Chapter 8.1: Sequences

- An indexed list $a_1, a_2, a_b, \ldots, a_m$ is a **sequence**.
  $a_n \in S$, where S is some set.
  m can be finite or infinite.

Chapter 8.1: Sequences

- An indexed list $a_1, a_2, a_b, \ldots, a_m$ is a **sequence**.
  $a_n \in S$, where S is some set.
  m can be finite or infinite.
- Index may start anywhere, but usually 0 (as in languages C, Java, Python) or one (as in Fortran, MATLAB, Julia)

Chapter 8.1: Sequences

- An indexed list $a_1, a_2, a_b, \ldots, a_m$ is a **sequence**.
  $a_n \in S$, where S is some set.
  m can be finite or infinite.
- Index may start anywhere, but usually 0 (as in languages C, Java, Python) or one (as in Fortran, MATLAB, Julia)
- Examples:
  $V_n = \#$ visitors to web site on day n of year, $1 \leq n \leq 365$
  $t_n =$ List of ordered tasks, $t_n =$ function.

Chapter 8.1: Sequences

- An indexed list $a_1, a_2, a_b, \ldots, a_m$ is a **sequence**.
  $a_n \in S$, where S is some set.
  m can be finite or infinite.
- Index may start anywhere, but usually 0 (as in languages C, Java, Python) or one (as in Fortran, MATLAB, Julia)
- Examples:
  $V_n = \#$ visitors to web site on day n of year, $1 \leq n \leq 365$
  $t_n =$ List of ordered tasks, $t_n =$ function.
- A sequence $\{a_n\}$ is:
  increasing if $a_n > a_{n-1}$ for all possible n
  decreasing if $a_n < a_{n-1}$ for all possible n.

Chapter 8.1: Sequences

- An indexed list $a_1, a_2, a_b, \ldots, a_m$ is a **sequence**.
  $a_n \in S$, where S is some set.
  m can be finite or infinite.
- Index may start anywhere, but usually 0 (as in languages C, Java, Python) or one (as in Fortran, MATLAB, Julia)
- Examples:
  $V_n = \#$ visitors to web site on day n of year, $1 \leq n \leq 365$
  $t_n =$ List of ordered tasks, $t_n =$ function.
- A sequence $\{a_n\}$ is:
  increasing if $a_n > a_{n-1}$ for all possible n
  decreasing if $a_n < a_{n-1}$ for all possible n.

Chapter 8.1: Arithmetic and geometric sequences

- In an **arithmetic sequence**, the difference of subsequent terms is the same.
  Example: $1, 4, 7, 10, 13, \ldots$ $(d = 3)$

Chapter 8.1: Arithmetic and geometric sequences

- In an **arithmetic sequence**, the difference of subsequent terms is the same.
  Example: $1, 4, 7, 10, 13, \ldots$ $(d = 3)$
  General formula: $a_n = a_0 + dn$, where a_0 is first term and d is difference $a_n - a_{n-1}$.
  For above example, $a_n = 1 + 3n$.
- In a **geometric sequence**, the ratio of subsequent terms is the same .
  Example: $1/3, -1/9, 1/27, -1/81, ...$

Chapter 8.1: Arithmetic and geometric sequences

- In an **arithmetic sequence**, the difference of subsequent terms is the same.
  Example: $1, 4, 7, 10, 13, \ldots$ $(d = 3)$
  General formula: $a_n = a_0 + dn$, where a_0 is first term and d is difference $a_n - a_{n-1}$.
  For above example, $a_n = 1 + 3n$.
- In a **geometric sequence**, the ratio of subsequent terms is the same .
  Example: $1/3, -1/9, 1/27, -1/81, ...$
  General formula: $a_n = a_0 r^n$, where a_0 is first term and d is ratio a_n/a_{n-1}.
- Re-indexing: replacing n by $n - n_0$ in a formula produces the same sequence, but with index starting at n_0 instead of zero.

Chapter 8.1: Arithmetic and geometric sequences

- In an **arithmetic sequence**, the difference of subsequent terms is the same.
  Example: $1, 4, 7, 10, 13, \ldots$ ($d = 3$)
  General formula: $a_n = a_0 + dn$, where a_0 is first term and d is difference $a_n - a_{n-1}$.
  For above example, $a_n = 1 + 3n$.
- In a **geometric sequence**, the ratio of subsequent terms is the same .
  Example: $1/3, -1/9, 1/27, -1/81, \ldots$
  General formula: $a_n = a_0 r^n$, where a_0 is first term and d is ratio a_n/a_{n-1}.
- Re-indexing: replacing n by $n - n_0$ in a formula produces the same sequence, but with index starting at n_0 instead of zero.
- Arithmetic sequences are increasing (decreasing) if $d > 0$ ($d < 0$). Geometric sequences are increasing if $r > 1$, decreasing if $0 < r < 1$, and neither if $r < 0$.
- Problem: Suppose the first Thursday in January is the 2nd. Write a formula for the sequence of all Thursdays in January.

Chapter 8.2 Recurrence relations

- One way to specify a sequence is by an explicit formula $a_n = f(n)$; another is by a **recurrence relation**, which is a formula of the form $a_n = f(a_{n-1})$ or more generally $a_n = f(a_{n-1}, a_{n-2}, a_{n-3}, \ldots)$.

Chapter 8.2 Recurrence relations

- One way to specify a sequence is by an explicit formula $a_n = f(n)$; another is by a **recurrence relation**, which is a formula of the form $a_n = f(a_{n-1})$ or more generally $a_n = f(a_{n-1}, a_{n-2}, a_{n-3}, \ldots)$.
- Also need to specify starting values , e.g. a_0 or sometimes more so that $f(a_{n-1}, a_{n-2}, a_{n-3}, \ldots)$ can be evaluated.

Chapter 8.2 Recurrence relations

- One way to specify a sequence is by an explicit formula $a_n = f(n)$; another is by a **recurrence relation**, which is a formula of the form $a_n = f(a_{n-1})$ or more generally $a_n = f(a_{n-1}, a_{n-2}, a_{n-3}, \ldots)$.
- Also need to specify starting values , e.g. a_0 or sometimes more so that $f(a_{n-1}, a_{n-2}, a_{n-3}, \ldots)$ can be evaluated.
- Example: An arithmetic sequence starting at a_0 and with difference d can be written in recurrence form as $a_n = a_{n-1} + d$.

Chapter 8.2 Recurrence relations

- One way to specify a sequence is by an explicit formula $a_n = f(n)$; another is by a **recurrence relation**, which is a formula of the form $a_n = f(a_{n-1})$ or more generally $a_n = f(a_{n-1}, a_{n-2}, a_{n-3}, \ldots)$.
- Also need to specify starting values , e.g. a_0 or sometimes more so that $f(a_{n-1}, a_{n-2}, a_{n-3}, \ldots)$ can be evaluated.
- Example: An arithmetic sequence starting at a_0 and with difference d can be written in recurrence form as $a_n = a_{n-1} + d$.
- Example: Suppose $a_n = na_{n-1}$ and $a_0 = 1$. Then $a_1 = 1$, $a_2 = 2$, $a_3 = 6$, $a_4 = 24$, etc. Thus $a_n = n!$.

