 7, 8.
3. 1 is the initial index. The initial term is 2. 5 is the final index. The final term is 8.
4. $\{a_k\}$ is an infinite sequence defined by $a_k = \frac{1}{k}$, for $k = 1, 2, 3, \dots$. $\{a_k\}$ starts with 1, 1/2, 1/3, $a_1 = 1$ is the initial term in the sequence.

**PARTICIPATION
ACTIVITY**

7.1.2: Sequence basics.

Define the sequence $\{a_k\}$, beginning with $a_1 = 7$:

$$7, 10, 13, 16, 19, 22, 25$$

1) What is a_2 ?

Check

[Show answer](#)

2) What is the index of the term 16 in the sequence?

Check

[Show answer](#)

3) What is the final index of the sequence?

Check[Show answer](#)

- 4) Consider the sequence defined by the formula $b_k = k^2$ for $k \geq 2$. What is the third term in the sequence?

Check[Show answer](#)

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Increasing and decreasing sequences

A sequence is **increasing** if for every two consecutive indices, k and $k + 1$, in the domain, $a_k < a_{k+1}$. A sequence is **non-decreasing** if for every two consecutive indices, k and $k + 1$, in the domain, $a_k \leq a_{k+1}$. Notice that an increasing sequence is always non-decreasing as well, because if $a_k < a_{k+1}$ then it is also true that $a_k \leq a_{k+1}$.

Figure 7.1.1: Increasing and non-decreasing sequences.




A sequence is **decreasing** if for every two consecutive indices, k and $k + 1$, in the domain, $a_k > a_{k+1}$. A sequence is **non-increasing** if for every two consecutive indices, k and $k + 1$, in the domain, $a_k \geq a_{k+1}$.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Figure 7.1.2: Decreasing and non-increasing sequences.

6, > 5, > 4, > 2 Decreasing *and* non-increasing

6, ≥ 4, ≥ 4, ≥ 2 Non-increasing *but not* decreasing



©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

PARTICIPATION ACTIVITY

7.1.3: Increasing and decreasing sequences.



Select the choice that describes the sequence.

1) 1, 2, 3, 3



- ☐ Non-decreasing
- ☐ Non-increasing
- ☐ Increasing

2) 4, 3.5, 2, -1



- ☐ Non-increasing but not decreasing
- ☐ Both non-increasing and non-decreasing
- ☐ Decreasing and non-increasing

3) 2, 2, 2, 2



- ☐ Neither non-increasing nor non-decreasing
- ☐ Non-increasing and non-decreasing
- ☐ Non-increasing but not non-decreasing

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Geometric sequences and arithmetic sequences are two types of sequences that arise frequently in discrete mathematics. Both types of sequences are described below.

Geometric sequences

Definition 7.1.1: Definition of a geometric sequence.

A **geometric sequence** is a sequence of real numbers where each term after the initial term is found by taking the previous term and *multiplying* by a fixed number called the **common ratio**. A geometric sequence can be finite or infinite.

PARTICIPATION ACTIVITY

7.1.4: Geometric sequences.



Animation captions:

1. Geometric sequence example: the initial term = 4, and the common ratio = $1/2$. The second term is the initial term times the common ratio: $4 \times \frac{1}{2} = 2$.
2. The next term is $2 \times \frac{1}{2} = 1$ and the one after that is $1 \times \frac{1}{2} = \frac{1}{2}$, etc

The infinite geometric sequence with initial term $a_0 = 1$ and common ratio $r = 1/2$ begins with:

$$1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \dots$$

The finite geometric sequence with initial term $a_0 = 5$, final index 6, and common ratio $r = -1$ is:

$$5, -5, 5, -5, 5, -5, 5$$

If the initial index of an infinite geometric sequence $\{s_k\}$ is 0, then the explicit formula defining the sequence is: $s_k = a \cdot r^k$, for $k \geq 0$. Here, $s_0 = a \cdot r^0 = a$ is the initial term in the sequence.

Example 7.1.1: Interest rates and geometric sequences.

Money in a bank account earning a fixed rate of interest can be expressed as a geometric sequence. Suppose \$1000 is stored in a bank account that earns 6% annual interest compounded monthly. Since the interest rate is annual and compounded monthly, $(6/12)\%$ of the current amount is added to the account each month. $a_0 = 1000$ is the initial balance in the account, and a_n is the balance in the account after n months of earning interest. Each month, the balance in the account increases by $(6/12)\% = 0.005$, which means that the balance is 1.005 times the amount that was in the account in the previous month. The sequence $\{a_n\}$ is a geometric sequence with $a_n = 1000 \cdot (1.005)^n$ for $n \geq 0$.

PARTICIPATION ACTIVITY

7.1.5: Geometric sequences.



- 1) Let $\{s_n\}$ be a geometric sequence that starts with an initial index of 0. The initial term is 2 and the common ratio is 5. What is s_2 ?

**Check**[Show answer](#)

- 2) Let $\{s_n\}$ be a geometric sequence that starts with an initial index of 0. The initial term is 16 and the common ratio is $1/2$. What is s_3 ?

**Check**[Show answer](#)

- 3) Consider the geometric sequence: 3, 6, 12,
What is the common ratio?

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022



Check**Show answer**

Arithmetic sequences

Definition 7.1.2: Definition of an arithmetic sequence.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

An **arithmetic sequence** is a sequence of real numbers where each term after the initial term is found by taking the previous term and *adding* a fixed number called the **common difference**. An arithmetic sequence can be finite or infinite.

**PARTICIPATION
ACTIVITY**

7.1.6: Arithmetic sequences.



Animation content:

undefined

Animation captions:

1. Arithmetic sequence example: the initial term = -1, and the common difference = 2.5. The second term is: $-1 + 2.5 = 1.5$.
2. The next term is $1.5 + 2.5 = 4$ and the one after that is $4 + 2.5 = 6.5$, etc.

The infinite arithmetic sequence with initial term $a_0 = 2$ and common difference $d = 3$ begins with:

2, 5, 8, 11,

The finite arithmetic sequence with initial term $a_0 = 3$, final index 5, and common difference $d = -2$ is:

3, 1, -1, -3, -5, -7

If the initial index of an arithmetic sequence $\{t_n\}$ is 0, then the explicit formula defining the sequence is: $t_n = t_0 + dn$, for $n \geq 0$, where d is the common difference .

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Example 7.1.2: Accumulated resources and arithmetic sequences.

Suppose a person inherits a collection of 500 baseball cards and decides to continue growing the collection at a rate of 10 additional cards each week. a_n is the number of cards in the collection after n weeks of collecting. Since the collection starts with 500 cards, $a_0 = 500$. The sequence $\{a_n\}$ is an arithmetic sequence with an initial value of 500 and a common difference of 10. After n weeks of collecting, $a_n = 500 + 10n$.

PARTICIPATION ACTIVITY

7.1.7: Arithmetic sequences.



- 1) Let $\{s_n\}$ be an arithmetic sequence that starts with an initial index of 0. The initial term is 3 and the common difference is -2. What is s_2 ?

**Check**[Show answer](#)

- 2) Consider the arithmetic sequence: 7, 4, 1, What is the next term in the sequence?

**Check**[Show answer](#)

CHALLENGE ACTIVITY

7.1.1: Sequences.



422102.2723990.qx3zqy7

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Start

Enter the next three numbers in the sequence of the positive integers

3, 4, 5, , 9.....

1	2	3	4	5	6
---	---	---	---	---	---

Check

Next

Additional exercises

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022



EXERCISE

7.1.1: Evaluating sequences.



Give the first ten terms of the following sequences. You can assume that the sequences start with an index of 1. Logs are to base 2.

Indicate whether the sequence is increasing, decreasing, non-increasing, or non-decreasing. The sequence may have more than one of those properties.

- (a) The n^{th} term is $\lceil \sqrt{n} \rceil$.
- (b) The first two terms in the sequence are 1. The rest of the terms are the sum of the two preceding terms.
- (c) The n^{th} term is the largest integer k such that $k! \leq n$.
- (d) The n^{th} term is $1/n$.
- (e) The n^{th} term is 3.
- (f) The n^{th} term is n^2 .
- (g) The n^{th} term is $\lceil \log n \rceil$.
- (h) The n^{th} term is $2^{\lceil \log n \rceil}$.
- (i) The n^{th} term is $\lceil \frac{-n}{2} \rceil$.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022



EXERCISE

7.1.2: Describing sequence properties.



For each of the sequences described below, indicate which of the following properties describes the sequence, keeping in mind that the sequence may have more than one of the properties, or none of the properties:

- Increasing
- Decreasing
- Non-increasing
- Non-decreasing

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

- (a) $a_n = n^2 - 2n$, for $n \geq 1$
- (b) $a_n = n^2 - 3n$, for $n \geq 1$
- (c) $a_n = n^2 - 4n$, for $n \geq 1$
- (d) $a_n = 2^n - n!$, for $n \geq 1$
- (e) $a_n = 2^n - 3^n$, for $n \geq 1$



EXERCISE

7.1.3: Arithmetic and geometric sequences.



Give the first six terms of the following sequences. You can assume that the sequences start with an index of 1.

- (a) A geometric sequence in which the first value is 2 and the common ratio is 3.
- (b) An arithmetic sequence in which the first value is 2 and the common difference is 3.
- (c) A geometric sequence in which the first value is 27 and the common ratio is $1/3$.
- (d) An arithmetic sequence in which the first value is 3 and the common difference is $-1/2$.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022



EXERCISE

7.1.4: Arithmetic sequences, geometric sequences and sequence properties.



- (a) What are the conditions on the common ratio r and initial value a , that would make the resulting geometric sequence increasing?
- (b) What are the conditions on the common ratio r and initial value a , that would make the resulting geometric sequence decreasing?
- (c) What are the conditions on the common difference d and initial value a , that would make the resulting arithmetic sequence increasing?
- (d) What are the conditions on the common difference d and initial value a , that would make the resulting arithmetic sequence decreasing?
- (e) Is it possible to have an arithmetic sequence that is non-decreasing but not increasing?
- (f) Is it possible to have a geometric sequence that is non-increasing but not decreasing?

7.2 Recurrence relations

Some sequences are most naturally defined by specifying one or more initial terms and then giving a rule for determining subsequent terms from earlier terms in the sequence. A rule that defines a term a_n as a function of previous terms in the sequence is called a **recurrence relation**. For example, in an arithmetic sequence, each number is obtained by adding a fixed value to the previous number.

Example 7.2.1: Recurrence relation defining an arithmetic sequence.

$$a_0 = a \quad (\text{initial value})$$

$$a_n = d + a_{n-1} \quad \text{for } n \geq 1 \quad (\text{recurrence relation})$$

Initial value = a . Common difference = d .

Geometric sequences are also naturally specified by recurrence relations. In a geometric sequence, each number is obtained by multiplying the previous number by a fixed constant.

Example 7.2.2: Recurrence relation defining a geometric sequence.

$$a_0 = a \quad (\text{initial value})$$

$$a_n = r \cdot a_{n-1} \quad \text{for } n \geq 1 \quad (\text{recurrence relation})$$

Initial value = a . Common ratio = r .

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Leonardo Fibonacci, a prominent mathematician in the middle ages, used a recurrence relation to define a sequence that is now known as the **Fibonacci sequence**. Fibonacci defined the sequence in an attempt to mathematically describe the population growth of rabbits. The colony starts with one pair of newborn rabbits. The rabbits must be at least one month old before they can reproduce. Every pair of reproducing rabbits gives birth to a new pair of rabbits, one male and one female over the course of a month.

Suppose that f_n is the number of pairs of rabbits at the end of month n . Assume that the first pair of rabbits obtained for the colony are born at the end of month 1, so $f_0 = 0$ and $f_1 = 1$. In month n , the colony has all the rabbits from the previous month (f_{n-1}) plus all the new rabbits born in that month. Since each pair of reproducing rabbits gives rise to a new pair, the number of new rabbits appearing in month n is the same as the number of pairs of rabbits that are able to reproduce in month n , which is f_{n-2} , the number of pairs that were alive at least one full month before the beginning of month n . Thus, the Fibonacci sequence is defined as:

$$\begin{aligned} f_0 &= 0 \\ f_1 &= 1 \\ f_n &= f_{n-1} + f_{n-2} \quad \text{for } n \geq 2 \end{aligned}$$

Interestingly, the Fibonacci sequence also arises naturally in computer science in analyzing certain data structures and algorithms for data search. Recurrence relations are used to describe phenomena in many diverse areas of business, engineering, and science. Recurrence relations are especially useful in computer science for modeling the running time of recursive algorithms.

PARTICIPATION ACTIVITY

7.2.1: The Fibonacci sequence.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Animation captions:

1. The initial values for the Fibonacci sequence are $f_0 = 0$ and $f_1 = 1$.
2. $f_2 = f_1 + f_0 = 1 + 0 = 1$
3. $f_3 = f_2 + f_1 = 1 + 1 = 2$
4. $f_4 = f_3 + f_2 = 2 + 1 = 3$

PARTICIPATION
ACTIVITY

7.2.2: Recurrence relations.



Define the sequence $\{b_n\}$ as:

$$b_0 = 1$$

$$b_1 = 1$$

$$b_n = b_{n-1} \cdot b_{n-2} + 2 \quad \text{for } n \geq 2$$

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

1) What is b_2 ?



Check

Show answer

2) What is b_3 ?



Check

Show answer

CHALLENGE
ACTIVITY

7.2.1: Recurrence Relations.



Note: You may want to use paper and pencil, and/or a calculator.

422102.2723990.qx3zqy7

Start

Given the definition of a sequence f_n whose domain is the set of all non-negative integers:

$$f_0 = 1$$

$$f_n = f_{n-1} + 2n \quad \text{for } n \geq 1$$

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

$$f_3 = \text{Ex: 5} \quad \updownarrow$$

1	2	3	4
---	---	---	---

Check

Next

The number of initial values required to define a sequence using a recurrence relation depends on which previous terms in the sequence are used to define the n^{th} term. In the Fibonacci sequence, f_n depends on the previous two terms, so the definition of the Fibonacci sequence includes the two initial values for the sequence. f_0 and f_1 are both required to define f_2 . If only f_0 was given, then the recurrence relation could not be applied to $f_1 = f_0 + f_{-1}$ because f_{-1} is not defined.

**PARTICIPATION
ACTIVITY**

7.2.3: Initial values for recurrence relations.



Each question below gives a recurrence relation. How many initial values must be given for the sequence to be well defined for all $n \geq 0$?

1) $c_n = 3 \cdot c_{n-1} + 2 \cdot c_{n-2}$



Check

[Show answer](#)

2) $d_n = (n-1) \cdot d_{n-2}$



Check

[Show answer](#)

3) $b_n = b_{n-3} \cdot b_{n-2}$



Check

[Show answer](#)

Fibonacci's rabbit colony is an example of a dynamical system. A **dynamical system** is a system that changes over time. Moreover, the state of the system at any point in time is determined by a set of well defined rules that depend on the past states of the system. In a **discrete time dynamical system**, time is divided into discrete time intervals and the state of the system stays fixed within each time interval. The state during one time interval is a function of the state in previous time intervals. Thus, the history of the system is defined by a sequence of states, indexed by the non-negative integers.

In the case of Fibonacci's rabbits, each time interval is a month and the state of the system is simply the number of pairs of rabbits. Dynamical systems are useful in studying phenomena in biology, engineering, physics, economics, and finance. Many dynamical systems naturally give rise

to recurrence relations that describe how the system changes over time.

Example 7.2.3: Outstanding balance on a car loan.

An individual takes out a \$20,000 car loan. The annual interest rate for the loan is 3%. He wishes to make a monthly payment of \$500. Define a_n to be the amount of outstanding debt after n months. Since the interest rate describes the annual interest, the percentage increase each month is actually $3\% / 12 = 0.25\%$. Thus, the multiplicative factor increase each month is 1.0025. The recurrence relation for $\{a_n\}$ is:

$$\begin{aligned} a_0 &= \$20,000 \\ a_n &= (1.0025) \cdot a_{n-1} - 500 \quad \text{for } n \geq 1 \end{aligned}$$

The $(1.0025) \cdot a_{n-1}$ term describes the increase in debt based on the interest rate. The -500 term describes the decrease due to the monthly payment. The first few values, to the nearest dollar, for the sequence $\{a_n\}$ are:

$$\begin{aligned} a_0 &= \$20,000 \\ a_1 &= \$19,550 \\ a_2 &= \$19,099 \\ a_3 &= \$18,647 \\ &\dots \end{aligned}$$

PARTICIPATION ACTIVITY

7.2.4: Defining sequences by recurrence relations.



Each question describes a dynamical system and the corresponding sequence. Select the recurrence relation that describes the sequence.



- 1) Let s_n be the salary of an employee in year n . The employee receives a \$1000 bonus in December. The salary for the next year is 5% more than the income received by the employee in the previous year (including salary and bonus).

- ☐ $s_n = 1.05 \cdot (s_{n-1}) + 1000$
☐ $s_n = 1.5 \cdot (s_{n-1} + 1000)$
☐ $s_n = 1.05 \cdot (s_{n-1} + 1000)$

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022



2) 25% of the yeast cells in a petri dish divide every hour. When a single cell divides, it becomes two cells instead of one and the number of cells goes up by one. Let y_n be the number of yeast cells in the petri dish after n hours.

- ☐ $y_n = (0.25) y_{n-1}$
- ☐ $y_n = (1.25) y_{n-1}$
- ☐ $y_n = (.25) y_{n-1} + y_{n-2}$

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Additional exercises



EXERCISE

7.2.1: Evaluating recursively defined sequences.



Give the first six terms of the following sequences.

- (a) The first term is 1 and the second term is 2. The rest of the terms are the product of the two preceding terms.
- (b) $a_1 = 1$, $a_2 = 5$, and $a_n = 2 \cdot a_{n-1} + 3 \cdot a_{n-2}$ for $n \geq 3$.
- (c) $g_1 = 2$ and $g_2 = 1$. The rest of the terms are given by the formula $g_n = n \cdot g_{n-1} + g_{n-2}$.
- (d) $c_1 = 4$, $c_2 = 5$, and $c_n = c_{n-1} \cdot c_{n-2}$ for $n \geq 3$.
- (e) $b_1 = 1$, $b_2 = 3$, and $b_n = b_{n-1} - 7 \cdot b_{n-2}$ for $n \geq 3$.
- (f) $d_1 = 1$, $d_2 = 1$, and $d_n = (d_{n-1})^2 + d_{n-2}$ for $n \geq 3$.
- (g) $f_1 = 0$, $f_2 = 2$, and $f_n = 5 \cdot f_{n-1} - 2 \cdot f_{n-2}$ for $n \geq 3$.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022



EXERCISE

7.2.2: Refinancing a loan.



Suppose someone takes out a home improvement loan for \$30,000. The annual interest on the loan is 6% and is compounded monthly. The monthly payment is \$600. Let a_n denote the amount owed at the end of the n th month. The payments start in the first month and are due the last day of every month.

- (a) Give a recurrence relation for a_n . Don't forget the base case.
- (b) Suppose that the borrower would like a lower monthly payment. How large does the monthly payment need to be to ensure that the amount owed decreases every month?

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022



EXERCISE

7.2.3: A variation on Fibonacci's rabbits.



- (a) Give a recurrence relation for the population of Fibonacci's rabbits if the rabbits must be at least two months old to reproduce.

The colony starts with one pair of newborn rabbits. Every pair of reproducing rabbits gives birth to a new pair of rabbits, one male and one female over the course of a month. Let g_n denote the number of pairs of rabbits at the end of month n . Assume that the first pair of rabbits obtained for the colony are born at the end of month 1.

7.3 Summations

Summation notation is used to express the *sum* of terms in a numerical sequence. Consider a sequence:

$$a_s, a_{s+1}, \dots, a_t$$

The notation to express the sum of the terms in that sequence is:

$$\sum_{i=s}^t a_i = a_s + a_{s+1} + \dots + a_t$$

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

In the summation $\sum_{i=s}^t a_i$, the variable i is called the **index** of the summation. The value s is the **lower limit** and t is the **upper limit** of the summation. Any variable name could be used for the index instead of i , but variables i , j , k , and l are the most common. The capital letter sigma Σ is used to denote the fact that the terms are to be added together. Summation notation can be used to sum up the terms in a sequence defined by an explicit formula:

$$a_n = n^3, \quad \text{for } n = 1, 2, 3, 4$$

Then:

$$\sum_{j=1}^4 j^3 = 1^3 + 2^3 + 3^3 + 4^3 = 1 + 8 + 27 + 64 = 100$$

The expression $\sum_{j=1}^4 j^3$ is called the **summation form** of the sum, and the expression $1^3 + 2^3 + 3^3 + 4^3$ is called the **expanded form** of the sum. When the expression to be summed has more than one term, it is important to use parentheses to indicate that all the terms are included in the summation as in $\sum_{j=1}^n (j + 1)$ because the expression $\sum_{j=1}^n j + 1$ is interpreted as $(\sum_{j=1}^n j) + 1$.

PARTICIPATION ACTIVITY

7.3.1: Computing summations.



Animation captions:

$$1. \sum_{j=-1}^2 (j^2 + 1) = ((-1)^2 + 1) + (0^2 + 1) + (1^2 + 1) + (2^2 + 1) = 10$$

PARTICIPATION ACTIVITY

7.3.2: Computing summations.



Give numerical values for the following summations.

$$1) \sum_{j=2}^5 (2j - 1)$$



Check

Show answer

$$2) \sum_{j=0}^2 (j + 1)^2$$



Check

Show answer

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Pulling out a final term from a summation

In working with summations, it is often useful to pull out or add in a final term to a summation:

$$\sum_{k=m}^n a_k = \sum_{k=m}^{n-1} a_k + a_n, \quad \text{for } n > m$$

The drawing below illustrates the idea with an example. The expanded form of the sum is given to show why the two expressions are equal:

Figure 7.3.1: Pulling out a final term from a summation.

$$\sum_{j=1}^n (j+1)^2 = (1+1)^2 + (2+1)^2 + \dots + ((n-1)+1)^2 + (n+1)^2$$

$$= \sum_{j=1}^{n-1} (j+1)^2 + (n+1)^2$$

The diagram shows a green bracket under the first four terms of the expanded sum, with an arrow pointing to the summation term in the second equation. A green arrow also points from the last term of the first equation to the separate term in the second equation.

**PARTICIPATION
ACTIVITY**

7.3.3: Removing a final term in a sum.



Select the choice whose value is equal to the given sum.

1) $\sum_{j=0}^n 2^j$



☐ $\sum_{j=0}^{n-1} 2^j + 2^j$

☐ $\sum_{j=0}^{n-1} 2^j + 2^{n-1}$

☐ $\sum_{j=0}^{n-1} 2^j + 2^n$

☐ $\sum_{j=0}^n 2^j + 2^n$

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

2) $\sum_{j=0}^{n+2} 2^{j-1}$

☐ $\sum_{j=0}^{n+1} 2^{j-1} + 2^n$

☐ $\sum_{j=0}^{n-1} 2^{j-1} + 2^{n+1}$

☐ $\sum_{j=0}^{n+1} 2^{j-1} + 2^{n+1}$

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

**CHALLENGE
ACTIVITY**

7.3.1: Pulling out a final term from a summation.

422102.2723990.qx3zqy7

Start

Fill in the missing values such that the final term is pulled out of the summation.

$$\sum_{i=1}^9 i = \sum_{i=1}^{\text{Ex: 5}} i + \text{Ex: 5}$$

1	2	3
---	---	---

Check

Next

Change of variables in summations

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

When a variable is used to denote the lower or upper limit of a sum, the value of the variable must be provided in order to evaluate the sum. In the summation below, the value of the sum depends on the variable n :

$$\sum_{j=1}^n (j+2)^3$$

By contrast, the variable j used for the index is internal to the sum and can be replaced with any other variable name:

$$\sum_{j=1}^n (j+2)^3 = \sum_{k=1}^n (k+2)^3$$

More complex substitutions can be done for the index variable which require that the upper and lower limit be adjusted. Consider the substitution $i = k + 2$. Here are the steps:

1. Determine the new lower limit: when $k = 1$, $i = k + 2 = 1 + 2 = 3$.
2. Determine the new upper limit: when $k = n$, $i = k + 2 = n + 2$.
3. Replace the variable in the terms: If $i = k + 2$ then $k = i - 2$. Replace k in the expression inside the sum with $i - 2$

$$(k+2)^3 = ((i-2)+2)^3 = i^3.$$

Putting it all together, we get

$$\sum_{k=1}^n (k+2)^3 = \sum_{i=3}^{n+2} i^3$$

To verify that the two sums are the same, verify that the expanded form for both sums is:

$$3^3 + 4^3 + \cdots + (n+1)^3 + (n+2)^3$$

The animation below illustrates with another example:

PARTICIPATION ACTIVITY

7.3.4: Change of variables in summations.



Animation captions:

1. In the summation $\sum_{j=1}^{17} 2^{j-1}$, substitute $k = j - 1$.
2. Substitute the initial index $j = 1$ into the expression for k to get the initial condition for k : $k = j - 1 = 1 - 1 = 0$.
3. The new sum has an initial index of $k = 0$: $\sum_{k=0}$.
4. Substitute the final index $j = 17$ into the expression for k to get the final condition for

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

$k : k = j - 1 = 17 - 1 = 16$. The new sum is now $\sum_{k=0}^{16}$.

5. The equality $k = j - 1$ implies $j = k + 1$. Substitute the expression for j in the summation term: $2^{j-1} = 2^{(k+1)-1} = 2^k$. The final sum is $\sum_{k=0}^{16} 2^k$.

PARTICIPATION ACTIVITY

7.3.5: Change of variables in summations.



©zyBooks 12/15/22 00:21 1361995

John Farrell

COLOSTATECS220SeaboltFall2022

Substitute the variable i with $j = i + 2$ in the summation below:

$$\sum_{i=3}^{21} \frac{1}{(i+3)} = \sum_{j=?}^? ?$$

- 1) What is the lower limit of the sum with index j ?



Check

[Show answer](#)

- 2) What is the upper limit of the sum with index j ?



Check

[Show answer](#)

- 3) Give an expression for i in terms of j .



$i =$

Check

[Show answer](#)

- 4) The term in the summation is a fraction $\frac{1}{(i+3)} = \frac{1}{(?)}$. Give an expression for the missing denominator in terms of j .



Check

[Show answer](#)

©zyBooks 12/15/22 00:21 1361995

John Farrell

COLOSTATECS220SeaboltFall2022

Closed forms for sums

A **closed form** for a sum is a mathematical expression that expresses the value of the sum without summation notation. There are closed form expressions for some, although not all, of the summations that arise naturally in scientific applications. For example, there is a closed form expression for the sum of terms in an arithmetic sequence:

Theorem 7.3.1: Closed form for the sum of terms in an arithmetic sequence.

For any integer $n \geq 1$:

$$\sum_{k=0}^{n-1} (a + kd) = an + \frac{d(n-1)n}{2}$$

For example, in the sum $5 + 8 + 11 + 14 + 17 + 20 + 23$, the initial term $a = 5$, the common difference $d = 3$, and the number of terms in the sum $n = 7$. The theorem says that:

$$5 + 8 + 11 + 14 + 17 + 20 + 23 = \sum_{j=0}^6 (5 + 3j) = 5 \cdot 7 + \frac{3 \cdot 6 \cdot 7}{2} = 98$$

There is also a closed form expression for the sum of terms in a geometric sequence:

Theorem 7.3.2: Closed form for the sum of terms in a geometric sequence.

For any real number $r \neq 1$ and any integer $n \geq 1$:

$$\sum_{k=0}^{n-1} a \cdot r^k = \frac{a(r^n - 1)}{r - 1}$$

For example, in the sum $3 + 6 + 12 + 24 + 48 + 96$, the initial term $a = 3$, the common ratio $r = 2$, and the number of terms in the sum $n = 6$. The theorem says that:

$$3 + 6 + 12 + 24 + 48 + 96 = \sum_{j=0}^5 3 \cdot 2^j = \frac{3(2^6 - 1)}{1} = 189$$

The two theorems above will be proven in the section on mathematical induction.

Example 7.3.1: Total sales of a company with exponential growth.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Consider a growing company whose sales are expected to grow at a rate of 10% each month. That is, if the company sells x widgets in one month, the company is expected to sell $1.1(x)$ widgets in the following month. If the company sells 1000 widgets in January, what can the company project its total sales to be for the entire year?

The sequence of number of widgets sold each month is a geometric sequence so the total sales can be obtained by using the closed form:

$$\sum_{k=0}^{n-1} a \cdot r^k = \frac{a(r^n - 1)}{r - 1}$$

The initial value $a = 1000$. The common ratio $r = 1.1$. The total number of months in a year is $n = 12$:

$$\sum_{k=0}^{11} 1000(1.1)^k = \frac{1000((1.1)^{12} - 1)}{1.1 - 1} \approx 21384$$

PARTICIPATION ACTIVITY

7.3.6: Computing sums of geometric and arithmetic sequences.



Give numerical values for the following summations.

1) $\sum_{k=0}^{199} k$



Check

[Show answer](#)

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

2) $\sum_{k=0}^9 2^k$



Check

[Show answer](#)

**CHALLENGE
ACTIVITY**

7.3.2: Summations.



Note: You may want to use paper and pencil, and/or a calculator.

422102.2723990.qx3zqy7

Start

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Enter the result of the summation.

$$\sum_{j=1}^3 (j - 2) =$$

Ex: 2

1	2	3	4	5
---	---	---	---	---

Check

Next

Additional exercises**EXERCISE**

7.3.1: Evaluating summations.



Evaluate the following summations.

(a) $\sum_{k=-1}^4 k^2$.

(b) $\sum_{k=0}^4 2^k$.

(c) $\sum_{k=-3}^2 k^3$.

(d) $\sum_{k=0}^3 3^k$.

(e) $\sum_{k=0}^{200} (2 + 3k)$.

(f) $\sum_{k=0}^{200} 2 \cdot (1.01)^k$.

(g) $\sum_{k=0}^{100} (3 + 5k)$.

(h) $\sum_{k=0}^{100} 3 \cdot (1.1)^k$.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022



EXERCISE

7.3.2: Expressing sums using summation notation.



Express the following sums using summation notation.

(a) $(-2)^5 + (-1)^5 + \dots + 7^5$

(b) $(-2) + (-1) + 0 + 1 + 2 + 3 + 4 + 5$

(c) $2^2 + 2^3 + 2^4 + 2^5 + 2^6 + 2^7 + 2^8$

(d) $0^3 + 1^3 + 2^3 + 3^3 + 4^3 + 5^3 + \dots + 17^3$

(e) The sum of the cubes of the first 15 positive integers.

(f) The sum of the squares of the odd integers between 0 and 100.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022



EXERCISE

7.3.3: Pulling out a final term from a summation.



For each of the following expressions, write down an equivalent expression where the last term in the sum is outside the summation.

(a) $\sum_{j=-2}^{18} 2^j$

(b) $\sum_{j=-2}^{100} (j^2 + 2^j + \frac{j}{2})$

(c) $\sum_{k=0}^n (k^2 - 4k + 1)$

(d) $\sum_{k=0}^{m+2} (k^2 - 4k + 1)$

(e) $\sum_{k=0}^{q-1} (k^2 + 4k + 3)$

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

**EXERCISE**

7.3.4: Variable substitution for indices of summations.



- (a) Substitute variable i for j , where $i = j + 2$, in the summation below:

$$\sum_{j=0}^n (j + 2)$$

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

- (b) Substitute variable j for k , where $j = k - 1$, in the summation below:

$$\sum_{k=0}^{n-1} 2^{k-2}$$

- (c) Substitute variable i for j , where $i = j - 1$, in the summation below:

$$\sum_{j=0}^{n+1} (j^2 - 2j + 1)$$

- (d) Substitute variable k for j , where $k = j - 4$, in the summation below:

$$\sum_{j=4}^{17} (2j + 4)$$

- (e) Substitute variable j for k , where $j = k + 5$, in the summation below:

$$\sum_{k=10}^{20} (6k - 4)$$

**EXERCISE**

7.3.5: Are the two summations equal?

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022



- (a) Indicate whether the following equality is true and justify your answer:

$$\sum_{j=0}^{100} j^3 = \sum_{j=1}^{100} j^3 .$$



EXERCISE

7.3.6: Building a car collection.



A Silicon Valley billionaire purchases 3 new cars for his collection at the end of every month. Let a_n denote the number of cars he has after n months. Let $a_0 = 23$.

- What is a_8 ?
- If he pays \$50 each month to have each car maintained, what is the total amount that he has paid for maintenance after 2 years? No need to calculate the actual number. Instead give a closed form (without the summation) mathematical expression for the number. Note that he purchases the new cars at the end of each month, so during the first month, he is only maintaining 23 cars.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022



EXERCISE

7.3.7: Raising rabbits.



A population of rabbits on a farm grows by 12% each year. Define a sequence $\{r_n\}$ describing the rabbit population at the end of each year. Suppose that the sequence starts with $r_0 = 30$.

- Give a mathematical expression for r_{12} . (You don't have to actually compute the number.)
- If each rabbit consumes 10 pounds of rabbit food each year, then how much rabbit food is consumed in 10 years? For simplicity, you can omit the food consumed by the baby rabbits born in a given year. For example, suppose the farm starts tabulating rabbit food on January 1, 2012 at which time the rabbit population is 30. You will count the food consumed by those 30 rabbits during 2012. You won't count the food consumed by the rabbits born in 2012 until after January 1, 2013. Again, you don't have to compute the number, but you do have to give a closed form (without the summation) mathematical expression for the number.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

7.4 Mathematical induction

Suppose that one day a genie grants you three wishes. You make two wishes and then for your third wish, you wish for three more wishes the next day. Given the fact that you can always use your third wish to renew your wishes for the next day, it is possible to prove that from that first day onward, you can have three wishes every day for the rest of your life. This is an example of the principle of mathematical induction (or just *induction* for short).

Induction is a proof technique that is especially useful for proving statements about elements in a

sequence. An inductive proof establishes that some statement parameterized by n is true, for any positive integer n . In the example with the three wishes, the sequence is the sequence of days after that first day of wishes. The fact that is being proven is that you are able to have three wishes on the n^{th} day for $n = 1, 2, \dots$

Figure 7.4.1: The two components of an inductive proof.

- The **base case** establishes that the theorem is true for the first value in the sequence.
 - The genie grants you three wishes on day 1
- The **inductive step** establishes that if the theorem is true for k , then the theorem also holds for $k + 1$.
 - If you have three wishes on day k , then you can get three wishes for day $k+1$ (by using the third wish to ask for three more wishes the next day).

The **principle of mathematical induction** states that if the base case (for $n = 1$) is true and inductive step is true, then the theorem holds for all positive integers.

Figure 7.4.2: Principle of mathematical induction.

Let $S(n)$ be a statement parameterized by a positive integer n . Then $S(n)$ is true for all positive integers n , if:

1. $S(1)$ is true (the base case).
2. For all $k \in \mathbf{Z}^+$, $S(k)$ implies $S(k+1)$ (the inductive step).

The inductive step is a compact way to state that an infinite sequence of implications are true:

For all $k \in \mathbf{Z}^+$, $S(k)$ implies $S(k+1)$ \leftrightarrow [$S(1)$ implies $S(2)$] and [$S(2)$ implies $S(3)$] and [$S(3)$ implies $S(4)$] and

In the statement " $S(k)$ implies $S(k+1)$ " of the inductive step, the supposition that $S(k)$ is true is called the **inductive hypothesis**. The animation below illustrates why the principle of induction works by showing how the two components of an inductive proof imply that $S(n)$ is true for any positive integer n :

PARTICIPATION
ACTIVITY

7.4.1: Why induction works.

**Animation content:**

undefined

Animation captions:

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

1. $P(1)$ is true, and for all $k \geq 1$, $P(k)$ implies $P(k + 1)$. Setting $k = 1$ means that $P(1)$ implies $P(2)$.
2. Therefore, $P(2)$ is also true.
3. For $k = 2$, $P(2)$ implies $P(3)$. Therefore, $P(3)$ is also true. For $k = 3$, $P(3)$ implies $P(4)$. Therefore, $P(4)$ is also true.
4. Since $P(k)$ implies $P(k + 1)$ for all $k \geq 1$, the process can be continued up to any $n \geq 1$.
5. Therefore, for all $n \geq 1$, $P(n)$ is true.

PARTICIPATION
ACTIVITY

7.4.2: The components of an inductive proof.



Theorem: For all $n \in \mathbf{Z}^+$, $Q(n)$ is true,

where $Q(n)$ is a statement that is parameterized by a positive integer n .

- 1) In an inductive proof of the theorem, what must be proven in the base case?
 - ☐ $Q(0)$ is true
 - ☐ $Q(1)$ is true
 - ☐ $Q(k)$ is true
 - ☐ $Q(n)$ is true
- 2) In an inductive proof of the theorem, what must be proven in the inductive step?

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

- ☐ For all positive integers k , $Q(k - 1)$ implies $Q(k)$

3) If the inductive step says that for all positive integers $Q(k)$ implies $Q(k+1)$, then what is the inductive hypothesis?

- ☐ For all positive integers k , $Q(k)$ implies $Q(k+1)$
- ☒ $Q(n)$
- ☐ For all positive integers k , $Q(k)$
- ☐ $Q(k)$

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

The next example will be a proof that for any positive integer n , the following equality is true:

$$P(n) : \sum_{j=1}^n j = \frac{n(n+1)}{2}$$

If n is assigned a particular value, then the values of both sides of the equation are determined. For example, if $n = 5$, the equality $P(5)$ is obtained by replacing n with 5 in $P(n)$:

$$P(5) : \sum_{j=1}^5 j = \frac{5(5+1)}{2}.$$

The left side of the equation is the sum $1 + 2 + 3 + 4 + 5$. The expression on the right side of the equation is $5(5+1)/2$. Now it is possible to check that both sides evaluate to the same number (in this case 15). The assertion that the equality holds for a particular n can be denoted by the predicate $P(n)$. The statement $P(n)$ may be true or false, depending on the value of n .

PARTICIPATION ACTIVITY

7.4.3: The components of an inductive proof.

Define the predicate $P(n)$ to be the statement:

$$\sum_{j=1}^n j = \frac{n(n+1)}{2}$$

Theorem: For every positive integer n , $P(n)$ is true.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

1) $P(n) : \sum_{j=1}^n j = \frac{n(n+1)}{2}$

Express the statement $P(3)$.

☐ $1 + 2 + \dots + n = \frac{3(3+1)}{2}$

☐ $1 + 2 + 3 + \dots + j = \frac{3(3+1)}{2}$

2) The theorem states that for every positive integer n , $P(n)$ is true.

In an inductive proof of the theorem, what would be proven for the base case?

☐ $\sum_{j=1}^n j = \frac{n(n+1)}{2}$

☐ $\sum_{j=1}^0 j = \frac{0(0+1)}{2}$

☐ $\sum_{j=1}^1 j = \frac{1(1+1)}{2}$

3) $P(n) : \sum_{j=1}^n j = \frac{n(n+1)}{2}$

What is $P(k+1)$?

☐ $\sum_{j=1}^{k+1} j = \frac{(k+1)(k+2)}{2}$

☐ $\sum_{j=1}^{k+1} j = \frac{k(k+1)}{2}$

☐ $\sum_{j=1}^{k+1} j = \frac{n(n+1)}{2}$

4) $P(n) : \sum_{j=1}^n j = \frac{n(n+1)}{2}$

In an inductive proof of the theorem, what would be proven in the inductive step?

☐ For any integer $k \geq 1$,
 $\sum_{j=1}^{k+1} j = \frac{(k+1)(k+2)}{2}$

☐ For any integer $k \geq 1$,
 if $\sum_{j=1}^k j = \frac{k(k+1)}{2}$ then
 $\sum_{j=1}^n j = \frac{n(n+1)}{2}$

☐ For any integer $k \geq 1$, if
 $\sum_{j=1}^k j = \frac{k(k+1)}{2}$ then
 $\sum_{j=1}^{k+1} j = \frac{(k+1)(k+2)}{2}$

©zyBooks 12/15/22 00:21 1361995
 John Farrell
 COLOSTATECS220SeaboltFall2022

©zyBooks 12/15/22 00:21 1361995
 John Farrell
 COLOSTATECS220SeaboltFall2022

The complete proof by induction is given below for the theorem that $P(n)$ is true for all positive

integers n . The proof begins with the statement "By induction on n ", which lets the reader know that the proof will be an inductive proof and the variable n is the parameter to which the principle of induction is applied. In the beginning of the inductive step, the proof states explicitly: "Suppose that for positive integer k , $P(k)$ is true, then we will show that $P(k + 1)$ is also true." The statement gives the reader an overview of the coming argument. It is also important to point out at which step the inductive hypothesis is being applied.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Proof 7.4.1: Proving an identity by induction.