Chapter 8.2 Recurrence relations

- One way to specify a sequence is by an explicit formula $a_n = f(n)$; another is by a **recurrence relation**, which is a formula of the form $a_n = f(a_{n-1})$ or more generally $a_n = f(a_{n-1}, a_{n-2}, a_{n-3}, \ldots)$.
- Also need to specify starting values , e.g. a_0 or sometimes more so that $f(a_{n-1}, a_{n-2}, a_{n-3}, \ldots)$ can be evaluated.
- Example: An arithmetic sequence starting at a_0 and with difference d can be written in recurrence form as $a_n = a_{n-1} + d$.
- Example: Suppose $a_n = na_{n-1}$ and $a_0 = 1$. Then $a_1 = 1$, $a_2 = 2$, $a_3 = 6$, $a_4 = 24$, etc. Thus $a_n = n!$.
- Fibonacci sequence: $a_n = a_{n-1} + a_{n-2}$, with $a_1 = a_2 = 1$.

Chapter 8.2 Recurrence relations

- One way to specify a sequence is by an explicit formula $a_n = f(n)$; another is by a **recurrence relation**, which is a formula of the form $a_n = f(a_{n-1})$ or more generally $a_n = f(a_{n-1}, a_{n-2}, a_{n-3}, \ldots)$.
- Also need to specify starting values , e.g. a_0 or sometimes more so that $f(a_{n-1}, a_{n-2}, a_{n-3}, \ldots)$ can be evaluated.
- Example: An arithmetic sequence starting at a_0 and with difference d can be written in recurrence form as $a_n = a_{n-1} + d$.
- Example: Suppose $a_n = na_{n-1}$ and $a_0 = 1$. Then $a_1 = 1$, $a_2 = 2$, $a_3 = 6$, $a_4 = 24$, etc. Thus $a_n = n!$.
- Fibonacci sequence: $a_n = a_{n-1} + a_{n-2}$, with $a_1 = a_2 = 1$. This produces sequence $1, 1, 2, 3, 5, 8, 13, 21, 34, \ldots$

Chapter 8.2 Recurrence relations

- One way to specify a sequence is by an explicit formula $a_n = f(n)$; another is by a **recurrence relation**, which is a formula of the form $a_n = f(a_{n-1})$ or more generally $a_n = f(a_{n-1}, a_{n-2}, a_{n-3}, \ldots)$.
- Also need to specify starting values , e.g. a_0 or sometimes more so that $f(a_{n-1}, a_{n-2}, a_{n-3}, \ldots)$ can be evaluated.
- Example: An arithmetic sequence starting at a_0 and with difference d can be written in recurrence form as $a_n = a_{n-1} + d$.
- Example: Suppose $a_n = na_{n-1}$ and $a_0 = 1$. Then $a_1 = 1$, $a_2 = 2$, $a_3 = 6$, $a_4 = 24$, etc. Thus $a_n = n!$.
- Fibonacci sequence: $a_n = a_{n-1} + a_{n-2}$, with $a_1 = a_2 = 1$. This produces sequence $1, 1, 2, 3, 5, 8, 13, 21, 34, \ldots$

Problems:

(1) Suppose $a_0 = 5$ and $a_n = -a_{n-1}$. What kind of sequence is generated? is there a non-recursive formula for a_n?

(2) Let $a_1 = 1$, and let $a_n = a_1 + a_2 + \ldots + \ldots + a_{n-1}$. Is there an obvious pattern?

Chapter 8.2 Enumerations using recurrence relations

Suppose you want to determine the number a_n of strings (of lowercase letters) of length n with non-adjacent letters.

Suppose you want to determine the number a_n of strings (of lowercase letters) of length n with non-adjacent letters.

Clearly, $a_1 = 26$.
For strings of length $n-1$, there are 25 ways of producing a string of length n, since the last two letters can't be the same.

Suppose you want to determine the number a_n of strings (of lowercase letters) of length n with non-adjacent letters.

Clearly, $a_1 = 26$.
For strings of length $n-1$, there are 25 ways of producing a string of length n, since the last two letters can't be the same.
Therefore $a_n = 25a_{n-1}$.

Suppose you want to determine the number a_n of strings (of lowercase letters) of length n with non-adjacent letters.

Clearly, $a_1 = 26$.
For strings of length $n-1$, there are 25 ways of producing a string of length n, since the last two letters can't be the same.
Therefore $a_n = 25a_{n-1}$.

This gives a geometric series. Problem: find non-recursive formula for a_n.

Suppose you want to determine the number a_n of strings (of lowercase letters) of length n with non-adjacent letters.

Clearly, $a_1 = 26$.
For strings of length $n-1$, there are 25 ways of producing a string of length n, since the last two letters can't be the same.
Therefore $a_n = 25a_{n-1}$.

This gives a geometric series. Problem: find non-recursive formula for a_n.
Answer: $a_n = 26(25)^{n-1}$.

Harder version: find the number a_n of strings of length n with non-adjacent vowels.

Harder version: find the number a_n of strings of length n with non-adjacent vowels.

As before, $a_1 = 26$. For a_2, there are two cases:
The first letter was a vowel, so the second must be a consonant.
This results in 5(21) possibilities.

Harder version: find the number a_n of strings of length n with non-adjacent vowels.

As before, $a_1 = 26$. For a_2, there are two cases:
The first letter was a vowel, so the second must be a consonant. This results in $5(21)$ possibilities.
Otherwise, the first letter is a consonant, and the second can be anything. This results in $21(26)$ possibilities. Thus $a_2 = 105 + 546 = 651$.

Harder version: find the number a_n of strings of length n with non-adjacent vowels.

As before, $a_1 = 26$. For a_2, there are two cases:
The first letter was a vowel, so the second must be a consonant. This results in $5(21)$ possibilities.
Otherwise, the first letter is a consonant, and the second can be anything. This results in $21(26)$ possibilities. Thus $a_2 = 105 + 546 = 651$.

In general, there are two cases for a string of length n:
(i) The last letter is a consonant. There are $21a_{n-1}$ ways of doing this.

Harder version: find the number a_n of strings of length n with non-adjacent vowels.

As before, $a_1 = 26$. For a_2, there are two cases:
The first letter was a vowel, so the second must be a consonant. This results in $5(21)$ possibilities.
Otherwise, the first letter is a consonant, and the second can be anything. This results in $21(26)$ possibilities. Thus $a_2 = 105 + 546 = 651$.

In general, there are two cases for a string of length n:
(i) The last letter is a consonant. There are $21a_{n-1}$ ways of doing this.
(ii) The last two letters are a consonant, and then a vowel. There are $21(5)a_{n-2}$ ways of doing this.

Harder version: find the number a_n of strings of length n with non-adjacent vowels.

As before, $a_1 = 26$. For a_2, there are two cases:
The first letter was a vowel, so the second must be a consonant. This results in 5(21) possibilities.
Otherwise, the first letter is a consonant, and the second can be anything. This results in 21(26) possibilities. Thus
$a_2 = 105 + 546 = 651$.

In general, there are two cases for a string of length n:
(i) The last letter is a consonant. There are $21a_{n-1}$ ways of doing this.
(ii) The last two letters are a consonant, and then a vowel. There are $21(5)a_{n-2}$ ways of doing this.
There are therefore

$$a_n = 21a_{n-1} + 105a_{n-2}$$

ways of constructing strings of length n.