Theorem: For every positive integer n ,

$$\sum_{j=1}^n j = \frac{n(n+1)}{2}$$

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Proof.

By induction on n .

Base case: $n = 1$.

When $n = 1$, the left side of the equation is $\sum_{j=1}^1 j = 1$.

When $n = 1$, the right side of the equation is $1(1+1)/2 = 1$.

Therefore, $\sum_{j=1}^1 j = \frac{1(1+1)}{2}$.

Inductive step: Suppose that for positive integer k , $\sum_{j=1}^k j = \frac{k(k+1)}{2}$, then we will show that

$$\sum_{j=1}^{k+1} j = \frac{(k+1)(k+2)}{2}$$

Starting with the left side of the equation to be proven:

$$\sum_{j=1}^{k+1} j = \sum_{j=1}^k j + (k+1) \quad \text{by separating out the last term}$$

$$= \frac{k(k+1)}{2} + (k+1) \quad \text{by the inductive hypothesis}$$

$$= \frac{k(k+1) + 2(k+1)}{2}$$

$$= \frac{(k+2)(k+1)}{2}$$

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022
by algebra

Therefore, $\sum_{j=1}^{k+1} j = \frac{(k+1)(k+2)}{2}$. ■

Video explaining the inductive step (2:14).

Sometimes the smallest n for which $S(n)$ holds is some constant c other than 1. Then the base case must show that $S(c)$ is true. The inductive step will establish that $S(k) \rightarrow S(k+1)$ for any $k \geq c$. An inductive proof assumes that $S(k)$ and $k \geq c$ are true and uses those two facts to establish that $S(k+1)$ is true.

For example, let $S(n)$ be the inequality $2^n \geq 3n$. $S(3)$ is not true because $2^3 = 8$ which is not greater than or equal to $3 \cdot 3 = 9$. The next inductive proof will show that for $n \geq 4$, $S(n)$ is true.

©zyBooks 12/15/22 00:21 1361995

John Farrell

COLOSTATECS220SeaboltFall2022

Proof 7.4.2: Proving an inequality by induction.

Theorem: For $n \geq 4$, $2^n \geq 3n$.

Proof.

By induction on n .

Base case: $n = 4$.

$$2^4 = 16 \geq 12 = 3 \cdot 4$$

Therefore, for $n = 4$, $2^n \geq 3n$.

Inductive step: We will show that for any integer $k \geq 4$, if $2^k \geq 3k$, then

$$2^{k+1} \geq 3(k+1).$$

Starting with the left side of the inequality to be proven:

$$2^{k+1} = 2 \cdot 2^k \quad \text{by algebra}$$

$$\geq 2 \cdot 3k \quad \text{by the inductive hypothesis}$$

$$= 3k + 3k$$

$$\geq 3k + 3 \quad \text{because } k \geq 4 \geq 1$$

$$= 3(k+1) \quad \text{by algebra}$$

©zyBooks 12/15/22 00:21 1361995

John Farrell

COLOSTATECS220SeaboltFall2022

Therefore, $2^{k+1} \geq 3(k+1)$ ■.

Video explaining the inductive step (3:42).

**PARTICIPATION
ACTIVITY**

7.4.4: Proving an inequality by induction.



Consider a proof of the following fact:

For all $n \geq 4$, $2^n \geq n^2$

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

1) What should be proven in the base case?

- ☐ For $n = 1$, $2^n \geq n^2$
- ☐ For $n = 4$, $2^n \geq n^2$
- ☐ $2^k \geq k^2$
- ☐ For every $k \geq 4$,
if $2^k \geq k^2$,
then $2^{k+1} \geq (k+1)^2$

2) What should be proven in the inductive step?

- ☐ For $n = 4$, $2^n \geq n^2$
- ☐ $2^k \geq k^2$
- ☐ For every $k \geq 4$,
if $2^k \geq k^2$,
then $2^{k+1} \geq (k+1)^2$

3) Below is an argument for the inductive step. In which line is the inductive hypothesis used?

$$2^{k+1} = 2 \cdot 2^k \quad (1)$$

$$\geq 2 \cdot k^2 \quad (2)$$

$$= k^2 + k^2 \quad (3)$$

$$\geq k^2 + 4k \quad \text{because } k \geq 4$$

$$= k^2 + 2k + 2k \quad (4)$$

$$\geq k^2 + 2k + 1 = (k+1)^2$$

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

- ☐ Line 1
- ☐ Line 2
- ☐ Line 3
- ☐ Line 4

Additional exercises

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022



EXERCISE

7.4.1: Components of an inductive proof.



Define $P(n)$ to be the assertion that:

$$\sum_{j=1}^n j^2 = \frac{n(n+1)(2n+1)}{6}$$

- (a) Verify that $P(3)$ is true.
- (b) Express $P(k)$.
- (c) Express $P(k+1)$.
- (d) In an inductive proof that for every positive integer n ,

$$\sum_{j=1}^n j^2 = \frac{n(n+1)(2n+1)}{6}$$

what must be proven in the base case?

- (e) In an inductive proof that for every positive integer n ,

$$\sum_{j=1}^n j^2 = \frac{n(n+1)(2n+1)}{6}$$

what must be proven in the inductive step?

- (f) What would be the inductive hypothesis in the inductive step from your previous answer?

- (g) Prove by induction that for any positive integer n ,

$$\sum_{j=1}^n j^2 = \frac{n(n+1)(2n+1)}{6}$$

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022



EXERCISE

7.4.2: Proving identities by induction.



Prove each of the following statements using mathematical induction.

(a)

Prove that for any positive integer n , $\sum_{j=1}^n j^3 = \left(\frac{n(n+1)}{2}\right)^2$

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

(b)

Prove that for any positive integer n , $\sum_{j=1}^n j \cdot 2^j = (n-1)2^{n+1} + 2$

(c)

Prove that for any positive integer n , $\sum_{j=1}^n j(j-1) = \frac{n(n^2-1)}{3}$

(d)

Prove that for any positive integer n , $\sum_{j=1}^n \frac{1}{j(j+1)} = 1 - \frac{1}{n+1}$

(e) Prove that for any non-negative integer n ,

$$\sum_{j=0}^n j \cdot 3^j = \frac{3}{4}[n \cdot 3^{n+1} - (n+1)3^n + 1]$$

(f) Prove that for any positive integer $n \geq 2$,

$$\left(1 - \frac{1}{2^2}\right) \left(1 - \frac{1}{3^2}\right) \dots \left(1 - \frac{1}{n^2}\right) = \frac{n+1}{2n}$$

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

**EXERCISE**

7.4.3: Proving inequalities by induction.



Prove each of the following statements using mathematical induction.

(a) Prove that for $n \geq 2$, $3^n > 2^n + n^2$.

(b) For any $n \geq 1$, the factorial function, denoted by $n!$, is the product of all the positive integers through n :

$$n! = 1 \cdot 2 \cdot 3 \cdots (n-1) \cdot n$$

Prove that for $n \geq 4$, $n! \geq 2^n$.

(c) Prove that for $n \geq 1$, $\sum_{j=1}^n \frac{1}{j^2} \leq 2 - \frac{1}{n}$

(d) Prove that for any non-negative integer $n \geq 3$, $3^n \geq n^3$.

(e) Prove that for any non-negative integer $n \geq 96$, $(1.1)^n \geq n^2$.
Hint: $(1.1)^{96} \approx 9412$, $96^2 = 9216$

(f) Prove that for any non-negative integer $n \geq 4$, $3^n \leq (n+1)!$.

(g) Prove that for any non-negative integer $n \geq 0$ and $a \geq 0$, $(1+na) \leq (1+a)^n$.

7.5 More inductive proofs

Divisibility proof by induction

The next example of a proof by induction proves that 3 evenly divides the value of a mathematical expression parameterized by an integer n . A quantity is evenly divisible by 3 if it can be expressed as $3 \cdot m$ for some integer m . Define the predicate $Q(n)$ to be the statement:

$$Q(n): 3 \text{ evenly divides } 2^{2^n} - 1.$$

The statement $Q(3)$ says that 3 evenly divides $2^{2^3} - 1$. $Q(3)$ is true because $2^{2^3} - 1 = 2^8 - 1 = 255$ and 3 does in fact evenly divide 255 (because $255 = 3 \cdot 85$).

The following example shows an inductive proof of the theorem that $Q(n)$ is true for all positive integers n .

Proof 7.5.1: Proving a divisibility proof by induction.

Theorem: For every positive integer n , 3 evenly divides $2^{2n} - 1$.

Proof.

By induction on n .

Base case: $n = 1$.

$2^{2 \cdot 1} - 1 = 2^2 - 1 = 3$. Since 3 evenly divides 3, the theorem holds for the case $n = 1$.

Inductive step: Suppose that for positive integer k , 3 evenly divides $2^{2k} - 1$. Then we will show that 3 evenly divides $2^{2(k+1)} - 1$.

By the inductive hypothesis, 3 evenly divides $2^{2k} - 1$, which means that $2^{2k} - 1 = 3m$ for some integer m . By adding 1 to both sides of the equation $3m = 2^{2k} - 1$, we get $2^{2k} = 3m + 1$ which is an equivalent statement of the inductive hypothesis.

We must show that $2^{2(k+1)} - 1$ can be expressed as 3 times an integer.

$$\begin{aligned}
 2^{2(k+1)} - 1 &= 2^{2k+2} - 1 \\
 &= 4 \cdot 2^{2k} - 1 && \text{by algebra} \\
 &= 4(3m + 1) - 1 && \text{by the inductive hypothesis} \\
 &= 3 \cdot 4m + 4 - 1 \\
 &= 3(4m + 1) && \text{by algebra}
 \end{aligned}$$

Since m is an integer, $(4m + 1)$ is also an integer. Therefore $2^{2(k+1)} - 1$ is equal to 3 times an integer which means that $2^{2(k+1)} - 1$ is divisible by 3. ■

**PARTICIPATION
ACTIVITY**

7.5.1: Divisibility proof by induction.

Theorem: For every positive integer n , 4 evenly divides $3^{2n} - 1$.

- 1) Suppose that an inductive proof of the theorem uses the following inductive hypothesis: For positive integer k , 4 evenly divides $3^{2k} - 1$. Which statement is equivalent to the

inductive hypothesis?

- ☐ For some integer m , $4m = 3^{2k}$.
- ☐ For some integer m , $3m = 3^{2k} - 1$.
- ☐ For some integer m , $4m + 1 = 3^{2k}$.

2) In the inductive step, the proof assumes that the inductive hypothesis is true and must prove that 4 evenly divides $3^{2(k+1)} - 1$. Which expression implies that 4 evenly divides $3^{2(k+1)} - 1$?

- ☐ $3^{2(k+1)} - 1 = 4(9m + 2)$, for some integer m .
- ☐ $3^{2(k+1)} = 4(9m + 2)$, for some integer m .
- ☐ $3^{2(k+1)} - 1 = 4(9m + 1/2)$, for some integer m .

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022



Using induction to prove an assertion about a recurrence relation

Induction is particularly well suited to proving facts about sequences defined by recurrence relations. The theorem below starts by defining a sequence with a recurrence relation and initial value. The theorem asserts that a particular explicit formula (given in the last line of the theorem statement) corresponds to the same sequence defined by the recurrence relation.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Theorem 7.5.1: Explicit formula for a sequence defined by a recurrence relation.

Define the sequence $\{g_n\}$ as:

- $g_0 = 1$.
- $g_n = 3 \cdot g_{n-1} + 2n$, for any $n \geq 1$.

Then for any $n \geq 0$,

$$g_n = \frac{5}{2} \cdot 3^n - n - \frac{3}{2}$$

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

The recurrence relation ($g_n = 3 \cdot g_{n-1} + 2n$) and initial value are given as part of the assumptions of the theorem and can be used at any point in the proof. In the inductive step of the proof, we assume the explicit formula is correct for term g_k and then prove that it is also correct for term g_{k+1} . The inductive step will therefore need to express g_{k+1} in terms of g_k . However, the recurrence relation defines g_n in terms of g_{n-1} . The proof, therefore, uses a slightly different form of the recurrence relation which is obtained by replacing the index n with $k + 1$. Note that with the change of variable, the first index for the recurrence is $k = 0$ instead of $n = 1$. The two statements of the recurrence relation are equivalent:

$$g_n = 3g_{n-1} + 2n \quad \text{for } n \geq 1$$

$$g_{k+1} = 3g_k + 2(k+1) \quad \text{for } k \geq 0$$

Both statements are equivalent to the infinite sequence of equations:

- $g_1 = 3 \cdot g_0 + 2 \cdot 1$
- $g_2 = 3 \cdot g_1 + 2 \cdot 2$
- $g_3 = 3 \cdot g_2 + 2 \cdot 3$
- etc.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Proof 7.5.2: Proving the explicit formula for a recurrence relation by induction.

Proof.

By induction on n .

Base case: $n = 0$.

We must show that $g_0 = \frac{5}{2} \cdot 3^0 - 0 - \frac{3}{2}$. Since $\frac{5}{2} \cdot 3^0 - 0 - \frac{3}{2} = 1$ and the initial condition states that $g_0 = 1$, the theorem holds for $n = 0$.

Inductive step: For any integer $k \geq 0$, assume that $g_k = \frac{5}{2} \cdot 3^k - k - \frac{3}{2}$ is true. Then we will show that

$$g_{k+1} = \frac{5}{2} \cdot 3^{k+1} - (k+1) - \frac{3}{2}$$

is true.

For any integer $k \geq 0$:

$$g_{k+1} = 3g_k + 2(k+1) \quad \text{by definition}$$

$$= 3 \left(\frac{5}{2} 3^k - k - \frac{3}{2} \right) + 2(k+1) \quad \text{by the inductive hypothesis}$$

$$= \frac{5}{2} \cdot 3 \cdot 3^k - 3k - 3 \cdot \frac{3}{2} + 2k + 2$$

$$= \frac{5}{2} 3^{k+1} - k - \frac{5}{2}$$

$$= \frac{5}{2} 3^{k+1} - (k+1) - \frac{3}{2} \quad \text{by algebra}$$

Therefore, $g_{k+1} = \frac{5}{2} \cdot 3^{k+1} - (k+1) - \frac{3}{2}$. ■

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

The next set of questions present an inductive proof of the following theorem:

Theorem 7.5.2: Explicit formula for a sequence defined by a recurrence relation.

Define the sequence $\{h_n\}$ as:

- $h_0 = 7$
- $h_n = (h_{n-1})^3$, for any $n \geq 1$

Then for any $n \geq 0$,

$$h_n = 7^{(3^n)}$$

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

**PARTICIPATION
ACTIVITY**

7.5.2: Using induction to prove an explicit formula for a sequence defined by a recurrence relation.



Select the correct statements to complete the proof of the theorem stated above.

Proof.

By induction on n .

Base case: $n = 0$.

We must show that $h_0 = 7^{(3^0)} = 7$. Since $7^{(3^0)} = 7^1 = 7$ and the initial condition states that $h_0 = 7$, the theorem holds for $n = 0$.

Inductive step: We will show that __ (A) __

The recurrence relation is equivalent to the statement __ (B) __.

$$h_{k+1} = (C)$$

by definition

$$= (7^{(3^k)})^3$$

by the inductive hypothesis

$$= 7^{3 \cdot (3^k)} = 7^{(3^{k+1})}$$

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Therefore, $h_{k+1} = 7^{(3^{k+1})}$. ■

1) Select the correct statement for A.



☐ For every $k \geq 0$, if $h_k = (h_{k-1})^3$ is true, then $h_k = 7^{(3^k)}$

☐ For every $k \geq 0$, if $h_k = 7^{(3^k)}$ is true, then $h_{k+1} = (h_k)^3$.

☐ For every $k \geq 0$, if $h_k = 7^{(3^k)}$

2) Select the correct statement for B.

The selection should be equivalent to the recurrence relation:

$$h_n = (h_{n-1})^3 \quad \text{for } n \geq 1$$

☐ $h_k = (h_{k-1})^3 \quad \text{for } k \geq 0$

☐ $h_{k+1} = (h_k)^3 \quad \text{for } k \geq 0$

☐ $h_{k+1} = (h_{k-1})^3 \quad \text{for } k \geq 0$

3) Select the correct term for C.

☐ $(h_k)^3$

☐ $7^{(3^{k+1})}$

☐ $7^{(3 \cdot 3^k)}$

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Inductive proofs showing the closed forms for sums of arithmetic and geometric sequences.

The theorem below gives a closed form for the sum of the first n terms in an arithmetic sequence. Here we present an inductive proof of the theorem.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Theorem 7.5.3: Closed form for the sum of terms in an arithmetic sequence.

For any integer $n \geq 1$:

$$\sum_{j=0}^{n-1} (a + jd) = an + \frac{d(n-1)n}{2}$$

Proof 7.5.3: Closed form for the sum of terms in an arithmetic sequence.

Proof: By induction on n .

The base case is proven for $n = 1$. In this case, the summation goes from $j = 0$ to $n - 1 = 0$. Therefore, there is only one term in the summation in which $j = 0$. The value of the summation for $n = 1$ is $a + 0 \cdot d = a$. Plugging in $n = 1$ into the expression on the right side of the equality also yields a value of a .

The inductive step assumes the theorem holds for positive integer k (the inductive hypothesis):

$$\sum_{j=0}^{k-1} (a + jd) = ak + \frac{d(k-1)k}{2}$$

and uses the inductive hypothesis to show that the theorem holds for $k+1$:

$$\sum_{j=0}^k (a + jd) = a(k+1) + \frac{dk(k+1)}{2}$$

The remainder of the proof is given in the following animation:

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

PARTICIPATION ACTIVITY

7.5.3: The inductive step in the proof of the closed form for the sum of an arithmetic sequence.



Animation captions:

1. Start with the left hand side of the fact to be proven and separate the last term from the summation: $\sum_{j=0}^k (a + jd) = (a + kd) + \sum_{j=0}^{k-1} (a + jd)$.
2. Apply the inductive hypothesis by replacing $\sum_{j=0}^k (a + jd)$ with $ka + \frac{dk(k-1)}{2}$ to get $(a + kd) + ka + \frac{dk(k-1)}{2}$.
3. Rearrange the expression using algebra to get $a + ka + \frac{2kd + dk(k-1)}{2}$.
4. Pull out common factors from terms to get $a(k+1) + \frac{d(2k + k^2 - k)}{2}$.
5. Simplifying further gives $a(k+1) + \frac{dk(k+1)}{2}$ which is equal to the initial expression $\sum_{j=0}^k (a + jd)$.

There is also a theorem giving an closed form expression for the sum of terms in a geometric sequence:

Theorem 7.5.4: Closed form for the sum of terms in a geometric sequence.

For any real number $r \neq 1$ and any integer $n \geq 1$:

$$\sum_{j=0}^{n-1} a \cdot r^j = \frac{a(r^n - 1)}{r - 1}$$

An inductive proof of the theorem is given in the animation below:

PARTICIPATION ACTIVITY

7.5.4: An inductive proof of the closed form for the sum of a geometric sequence.



Animation content:

undefined

Animation captions:

1. The base case proves the theorem for $n = 1$. $\sum_{j=0}^0 ar^j = ar^0 = a$. Also, $\frac{a(r^1 - 1)}{r - 1} = a$.

The two expressions are equal.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

2. The inductive step assumes that $\sum_{j=0}^{k-1} ar^j = \frac{a(r^k - 1)}{r - 1}$.

3. The inductive step proves that $\sum_{j=0}^k ar^j = \frac{a(r^{k+1} - 1)}{r - 1}$.

4. The inductive step starts with the left side of the equality to be proven and pulls out the last term from the summation: $\sum_{j=0}^k ar^j = ar^k + \sum_{j=0}^{k-1} ar^j$.

5. The inductive hypothesis is applied by replacing $\sum_{j=0}^{k-1} ar^j$ with $\frac{a(r^k - 1)}{r - 1}$ to get

$$ar^k + \frac{a(r^k - 1)}{r - 1}.$$

6. The fractions are combined and simplified to get $\frac{a[r^{k+1} - r^k + r^k - 1]}{r - 1}$.

7. The two middle terms in the numerator cancel, resulting in $\frac{a(r^{k+1} - 1)}{r - 1}$, which is equal to

$$\text{the initial expression } \sum_{j=0}^k ar^j.$$

PARTICIPATION ACTIVITY

7.5.5: Induction proofs for sums of arithmetic and geometric sequences.



1) In the proof of the closed form for the sum of an arithmetic sequence, the inductive hypothesis is:

$$\sum_{j=0}^{k-1} (a + jd) = ak + \frac{d(k-1)k}{2}.$$

Select the expression that is equivalent to the expression below by applying the inductive hypothesis:

$$(a + kd) + \sum_{j=0}^{k-1} (a + jd)$$



©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

☐ $(a + kd) + ak + \frac{d(k-1)k}{2}$

☐ $(a + kd) + ak + \frac{d(k+1)k}{2}$

2) In the inductive proof of the sum of a geometric sequence, the inductive hypothesis is that for positive integer

k , $\sum_{j=0}^{k-1} ar^j = \frac{a(r^k - 1)}{r - 1}$. What

must be proven assuming the inductive hypothesis is true?

☐ $\sum_{j=0}^{k-1} ar^j = \frac{a(r^k - 1)}{r - 1}$.

☐ $\sum_{j=0}^k ar^j = (ar^k) + \sum_{j=0}^{k-1} ar^j$

☐ $\sum_{j=0}^k ar^j = \frac{a(r^{k+1} - 1)}{r - 1}$.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

An inductive proof for set operations

Induction is a useful technique for proving theorems about other mathematical objects, beyond sequences and sums. The animation below uses induction to prove a generalized version of de Morgan's law for sets. The theorem assumes that de Morgan's law is true for two sets and shows that de Morgan's law can be extended to an arbitrary number of sets.

PARTICIPATION ACTIVITY

7.5.6: Inductive proof of the generalized de Morgan's law for sets.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Animation captions:

1. Inductive proof of a Generalized De Morgan's law for sets. The base case for $n = 2$ is just De Morgan's law: $\overline{A_1 \cap A_2} = \overline{A_1} \cup \overline{A_2}$.
2. The inductive step assumes the theorem for k :
 $\overline{A_1 \cap A_2 \cap \dots \cap A_k} = \overline{A_1} \cup \overline{A_2} \cup \dots \cup \overline{A_k}$ and proves the theorem for $k+1$.
3. B is defined to be $A_1 \cap A_2 \cap \dots \cap A_k$. Therefore,

$$\overline{A_1 \cap A_2 \cap \dots \cap A_k \cap A_{k+1}} = \overline{B \cap A_{k+1}}.$$

4. $\overline{B \cap A_{k+1}} = \overline{B} \cup \overline{A_{k+1}}$ by De Morgan's law.

5. $\overline{B} \cup \overline{A_{k+1}} = \overline{A_1 \cap A_2 \cap \dots \cap A_k} \cup \overline{A_{k+1}}$ by the definition of B.

6. The result of applying the inductive hypothesis to the last expression is

$$\overline{A_1} \cup \overline{A_2} \cup \dots \cup \overline{A_k}, \text{ which is equal to the first expression } \overline{A_1 \cap A_2 \cap \dots \cap A_k}.$$

©zyBooks 12/15/22 00:21 1361995

John Farrell

COLOSTATECS220SeaboltFall2022

PARTICIPATION ACTIVITY

7.5.7: Induction proof for generalized de Morgan's law for sets.



1) What is the inductive hypothesis in the inductive proof for the generalized de Morgan's law for sets?



☐ $\overline{A_1 \cap A_2 \cap \dots \cap A_{k-1}} = \overline{A_1} \cup \overline{A_2} \cup \dots \cup \overline{A_{k-1}}$

☐ $\overline{A_1 \cap A_2 \cap \dots \cap A_k} = \overline{A_1} \cup \overline{A_2} \cup \dots \cup \overline{A_k}$

☐ $\overline{A_1 \cap A_2 \cap \dots \cap A_{k+1}} = \overline{A_1} \cup \overline{A_2} \cup \dots \cup \overline{A_{k+1}}$

Additional exercises



EXERCISE

7.5.1: Proving divisibility results by induction.



Prove each of the following statements using mathematical induction.

(a) Prove that for any positive integer n , 4 evenly divides $3^{2n}-1$.

(b) Prove that for any positive integer n , 6 evenly divides $7^n - 1$.

(c) Prove that for any positive integer n , 4 evenly divides $11^n - 7^n$.

(d) Prove that for any positive integer n , 7 evenly divides $9^n - 2^n$.

(e) Prove that for any positive integer n , 2 evenly divides $n^2 - 5n + 2$.

(f) Prove that for any positive integer n , 3 evenly divides $n^3 - 4n + 6$.

©zyBooks 12/15/22 00:21 1361995

John Farrell

COLOSTATECS220SeaboltFall2022

**EXERCISE**

7.5.2: Proving generalized laws by induction for logical expressions.



Prove each of the following statements using mathematical induction.

- (a) Prove the following generalized version of DeMorgan's law for logical expressions:

For any integer $n \geq 2$, $\neg(x_1 \wedge x_2 \wedge \dots \wedge x_n) = \neg x_1 \vee \neg x_2 \vee \dots \vee \neg x_n$

You can use DeMorgan's law for two variables in your proof:

$$\neg(x_1 \wedge x_2) = \neg x_1 \vee \neg x_2$$

- (b) Prove the following generalization of the Distributive law for logical expressions.

For any integer $n \geq 2$,

$$y \vee (x_1 \wedge x_2 \wedge \dots \wedge x_n) = (y \vee x_1) \wedge (y \vee x_2) \wedge \dots \wedge (y \vee x_n)$$

You can use the Distributive law for two variables in your proof:

$$y \vee (x_1 \wedge x_2) = (y \vee x_1) \wedge (y \vee x_2)$$



EXERCISE

7.5.3: Proving explicit formulas for recurrence relations by induction.



Prove each of the following statements using mathematical induction.

(a) Define the sequence $\{c_n\}$ as follows:

- $c_0 = 5$
- $c_n = (c_{n-1})^2$ for $n \geq 1$

Prove that for $n \geq 0$, $c_n = 5^{2^n}$.

Note that in the explicit formula for c_n , the exponent of 5 is 2^n .

(b) Define the sequence $\{b_n\}$ as follows:

- $b_0 = 1$
- $b_n = 2b_{n-1} + 1$ for $n \geq 1$

Prove that for $n \geq 0$, $b_n = 2^{n+1} - 1$.

(c) Define the sequence $\{a_n\}$ as follows:

- $a_1 = 6$
- $a_n = 2 \cdot a_{n-1} + 2n$ for $n \geq 2$

Prove that for any positive integer $n \geq 1$, $a_n = -2n - 4 + 6 \cdot 2^n$.

(d) Define the sequence $\{g_n\}$ as follows:

- $g_0 = 0$
- $g_n = g_{n-1} + n + 1$ for $n \geq 1$

Prove that for any integer $n \geq 0$, $g_n = \frac{n(n+3)}{2}$.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

7.6 Strong induction and well-ordering

The Fibonacci sequence is defined by initial values $f_0 = 0$, $f_1 = 1$ and the recurrence relation for $n \geq 2$:

$$f_n = f_{n-1} + f_{n-2}$$

Suppose we wanted to prove an upper bound on the Fibonacci numbers, such as $f_n \leq 2^n$, for all $n \geq 0$. In the standard form of induction, the inductive step would use the fact that $f_k \leq 2^k$ to argue that $f_{k+1} \leq 2^{k+1}$. However, since f_{k+1} is defined in terms of f_k and f_{k-1} , the proof requires a bound on both f_k and f_{k-1} in order to upper bound f_{k+1} . The inductive hypothesis for standard induction only provides the bound on f_k .

For the upper bound on terms of the Fibonacci sequence, a stronger version of induction is required. The **principle of strong induction** assumes that the fact to be proven holds for all values

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

less than or equal to k and proves that the fact holds for $k+1$. By contrast the standard form of induction only assumes that the fact holds for k in proving that it holds for $k+1$. Define $S(n)$ to be the assertion that $f_n \leq 2^n$, where f_n is defined by Fibonacci's initial values and recurrence relation. The proof that $S(n)$ is true for all non-negative integers n has two components:

- **Base case:** $S(0)$ and $S(1)$ are true.
- **Inductive step:** For every $k \geq 1$, $(S(0) \wedge S(1) \wedge \dots \wedge S(k))$ implies $S(k+1)$.

In strong induction the inductive hypothesis is $S(0) \wedge S(1) \wedge \dots \wedge S(k)$. In regular (or weak) induction, the inductive hypothesis is just $S(k)$. Sometimes the inductive hypothesis is expressed using a universal quantifier. The following two ways to express the inductive hypothesis are equivalent:

$S(0) \wedge S(1) \wedge \dots \wedge S(k)$ is true. \leftrightarrow For every j in the range 0 through k , $S(j)$ is true.

Figure 7.6.1: Expanded expression of the inductive step for weak and strong induction.

Inductive step for weak induction

For all $k \geq 1$, $S(k) \rightarrow S(k+1)$

$k = 1$: $S(1) \rightarrow S(2)$

$k = 2$: $S(2) \rightarrow S(3)$

$k = 3$: $S(3) \rightarrow S(4)$

\vdots

\ddots

Inductive step for strong induction

For all $k \geq 1$, $(S(0) \wedge S(1) \dots \wedge S(k)) \rightarrow S(k+1)$

$k = 1$: $(S(0) \wedge S(1)) \rightarrow S(2)$

$k = 2$: $(S(0) \wedge S(1) \wedge S(2)) \rightarrow S(3)$

$k = 3$: $(S(0) \wedge S(1) \wedge S(2) \wedge S(3)) \rightarrow S(4)$

\vdots

\ddots

PARTICIPATION ACTIVITY

7.6.1: Using strong induction.



Suppose that the inductive step of a strong induction proof shows that for $k \geq 1$, if $(S(0) \wedge S(1) \wedge \dots \wedge S(k))$ is true, then $S(k+1)$ is true.

- 1) What statement is equivalent to the inductive hypothesis?

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022



- ☐ For $k \geq 1$ and any j in the range from 0 through k , $S(j)$ is true.
- 2) If you assume that the inductive hypothesis is true, can you conclude that $S(k-2)$ is true?
- ☐ For $k \geq 1$ and any j in the range from 0 through $k+1$, $S(j)$ is true.

- ☐ Yes
☐ $S(j)$ is true.
☐ No

- 3) If you assume that the inductive hypothesis is true, can you conclude that $S(k-1)$ is true?

- ☐ Yes
☐ No

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

There is no simple rule to determine when strong induction is required instead of the standard form of induction. The best method is to try proving the inductive step using only the preceding value (i.e., try and prove the statement for $k + 1$ using only the statement for k). If that fails, then try a stronger inductive hypothesis. The inductive proof showing a bound on the Fibonacci numbers given below would fail with standard induction because bounds on f_k and f_{k-1} are required to prove the bound for f_{k+1} .

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Proof 7.6.1: Upper bound for Fibonacci numbers proven using strong induction.

Theorem: For $n \geq 0$, $f_n \leq 2^n$.

Proof.

By strong induction on n .

Base case:

- $n = 0$: $f_0 = 0 \leq 2^0$
- $n = 1$: $f_1 = 1 \leq 2^1$

Inductive step:

For $k \geq 1$, suppose that for any j in the range from 0 through k , $f_j \leq 2^j$. We will prove that $f_{k+1} \leq 2^{k+1}$.

Since $k \geq 1$, then $k-1 \geq 0$. Therefore both k and $k-1$ fall in the range from 0 through k , and by the inductive hypothesis, $f_{k-1} \leq 2^{k-1}$ and $f_k \leq 2^k$.

$$f_{k+1} = f_k + f_{k-1} \quad \text{by definition}$$

$$\leq 2^k + 2^{k-1} \quad \text{by the inductive hypothesis}$$

$$\leq 2^k + 2^{k-1} + 2^{k-1} \quad \text{since } 2^{k-1} \geq 0$$

$$= 2^k + 2 \cdot 2^{k-1}$$

$$= 2^k + 2^k = 2 \cdot 2^k = 2^{k+1} \quad \text{by algebra}$$

Therefore, $f_{k+1} \leq 2^{k+1}$. ■

Generalized strong induction: multiple base cases

The predicate $S(n)$ may be true only for n greater than or equal to a particular value denoted below by the constant a . Also, the base case may require proving the assertion directly for more than one value. The **base case** for a proof by strong induction establishes that $S(n)$ holds for $n = a$ through b , where a and b are constants. The **inductive step** in a proof by strong induction assumes that $S(j)$ is true for all values of j in the range from a through some integer $k \geq b$ and then proves that theorem holds for $k+1$.

Figure 7.6.2: Principle of strong mathematical induction.

Let $S(n)$ be a statement parameterized by n , a positive integer. Let $a \leq b$ be integers. Then $S(n)$ is true for all $n \geq a$, if the following two conditions hold:

1. $S(a), S(a+1), \dots, S(b)$ are true (the base case).
2. For all $k \geq b$, if $(S(a) \wedge S(a+1) \wedge \dots \wedge S(k))$ is true, then $S(k+1)$ is also true (the inductive step).

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Figure 7.6.3: Expanded expression of the inductive step with an arbitrary range for the base case.

Here is the inductive step in expanded form. The base case must establish that the inductive hypothesis is true for the smallest value for k .

Base case: $(S(a) \wedge S(a+1) \wedge \dots \wedge S(b))$

Inductive step: For all $k \geq b$, $(S(a) \wedge S(a+1) \dots \wedge S(k)) \rightarrow S(k+1)$

$k = b$: $(S(a) \wedge S(a+1) \wedge \dots \wedge S(b)) \rightarrow S(b+1)$

$k = b+1$: $(S(a) \wedge S(a+1) \wedge \dots \wedge S(b) \wedge S(b+1)) \rightarrow S(b+2)$

$k = b+2$: $(S(a) \wedge S(a+1) \wedge \dots \wedge S(b) \wedge S(b+1) \wedge S(b+2)) \rightarrow S(b+3)$

\vdots

\ddots

PARTICIPATION ACTIVITY

7.6.2: The principle of strong induction with general range for the base case.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022



- 1) Consider an inductive proof in which the inductive step is:

For all $k \geq 6$, if $P(3), P(4), \dots, P(k)$ are all true, then $P(k+1)$ is true.

What facts must be proven in the base case?

- ☐ $P(3)$ only.
- ☐ $P(3)$, $P(4)$, and $P(5)$.
- ☐ $P(3)$, $P(4)$, $P(5)$, and $P(6)$.

2) Select the choice that is equivalent to the following statement:

For $k \geq 7$, if $P(j)$ is true for all j in the range 2 through k , then $P(k+1)$ is true.

- ☐ For $k \geq 7$,
 $(P(2) \wedge P(3) \wedge \dots \wedge P(k))$
implies $P(k+1)$
- ☐ For $k \geq 7$,
 $(P(2) \wedge P(3) \wedge \dots \wedge P(k))$
implies $P(k+1)$
- ☐ For $k \geq 7$,
 $(P(7) \wedge P(8) \wedge \dots \wedge P(k))$
implies $P(k+1)$

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Strong induction example: buying cans of juice in 3-packs or 4-packs.

Suppose that cans of juice come in packs of 3 or 4. We would like to be able to buy n cans of juice by purchasing a combination of 3-packs or 4-packs. For which values of n is this possible? Let $P(n)$ be the assertion that it is possible to buy n cans of juice by purchasing only 3-packs or 4-packs. $P(7)$ is true because we can purchase one 3-pack and one 4-pack. $P(5)$ is not true because there is no way to combine 3-packs and 4-packs to get five cans of juice. We will use strong induction to prove that for any $n \geq 6$, $P(n)$ is true. Here is an outline of the proof:

- **Base case:** Prove $P(6)$, $P(7)$, and $P(8)$ directly.
- **Inductive step:** For $k \geq 8$, assume that that $P(j)$ is true for any j in the range 6 through k , and prove that $P(k+1)$ is true.
 - Argue that $P(k-2)$ is true by showing that $k-2$ falls in the range 6 through k , covered in the inductive hypothesis. The argument uses the fact that $k \geq 8$ and therefore $k-2 \geq 6$.
 - Since $P(k-2)$ is true, $k-2$ cans of juice can be bought by combining 3-packs and 4-packs. By adding one 3-pack to the $k-2$ cans of juice, there will be $k-2+3 = k+1$ cans of juice. Therefore $P(k+1)$ is true.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Animation captions:

1. The base case starts with $n = 6$. Purchasing two 3-packs of juice results in 6 cans. Therefore, $P(6)$ is true.
2. $n = 7$ is next. Purchasing one 3-pack and one 4-pack results in 7 cans. Therefore, $P(7)$ is true.
3. $n = 8$ is next. Purchasing two 4-packs results in 8 cans. Therefore, $P(8)$ is true.
4. The inductive hypothesis is that for $k \geq 8$, $P(j)$ is true for any j in the range 6 through k .
5. The inductive step proves $P(k + 1)$. Start with $k - 2$ cans of juice.
6. Since $k \geq 8$, then $k - 2 \geq 6$. Therefore, $k - 2$ is in the range 6 through k and by the inductive hypothesis, $k - 2$ cans of juice can be purchased.
7. When an additional 3-pack is purchased, the number of cans is $(k - 2) + 3 = k + 1$. Therefore, $P(k + 1)$ is true.

How many base cases?

How did we know to use three base cases in the example with the cans of 3-packs and 4-packs of juice? To some extent, determining how many base cases are required in a strong induction proof requires some trial and error, but there are some basic principles that can be applied. In proving that $P(k + 1)$ is true, we needed to assume that $P(k - 2)$ is true in order to add one 3-pack to get $k + 1$ cans of juice. If the base cases had only included $P(6)$ and $P(7)$, then we would have been unable to prove the inductive step for $k = 7$: $(P(6) \wedge P(7)) \rightarrow P(8)$. In proving $P(8)$, we needed to assume $P(8 - 3) = P(5)$, but $P(5)$ is not included in the base cases. Note that, the inductive step could have added a 4-pack at the end instead of a 3-pack but using a 4-pack would have required 4 consecutive base cases (6 through 9) instead of 3 consecutive base cases.

PARTICIPATION ACTIVITY

7.6.4: Strong induction.



Consider again the problem of purchasing cans of juice. Suppose now that juice is sold in 2-packs or 5-packs. Let $Q(n)$ be the statement that it is possible to buy n cans of juice by combining 2-packs and 5-packs. This problem discusses using strong induction to prove that for any $n \geq 4$, $Q(n)$ is true.

- 1) The inductive step will prove that for $k \geq 5$, $Q(k+1)$ is true. What is the inductive hypothesis?

- ☐ $Q(j)$ is true for any $j = 4, 5, \dots, k$.
- ☐ $Q(j)$ is true for any $j = 4, 5, \dots, k+1$.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022



- ☐ $Q(j)$ is true for any
 $j = 5, \dots, k+1$.

2) The inductive step will prove that for $k \geq 5$, $Q(k+1)$ is true. What part of the inductive hypothesis is used in the proof?



- ☐ $Q(k-2)$
☐ $Q(k-1)$
☐ $Q(k)$

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

3) In the base case, for which values of n should $Q(n)$ be proven directly?



- ☐ $n = 3, n = 4$, and $n = 5$.
☐ $n = 4$ and $n = 5$.
☐ $n = 4, n = 5$, and $n = 6$.

The next example comes from number theory. The assertion is that every positive integer greater than 1 can be written as a product of prime numbers. For example:

$$\begin{aligned}20 &= 2 \cdot 2 \cdot 5 \\156 &= 2 \cdot 2 \cdot 3 \cdot 13 \\71 &= 71\end{aligned}$$

The last example, 71, is itself prime so it is the product of only one prime number.

Theorem 7.6.1: Prime factorization of integers greater than 1.

For any integer $n \geq 2$, n can be expressed as a product of prime numbers.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Proof 7.6.2: Prime factorization of integers greater than 1.

Proof.

By strong induction on n .

Base case: $n = 2$. Since 2 is a prime number, it already is a product of one prime number: 2.

Inductive step: Assume that for $k \geq 2$, any integer j in the range from 2 through k can be expressed as a product of prime numbers. We will show that $k + 1$ can be expressed as a product of prime numbers.

If $k + 1$ is prime, then it is a product of one prime number, $k + 1$. If $k + 1$ is not prime, $k + 1$ is composite and can be expressed as the product of two integers, a and b , that are each at least 2. We need to show that both a and b are at most k in order to apply in the inductive hypothesis.