Chapter 8.3: Summation

- Recall Sigma notation: $a_s + a_{s+1} + \ldots + a_t = \Sigma_{j=s}^{t} a_j$

Chapter 8.3: Summation

- Recall Sigma notation: $a_s + a_{s+1} + \ldots + a_t = \Sigma_{j=s}^{t} a_j$
- A "closed form" expression is one without an arbitrary number of terms, i.e. a formula rather than Sigma notation.

Chapter 8.3: Summation

- Recall Sigma notation: $a_s + a_{s+1} + \ldots + a_t = \Sigma_{j=s}^{t} a_j$
- A "closed form" expression is one without an arbitrary number of terms, i.e. a formula rather than Sigma notation.
- Examples
  (arithmetic series): $1 + 2 + 3 + \ldots + n = \Sigma_{j=1}^{n} j = n(n+1)/2$
  (geometric series): $\Sigma_{j=0}^{n} ar^j = a(1 - r^{n+1})/(1 - r)$

Chapter 8.3: Summation

- Recall Sigma notation: $a_s + a_{s+1} + \ldots + a_t = \Sigma_{j=s}^{t} a_j$
- A "closed form" expression is one without an arbitrary number of terms, i.e. a formula rather than Sigma notation.
- Examples
  (arithmetic series): $1 + 2 + 3 + \ldots + n = \Sigma_{j=1}^{n} j = n(n+1)/2$
  (geometric series): $\Sigma_{j=0}^{n} ar^j = a(1 - r^{n+1})/(1 - r)$
- Usual rules (Communative, associative, distributive) apply.
  For example, for the arithmetic sequence
  $a + (a + d) + (a + 2d) + \ldots + (a + (n-1)d) = \Sigma_{k=0}^{n-1}(a + kd)$
  $= \Sigma_{k=0}^{n-1} a + d\Sigma_{k=0}^{n-1} k = an + d\frac{n(n-1)}{2}$.

Chapter 8.3: Rewriting summations

- To write a sum with different bounds on the index, use the substitution $j = k + (j_0 - k_0)$, where j_0 and k_0 are lower bounds for indices j, k.

Chapter 8.3: Rewriting summations

- To write a sum with different bounds on the index, use the substitution $j = k + (j_0 - k_0)$, where j_0 and k_0 are lower bounds for indices j, k.
- The old index j must be replaced everywhere in the expression.

Chapter 8.3: Rewriting summations

- To write a sum with different bounds on the index, use the substitution $j = k + (j_0 - k_0)$, where j_0 and k_0 are lower bounds for indices j, k.
- The old index j must be replaced everywhere in the expression.
- For example, suppose we want the lower index in $\sum_{j=s}^{t} a_j$ to be zero.

Chapter 8.3: Rewriting summations

- To write a sum with different bounds on the index, use the substitution $j = k + (j_0 - k_0)$, where j_0 and k_0 are lower bounds for indices j, k.
- The old index j must be replaced everywhere in the expression.
- For example, suppose we want the lower index in $\sum_{j=s}^{t} a_j$ to be zero. Choose $j = k + s$, so that when $j = s$, $k = 0$. Replace j everywhere:

$$\sum_{k=0}^{t-s} a_{k+s}$$

Chapter 8.3: Rewriting summations

- To write a sum with different bounds on the index, use the substitution $j = k + (j_0 - k_0)$, where j_0 and k_0 are lower bounds for indices j, k.
- The old index j must be replaced everywhere in the expression.
- For example, suppose we want the lower index in $\Sigma_{j=s}^{t} a_j$ to be zero. Choose $j = k + s$, so that when $j = s$, $k = 0$. Replace j everywhere:

$$\Sigma_{k=0}^{t-s} a_{k+s}$$

To decompose sums, split Σ notation into two terms as

$$\Sigma_{j=s}^{t} a_j = \Sigma_{j=s}^{m} a_j + \Sigma_{j=m+1}^{t} a_j.$$

Chapter 8.3: Rewriting summations

- To write a sum with different bounds on the index, use the substitution $j = k + (j_0 - k_0)$, where j_0 and k_0 are lower bounds for indices j, k.
- The old index j must be replaced everywhere in the expression.
- For example, suppose we want the lower index in $\Sigma_{j=s}^{t} a_j$ to be zero. Choose $j = k + s$, so that when $j = s$, $k = 0$. Replace j everywhere:

$$\Sigma_{k=0}^{t-s} a_{k+s}$$

To decompose sums, split Σ notation into two terms as

$$\Sigma_{j=s}^{t} a_j = \Sigma_{j=s}^{m} a_j + \Sigma_{j=m+1}^{t} a_j.$$

In particular, it will be useful to split off the last term in a sum, e.g.

$$\Sigma_{j=1}^{n+1} f(j) = \Sigma_{j=1}^{n} f(j) + f(n+1)$$

Chapter 8.3: Summation, problems

(1) Write the sum of even numbers between 0 and 1000 in Σ notation.

(2) Rewrite the summation so that the lower bound on the index is one:

$$\Sigma_{j=5}^{13} \frac{j+4}{2^j}$$

Chapter 8.4: Proof by induction

- Suppose that we want to verify the truth of a sequence of propositions $S_1, S_2, \ldots$.

Chapter 8.4: Proof by induction

- Suppose that we want to verify the truth of a sequence of propositions $S_1, S_2, \ldots$.
- Consider a case where $S_n \rightarrow S_{n+1}$ for every value of n, i.e. $S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow \ldots$.

Chapter 8.4: Proof by induction

- Suppose that we want to verify the truth of a sequence of propositions $S_1, S_2, \ldots$.
- Consider a case where $S_n \rightarrow S_{n+1}$ for every value of n, i.e. $S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow \ldots$.
- If we verify that S_1 is true, then S_2 is also true by modus ponens.

Chapter 8.4: Proof by induction

- Suppose that we want to verify the truth of a sequence of propositions $S_1, S_2, \ldots$.
- Consider a case where $S_n \rightarrow S_{n+1}$ for every value of n, i.e. $S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow \ldots$.
- If we verify that S_1 is true, then S_2 is also true by modus ponens.
- Repeating the same argument, it follows that $S_2, S_3, \ldots$ all must be true.

Chapter 8.4: Proof by induction

- Suppose that we want to verify the truth of a sequence of propositions $S_1, S_2, \ldots$.
- Consider a case where $S_n \rightarrow S_{n+1}$ for every value of n, i.e. $S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow \ldots$.
- If we verify that S_1 is true, then S_2 is also true by modus ponens.
- Repeating the same argument, it follows that $S_2, S_3, \ldots$ all must be true.
- We must therefore verify two things:
  (1) That S_1 is true ("base case")
  (2) That $S_n \rightarrow S_{n+1}$ ("inductive step")
  This is known as the *method of induction*.

Chapter 8.4: Proof by induction, examples

Verify $\sum_{j=1}^{n} j^2 = \frac{n(n+1)(2n+1)}{6}$ for all $n = 1, 2, 3, \ldots$.

Chapter 8.4: Proof by induction, examples

Verify $\sum_{j=1}^{n} j^2 = \frac{n(n+1)(2n+1)}{6}$ for all $n = 1, 2, 3, \ldots$.

Proof: By induction on n.

Chapter 8.4: Proof by induction, examples

Verify $\Sigma_{j=1}^{n} j^2 = \frac{n(n+1)(2n+1)}{6}$ for all $n = 1, 2, 3, \ldots$.

Proof: By induction on n.
(Base case) For $n = 1$, we have $\Sigma_{j=1}^{1} j^2 = 1 = 1(1+1)(2(1)+1)/6$.