Since $k+1 = a \cdot b$, then $a = (k+1)/b$. Furthermore, since $b \geq 2$, then $a = (k+1)/b < k+1$. If a is an integer which is strictly less than $k + 1$, then $a \leq k$. The symmetric argument can be used to show that $b = (k+1)/a \leq k$. Thus a and b both fall in the range from 2 through k which means that the inductive hypothesis can then be applied and they can each be expressed as a product of primes:

$$\begin{aligned} a &= p_1 \cdot p_2 \cdots p_l \\ b &= q_1 \cdot q_2 \cdots q_m \end{aligned}$$

Now $k + 1$ can be expressed as a product of primes:

$$k + 1 = a \cdot b = (p_1 \cdot p_2 \cdots p_l) \cdot (q_1 \cdot q_2 \cdots q_m).$$



PARTICIPATION ACTIVITY

7.6.5: Applying the inductive hypothesis.



- 1) In the proof that every integer greater than 1 can be expressed as a product of primes, it is necessary to apply the inductive hypothesis to integers a and b . What must be proven about a and b in order to apply the inductive hypothesis?

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022



- ☐ a and b are greater than 1.
- ☐ a and b are in the range from 2 through $k + 1$.
- ☐ a and b are in the range from 2

The principle of induction is closely related to another principle regarding the integers called the well-ordering principle:

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Figure 7.6.4: The well-ordering principle.

The **well-ordering principle** says that any non-empty subset of the non-negative integers has a smallest element.

It can be shown that the principle of mathematical induction, the principle of strong induction, and the well-ordering principle are all equivalent. In other words, using any one of the principles as an assumption, the other two principles can be proven. Below is a proof showing that if the well-ordering principle is true, then the principle of mathematical induction is true. The principle of induction is stated again below for review:

Figure 7.6.5: The principle of mathematical induction.

Let $P(n)$ be a predicate that is parameterized by non-negative integers n . If the following two conditions hold:

- **Base case:** $P(1)$ is true
- **Inductive step:** For all $k \geq 1$, $P(k)$ implies $P(k+1)$

then for all $n \geq 1$, $P(n)$ is true.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Proof 7.6.3: Well-ordering implies the principle of mathematical induction.

Proof.

We will show that the well-ordering principle implies the contrapositive of the principle of mathematical induction. In particular, we will show that if the well-ordering principle is true and the conclusion of the principle of mathematical induction is false then at least one of the two hypotheses for induction must fail.

Suppose that the principle of mathematical induction is true and that $P(n)$ is false for some $n \geq 1$. Consider the set of integers n such that $n \geq 1$ and $P(n)$ is not true. By the well-ordering principle, there must be a smallest element m in that set. Note that $m \neq 0$ because 0 is not a member of the set. (One of the criteria for an integer n to be included in the set is that $n \geq 1$.) If $m = 1$, then $P(1)$ is not true and the base case fails. If $m \geq 2$, then $P(m - 1)$ is true but $P(m)$ is false. Letting $k = m - 1$, $k \geq 1$ and $P(k)$ is true but $P(k + 1)$ is not true. Therefore, the inductive step fails for some $k \geq 1$. Thus, if the statement $P(n)$ is false for some $n \geq 1$, then one of the two conditions for the principle of mathematical induction must be false. ■

The well-ordering principle can be used to prove some facts directly. Typically, a proof that uses the well-ordering principle is a proof by contradiction. There is a statement $P(n)$, parameterized by an integer n and the assertion to be proven is that $P(n)$ is true for all $n \geq b$ for some constant b . Define a set that consists of all integers which violate the assertion to be proven. The set contains all integers n such that $n \geq b$ and $P(n)$ is false. The well-ordering principle says that if the set is non-empty then it must have a smallest element. The details of the proof of course depend on the nature of the statement $P(n)$, but typically a contradiction arises by examining the smallest element in the set. Often the contradiction involves showing that there must be a yet a smaller value for which $P(n)$ is false. If the set of all integers $n \geq b$ such that $P(n)$ is false can not have a smallest element, the set must be empty and therefore $P(n)$ must be true for all $n \geq b$. We illustrate the use of the well-ordering principle in proving the following theorem:

Theorem 7.6.2: The Division Algorithm.

Let n be an integer and d a positive integer. There are unique integers q and r such that $n = qd + r$ and $0 \leq r < d$.

The number q is the integer quotient when n is divided by d and r is the remainder. It turns out that given n and d , the numbers q and r are unique. For now, the well-ordering principle will be used

simply to show that they exist.

Proof 7.6.4: Proof of the Division Algorithm using the well-ordering principle.

Proof.

Consider the set S of all non-negative integers of the form $(n - td)$, where t is an integer. The set is non-empty because t can be chosen so that $-td$ is as large as desired and large enough to make $(n - td)$ non-negative. Let r be the smallest number in the set S and let q be the value for t which achieves the minimum: $r = n - qd$. By definition, r is non-negative and therefore $r \geq 0$. The fact that $r < d$ will be proven by contradiction. Suppose that $r \geq d$. Then $r - d \geq 0$.

$$r - d = (n - qd) - d = n - (q + 1)d$$

Since $r - d$ can be written in the form $(n - td)$ by selecting $t = q + 1$, r was not the smallest non-negative number in the set of integers of the form $n - td$. A contradiction. ■

PARTICIPATION ACTIVITY

7.6.6: Using the well-ordering principle.



The well-ordering principle can be used to prove any fact which can be proven by standard or strong induction.

Consider again the problem of purchasing cans of juice. Suppose that juice is sold in 2-packs or 5-packs. Let $Q(n)$ be the statement that it is possible to buy n cans of juice by combining 2-packs and 5-packs. We can use the principle of well-ordering to show that for $n \geq 4$, $Q(n)$ is true.

1) To which set should the well-ordering principle be applied?



- ☐ The set of all $n \geq 1$ such that $Q(n)$ is false.
- ☐ The set of all $n \geq 4$ such that $Q(n)$ is true.
- ☐ The set of all $n \geq 4$ such that $Q(n)$ is false.

2) Let m be the smallest value of the set from question 1. Select the statement that correctly describes m .



- ☐ $Q(m)$ is false, but for any j in the range from 4 through $m - 1$, $Q(j)$ is true.
- ☐ For any j in the range from 4 through m , $Q(j)$ is true.
- ☐ $Q(m)$ is true, but for any j in the range from 4 through $m - 1$, $Q(j)$ is false.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022



3) What is the contradiction that results if $m = 4$ or $m = 5$?

- ☐ Since it is possible to buy 2 or 3 cans, just add an extra 2-pack to get 4 or 5 cans.
- ☐ It is possible to buy 4 cans (two 2-packs) and 5 cans (one 5-pack). Therefore, $Q(4)$ and $Q(5)$ are both true.

4) What is the contradiction that results if $m \geq 6$?



- ☐ It is possible to buy $m - 2$ cans. m cans can be purchased by buying $m - 2$ cans and adding a 2-pack.
- ☐ $Q(6)$ is true because it is possible to buy 6 cans by purchasing three 2-packs.

Additional exercises

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

**EXERCISE**

7.6.1: Proofs by strong induction - combining stamps.



Prove each of the following statements using strong induction.

- Prove that any amount of postage worth 8 cents or more can be made from 3-cent or 5-cent stamps.
- Prove that any amount of postage worth 24 cents or more can be made from 7-cent or 5-cent stamps.
- Prove that any amount of postage worth 12 cents or more can be made from 3-cent or 7-cent stamps.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

**EXERCISE**

7.6.2: Proofs by strong induction - explicit formulas for recurrence relations.



Prove each of the following statements using strong induction.

- The Fibonacci sequence is defined as follows:

- $f_0 = 0$
- $f_1 = 1$
- $f_n = f_{n-1} + f_{n-2}$, for $n \geq 2$

Prove that for $n \geq 0$,

$$f_n = \frac{1}{\sqrt{5}} \left[\left(\frac{1 + \sqrt{5}}{2} \right)^n - \left(\frac{1 - \sqrt{5}}{2} \right)^n \right]$$

- Define the sequence $\{h_n\}$ as follows:

- $h_0 = 5/3$
- $h_1 = 11/3$
- $h_n = 3h_{n-1} + 4h_{n-2} + 6n$, for $n \geq 2$

Prove that for $n \geq 0$,

$$h_n = 2 \cdot 4^n + \frac{3}{2}(-1)^n - n - \frac{11}{6}$$

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

- Define the sequence $\{g_n\}$ as follows:

- $g_0 = 51$
- $g_1 = 348$
- $g_n = 5g_{n-1} - 6g_{n-2} + 20 \cdot 7^n$ for $n \geq 2$

Prove that for $n \geq 0$, $g_n = 2^n + 3^n + 7^{n+2}$.



EXERCISE

7.6.3: Proving upper bounds for recurrence relations.



Prove each of the following statements using strong induction.

(a) Define the sequence $\{a_n\}$ as follows:

- $a_1 = a_2 = a_3 = 1$
- $a_n = a_{n-1} + a_{n-2} + a_{n-3}$ for $n \geq 4$

Prove that for all $n \geq 1$, $a_n < 2^n$.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

(b) Define the sequence $\{a_n\}$ as follows:

- $a_0 = a_1 = 2$
- $a_n = (a_{n-1})^2 a_{n-2}$, for $n \geq 2$

Prove that for all $n \geq 0$, $a_n \leq 2^{3^n}$



EXERCISE

7.6.4: Number representation with induction.



(a) Prove that every positive integer can be expressed as the sum of distinct non-negative integer powers of 2. In other words, prove that for every positive integer n , there are non-negative integers b_0, b_1, \dots, b_r that are all distinct ($b_0 < b_1 < \dots < b_r$) such that

$$n = 2^{b_0} + 2^{b_1} + 2^{b_2} + \dots + 2^{b_r}.$$

(Hint: in order to express $k + 1$ in the inductive step, start with finding a number b such that $2^b \leq k + 1 < 2^{b+1}$)



EXERCISE

7.6.5: Proof by the well-ordering principle.



Prove each of the following statements using the principle of well-ordering.

(a) Define the sequence $\{g_n\}$ as follows:

- $g_0 = 51$
- $g_1 = 348$
- $g_n = 5g_{n-1} - 6g_{n-2} + 20 \cdot 7^n$ for $k \geq 2$

Prove that for $n \geq 0$, $g_n = 2^n + 3^n + 7^{n+2}$.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

(b) Prove that any amount of postage worth 8 cents or more can be made from 3-cent or 5-cent stamps.

7.7 Recursive algorithms

A **recursive algorithm** is an algorithm that calls itself. Like recursively defined sequences and structures, a recursively defined algorithm has a base case in which the output is computed directly on an input of small size or value. On a larger input, the algorithm calls itself on an input of smaller size and uses the result to construct a solution to the larger input. An algorithm's calls to itself are known as **recursive calls**. Here is an example of a recursive algorithm to compute $n!$. **Comments** in the pseudo-code, which begin with "//", explain the purpose of certain steps and are not part of the algorithm itself.

Example 7.7.1: Recursive algorithm to compute the factorial function.

Factorial(n)

Input: A non-negative integer n .

Output: $n!$

If ($n = 0$), Return(1)

$r := \text{Factorial}(n - 1)$ // The recursive call

Return($r * n$)

PARTICIPATION ACTIVITY

7.7.1: An execution of the recursive algorithm to compute $4!$.



Animation captions:

1. A call to Factorial with input $n = 4$ results in a call to Factorial with input $n-1 = 3$.
2. The call Factorial(3) results in the call Factorial(2), which calls Factorial(1), which calls Factorial(0). When the input n is 0, Factorial returns 1.
3. The call Factorial(0) returns 1 to Factorial(1). Factorial(n) returns $n \cdot \text{Factorial}(n-1)$, so Factorial(1) returns $1 \cdot 1 = 1$.
4. Factorial(2) returns $2 \cdot 1 = 2$. Factorial(3) returns $3 \cdot 2 = 6$. The original call to Factorial(4) returns $4 \cdot 6 = 24$.

The algorithm given below takes in as input a set A and returns the power set of A . Recall that the power set of A (denoted $P(A)$) is the set of all subsets of A . The elements of $P(A)$ are subsets of A .

Note that the details of how the set is actually represented and manipulated on a computer are omitted, but there are enough details to run through the algorithm on paper. The base case for the algorithm is when the input A is the empty set. The power set of the empty set is the set containing the empty set as its only element. Then for the recursive step, the algorithm removes an element x from A to obtain A' whose cardinality is strictly smaller than A . The algorithm is then run recursively on A' . All the elements in the power set of A' are also in the power set of A . In addition, the power set of A contains each set in $P(A')$ with the element x added back in.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Example 7.7.2: Recursive algorithm to compute the power set of a set.

PowerSet(A)

Input: A set A .

Output: The power set of A .

If ($A = \emptyset$), Return($\{\emptyset\}$)

Select an element $a \in A$

$A' := A - \{a\}$

$P := \text{PowerSet}(A')$ //the recursive
call

$P' := P$

For each $S \in P'$

 Add $S \cup \{a\}$ to P

End-for

Return(P)

PARTICIPATION ACTIVITY

7.7.2: An execution of the recursive algorithm to compute the power set of $\{a, b, c\}$.



Animation captions:

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

1. A call to $\text{PowerSet}(\{a, b, c\})$ removes a to get the set $\{b, c\}$ and then calls $\text{PowerSet}(\{b, c\})$.
2. A call to $\text{PowerSet}(\{b, c\})$ removes b to get the set $\{c\}$ and then calls $\text{PowerSet}(\{c\})$ which then calls $\text{PowerSet}(\emptyset)$. $\text{PowerSet}(\emptyset)$ returns $\{\emptyset\}$.
3. When $\{\emptyset\}$ is returned to $\text{PowerSet}(\{c\})$, the algorithm adds $\emptyset \cup \{c\} = \{c\}$ to the power set and returns $\{\emptyset, \{c\}\}$.
4. $\text{PowerSet}(\{b, c\})$ adds b to all the sets and returns $\{\emptyset, \{c\}, \{b\}, \{b, c\}\}$. The $\text{PowerSet}(\{a, b, c\})$

adds a to all the sets and returns $\{\emptyset, \{c\}, \{b\}, \{b,c\}, \{a\}, \{a,c\}, \{a,b\}, \{a,b,c\}\}$

The recursive algorithm below takes as input a string s and outputs the string with all the characters reversed. The algorithm removes the first character from the string, recursively reverses the rest of the string and then adds the first character back at the end of the string.

Example 7.7.3: Recursive algorithm to reverse a string.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

ReverseString(s)

Input: A string s.

Output: The reverse of s.

If s is the empty string, Return(s)

Let c be the first character in s

Remove c from s

s' := ReverseString(s)

s := string s' with c added to the
end

Return(s)

PARTICIPATION ACTIVITY

7.7.3: An execution of the recursive algorithm to reverse the string 'POT'.



Animation captions:

1. ReverseString("POT") removes the first char P to get "OT" and calls ReverseString("OT").
2. Then O is removed and ReverseString("T") is called. Then T is removed and ReverseString is called with the empty string which returns the empty string.
3. ReverseString("T") adds T to the end of the empty string to get "T".
4. The string "T" is returned to the call ReverseString("OT") and the O is added to the end of "T" to get "TO".
5. The string "TO" is returned to the call ReverseString("POT") and the P is added to the end of "TO" to get "TOP", which is the final return value.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

**PARTICIPATION
ACTIVITY**

7.7.4: A recursive algorithm to compute an exponential.



Here is a recursive algorithm to compute r^n with some lines missing. The input r can be any real number. The input n is assumed to be a non-negative integer.

Exponent(r, n)

Input: Real number r and a non-negative integer n .

Output: r^n

----- //The base case

$p :=$ Exponent(-----) //The recursive call

Return($r \cdot p$)

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

1) What missing input values should be used in the recursive call?



- ☐ ($r, n-1$)
- ☐ ($r-1, n-1$)
- ☐ ($r-1, n$)

2) What should go in the first blank (labeled "The base case")?



- ☐ If ($r=0$), return(1)
- ☐ If ($n=1$), return(r)
- ☐ If ($n=0$), return(1)

In the above examples, each algorithm has one recursive call. It is also possible for an algorithm to call itself multiple times. Consider the following recursive algorithm to compute Fibonacci numbers:

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Example 7.7.4: Recursive algorithm to compute Fibonacci numbers.

RecursiveFibonacci(n)

Input: a non-negative integer n.

Output: the Fibonacci number with index n.

If (n=0) Return(0)

If (n=1) Return(1)

f = RecursiveFibonacci(n-1) + RecursiveFibonacci(n-2)

Return(f)

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

For $n = 0$ or $n = 1$, the algorithm returns the correct value without any recursive calls. When $n \geq 2$, the call to `Fibonacci(n)` generates two more recursive calls. The recursive algorithm above is not the most efficient way to compute Fibonacci numbers. As demonstrated in the animation below, the algorithm performs many redundant calculations:

PARTICIPATION ACTIVITY

7.7.5: A recursive algorithm to compute Fibonacci numbers.



Animation captions:

1. RecursiveFibonacci(5) calls RecFib(4) and RecFib(3).
2. RecFib(3) calls RecFib(2) and RecFib(1). RecFib(1) returns from the Base Case.
3. RecFib(2) calls RecFib(1) and RecFib(0) both of which return from the Base Case.
4. The call to RecFib(4) made by the original call to RecursiveFibonacci(5) results in eight more recursive calls.
5. RecFib(1) is computed a total of 5 times and RecFib(0) is computed 3 times.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

It is much more efficient to compute Fibonacci numbers by using a for-loop and storing the last two numbers in the sequence. The efficiency comes from the fact that the algorithm stores and reuses partial results instead of recomputing.

Example 7.7.5: Non-recursive algorithm to compute Fibonacci numbers.

Non-RecursiveFibonacci(n)

Input: A non-negative integer n.

Output: The Fibonacci number with index n.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

If (n = 0), Return(0)

If (n = 1), Return(1)

last = 1

oneBeforeLast = 0

For k = 2 to n

current := last + oneBeforeLast

oneBeforeLast := last

last := current

End-for

Return(current)

PARTICIPATION ACTIVITY

7.7.6: Computing Fibonacci numbers.



- 1) How many addition operations are performed in RecursiveFibonacci(5)?



Check

[Show answer](#)

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

- 2) How many addition operations are performed in Non-RecursiveFibonacci(5)?



Check[Show answer](#)

Additional exercises

**EXERCISE**

7.7.1: Recursively computing sums of cubes.



©zyBooks 12/15/22 00:21 1361995

John Farrell

COLOSTATECS220SeaboltFall2022

- (a) Give a recursive algorithm to compute the sum of the cubes of the first n positive integers. The input to the algorithm is a positive integer n . The output is $\sum_{j=1}^n j^3$. The algorithm should be recursive, it should not compute the sum using a closed form expression or an iterative loop.

**EXERCISE**7.7.2: Recursively computing the sum of the first n positive odd integers.

- (a) Give a recursive algorithm which takes as input a positive integer n and returns the sum of the first n positive odd integers.

**EXERCISE**

7.7.3: Recursively computing the maximum and minimum of a sequence.



The input to the maximum and minimum problems is a sequence of numbers, $a_1 \dots a_n$, where n , the length of the sequence, is a positive integer. The function $\text{length}(a_1 \dots a_n)$ returns n , the length of the sequence. If $n > 1$, you can also create a new sequence $a_1 \dots a_{n-1}$, which is the original sequence $a_1 \dots a_n$, with the last number a_n omitted.

- (a) Give a recursive algorithm which takes as input a sequence of numbers and returns the minimum (i.e., smallest) number in the sequence. Your algorithm should not use an iterative loop.
- (b) Give a recursive algorithm which takes as input a sequence of numbers and returns the maximum (i.e., largest) number in the sequence. Your algorithm should not use an iterative loop.

©zyBooks 12/15/22 00:21 1361995

John Farrell

COLOSTATECS220SeaboltFall2022

**EXERCISE**

7.7.4: Reversing a string and removing blanks.



- (a) Give a recursive algorithm that takes as input a string s , removes the blank characters and reverses the string. For example, on input "Hello There", the algorithm should return "erehTolleH". The function $\text{IsBlank}(c)$ returns a boolean value indicating whether the character c is the blank character. Your algorithm should not use an iterative loop.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

**EXERCISE**

7.7.5: Product of primes.



- (a) Give a recursive algorithm that takes as input an integer n which is greater than 1. The algorithm should return the product of all the primes in the range 2 through n . For example, on input 7, the algorithm will return 210, which is the product of 2, 3, 5, and 7. The function $\text{IsPrime}(x)$ takes a positive integer x and returns a boolean value indicating whether x is prime.

**EXERCISE**

7.7.6: Recursively computing the product of two non-negative integers.



- (a) Give a recursive algorithm that takes as input two non-negative integers x and y and returns the product of x and y . The only arithmetic operations your algorithm can perform are addition or subtraction. Your algorithm should have no iterative loops.

**EXERCISE**

7.7.7: Recursively computing the sum of two non-negative integers.



- (a) Give a recursive algorithm that takes as input two non-negative integers x and y and returns the sum of x and y . The only arithmetic operations your algorithm can perform are $\text{Increment}(x)$ which returns $x+1$ and $\text{Decrement}(x)$ which returns $x-1$. Your algorithm should have no iterative loops.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022



EXERCISE

7.7.8: Recursively computing the set of all binary strings of a fixed length.



- (a) Give a recursive algorithm that takes as input a non-negative integer n and returns a set containing all binary strings of length n . Here are the operations on strings and sets you can use:

- Initialize an empty set S .
- Add a string x to S .
- $y := 0x$. This operation adds a 0 to the beginning of string x and assigns the result to string y .
- $y := 1x$. This operation adds a 1 to the beginning of string x and assigns the result to string y .
- $\text{return}(S)$
- A looping structure that performs an operation on every string in a set S :
 for every x in S ,
 // perform some sequence of steps with string x .

©zyBooks 12/15/22 00:21 1361995
 John Farrell
 COLOSTATECS220SeaboltFall2022



EXERCISE

7.7.9: Recursively computing a number raised to an exponent that is a power of 2.



- (a) Give a recursive algorithm whose input is a real number r and a non-negative integer n , and whose output is $r^{(2^n)}$. **Note that the exponent of r is 2^n .** Your algorithm should only use addition and multiplication operations and should not contain any iterative loops.

7.8 Induction and recursive algorithms

Induction is a useful proof technique to prove facts about recursive algorithms such as establishing that a recursive algorithm gives the correct output for every possible valid input. The first step is to prove that the base case works correctly. The inductive step then shows that the algorithm returns the correct value assuming that all of the recursive calls return the correct values.

An inductive proof can be used to show that the recursive algorithm `ReverseString` returns the reverse of the input string.

©zyBooks 12/15/22 00:21 1361995
 John Farrell
 COLOSTATECS220SeaboltFall2022

**PARTICIPATION
 ACTIVITY**
7.8.1: Overview of the proof of correctness for `ReverseString`.

Animation captions:

1. On a call to `ReverseString(s)`, if s is empty (the base case), then the reverse of s is s .
2. The characters in s are numbered $c_1 c_2 c_3 \dots c_k c_{k+1}$. When the first character is removed from s , the result is $c_2 c_3 \dots c_k c_{k+1}$.
3. By the inductive hypothesis, the recursive call returns the correct output, $s' = c_{k+1} c_k \dots c_3 c_2$.
4. The algorithm returns s' with c_1 added to the end, which is $c_{k+1} c_k \dots c_3 c_2 c_1$, the correct reverse of s .

©zyBooks 12/15/22 00:21 1361995
John Farrell

COLOSTATECS220SeaboltFall2022

Theorems proven by induction are usually parameterized by an index, such as "For all $n \geq 0$, $P(n)$ is true". The inductive step assumes that $P(k)$ is true and then proves that $P(k+1)$ is true. The most natural statement of the fact that `ReverseString` works correctly does not have an explicit parameter n :

For every input string s , `ReverseString` returns the reverse of string s .

Therefore, the proof starts out with the statement: "By induction on the length of the string", which means that the length of the string will be the parameter n . The theorem being proven is implicitly:

If s is a string of length n , then `ReverseString(s)` returns the reverse of string s .

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Proof 7.8.1: Proof of correctness for ReverseString.

Theorem: For every input string s , ReverseString returns the reverse of string s .

Proof.

- By induction on the length of the string.
- **Base case:** s is a string of length 0. Since s has no characters, the reverse of s is s , which is the string returned by the algorithm.
- **Inductive step:** Assume that for $k \geq 0$, ReverseString returns the correct output for any string of length k . Then we will prove that ReverseString returns the correct output for any string of length $k+1$.
 - Let s be a string of length $k + 1$. The characters in s are numbered as $s = c_1c_2\ldots c_kc_{k+1}$.
 - The algorithm ReverseString makes a recursive call whose input is the string s with the first character removed: $c_2\ldots c_kc_{k+1}$.
 - The string $c_2\ldots c_kc_{k+1}$ has k characters, so by the inductive hypothesis, the recursive call to ReverseString correctly returns the reverse of string $c_2\ldots c_kc_{k+1}$, which is $c_{k+1}c_k\ldots c_2$.
 - The algorithm ReverseString on input s , returns $c_{k+1}c_k\ldots c_2$ with c_1 added to the end: $c_{k+1}c_k\ldots c_2c_1$, which is in fact the original string s with the characters reversed. ■

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

An inductive proof is also used to show that the recursive algorithm PowerSet works correctly. The input is a finite set A and the output should be the power set of A , the set whose elements are all the subsets of A . The proof of correctness must show that if PowerSet(A) returns P , then P contains all of the subsets of A . In addition, the proof must establish that P does not have any additional elements that are not subsets of A . In other words, the proof must show that $X \subseteq A$ if and only if $X \in P$.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Figure 7.8.1: Recursive algorithm to compute the power set of a set.

PowerSet(A)

Input: Finite set A.

Output: The power set of A.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

1. If $A = \emptyset$, Return($\{\emptyset\}$)
2. Select an element $a \in A$
3. $A' := A - \{a\}$
4. $P := \text{PowerSet}(A')$ //the recursive call
5. $P' := P$
6. For each $S \in P'$
7. Add $S \cup \{a\}$ to P
8. Return(P)

**PARTICIPATION
ACTIVITY**

7.8.2: Proof of correctness for PowerSet.



Theorem: For any finite set A, PowerSet(A) returns the correct power set of A.

- 1) The first statement of the proof is "By induction on the size of the input set A." Select the phrase to fill in the blank to get a restatement of the theorem parameterized by n:



Let A be _____. The PowerSet(A) returns the correct power set of A.

- ☐ The empty set.
- ☐ A finite set.
- ☐ A finite set of size n.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

- 2) The base case is proven for $n = 0$.
What is the set A, when $n = 0$?



☐ the empty set

☐ an arbitrary set

☐ an arbitrary finite set

3) The inductive hypothesis assumes that for $k \geq 0$, the theorem holds for any set with k elements. What must be proven in the inductive step?



☐ If A has 0 elements then $\text{PowerSet}(A)$ returns the correct power set of A .

☐ If A has k elements then $\text{PowerSet}(A)$ returns the correct power set of A .

☐ If A has $k + 1$ elements then $\text{PowerSet}(A)$ returns the correct power set of A .

4) Suppose that a set A has $k + 1$ elements and $a \in A$. Then what is the size of the set $A - \{a\}$?



☐ $k - 1$

☐ k

☐ $k + 1$

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Proof 7.8.2: Proof of correctness for PowerSet.

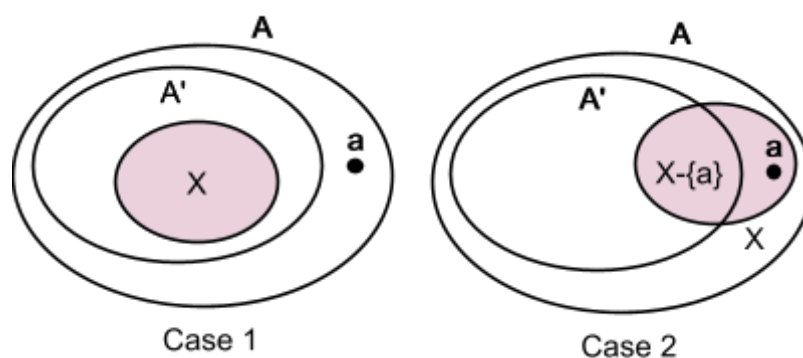
Theorem: For any finite set A , $\text{PowerSet}(A)$ returns the correct power set of A .

Proof.

By induction on the size of the input set A .

Base case: If $|A| = 0$, then $A = \emptyset$. On input $A = \emptyset$, $\text{PowerSet}(A)$ returns $\{\emptyset\}$ which is exactly the power set of A .

Inductive step: Assume that PowerSet works correctly on an input set of size k and prove that PowerSet works correctly on an input set of size $k + 1$. Let A be a set with $k + 1$ elements. The algorithm $\text{PowerSet}(A)$ removes an element a from the set A to obtain a new set A' with one less element, so $A' = A - \{a\}$, and $|A'| = k + 1 - 1 = k$. Let P' be the set returned by $\text{PowerSet}(A')$ and P the set returned by $\text{PowerSet}(A)$. The inductive hypothesis says that P' is exactly the power set of A' . We will use the inductive hypothesis to show that P is the power set of A . That is $X \subseteq A$ if and only if $X \in P$. The first direction is to prove that if $X \subseteq A$, then $X \in P$. There are two cases pictured below:



Case 1: $a \notin X$. Then X is also a subset of A' . Since X is in the power set of A' , by the inductive hypothesis, $X \in P'$. The algorithm only adds more subsets to P' to get P , so X is also in P .

Case 2: $a \in X$. Then $X - \{a\}$ is a subset of $A - \{a\} = A'$. By the inductive hypothesis, $X - \{a\}$ is included in P' . Therefore, in Step 7, the algorithm includes $(X - \{a\}) \cup \{a\} = X$ in P .

Therefore, if $X \subseteq A$, then $X \in P$.

The final step is to show that if $X \in P$, then $X \subseteq A$. Any set added to P is either in P' , and therefore a subset of A' , or is $S \cup \{a\}$, where S is a subset of A' . Therefore if a set is added to P , then the set is a subset of A .

■

FastExpRec is a recursive algorithm that computes r^n , where r can be any real number and n is a non-negative integer. FastExpRec is much faster than the algorithm that just multiplies r times itself $n - 1$ times. However, the fact that FastExpRec works is much less intuitive. An inductive proof is used to show FastExpRec works correctly.

The correctness of the algorithm FastExpRec, hinges on the fact that if n is even then $n = 2x$ for some integer x , and if n is odd, then $n = 2x + 1$, for some integer x . In either case $\lfloor n/2 \rfloor = x$:

$$\left\lfloor \frac{2x}{2} \right\rfloor = \lfloor x \rfloor = x$$

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

$$\left\lfloor \frac{2x + 1}{2} \right\rfloor = \left\lfloor x + \frac{1}{2} \right\rfloor = x$$

In the inductive step of the proof, we will assume that the algorithm returns the correct value for any x in the range $0 \dots k$, and will prove that the algorithm returns the correct value for $k+1$. The proof of the inductive step will replace the variable n from the algorithm with $k+1$.

PARTICIPATION ACTIVITY

7.8.3: Overview of the proof of correctness for FastExpRec.



Animation content:

undefined

Animation captions:

1. On a call to FastExpRec(r, n), the smallest value for n is 0 because n must be non-negative. FastExpRec($r, 0$) returns $r^0 = 1$.
2. $x = \lfloor n/2 \rfloor$. If n is even, $n = 2x$. If n is odd, $n = 2x+1$. In the proof, n is replaced by $k+1$.
3. By the inductive hypothesis, FastExpRec(r, x) returns r^x , for any x in the range from 0 to k .
4. If n is even, the algorithm returns the square of FastExpRec(r, x), which is $(r^x)^2 = r^{2x} = r^n = r^{k+1}$.
5. If n is odd, the algorithm returns the square of FastExpRec(r, x) times r , which is $(r^x)^2 r = r^{2x} r = r^{2x+1} = r^n = r^{k+1}$.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Proof 7.8.3: Proof of correctness for FastExpRec.

Theorem: On input n and r , where r is a real number and n is a non-negative integer, FastExpRec returns r^n .

Proof.

By induction on n .

Base case: $n = 0$. When $n = 0$, $r^n = 1$. The algorithm returns 1.

Inductive Step: We show that if FastExpRec(r, x) returns r^x , for any x in the range from 0 through k , then FastExpRec($r, k + 1$) returns r^{k+1} . There are two cases, depending on whether $k + 1$ is odd or even.

Case 1: $k + 1$ is even. Then $k + 1 = 2x$, for some integer x and $\lfloor (k + 1)/2 \rfloor = x$. The integer x is non-negative and less than $k + 1$, so x falls in the range from 0 through k . By the inductive hypothesis, the recursive call to FastExpRec(r, x) returns r^x . The call to FastExpRec($k + 1$) returns

$$y^2 = (\text{FastExpRec}(r, x))^2 = (r^x)^2 = r^{2x} = r^{k+1}$$

Case 2: $k + 1$ is odd. Then $k + 1 = 2x + 1$, for some integer x and $\lfloor (k + 1)/2 \rfloor = \lfloor (2x + 1)/2 \rfloor = \lfloor x + (1/2) \rfloor = x$. The integer x is non-negative and less than $k + 1$, so x falls in the range from 0 through k . By the inductive hypothesis, the recursive call to FastExpRec(r, x) returns r^x . The call to FastExpRec($k + 1$) returns

$$y^2 \cdot r = (\text{FastExpRec}(r, x))^2 \cdot r = (r^x)^2 \cdot r = r^{2x} \cdot r = r^{2x+1} = r^{k+1}$$

Therefore, the call to FastExpRec($r, k + 1$) returns r^{k+1} . ■

PARTICIPATION ACTIVITY

7.8.4: Proof of correctness of Exponent.



The algorithm Exponent is a recursive algorithm which computes r^n in a different way than FastExpRec. The input r can be any real number. The input n is assumed to be a non-negative integer.

- 1) The correctness of algorithm Exponent(r, n) is proven by induction on n . What would be proven in the base case?

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022



Exponent(r, n)

Input: Real number r and a non-negative integer n .

Output: r^n

If ($n = 0$), Return(1) //The base case

$p := \text{Exponent}(r, n-1)$

Return($r \cdot p$)

☐ Exponent($0, n$)
returns 0

☐ Exponent($r, 0$)
returns 1

2) Suppose that the inductive hypothesis is that

☐ Exponent($r, 1$)
returns r

☐ Exponent(r, k)
returns r^k . What fact

must be proven in

the inductive step?

☐ Exponent($r, k+1$)
returns r^{k+1} .

☐ Exponent($r+1, k$)
returns $(r+1)^k$.

☐ Exponent($r, k-1$)
returns r^{k-1} .



3) The inductive step shows that the value returned by Exponent($r, k+1$) is:

$$p \cdot r = \text{Exponent}(r, k) \cdot r = r^k \cdot r = r^{k+1}$$

Which equality makes use of the inductive hypothesis?

☐ $p \cdot r = \text{Exponent}(r, k) \cdot r$

☐ $\text{Exponent}(r, k) \cdot r = r^k \cdot r$

☐ $r^k \cdot r = r^{k+1}$

Additional exercises

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

**EXERCISE**

7.8.1: Recursively computing sums of cubes, cont.



- (a) Use induction to prove that the algorithm to compute the sum of the cubes of the first n positive integers (shown below) returns the correct value for every positive integer input.

SumCube(n)

Input: A positive integer n .

Output: $1^3 + 2^3 + \dots + n^3$.

If ($n = 1$), Return(1)

$s := \text{SumCube}(n - 1)$ // The recursive call

Return($n^3 + s$)

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

**EXERCISE**7.8.2: Recursively computing the sum of the first n positive odd integers, cont.

- (a) Use induction to prove that your algorithm to compute the sum of the first n positive odd integers returns the correct value for every positive integer input.

**EXERCISE**

7.8.3: Recursively computing the maximum and minimum of a sequence, cont.



- (a) Use induction to prove that your algorithm to compute the minimum of a sequence outputs the correct value for every non-empty sequence of numbers.
(Hint: the minimum value of a sequence $a_1 \dots a_n$ is the unique value x such that $x = a_j$ for some $1 \leq j \leq n$ and $x \leq a_i$ for every $1 \leq i \leq n$.)
- (b) Use induction to prove that your algorithm to compute the maximum of a sequence outputs the correct value for every non-empty sequence of numbers.
(Hint: the maximum value of a sequence $a_1 \dots a_n$ is the unique value x such that $x = a_j$ for some $1 \leq j \leq n$ and $x \geq a_i$ for every $1 \leq i \leq n$.)

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

**EXERCISE**

7.8.4: Reversing a string and removing blanks, cont.



- (a) Use induction to prove that your algorithm to reverse a string and remove blank characters outputs the correct string for every possible input string.

**EXERCISE**

7.8.5: Product of primes, cont.



- (a) Use induction to prove that your algorithm to compute the product of primes in the range 2 through n computes the correct value for every integer n which is strictly greater than one.

**EXERCISE**

7.8.6: Recursively computing the product of two non-negative integers, cont.



- (a) Use induction to prove that your algorithm that recursively computes the product of x and y returns the correct output for every pair (x, y) where x and y are non-negative integers.

**EXERCISE**

7.8.7: Recursively computing the sum of two non-negative integers, cont.



- (a) Use induction to prove that your algorithm that recursively computes the sum of x and y returns the correct output for every pair (x, y) where x and y are non-negative integers.

**EXERCISE**

7.8.8: Recursively computing the set of all binary strings of a fixed length, cont.



- (a) Use induction to prove that your algorithm to compute the set of all binary strings of length n returns the correct set for every input n , where n is a non-negative integer.

**EXERCISE**

7.8.9: Recursively computing a number raised to an exponent that is a power of 2, cont.



- (a) Use induction to prove that your algorithm to compute $r^{(2^n)}$ returns the correct value for every input pair (r, n) , where r is a real number and n is a non-negative integer.



EXERCISE

7.8.10: A fast algorithm to multiply two non-negative integers.



- (a) The following algorithm takes as input two non-negative integers, x and y , and returns the product of x and y .

FastMult(x, y)

Input: Two non-negative integers, x and y

Output: The product of x and y

If ($y = 0$), Return(0)

$z := \lfloor y/2 \rfloor$

$p := \text{FastMult}(x, z)$ // The recursive call

If (y is even)

Return($2 \cdot p$)

Else

Return($2 \cdot p + x$)

End-if

Use induction to prove that the algorithm returns the correct value.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

7.9 Recursive definitions

The **factorial** function $f(n) = n!$ for $n \geq 0$ can be defined as: $f(n) = n! = n \cdot (n - 1) \cdots 1$

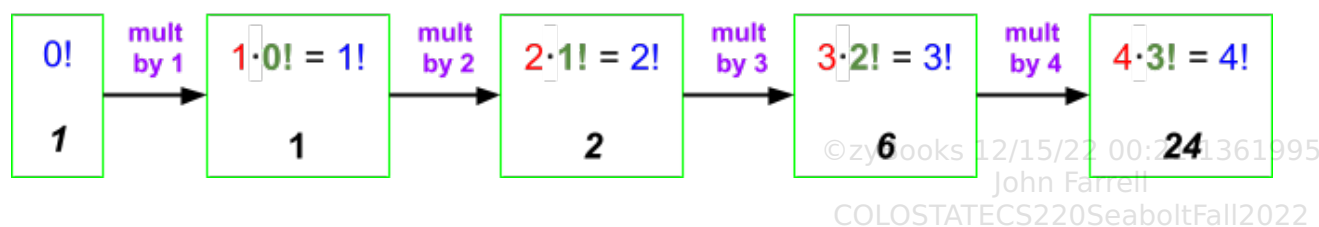
A disadvantage of the definition given above is that the reader is required to infer what goes between the $(n - 1)$ and the 1 terms. Also, the value of $0!$ is not clear from the definition. A more precise definition is $n! = f(n)$ such that:

$$\begin{aligned} f(0) &= 1 \\ f(n) &= n \cdot f(n - 1) \quad \text{for } n \geq 1 \end{aligned}$$

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

The second definition is an example of a recursive definition. In a **recursive definition** of a function, the value of the function is defined in terms of the output value of the function on smaller input values. One can use the recursive definition for $n!$ to determine the value of the factorial function on a particular value for n by starting at 0 , multiplying $0!$ by 1 to get the value of $1!$ then multiplying by 2 to get $2!$, and so on, until the desired $n!$ has been reached. The process is called recursion.

Recursion is the process of computing the value of a function using the result of the function on smaller input values.