Chapter 8.4: Proof by induction, examples

Verify $\Sigma_{j=1}^{n} j^2 = \frac{n(n+1)(2n+1)}{6}$ for all $n = 1, 2, 3, \ldots$.

Proof: By induction on n.
(Base case) For $n = 1$, we have $\Sigma_{j=1}^{1} j^2 = 1 = 1(1+1)(2(1)+1)/6$.
(Inductive hypothesis) Suppose that the formula is true for $n = k$.
We will show it works for $n = k + 1$.

Chapter 8.4: Proof by induction, examples

Verify $\Sigma_{j=1}^{n} j^2 = \frac{n(n+1)(2n+1)}{6}$ for all $n = 1, 2, 3, \ldots$.

Proof: By induction on n.
(Base case) For $n = 1$, we have $\Sigma_{j=1}^{1} j^2 = 1 = 1(1+1)(2(1)+1)/6$.
(Inductive hypothesis) Suppose that the formula is true for $n = k$.
We will show it works for $n = k + 1$.
$\Sigma_{j=1}^{k+1} j^2 = \Sigma_{j=1}^{k} j^2 + (k+1)^2$

Chapter 8.4: Proof by induction, examples

Verify $\Sigma_{j=1}^{n} j^2 = \frac{n(n+1)(2n+1)}{6}$ for all $n = 1, 2, 3, \ldots$.

Proof: By induction on n.
(Base case) For $n = 1$, we have $\Sigma_{j=1}^{1} j^2 = 1 = 1(1+1)(2(1)+1)/6$.
(Inductive hypothesis) Suppose that the formula is true for $n = k$.
We will show it works for $n = k + 1$.
$\Sigma_{j=1}^{k+1} j^2 = \Sigma_{j=1}^{k} j^2 + (k+1)^2$
$= k(k+1)(2k+1)/6 + (k+1)^2$ (by the inductive hypothesis)

Chapter 8.4: Proof by induction, examples

Verify $\Sigma_{j=1}^{n} j^2 = \frac{n(n+1)(2n+1)}{6}$ for all $n = 1, 2, 3, \ldots$.

Proof: By induction on n.
(Base case) For $n = 1$, we have $\Sigma_{j=1}^{1} j^2 = 1 = 1(1+1)(2(1)+1)/6$.
(Inductive hypothesis) Suppose that the formula is true for $n = k$.
We will show it works for $n = k + 1$.
$\Sigma_{j=1}^{k+1} j^2 = \Sigma_{j=1}^{k} j^2 + (k+1)^2$
$= k(k+1)(2k+1)/6 + (k+1)^2$ (by the inductive hypothesis)
$= (k+1)(2k^2 + k + 6k + 6)/6 = (k+1)(k+2)(2(k+1)+1)/6$,
which is the summation formula for $n = k + 1$.

Chapter 8.4: Proof by induction, examples

Show that for integers $n \geq 3$, $3^n \geq n^3$.

Proof: By induction on n.

Chapter 8.4: Proof by induction, examples

Show that for integers $n \geq 3$, $3^n \geq n^3$.

Proof: By induction on n.
(Base case) For $n = 3$, we have $3^n = n^3$.

Chapter 8.4: Proof by induction, examples

Show that for integers $n \geq 3$, $3^n \geq n^3$.

Proof: By induction on n.
(Base case) For $n = 3$, we have $3^n = n^3$.

(Inductive hypothesis) Suppose that the inequality holds for $n = k$, i.e. $3^k \geq k^3$. We will show it holds for $n = k + 1$, that is, $3^{k+1} \geq (k+1)^3 = k^3 + 3k^2 + 3k = 1$.

Chapter 8.4: Proof by induction, examples

Show that for integers $n \geq 3$, $3^n \geq n^3$.

Proof: By induction on n.
(Base case) For $n = 3$, we have $3^n = n^3$.

(Inductive hypothesis) Suppose that the inequality holds for $n = k$, i.e. $3^k \geq k^3$. We will show it holds for $n = k + 1$, that is, $3^{k+1} \geq (k+1)^3 = k^3 + 3k^2 + 3k = 1$.

By the inductive hypothesis, $3^{k+1} = 3(3^k) \geq 3k^3$.

Chapter 8.4: Proof by induction, examples

Show that for integers $n \geq 3$, $3^n \geq n^3$.

Proof: By induction on n.
(Base case) For $n = 3$, we have $3^n = n^3$.

(Inductive hypothesis) Suppose that the inequality holds for $n = k$, i.e. $3^k \geq k^3$. We will show it holds for $n = k + 1$, that is, $3^{k+1} \geq (k+1)^3 = k^3 + 3k^2 + 3k = 1$.

By the inductive hypothesis, $3^{k+1} = 3(3^k) \geq 3k^3$.
$= k^3 + k^3 + k^3$

Chapter 8.4: Proof by induction, examples

Show that for integers $n \geq 3$, $3^n \geq n^3$.

Proof: By induction on n.
(Base case) For $n = 3$, we have $3^n = n^3$.

(Inductive hypothesis) Suppose that the inequality holds for $n = k$, i.e. $3^k \geq k^3$. We will show it holds for $n = k + 1$, that is, $3^{k+1} \geq (k+1)^3 = k^3 + 3k^2 + 3k = 1$.

By the inductive hypothesis, $3^{k+1} = 3(3^k) \geq 3k^3$.
$= k^3 + k^3 + k^3$
$\geq k^3 + 3k^2 + 9k$

Chapter 8.4: Proof by induction, examples

Show that for integers $n \geq 3$, $3^n \geq n^3$.

Proof: By induction on n.
(Base case) For $n = 3$, we have $3^n = n^3$.

(Inductive hypothesis) Suppose that the inequality holds for $n = k$, i.e. $3^k \geq k^3$. We will show it holds for $n = k + 1$, that is, $3^{k+1} \geq (k+1)^3 = k^3 + 3k^2 + 3k = 1$.

By the inductive hypothesis, $3^{k+1} = 3(3^k) \geq 3k^3$.
$= k^3 + k^3 + k^3$
$\geq k^3 + 3k^2 + 9k$
$\geq k^3 + 3k^2 + 3k + 1 = (k+1)^3$.

Chapter 8.4: Proof by induction, examples

Show that for integers $n \geq 3$, $3^n \geq n^3$.

Proof: By induction on n.
(Base case) For $n = 3$, we have $3^n = n^3$.

(Inductive hypothesis) Suppose that the inequality holds for $n = k$, i.e. $3^k \geq k^3$. We will show it holds for $n = k + 1$, that is, $3^{k+1} \geq (k+1)^3 = k^3 + 3k^2 + 3k = 1$.

By the inductive hypothesis, $3^{k+1} = 3(3^k) \geq 3k^3$.
$= k^3 + k^3 + k^3$
$\geq k^3 + 3k^2 + 9k$
$\geq k^3 + 3k^2 + 3k + 1 = (k+1)^3$.

Chapter 8.4: Proof by induction, examples

Recall: an integer n is even if it can be written as $2k$ and odd if it can be written as $2k + 1$ for some $k \in \mathbb{Z}$.

Chapter 8.4: Proof by induction, examples

Recall: an integer n is even if it can be written as $2k$ and odd if it can be written as $2k + 1$ for some $k \in \mathbb{Z}$.