Figure 7.9.1: Using the recursive definition of $n!$ to compute $4!$ 

We can use recursion to define functions, sequences and sets. For example, a recurrence relation (along with initial values) is a recursive definition of a sequence because the recurrence relation shows how to compute the value of a term as a function of terms with smaller indices (i.e., terms that occur earlier in the sequence). A sequence is just a special kind of function in which the domain is a consecutive set of integers. The notation for a sequence is slightly different since the input is specified with a subscript: $f_n = f(n)$. Here is a recurrence relation for sequence $\{f_n\}$ that is equivalent to the recursive definition for the factorial function:

$$\begin{aligned} f_0 &= 1 \\ f_n &= n \cdot f_{n-1} \quad \text{for } n \geq 1 \end{aligned}$$

PARTICIPATION ACTIVITY

7.9.1: Computing the value of functions defined recursively.



- 1) Given the definition of a function g whose domain is the set of all non-negative integers:



$$\begin{aligned} g(0) &= 0 \\ g(n) &= g(n-1) + n^3 \quad \text{for } n \geq 1 \end{aligned}$$

What is $g(3)$?

Check

[Show answer](#)

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

- 2) Given the definition of a function h whose domain is the set of all non-negative integers:



$$\begin{aligned} h(0) &= 1 \\ h(n) &= (n^2 + 1) \cdot h(n-1) \quad \text{for } n \geq 1 \end{aligned}$$

What is $h(2)$?

Check

[Show answer](#)

**PARTICIPATION
ACTIVITY**

7.9.2: Finding recursive definitions for functions on non-negative integers.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Find the recursive definition that matches the non-recursive definition given for each function.

1) $g(n) = \sum_{j=0}^n j^2$



- ☐ $g(0) = 0$
 $g(n) = n^2 + (g(n-1))^2$ for $n \geq 1$
- ☐ $g(0) = 0$
 $g(n) = n^2 + g(n-1)$ for $n \geq 1$
- ☐ $g(0) = 0$
 $g(n) = n^2 \cdot g(n-1)$ for $n \geq 1$

2) $h(n) = (n!)^2$



- ☐ $h(0) = 1$
 $h(n) = n^2 \cdot h(n-1)$ for $n \geq 1$
- ☐ $h(0) = 1$
 $h(n) = n^2 \cdot (h(n-1))^2$ for $n \geq 1$
- ☐ $h(0) = 1$
 $h(n) = n \cdot (h(n-1))^2$ for $n \geq 1$

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Recursively defined sets

Certain kinds of sets are most naturally specified with recursive definitions. A recursive definition of a set shows how to construct elements in the set by putting together smaller elements.

Definition 7.9.1: Components of a recursive definition of a set.

- A **basis** explicitly states that one or more specific elements are in the set.
- A **recursive rule** shows how to construct additional elements in the set from elements already known to be in the set. (There is often more than one recursive rule).
- An **exclusion statement** states that an element is in the set only if it is given in the basis or can be constructed by applying the recursive rules repeatedly to elements given in the basis.

Consider the task of specifying the correct syntax of a programming language, such as formally specifying that the parentheses within a statement must be properly nested. For simplicity, we disregard the rest of the characters in the statement and just focus on strings of left and right parentheses. For example, the string `(())` is properly nested, but the string `())` is not. Formally specifying which strings are and are not properly nested is important because a compiler must systematically check that an expression is in the correct format, and must otherwise issue an appropriate error statement. Below is a recursive definition for the set of all strings of parentheses that are properly nested:

Figure 7.9.2: Recursive definition of the set of all properly nested parentheses.

- Basis: The sequence `()` is properly nested.
- Recursive rules: If `u` and `v` are properly-nested sequences of parentheses then:
 1. `(u)` is properly nested.
 2. `uv` is properly nested.
- Exclusion statement: a string is properly nested only if it is given in the basis or can be constructed by applying the recursive rules to strings in the basis.

Since the exclusion statement is essentially the same for every recursively defined set, it is often excluded (but implied) in a recursive definition of a set.

The basis states that the string `()` is properly nested. Rule 1 can be applied to `()` to show that `(())` is also properly nested. Rule 2 can be applied by taking two copies of `()` and concatenating them to

get the string $()()$ which is also properly nested. Properly nested strings of length 6 can be obtained by taking either one of the properly nested string of length 4 and applying rule 1: $((())())$ and $((())())$. In addition, a properly nested string of length 2 can be concatenated with a properly nested string of length 4 to get properly nested strings of length 6: $()((()))$, $((())())()$, $()()()$.

Notice that in this definition, there is more than one way to apply the recursive rules to get the same string. For example, the string $()()()$ can be obtained by concatenating $()$ and $()()$ in either order: $()()()$ or $()()()$. In some situations, it is important to have a set of recursive rules in which there is only one way to construct each element in the set using the recursive rules.

**PARTICIPATION
ACTIVITY**

7.9.3: Constructing a properly nested string or parentheses recursively.

**Animation content:**

undefined

Animation captions:

1. The string $((())())()$ is properly nested because $((())())()$ can be constructed by repeatedly applying the recursive rules to the string $()$ in the basis.
2. The string $((()))$ is not properly nested. It is proven elsewhere that there is no way to construct the string using the recursive rules.
3. For example, if we start by constructing the string $((()))$, a right parenthesis cannot be added to the end because there is no matching left parenthesis.

**PARTICIPATION
ACTIVITY**

7.9.4: Identifying properly nested parentheses.



Which strings of parentheses are properly nested?

1) $()()$ 

- ☐ Properly nested
☐ Not properly nested

2) $((()))$ 

- ☐ Properly nested
☐ Not properly nested

3) $((())())$ 

☐ Properly nested

**PARTICIPATION
ACTIVITY**

7.9.5: Applying recursive rules to get properly nested parentheses.



What was the last recursive rule applied in constructing each of the following properly nested string or parentheses?

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Recursive rules:

1. (u) is properly nested.
2. uv is properly nested.

1) (())()



- ☐ Rule 1
- ☐ Rule 2

2) (())()()



- ☐ Rule 1
- ☐ Rule 2

3) (((())()()))



- ☐ Rule 1
- ☐ Rule 2

Example: binary strings

The set B^k , where $B = \{0, 1\}$, is defined to be the set of all binary strings of length k . The set of all binary strings without any restriction on length (denoted by B^*) is an infinite set. The **empty string** (denoted by the symbol λ) is the unique string whose length is 0. Since B^0 is the set of all binary strings of length 0, $B^0 = \{\lambda\}$. B^* includes all of B^k for any $k \geq 0$. One way to define B^* is by an infinite union:

$$B^* = B^0 \cup B^1 \cup B^2 \cup \dots$$

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

The recursive definition of B^* does not require using an infinite union of finite sets. The recursive rule constructs a longer string from a shorter string by concatenating 0 or 1. The recursive definition of B^* is:

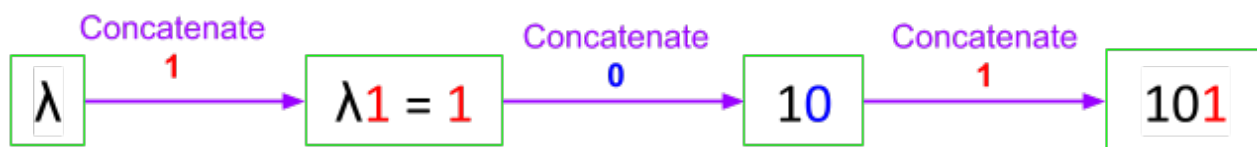
- Base case: $\lambda \in B^*$
- Recursive rule: if $x \in B^*$ then,

- $x_0 \in B^*$
- $x_1 \in B^*$

Every binary string can be constructed by starting with λ and repeatedly concatenating 0 or 1 to the end of the string. The diagram below shows an example:

Figure 7.9.3: Constructing a binary string using the recursive definition.

Creating: 101



PARTICIPATION ACTIVITY

7.9.6: How many applications of the recursive rule to build a string?



- 1) How many applications of the recursive rule given in the definition of B^* are required to construct the string 1?



Check

Show answer

- 2) How many applications of the recursive rule given in the definition of B^* are required to construct the string 11001?



Check

Show answer

Recursive definition for the length of a binary string

The length $|x|$ of a binary string x is a function that maps every binary string to a non-negative integer. The length of a binary string can be defined recursively based on the recursive definition of $\{0,1\}^*$.

- Base case: $|\lambda| = 0$
- Recursive rule: if $x \in B^*$ then,
 - $|x0| = |x| + 1$
 - $|x1| = |x| + 1$

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

PARTICIPATION ACTIVITY

7.9.7: Applying the recursive definition for the length of a binary string.



- 1) According to the recursive definition for the length of a string, what is $|001|$?



Check

Show answer

- 2) If $|x| = 7$, then what is $|x0|$?



Check

Show answer

Recursive definition for perfect binary trees

The next example is a recursive definition for a set of mathematical objects that are not strings. A tree has **vertices** (denoted by a circle) and **edges** (denoted by line segments) which connect pairs of vertices. Not every collection of vertices and edges is a tree. A formal definition of trees along with their properties is given elsewhere in this material. Here we give a recursive definition for a particular class of trees called perfect binary trees. Each perfect binary tree has a designated vertex called the **root**.

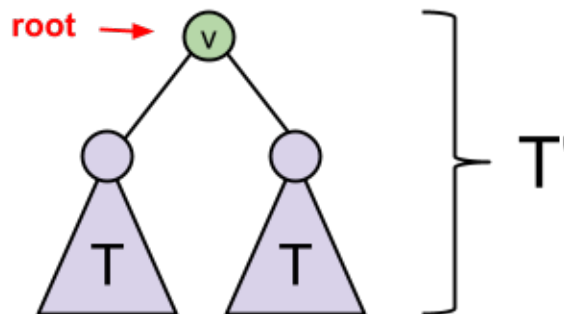
©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Definition 7.9.2: Recursive definition for the set of perfect binary trees.

Basis: A single vertex with no edges is a perfect binary tree. The root is the only vertex in the tree.

root → 

Recursive rule: If T is a perfect binary tree, then a new perfect binary tree T' can be constructed by taking two copies of T , adding a new vertex v and adding edges between v and the roots of each copy of T . The new vertex v is the root of T' .



PARTICIPATION ACTIVITY

7.9.8: Using the recursive rule to construct perfect binary trees from smaller ones.



Animation captions:

1. The tree T consisting of a single node is a perfect binary tree.
2. To create a new tree, T is duplicated.
3. A new vertex v is added, along with edges from v to the roots of the two copies of T .
4. The result is a perfect binary tree with three vertices. Call the new tree T .
5. T is duplicated and a new vertex v is added, along with edges from v to the roots of the two copies of T , resulting in a perfect binary tree with 7 vertices.

PARTICIPATION ACTIVITY

7.9.9: How many applications of the recursive rule to build a perfect binary tree?



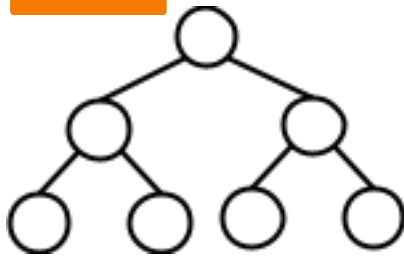
How many recursive steps need to be applied in order to obtain the following perfect binary trees?

1) 



Check[Show answer](#)

2)

**Check**[Show answer](#)

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

**CHALLENGE
ACTIVITY**

7.9.1: Recursive definitions.



422102.2723990.qx3zqy7

Start

The domain of function f is the set of all positive integers and the range is the set of all integers.

$$f(1) = 5$$

$$f(n) = (n - 1) \cdot f(n - 1) + 2 \quad \text{for } n \geq 2$$

What is the value of $f(3)$?

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

1

2

3

Check**Next****Additional exercises**

**EXERCISE**

7.9.1: Properly nested parentheses and curly braces.



- (a) Give a recursive definition for strings of properly nested parentheses and curly braces. For example, $\{\}\{\}\{\}$ is properly nested but $\{\}\}$ is not properly nested. The empty string should not be included in your definition.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

**EXERCISE**

7.9.2: Devising recursive definitions for sets of strings.



Let $A = \{a, b\}$.

- (a) Give a recursive definition for A^* .
- (b) The set A^+ is the set of strings over the alphabet $\{a, b\}$ of length at least 1. That is $A^+ = A^* - \{\lambda\}$. Give a recursive definition for A^+ .
- (c) Let S be the set of all strings from A^* in which there is no b before an a . For example, the strings λ , aa , bbb , and $aabbbb$ all belong to S , but $aabab \notin S$. Give a recursive definition for the set S . (Hint: a recursive rule can concatenate characters at the beginning or the end of a string.)
- (d) For $x \in A^*$, let $bCount(x)$ be the number of occurrences of the character b in x . Give a recursive definition for $bCount$.

**EXERCISE**

7.9.3: Recursive definition for the set of strings over a finite set of characters.



Let A be an arbitrary finite set of characters. For example A could be $\{v, w, x, y, z\}$ or $\{0, 1, 2\}$.

- (a) Give a recursive definition for A^* .
- (b) Give a recursive definition for $|x|$, the length of a string $x \in A^*$.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

**EXERCISE**

7.9.4: Recursive definitions for subsets of binary strings.



Give a recursive definition for each subset of the binary strings. A string x should be in the recursively defined set if and only if x has the property described.

- (a) The set S consists of all strings with an even number of 1's.
- (b) The set S is the set of all binary strings that are palindromes. A string is a palindrome if it is equal to its reverse. For example, 0110 and 11011 are both palindromes.
- (c) The set S consists of all strings that have the same number of 0's and 1's.

**EXERCISE**

7.9.5: Multiple copies of a string.



- (a) If x is a string and j is a non-negative integer, then x^j is defined to be j copies of the string concatenated together. For example, if $x = 101$, then $x^4 = 101101101101$, and $x^0 = \lambda$. Give a recursive definition for x^j .

**EXERCISE**

7.9.6: Recursive definitions for subsets of integers.



- (a) A subset S of the integers is defined recursively as follows:

- Base case: $2 \in S$
- Recursive rule: if $k \in S$, then
 - $k + 5 \in S$
 - $k - 5 \in S$

List the elements of S whose absolute value is less than 20.

- (b) A subset T of the integers is defined recursively as follows:

- Base case: $2 \in T$
- Recursive rule: if $k \in T$, then
 - $k + 5 \in T$

List the elements of T whose absolute value is less than 20.

- (c) Give a recursive definition for E , the set of all even integers. Recall that 0 is an even integer. Also an even integer can be negative, such as -2 and -4.
- (d) Give a recursive definition for O , the set of all odd integers. Recall that an odd integer can be negative, such as -1 and -3.



EXERCISE

7.9.7: Interpreting a recursive definition of a set of strings.



- (a) The recursive definition given below defines a set S of strings over the alphabet $\{a, b\}$:

- Base case: $\lambda \in S$ and $a \in S$
- Recursive rule: if $x \in S$ then,
 - $xb \in S$
 - $xba \in S$

List all the strings of length at most 3 in S .

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022



EXERCISE

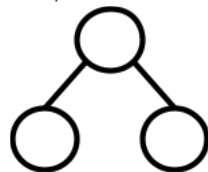
7.9.8: The height of a perfect binary tree.



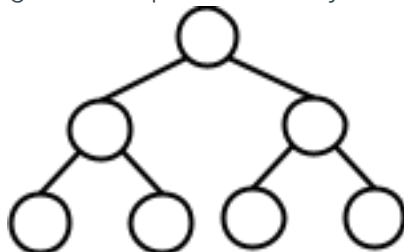
The function h maps every perfect binary tree to a non-negative integer. $h(T)$ is called the height of tree T . The function h is defined recursively as follows:

- If T is the perfect binary tree that consists of a single vertex, then $h(T) = 0$.
- Suppose that the tree T' is constructed by taking two copies of perfect binary tree T , adding a new vertex v and adding edges between v and the roots of each copy of T . Then $h(T') = h(T) + 1$.

- (a) What is the height of the perfect binary tree shown below?



- (b) What is the height of the perfect binary tree shown below?



- (c) Describe the function h in words

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

**EXERCISE**

7.9.9: Recursive definition for the number of vertices in a perfect binary tree.



- (a) The function v maps every perfect binary tree to a positive integer. $v(T)$ is equal to the number of vertices in T . Give a recursive definition for $v(T)$.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022



EXERCISE

7.9.10: A recursive definition for full binary trees.



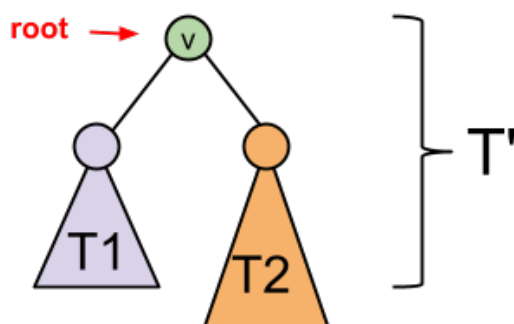
Here is a definition for a set of trees called full binary trees.

Basis: A single vertex with no edges is a full binary tree. The root is the only vertex in the tree.

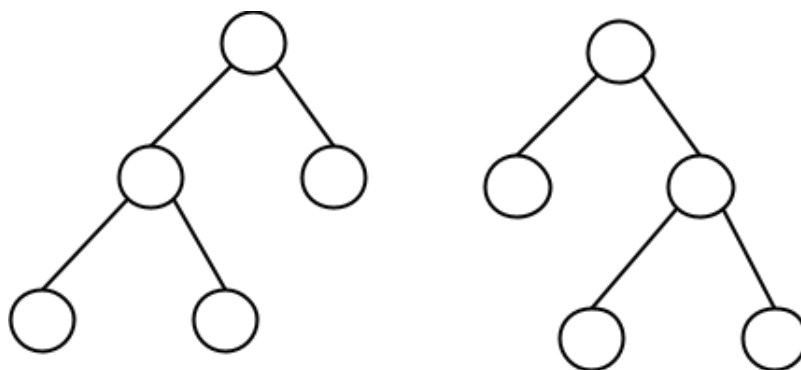


©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Recursive rule: If T_1 and T_2 are full binary trees, then a new tree T' can be constructed by first placing T_1 to the left of T_2 , adding a new vertex v at the top and then adding an edge between v and the root of T_1 and an edge between v and the root of T_2 . The new vertex v is the root of T' .



Note that it makes a difference which tree is placed on the left and which tree is placed on the right. For example, the two trees below are considered to be different full binary trees:



(a) Draw all possible full binary trees with 3 or fewer vertices.

(b) Draw all possible full binary trees with 5 vertices.

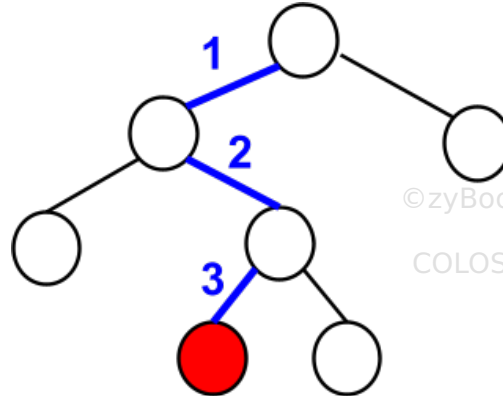
(c) Draw all possible full binary trees with 7 vertices.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

(d) The function v maps every full binary tree to a positive integer. $v(T)$ is equal to the number of vertices in T . Give a recursive definition for $v(T)$.

(e) The function h maps every full binary tree to a non-negative integer. $h(T)$ is called the height of tree T and is defined to be the maximum number of edges reached in a path that starts at the vertex and always travels downward in the tree. For example, the tree shown below has height 3 because the number of edges reached in a path

the tree shown below has height 3 because the number of edges reached in a path from the root to the red vertex is 3. Furthermore, there is no path from the root that reaches four edges and always travels downward.



©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Give a recursive definition for $h(T)$ for every full binary tree T .

(Hint: you may need the max function. For any two numbers, x and y , $\max(x, y) = x$ if $x \geq y$ and $\max(x, y) = y$ if $y > x$.)

7.10 Structural induction

Induction is a powerful technique to prove that all the elements of a recursively defined set have a particular property. **Structural induction** is a type of induction used to prove theorems about recursively defined sets that follows the structure of the recursive definition.

As an example, we prove that every properly nested string of left and right parentheses is balanced. A string of parentheses is **balanced** if the number of left parentheses is equal to the number of right parentheses. For example, in the string $((()))$, the number of left parentheses and the number of right parentheses is 3. We will use structural induction to prove that any string of properly nested parentheses is balanced. It will be useful to define some notation to use in the proof. If x is a string of left and right parentheses, then $\text{left}[x]$ is equal to the number of left parentheses in x and $\text{right}[x]$ is the number of right parentheses in x . The functions "left" and "right" map every string of parentheses to non-negative integers, regardless of whether the string is balanced or not. Here are some examples to illustrate. The colors highlight left and right parentheses for readability.

- $\text{right}[())()] = 3$
- $\text{left}[())()] = 0$
- $\text{left}[((()))] = 3$

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

PARTICIPATION ACTIVITY

7.10.1: The left and right functions for strings of parentheses.



Evaluate each of the following:

1) $\text{right}[())]$



Check
[Show answer](#)

 2) $\text{left}[()>()()]$


Check
[Show answer](#)

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

3) If u is a string of parentheses such that $\text{left}[u] = 3$, then what is $\text{left}[(u)]$?



Check
[Show answer](#)

4) If u and v are strings of parentheses such that $\text{right}[u] = 2$ and $\text{right}[v] = 5$, then what is $\text{right}[uv]$?



Check
[Show answer](#)

Let P denote the set of properly nested parentheses. The set P is recursively defined below.

Definition 7.10.1: Recursive definition for the set of properly nested parentheses.

- Basis: $() \in P$.
- Recursive rules: If $u \in P$ and $v \in P$ then:

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

1. $(u) \in P$.
2. $uv \in P$.

We will use structural induction to show that if x is a string of properly nested parentheses ($x \in P$), then x has the property of being balanced ($\text{left}[x] = \text{right}[x]$). Note that the proof does not imply the converse of the statement: if a string of parentheses has the same number of left and right parentheses then the string is properly nested. In fact the converse of the statement is not true since the string $"()"$ is balanced but not properly nested.

The animation below shows an outline of the proof along with the elements of a structural induction proof.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

PARTICIPATION ACTIVITY

7.10.2: Outline for a proof using structural induction.



Animation content:

Animation captions:

1. Show that for every x in a recursively defined set P , x has a particular property. (Show that every x in P is balanced.)
2. Show that every element in the basis satisfies the property. (Show that $"()"$ is balanced.)
3. Inductive step: The recursive rules show how to construct larger elements in P using smaller elements from P . Assume the property for the smaller elements and prove for the larger ones.
4. Case 1: show that if u is balanced, then (u) is balanced. Case 2: show that if u and v are balanced, then uv is balanced.

PARTICIPATION ACTIVITY

7.10.3: Identifying elements of a proof by structural induction.



The set S is a set of strings defined recursively as follows

- Basis: $\lambda \in S$
- Recursive rules: if $x \in S$ and $y \in S$ then,
 - $xa \in S$ (Rule 1)
 - $xbb \in S$ (Rule 2)
 - $xy \in S$ (Rule 3)

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Consider a proof that every string in S has an even number of b 's. Select whether each proof part belongs in the base case or the inductive step.

- 1) Assume that x has an even number of b 's. Show that xbb also has an even



number of b's.

☒ Base case

☐ Inductive step

2) Show that λ has an even number of b's.

☐ Base case

☐ Inductive step

3) Assume that x and y have an even number of b's. Prove that xy also has an even number of b's.

☐ Base case

☐ Inductive step

4) Assume that x has an even number of b's. Prove that xa also has an even number of b's.

☐ Base case

☐ Inductive step

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Often the inductive step of an inductive proof is the most difficult part of the proof. The animation below illustrates how the inductive step of the proof that a string of properly nested parentheses is balanced works on some particular examples. The full proof is given afterwards.

PARTICIPATION ACTIVITY

7.10.4: Examples showing the inductive step of a structural induction proof.

Animation content:

undefined

Animation captions:

1. Suppose a string x is obtained by applying Rule 1 to properly nested string u . The string $u = (()())$ has three left parentheses, so $\text{left}[u] = 3$.
2. The string u also has three right parentheses, so $\text{right}[u] = 3$.
3. If x is obtained by applying recursive rule 1 to u , then $x = (u) = (()())()$.
4. $\text{left}[x] = \text{left}[(()())()]$. The string $(()())()$ has one more left parenthesis than $(())()$.
5. Therefore, $\text{left}[(()())()] = 1 + \text{left}[()()] = 1 + 3 = 1 + \text{right}[()()]$, because $\text{left}[u] = 3 = \text{right}[u]$.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

6. $1 + \text{right}[(())()] = \text{right}[(()())]$, because $(())()$ has one more right parenthesis than $(())()$.
7. Now consider Rule 2. $u = ()$. $v = ()$. $\text{left}[(())] = 2 = \text{right}[(())]$. Also, $\text{left}[] = 1 = \text{right}[]$.
8. x is obtained by applying Rule 2 to u and v : $x = uv = (())()$.
9. $\text{left}[x] = \text{left}[(())()] = \text{left}[(())] + \text{left}[]$.
10. $\text{left}[(())] + \text{left}[] = 2 + 1 = \text{right}[(())] + \text{right}[]$.
11. $\text{right}[(())] + \text{right}[] = \text{right}[(())()]$.

©zyBooks 12/15/22 00:21 1361995

John Farrell

COLOSTATECS220SeaboltFall2022

Proof 7.10.1: Properly nested parentheses are balanced.

Theorem: If string $x \in P$, then $\text{left}[x] = \text{right}[x]$.

Proof.

By induction.

Base case: $() \in P$. $\text{left}[] = \text{right}[] = 1$.

Inductive step: If $x \in P$, then x was constructed by applying a sequence of recursive rules starting with the string $()$ given in the basis. We consider two cases, depending on the last recursive rule that was applied to construct x .

Case 1: Rule 1 is the last rule applied to construct x . Then $x = (u)$, where $u \in P$. We assume that $\text{left}[u] = \text{right}[u]$ and prove that $\text{left}[x] = \text{right}[x]$. Then

$$\begin{aligned}
 \text{left}[x] &= \text{left}[(u)] && \text{because } x = (u) \\
 &= 1 + \text{left}[u] && (u) \text{ has one more "(" than } u \\
 &= 1 + \text{right}[u] && \text{by the inductive hypothesis} \\
 &= \text{right}[(u)] && (u) \text{ has one more ")" than } u \\
 &= \text{right}[x] && \text{because } x = (u)
 \end{aligned}$$

Case 2: rule 2 is the last rule applied to construct x . Then $x = uv$, where $u \in P$ and $v \in P$. We assume that $\text{left}[u] = \text{right}[u]$ and $\text{left}[v] = \text{right}[v]$ and then prove that $\text{left}[x] = \text{right}[x]$. Then

$$\begin{aligned}
 \text{left}[x] &= \text{left}[uv] && \text{because } x = uv \\
 &= \text{left}[u] + \text{left}[v] \\
 &= \text{right}[u] + \text{right}[v] && \text{by the inductive hypothesis} \\
 &= \text{right}[uv] \\
 &= \text{right}[x] && \text{because } x = uv
 \end{aligned}$$

Therefore, $\text{left}[x] = \text{right}[x]$. ■

Structural induction and perfect binary trees

The next example uses structural induction to prove the following theorem for perfect binary trees.

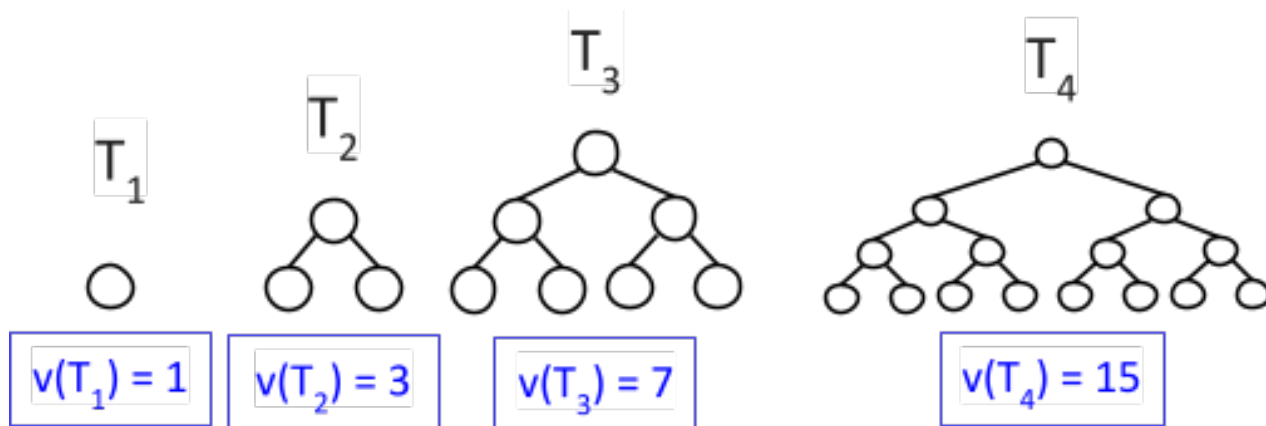
Theorem 7.10.1: Number of vertices in a perfect binary tree.

Let T be a perfect binary tree. Then the number of vertices in T is $2^k - 1$ for some positive integer k .

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

If T is a perfect binary tree T , then let $v(T)$ denote the number of vertices in T . The figure below shows a few examples of perfect binary trees and the number of vertices in each.

Figure 7.10.1: Number of vertices in perfect binary trees.



PARTICIPATION ACTIVITY

7.10.5: Using structural induction to prove a theorem about perfect binary trees.



This question outlines a proof of the theorem that the number of vertices in a perfect binary tree is $2^k - 1$ for some positive integer k .

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

- 1) The basis in the recursive definition of perfect binary trees says that T_1 , the tree with one vertex and no edges, is a perfect binary tree. What fact must be proven in the base case for the proof?

☐ $v(T_1) = 1 = 2^1 - 1$.



☐ $v(T_1) = 0 = 2^0 - 1.$

- 2) Let T' be a perfect binary tree created by at least one application of the recursive rule using a smaller perfect binary tree T . Select the statement that corresponds to the inductive hypothesis.



- ☐ $v(T') = 2^k - 1$, for some positive integer k .
- ☐ $v(T') = 2 \cdot v(T) - 1.$
- ☐ $v(T) = 2^k - 1$, for some positive integer k .

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

- 3) The recursive rule constructs T' by taking two copies of T , adding a vertex v , and adding edges between v and the roots of each copy of T . Select which statement is true about the relationship between the number of vertices in T and the number of vertices in T' .



- ☐ $v(T) = 2 \cdot v(T') + 1.$
- ☐ $v(T') = 2 \cdot v(T) + 1.$
- ☐ $v(T') = v(T) + 1.$

- 4) Here is the argument for the inductive step. In which line is the inductive hypothesis applied?



$$v(T') = 2 \cdot v(T) + 1; \quad (1)$$

$$= 2(2^k - 1) + 1; \quad (2)$$

$$= 2^{k+1} - 1 \quad (3)$$

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

- ☐ 1
- ☐ 2
- ☐ 3

Proof 7.10.2: Proof for the number of vertices in a perfect binary tree using structural induction.

Proof.

By induction.

Base case: $2^1 - 1 = 1$, so the tree with one vertex has $2^k - 1$ leaves, with $k = 1$.

Inductive step: Let T' be a perfect binary tree that is created with one or more applications of the recursive rule. The last recursive rule that is applied to create T' takes a perfect binary tree T , duplicates T and adds a new vertex v with edges to each of the roots of the two copies of T . We assume that $v(T) = 2^k - 1$, for some positive integer k and prove that $v(T') = 2^j - 1$ for some positive integer j .

The number of vertex in T' is twice the number of vertices in T (because of the two copies of T) plus 1 (because of the vertex v that is added), so $v(T') = 2 \cdot v(T) + 1$. By the inductive hypothesis, $v(T) = 2^k - 1$ for some positive integer k . Therefore

$$v(T') = 2 \cdot v(T) + 1 = 2(2^k - 1) + 1 = 2 \cdot 2^k - 2 + 1 = 2^{k+1} - 1$$

Since k is a positive integer, $j = k+1$ is also a positive integer. Therefore the number of vertices in T' is $2^j - 1$, where j is a positive integer. ■

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Additional exercises



EXERCISE

7.10.1: Properly nested parentheses and curly braces, cont.



- (a) Use structural induction to prove that for properly nested parentheses and curly braces:
1. the number of left parentheses is the same as the number of right parentheses, and
 2. the number of left curly braces is equal to the number of right curly braces.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022



EXERCISE

7.10.2: Proving facts about recursively defined sets of strings.



(a) Consider a set of strings defined recursively as follows:

- Base case: $a \in S$
- Recursive rule: if $x \in S$ then,
 - $xb \in S$ (Rule 1)
 - $xa \in S$ (Rule 2)

Prove that every string in S begins with the character a .

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

(b) Consider a set of strings defined recursively as follows:

- Base case: $a \in S$
- Recursive rules: if $x \in S$ then,
 - $xb \in S$ (Rule 1)
 - $bx \in S$ (Rule 2)

Prove that every string in S contains exactly one a .

(c) Consider a set of strings defined recursively as follows:

- Base case: $a \in S$
- Recursive rules: if $x \in S$ then,
 - $xb \in S$ (Rule 1)
 - $bx \in S$ (Rule 2)
 - $axa \in S$ (Rule 3)
 - $xaa \in S$ (Rule 4)

Prove that every string in S contains an odd number of a 's.

Note that the set S as defined does not contain every string over the alphabet $\{a, b\}$ that contains an odd number of a 's. Therefore, the converse of this statement (if x has an odd number of a 's then $a \in S$) is not true. For example, there is no way to create the string "aaababa" by applying the recursive rules to the string "a" given in the base case.

(d) Consider a set of strings defined recursively as follows:

- Base case: $\lambda \in S$
- Recursive rules: if $x \in S$ and $y \in S$ then,
 - $axb \in S$ (Rule 1)
 - $bx a \in S$ (Rule 2)
 - $xy \in S$ (Rule 3)

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Prove that every string in S contains the same number of a 's and b 's.

Note that your proof does not necessarily imply that every string that has the same number of a 's and b 's is in S .



EXERCISE

7.10.3: Characterizing the strings in a recursively defined set.



The recursive definition given below defines a set S of strings over the alphabet $\{a, b\}$:

- Base case: $\lambda \in S$ and $a \in S$
- Recursive rule: if $x \in S$ then,
 - $xb \in S$ (Rule 1)
 - $xba \in S$ (Rule 2)

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

This problem asks you to prove that the set S is exactly the set of strings over $\{a, b\}$ which do not contain two or more consecutive a 's. In other words, you will prove that $x \in S$ if and only if x does not contain two consecutive a 's. The two directions of the "if and only if" are proven separately.

- Use structural induction to prove that if a string $x \in S$, then x does not have two or more consecutive a 's.
- Use strong induction on the length of a string x to show that if x does not have two or more consecutive a 's, then $x \in S$. Specifically, prove the following statement parameterized by n :
For any $n \geq 0$, let x be a string of length n over the alphabet $\{a, b\}$ that does not have two or more consecutive a 's, then $x \in S$.



EXERCISE

7.10.4: Multiple copies of a string, cont.



- If x is a string and j is a non-negative integer, then x^j is defined to be j copies of the string concatenated together. For example, if $x = 101$, then $x^4 = 101101101101$. Use the recursive definition for x^j , where x is a binary string and j is a non-negative integer, to prove that $|x^j| = j \cdot |x|$.
You can use the fact that for any two binary strings, x and y , $|xy| = |x| + |y|$.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022



EXERCISE

7.10.5: Inductive proofs about a subset of the integers.



A subset T of the integers is defined recursively as follows:

- Base case: $2 \in T$
- Recursive rule: if $k \in T$, then
 - $k + 5 \in T$

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

This problem asks you to prove that T is exactly the set of integers that can be expressed as $5m+2$, where m is a non-negative integer. In other words, you will prove that $x \in T$ if and only if $x = 5m+2$, for some non-negative integer m . The two directions of the "if and only if" are proven separately.

- Use structural induction to prove that if $k \in T$, then $k = 5m + 2$, for some non-negative integer m .
- Use structural induction to prove that if $k = 5m + 2$, for some non-negative integer m , then $k \in T$.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022



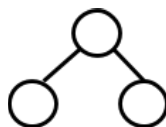
EXERCISE

7.10.6: Proving facts about perfect binary trees.



Here is a recursive definition for a set of trees. The set of trees called TREES.

Base case: The tree shown below is in TREES.



©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Recursive rule: If T is in TREES, then a new element of TREES called T' can be constructed by taking two copies of T , adding a new vertex v and adding edges between v and the roots of each copy of T . The new vertex v is the root of T' .

Note that the definition of TREES is the same as the definition for perfect binary trees except that the tree consisting of a single vertex is not included in the set TREES.

- (a) The degree of a vertex is the number of edges connected to that vertex. Define $d(v)$ to be the degree of vertex v .

Let T be tree in TREES and let r be the root of T . Then

- $d(r) = 2$
- For every vertex $v \neq r$, $d(v) = 1$ or $d(v) = 3$.

- (b) Let T be a tree in TREES. Let $L(T)$ be the number of vertices in T such that $d(v) = 1$. Let $E(T)$ be the number of vertices in T such that $d(v) > 1$. Then $L(T) = E(T) + 1$. You can use the fact proven in the previous question.



EXERCISE

7.10.7: The height of a perfect binary tree, cont.



- (a) Let T be a perfect binary tree. This problem makes use of the recursive definitions for $v(T)$, the number of vertices in T and $h(T)$ the height of T . Prove that $h(T) = \log_2(v(T) + 1) - 1$. (Hint: you may use the fact that $v(T)$ is always positive and for any positive number x , $\log_2 x + 1 = \log_2(2x)$.)

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022



EXERCISE

7.10.8: A recursive definition for full binary trees, cont.



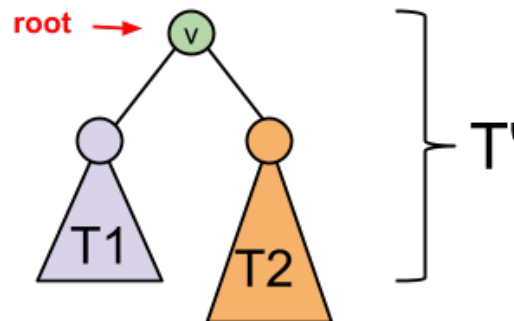
Here is a definition for a set of trees called full binary trees.

Basis: A single vertex with no edges is a full binary tree. The root is the only vertex in the tree.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022



Recursive rule: If T_1 and T_2 are full binary trees, then a new tree T' can be constructed by first placing T_1 to the left of T_2 , adding a new vertex v at the time and then adding an edge between v and the root of T_1 and an edge between v and the root of T_2 . The new vertex v is the root of T' .



- Prove that a full binary tree has an odd number of vertices. An integer x is odd if $x = 2k + 1$, for some integer k .
- Let T be a full binary tree. This problem makes use of the recursive definitions for $v(T)$, the number of vertices in T and $h(T)$ the height of T .
Prove that $v(T) \geq 2 \cdot h(T) + 1$
(Hint: you may use the fact that $h(T)$ is always non-negative and for any two non-negative numbers, x and y , $x + y \geq \max(x, y)$. This follows from the fact that if $x \geq 0$ and $y \geq 0$, then $x + y \geq x$ and $x + y \geq y$.)

7.11 Make Postage and Permutations

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

For this assignment you will be implementing two functions `make_postage` and `permutations`.

`makepostage(money)`: This recursive function returns the number of three cent and five cent stamps that are needed to make postage of the given amount following the inductive proof presented in class. The return value is a tuple `(numfive, numthree)` that indicates the number of five cent and three cent stamps, respectively that together make up the given amount. For example, `makepostage(15)` will return the tuple `(0, 5)`.

permutations(s): return all the permutations of a tuple of elements as a set. For example, permutations((1,2)) should return the set {(1, 2), (2, 1)}

422102.2723990.qx3zqy7

**LAB
ACTIVITY**

7.11.1: Make Postage and Permutations

0 / 100



main.py

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022
[Load default template...](#)

```
1
2 def make_postage(money) :
3
4
5 def permutations(s) :
6
7
8 if __name__=='__main__' :
9     amount = input()
10    print("change for ", amount, " : ", make_postage(int(amount)))
11    print("permutations: ", permutations((1, 2)))
```

Develop mode**Submit mode**

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

Run program

Input (from above)

**main.py**
(Your program)

0

Program output displayed here

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

Coding trail of your work [What is this?](#)

History of your effort will appear here once you begin working on this zyLab.

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

©zyBooks 12/15/22 00:21 1361995
John Farrell
COLOSTATECS220SeaboltFall2022