Theorem: every positive integer n is even or odd.

Chapter 8.4: Proof by induction, examples

Recall: an integer n is even if it can be written as $2k$ and odd if it can be written as $2k + 1$ for some $k \in \mathbb{Z}$.

Theorem: every positive integer n is even or odd.

Proof: By induction on n.
(Base case) For $n = 1$, we have $n = 2(0) + 1$, so n is odd.

Chapter 8.4: Proof by induction, examples

Recall: an integer n is even if it can be written as $2k$ and odd if it can be written as $2k + 1$ for some $k \in \mathbb{Z}$.

Theorem: every positive integer n is even or odd.

Proof: By induction on n.
(Base case) For $n = 1$, we have $n = 2(0) + 1$, so n is odd.

(Inductive hypothesis) Suppose that a positive integer n is even or odd. We will show that that $n + 1$ is even or odd.

Chapter 8.4: Proof by induction, examples

Recall: an integer n is even if it can be written as $2k$ and odd if it can be written as $2k+1$ for some $k \in \mathbb{Z}$.

Theorem: every positive integer n is even or odd.

Proof: By induction on n.
(Base case) For $n = 1$, we have $n = 2(0) + 1$, so n is odd.

(Inductive hypothesis) Suppose that a positive integer n is even or odd. We will show that that $n+1$ is even or odd.
Case 1: n is even. Then $n = 2k$, and $n + 1 = 2k + 1$, so $n + 1$ must be odd.

Chapter 8.4: Proof by induction, examples

Recall: an integer n is even if it can be written as $2k$ and odd if it can be written as $2k+1$ for some $k \in \mathbb{Z}$.

Theorem: every positive integer n is even or odd.

Proof: By induction on n.
(Base case) For $n = 1$, we have $n = 2(0) + 1$, so n is odd.

(Inductive hypothesis) Suppose that a positive integer n is even or odd. We will show that that $n+1$ is even or odd.
Case 1: n is even. Then $n = 2k$, and $n + 1 = 2k + 1$, so $n + 1$ must be odd.
Case 2: n is odd. Then $n = 2k + 1$, so that $n + 1 = 2(k + 1)$, thus $n + 1$ is even.

Chapter 8.4: Proof by induction, problems

(1) Show $\Sigma_{j=0}^{n} 2^j = 2^{n+1} - 1$ for integers $n = 0, 1, 2, \ldots$.

(2) Show that for integers $n \geq 4$, $n! > 2^n$.
Hint: notice $(k+1)! = (k+1)k!$, and $k + 1 \geq 2$ when $k \geq 4$.

Chapter 8.5 More proof by induction

- We can use proof by induction on a wide variety of problems involving properties of numbers, logical statements, and sets.

Chapter 8.5 More proof by induction

- We can use proof by induction on a wide variety of problems involving properties of numbers, logical statements, and sets.
- This idea can be applied whenever there is a indexed list of propositions $S_1, S_2, \ldots$ which must be proved. The index is not always obvious.

Chapter 8.5 More proof by induction

- We can use proof by induction on a wide variety of problems involving properties of numbers, logical statements, and sets.
- This idea can be applied whenever there is a indexed list of propositions $S_1, S_2, \ldots$ which must be proved. The index is not always obvious.
- All proofs by induction have the same structure:
  By induction on (index)
  (Base case) ...
  (Inductive hypothesis)...
  (Body of proof) At some point the inductive hypothesis MUST be invoked.

Chapter 8.5 More induction, examples

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

Proof: by induction on n.
(Base case) For $n = 1$, $3^{2(1)} - 1 = 8$, which is divisible by 4.

Chapter 8.5 More induction, examples

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

Proof: by induction on n.
(Base case) For $n = 1$, $3^{2(1)} - 1 = 8$, which is divisible by 4.

(Inductive hypothesis) Suppose for some positive integer n, $3^{2n} - 1$ is divisible by 4. We will show that $3^{2(n+1)} - 1$ is divisible by 4.

Chapter 8.5 More induction, examples

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

Proof: by induction on n.
(Base case) For $n = 1$, $3^{2(1)} - 1 = 8$, which is divisible by 4.

(Inductive hypothesis) Suppose for some positive integer n, $3^{2n} - 1$ is divisible by 4. We will show that $3^{2(n+1)} - 1$ is divisible by 4.

We can write $3^{2(n+1)} - 1 = 9 \cdot 3^{2n} - 1$. By the inductive hypothesis, $3^{2n} - 1 = 4m$ for some integer m, or $3^{2n} = 4m + 1$.

Chapter 8.5 More induction, examples

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

Proof: by induction on n.
(Base case) For $n = 1$, $3^{2(1)} - 1 = 8$, which is divisible by 4.

(Inductive hypothesis) Suppose for some positive integer n, $3^{2n} - 1$ is divisible by 4. We will show that $3^{2(n+1)} - 1$ is divisible by 4.

We can write $3^{2(n+1)} - 1 = 9 \cdot 3^{2n} - 1$. By the inductive hypothesis, $3^{2n} - 1 = 4m$ for some integer m, or $3^{2n} = 4m + 1$.
It follows that
$3^{2(n+1)} - 1 = 9(4m + 1) - 1 = 36m + 8 = 4(9m + 2)$,
which is divisible by 4 .

Chapter 8.5 More induction, examples

Theorem: Suppose that S is a finite set with cardinality n. Then S has 2^n subsets.

Proof: By induction on n.
(Base case) if $n = 0$, there is one $(= 2^0)$ subset, the empty set.

Chapter 8.5 More induction, examples

Theorem: Suppose that S is a finite set with cardinality n. Then S has 2^n subsets.

Proof: By induction on n.
(Base case) if $n = 0$, there is one $(= 2^0)$ subset, the empty set.
(Inductive hypothesis) Suppose that any set with $|S| = n$ has 2^n subsets. Then a set S' of cardinality $n + 1$ has 2^{n+1} subsets.

Chapter 8.5 More induction, examples

Theorem: Suppose that S is a finite set with cardinality n. Then S has 2^n subsets.

Proof: By induction on n.
(Base case) if $n = 0$, there is one $(= 2^0)$ subset, the empty set.
(Inductive hypothesis) Suppose that any set with $|S| = n$ has 2^n subsets. Then a set S' of cardinality $n + 1$ has 2^{n+1} subsets.
Choose any element $a \in S'$, and write S' as the disjoint union of a set S and $\{a\}$. By the inductive hypothesis, the set S has 2^n subsets. For any subset A' of S' either:
(i) A' contains element a. Then $A' = A \cup \{a\}$, where $A \subset S$.
There are 2^n such sets.

Chapter 8.5 More induction, examples

Theorem: Suppose that S is a finite set with cardinality n. Then S has 2^n subsets.

Proof: By induction on n.
(Base case) if $n = 0$, there is one $(= 2^0)$ subset, the empty set.
(Inductive hypothesis) Suppose that any set with $|S| = n$ has 2^n subsets. Then a set S' of cardinality $n + 1$ has 2^{n+1} subsets.
Choose any element $a \in S'$, and write S' as the disjoint union of a set S and $\{a\}$. By the inductive hypothesis, the set S has 2^n subsets. For any subset A' of S' either:
(i) A' contains element a. Then $A' = A \cup \{a\}$, where $A \subset S$.
There are 2^n such sets.
(ii) A' does not contain element a. Then A' must be a subset of S, and there are 2^n such sets.
In total, there must be $2^n + 2^n = 2^{n+1}$ subsets.

Chapter 8.5 More induction, examples

Consider the recursion relation:

$$a_1 = 6, \quad a_n = 2a_{n-1} + 2n, \quad n \geq 2$$

.

Chapter 8.5 More induction, examples

Consider the recursion relation:

$$a_1 = 6, \quad a_n = 2a_{n-1} + 2n, \quad n \geq 2$$

.

Theorem: for any positive integer n, $a_n = 6 \cdot 2^n - 2n - 4 = f(n)$.

Chapter 8.5 More induction, examples

Consider the recursion relation:

$$a_1 = 6, \quad a_n = 2a_{n-1} + 2n, \quad n \geq 2$$

.

Theorem: for any positive integer n, $a_n = 6 \cdot 2^n - 2n - 4 = f(n)$.

Proof: By induction on n.
(Base case) For $n = 1$, $f(1) = 6 \cdot 2^1 - 2(1) - 4 = 6$.

Chapter 8.5 More induction, examples

Consider the recursion relation:

$$a_1 = 6, \quad a_n = 2a_{n-1} + 2n, \quad n \geq 2$$

.

Theorem: for any positive integer n, $a_n = 6 \cdot 2^n - 2n - 4 = f(n)$.

Proof: By induction on n.
(Base case) For $n = 1$, $f(1) = 6 \cdot 2^1 - 2(1) - 4 = 6$.
(Inductive hypothesis) Suppose that $a_n = f(n)$ for some $n \geq 2$.
We will show that $a_{n+1} = f(n+1)$.

Chapter 8.5 More induction, examples

Consider the recursion relation:

$$a_1 = 6, \quad a_n = 2a_{n-1} + 2n, \quad n \geq 2$$

.

Theorem: for any positive integer n, $a_n = 6 \cdot 2^n - 2n - 4 = f(n)$.

Proof: By induction on n.
(Base case) For $n = 1$, $f(1) = 6 \cdot 2^1 - 2(1) - 4 = 6$.
(Inductive hypothesis) Suppose that $a_n = f(n)$ for some $n \geq 2$.
We will show that $a_{n+1} = f(n+1)$. By the recursive definition,
$a_{n+1} = 2a_n + 2(n+1)$

Chapter 8.5 More induction, examples

Consider the recursion relation:

$$a_1 = 6, \quad a_n = 2a_{n-1} + 2n, \quad n \geq 2$$

.

Theorem: for any positive integer n, $a_n = 6 \cdot 2^n - 2n - 4 = f(n)$.

Proof: By induction on n.
(Base case) For $n = 1$, $f(1) = 6 \cdot 2^1 - 2(1) - 4 = 6$.
(Inductive hypothesis) Suppose that $a_n = f(n)$ for some $n \geq 2$.
We will show that $a_{n+1} = f(n+1)$. By the recursive definition,
$a_{n+1} = 2a_n + 2(n+1)$
$= 2(6 \cdot 2^n - 2n - 4) + 2(n+1)$ (using the inductive hypothesis)

Chapter 8.5 More induction, examples

Consider the recursion relation:

$$a_1 = 6, \quad a_n = 2a_{n-1} + 2n, \quad n \geq 2$$

.

Theorem: for any positive integer n, $a_n = 6 \cdot 2^n - 2n - 4 = f(n)$.

Proof: By induction on n.
(Base case) For $n = 1$, $f(1) = 6 \cdot 2^1 - 2(1) - 4 = 6$.
(Inductive hypothesis) Suppose that $a_n = f(n)$ for some $n \geq 2$.
We will show that $a_{n+1} = f(n+1)$. By the recursive definition,
$a_{n+1} = 2a_n + 2(n+1)$
$= 2(6 \cdot 2^n - 2n - 4) + 2(n+1)$ (using the inductive hypothesis)
$= 12 \cdot 2^n - 2n - 6 = 6 \cdot 2^{n+1} - 2(n+1) - 4 = f(n+1)$.

Chapter 8.5 More induction, examples

Boolean distributive law for an arbitrary number of terms:

$$y(x_1 + x_2 + \ldots + x_n) = \sum_{k=1}^{n} yx_k$$

Chapter 8.5 More induction, examples

Boolean distributive law for an arbitrary number of terms:

$$y(x_1 + x_2 + \ldots + x_n) = \sum_{k=1}^{n} yx_k$$

Proof: By induction on n.
(Base case) For $n = 2$, $y(x_1 + x_2) = yx_1 + yx_2$ using the ordinary two-term distributive law.

Chapter 8.5 More induction, examples

Boolean distributive law for an arbitrary number of terms:

$$y(x_1 + x_2 + \ldots + x_n) = \sum_{k=1}^{n} yx_k$$

Proof: By induction on n.
(Base case) For $n = 2$, $y(x_1 + x_2) = yx_1 + yx_2$ using the ordinary two-term distributive law.

(Inductive hypothesis) Suppose that
$y(x_1 + x_2 + \ldots + x_n) = \sum_{k=1}^{n} yx_k$ for some n. We will show this works for $n + 1$ terms.

Chapter 8.5 More induction, examples

Boolean distributive law for an arbitrary number of terms:

$$y(x_1 + x_2 + \ldots + x_n) = \sum_{k=1}^{n} yx_k$$

Proof: By induction on n.
(Base case) For $n = 2$, $y(x_1 + x_2) = yx_1 + yx_2$ using the ordinary two-term distributive law.

(Inductive hypothesis) Suppose that
$y(x_1 + x_2 + \ldots + x_n) = \sum_{k=1}^{n} yx_k$ for some n. We will show this works for $n + 1$ terms.

We can write

$$\sum_{k=1}^{n+1} yx_k = \sum_{k=1}^{n} yx_k + yx_{n+1}.$$

Chapter 8.5 More induction, examples

Boolean distributive law for an arbitrary number of terms:

$$y(x_1 + x_2 + \ldots + x_n) = \sum_{k=1}^{n} yx_k$$

Proof: By induction on n.
(Base case) For $n = 2$, $y(x_1 + x_2) = yx_1 + yx_2$ using the ordinary two-term distributive law.

(Inductive hypothesis) Suppose that $y(x_1 + x_2 + \ldots + x_n) = \sum_{k=1}^{n} yx_k$ for some n. We will show this works for $n + 1$ terms.

We can write

$$\sum_{k=1}^{n+1} yx_k = \sum_{k=1}^{n} yx_k + yx_{n+1}.$$

By the inductive hypothesis and the two-term distributive law,

$$\sum_{k=1}^{n} yx_k + yx_{n+1} = y(x_1 + x_2 + \ldots + x_n) + yx_{n+1} = y(x_1 + x_2 + \ldots + x_{n+1}).$$

Chapter 8.5 More induction, problems

(1) Prove: If $x > 1$ then $x^n > 1$ for all positive integers n.

(2) For the recursion relation $a_n = a_{n-1} + n + 1, a_0 = 0$, show that $a_n = n(n+3)/2$.

(3) Consider the set $B = \{0, 1\}$. Then for any positive integer n, $B^n = B \times B \times \ldots \times B$ (n times) has 2^n elements.
Hint: for elements in B^{n+1}, consider two cases where the last component is one or zero.

Chapter 8.6: Strong induction

- There are two logical forms for induction: weak and strong.

Chapter 8.6: Strong induction

- There are two logical forms for induction: weak and strong.
- Up to this point, we have used the weak form

$$S_1 \to S_2 \to S_3 \to \dots.$$

This requires us to verify each conditional $S_n \to S_{n+1}$.

Chapter 8.6: Strong induction

- There are two logical forms for induction: weak and strong.
- Up to this point, we have used the weak form

$$S_1 \to S_2 \to S_3 \to \dots.$$

  This requires us to verify each conditional $S_n \to S_{n+1}$.
- Strong induction by contrast has the form

$$S_1 \to S_2, \quad (S_1 \wedge S_2) \to S_3, \quad (S_1 \wedge S_2 \wedge S_3) \to S_4, \dots$$

Chapter 8.6: Strong induction

- There are two logical forms for induction: weak and strong.
- Up to this point, we have used the weak form

$$S_1 \to S_2 \to S_3 \to \dots.$$

  This requires us to verify each conditional $S_n \to S_{n+1}$.
- Strong induction by contrast has the form

$$S_1 \to S_2, \quad (S_1 \wedge S_2) \to S_3, \quad (S_1 \wedge S_2 \wedge S_3) \to S_4, \dots$$

- Therefore, to prove that S_{n+1} is true, need to assume (and use the fact) that ALL $S_1, S_2, \dots, S_n$ are true.

Chapter 8.6: Strong induction,example

Theorem: Every integer $n \geq 2$ can be written as product of primes.

Proof: By induction on n.
(Base case) For $n = 2$, factorization is immediate.

Chapter 8.6: Strong induction,example

Theorem: Every integer $n \geq 2$ can be written as product of primes.

Proof: By induction on n.
(Base case) For $n = 2$, factorization is immediate.
(Inductive hypothesis) Suppose that **all** integers ≥ 2 and $\leq n$ can be written as the product of primes.

Chapter 8.6: Strong induction,example

Theorem: Every integer $n \geq 2$ can be written as product of primes.

Proof: By induction on n.
(Base case) For $n = 2$, factorization is immediate.
(Inductive hypothesis) Suppose that **all** integers ≥ 2 and $\leq n$ can be written as the product of primes.
For the integer $n + 1$, there are two cases:

Chapter 8.6: Strong induction,example

Theorem: Every integer $n \geq 2$ can be written as product of primes.

Proof: By induction on n.
(Base case) For $n = 2$, factorization is immediate.
(Inductive hypothesis) Suppose that **all** integers ≥ 2 and $\leq n$ can be written as the product of primes.
For the integer $n + 1$, there are two cases:
(1) $n + 1$ is prime, and the prime factorization is immediate.

Chapter 8.6: Strong induction,example

Theorem: Every integer $n \geq 2$ can be written as product of primes.

Proof: By induction on n.
(Base case) For $n = 2$, factorization is immediate.
(Inductive hypothesis) Suppose that **all** integers ≥ 2 and $\leq n$ can be written as the product of primes.
For the integer $n + 1$, there are two cases:
(1) $n + 1$ is prime, and the prime factorization is immediate.
(2) $n + 1$ is composite. Then $n + 1$ must have factors p and q, each less than $n + 1$. By the inductive hypothesis, both p and q have a prime factorization:

$$p = p_1 p_2 \cdots p_k, \quad q = q_1 q_2 \cdots q_k$$

Chapter 8.6: Strong induction,example

Theorem: Every integer $n \geq 2$ can be written as product of primes.

Proof: By induction on n.
(Base case) For $n = 2$, factorization is immediate.
(Inductive hypothesis) Suppose that **all** integers ≥ 2 and $\leq n$ can be written as the product of primes.
For the integer $n + 1$, there are two cases:
(1) $n + 1$ is prime, and the prime factorization is immediate.
(2) $n + 1$ is composite. Then $n + 1$ must have factors p and q, each less than $n + 1$. By the inductive hypothesis, both p and q have a prime factorization:

$$p = p_1 p_2 \cdots p_k, \quad q = q_1 q_2 \cdots q_k$$

Therefore $n + 1 = p_1 p_2 \cdots p_k q_1 q_2 \cdots q_k$, which is a prime factorization.

Chapter 8.6: Strong induction, Multiple base cases

Recall the Fibonacci sequence is given by the recursion relation $F_{n+2} = F_{n+1} + F_n, \quad F_1 = F_2 = 1.$

Chapter 8.6: Strong induction, Multiple base cases

Recall the Fibonacci sequence is given by the recursion relation $F_{n+2} = F_{n+1} + F_n, \quad F_1 = F_2 = 1.$

Theorem: $F_n = [(\frac{1+\sqrt{5}}{2})^n - (\frac{1-\sqrt{5}}{2})^n]/\sqrt{5}$

Proof: By induction on n.

(Base cases) For $n = 1, 2$, the formula can be checked directly.

Chapter 8.6: Strong induction, Multiple base cases

Recall the Fibonacci sequence is given by the recursion relation $F_{n+2} = F_{n+1} + F_n, \quad F_1 = F_2 = 1.$

Theorem: $F_n = [(\frac{1+\sqrt{5}}{2})^n - (\frac{1-\sqrt{5}}{2})^n]/\sqrt{5}$

Proof: By induction on n.

(Base cases) For $n = 1, 2$, the formula can be checked directly.

(Inductive hypothesis) Suppose ok for **all** integers $\leq n$.

Chapter 8.6: Strong induction, Multiple base cases

Recall the Fibonacci sequence is given by the recursion relation
$F_{n+2} = F_{n+1} + F_n, \quad F_1 = F_2 = 1.$
Theorem: $F_n = [(\frac{1+\sqrt{5}}{2})^n - (\frac{1-\sqrt{5}}{2})^n]/\sqrt{5}$
Proof: By induction on n.
(Base cases) For $n = 1, 2$, the formula can be checked directly.
(Inductive hypothesis) Suppose ok for **all** integers $\leq n$.
Using the recursion relation and the inductive hypothesis,
$F_{n+1} = F_n + F_{n-1} =$
$\frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right] + \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^{n-1} - \left(\frac{1-\sqrt{5}}{2}\right)^{n-1}\right].$
Rearranging,
$\frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n + \left(\frac{1+\sqrt{5}}{2}\right)^{n-1}\right] - \frac{1}{\sqrt{5}}\left[\left(\frac{1-\sqrt{5}}{2}\right)^n + \left(\frac{1-\sqrt{5}}{2}\right)^{n-1}\right] =$
$\frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^{n-1}(1 + \frac{1+\sqrt{5}}{2}) - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^{n-1}(1 + \frac{1-\sqrt{5}}{2}) =$
$\frac{1}{\sqrt{5}}(\frac{1+\sqrt{5}}{2})^{n+1} - \frac{1}{\sqrt{5}}(\frac{1-\sqrt{5}}{2})^{n+1}$, using fact $(\frac{1\pm\sqrt{5}}{2})^2 = \frac{3\pm\sqrt{5}}{2}$.

Chapter 8.6: Strong induction, example

Theorem (binary representation of integers). For a positive integer n, $n = 2^{b_0} + 2^{b_1} + \ldots + 2^{b_j}$, where $0 \leq b_0 < b_1 < \ldots$.

Chapter 8.6: Strong induction, example

Theorem (binary representation of integers). For a positive integer n, $n = 2^{b_0} + 2^{b_1} + \ldots + 2^{b_j}$, where $0 \leq b_0 < b_1 < \ldots$.

Proof: By induction on n.

(Base case) The integer $n = 1 = 2^0$.

Chapter 8.6: Strong induction, example

Theorem (binary representation of integers). For a positive integer n, $n = 2^{b_0} + 2^{b_1} + \ldots + 2^{b_j}$, where $0 \leq b_0 < b_1 < \ldots$.

Proof: By induction on n.

(Base case) The integer $n = 1 = 2^0$.

(Inductive hypothesis) Suppose that all integers $\leq n$ can be written as above.

Chapter 8.6: Strong induction, example

Theorem (binary representation of integers). For a positive integer n, $n = 2^{b_0} + 2^{b_1} + \ldots + 2^{b_j}$, where $0 \leq b_0 < b_1 < \ldots$.

Proof: By induction on n.

(Base case) The integer $n = 1 = 2^0$.

(Inductive hypothesis) Suppose that all integers $\leq n$ can be written as above.

For integer $n + 1$ either:

(1) $n + 1 = 2^b$, and we are done.

Chapter 8.6: Strong induction, example

Theorem (binary representation of integers). For a positive integer n, $n = 2^{b_0} + 2^{b_1} + \ldots + 2^{b_j}$, where $0 \leq b_0 < b_1 < \ldots$.

Proof: By induction on n.

(Base case) The integer $n = 1 = 2^0$.

(Inductive hypothesis) Suppose that all integers $\leq n$ can be written as above.

For integer $n+1$ either:

(1) $n+1 = 2^b$, and we are done.

(2) There exists a non-negative integer b so that $2^b \leq n+1 \leq 2^{b+1}$. Observe that $n+1-2^b \leq 2^{b+1} - 2^b = 2^b \leq n$. Then by the inductive hypothesis, $n+1-2^b = 2^{b_0} + 2^{b_1} + \ldots + 2^{b_j}$ with $b_j < b$, and consequently $n+1 = 2^{b_0} + 2^{b_1} + \ldots + 2^{b_j} + 2^b$.

Chapter 8.6: Strong induction, problem

Consider the recursion relation

$$a_n = a_{n-1} + a_{n-2} + a_{n-3}, \quad a_1 = a_2 = a_3 = 1.$$

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

Chapter 8.6: Strong induction, a final example

Consider a chocolate bar, composed of $n \times m$ small squares. How many vertical or horizontal breaks are needed to subdivide the bar into squares?

Chapter 8.6: Strong induction, a final example

Consider a chocolate bar, composed of $n \times m$ small squares. How many vertical or horizontal breaks are needed to subdivide the bar into squares?

Theorem: The number of breaks needed is $nm - 1 = P(n, m)$.

Proof: By induction on nm.
(Base cases) For $nm = 1$, no breaks are needed.

Chapter 8.6: Strong induction, a final example

Consider a chocolate bar, composed of $n \times m$ small squares. How many vertical or horizontal breaks are needed to subdivide the bar into squares?

Theorem: The number of breaks needed is $nm - 1 = P(n, m)$.

Proof: By induction on nm.
(Base cases) For $nm = 1$, no breaks are needed.
(Inductive hypothesis) Suppose that for any bar with $nm \leq k$ squares, the number of breaks needed is $P(n, m)$.

Chapter 8.6: Strong induction, a final example

Consider a chocolate bar, composed of $n \times m$ small squares. How many vertical or horizontal breaks are needed to subdivide the bar into squares?

Theorem: The number of breaks needed is $nm - 1 = P(n, m)$.

Proof: By induction on nm.
(Base cases) For $nm = 1$, no breaks are needed.
(Inductive hypothesis) Suppose that for any bar with $nm \leq k$ squares, the number of breaks needed is $P(n, m)$.
Consider a bar with $k + 1 = nm$ squares. Without loss of generality, the first break is along the rows, so that the bar is divided into two bars with $n_1 \times m$ and $n_2 \times m$ squares, where $n_1 + n_2 = n$.

Chapter 8.6: Strong induction, a final example

Consider a chocolate bar, composed of $n \times m$ small squares. How many vertical or horizontal breaks are needed to subdivide the bar into squares?

Theorem: The number of breaks needed is $nm - 1 = P(n, m)$.

Proof: By induction on nm.
(Base cases) For $nm = 1$, no breaks are needed.
(Inductive hypothesis) Suppose that for any bar with $nm \leq k$ squares, the number of breaks needed is $P(n, m)$.
Consider a bar with $k + 1 = nm$ squares. Without loss of generality, the first break is along the rows, so that the bar is divided into two bars with $n_1 \times m$ and $n_2 \times m$ squares, where $n_1 + n_2 = n$.
By the inductive hypothesis, the total number of breaks needed is

$$P(n_1, m) + P(n_2, m) + 1 = (n_1 m - 1) + (n_2 m - 1) + 1 = nm - 1.$$

Chapter 8.8: Recursive definitions

- A function which is defined in terms of itself is called **recursive**.

Chapter 8.8: Recursive definitions

- A function which is defined in terms of itself is called **recursive**.
- A recursion relation $a_{n+1} = f(a_n)$ is an example of this of this: $a_{n+1} = f(f(f(\ldots(a_0)))$.

Chapter 8.8: Recursive definitions

- A function which is defined in terms of itself is called **recursive**.
- A recursion relation $a_{n+1} = f(a_n)$ is an example of this of this: $a_{n+1} = f(f(f(\ldots(a_0)))$.
- Notice this process must eventually terminate once a known value is encountered.

Chapter 8.8: Recursive definitions

- A function which is defined in terms of itself is called **recursive**.
- A recursion relation $a_{n+1} = f(a_n)$ is an example of this of this: $a_{n+1} = f(f(f(\ldots(a_0)))$.
- Notice this process must eventually terminate once a known value is encountered.

Numerical examples:
(1) Sums can be computed recursively, for example:

$$f(n) = \sum_{j=1}^{n} e^{-j}, \quad \text{or,} \quad f(n) = f(n-1) + e^{-n^2},\ f(0) = 0.$$

Chapter 8.8: Recursive definitions

- A function which is defined in terms of itself is called **recursive**.
- A recursion relation $a_{n+1} = f(a_n)$ is an example of this of this: $a_{n+1} = f(f(f(\ldots(a_0)))$.
- Notice this process must eventually terminate once a known value is encountered.

Numerical examples:
(1) Sums can be computed recursively, for example:

$$f(n) = \sum_{j=1}^{n} e^{-j}, \quad \text{or,} \quad f(n) = f(n-1) + e^{-n^2},\ f(0) = 0.$$

(2) Want a function of a positive integer x that gives 1 if it is even, and 0 if it is odd:

$$f(x) = 1 - f(x-1), \quad f(0) = 1.$$

Chapter 8.8: Recursive definitions of sets

- Complicated sets cannot be easily specified by direct conditions (given for example, in set-builder notation).

Chapter 8.8: Recursive definitions of sets

- Complicated sets cannot be easily specified by direct conditions (given for example, in set-builder notation).
- Recursive set definitions have ingredients:
  (1) Base case(s), which specify the simplest elements
  (2) Recursion rules, which specify how new elements are built from simpler ones
  (3) Exclusion rule(s), specifying which elements are excluded.

Example: The set S is prime numbers.

Chapter 8.8: Recursive definitions of sets

- Complicated sets cannot be easily specified by direct conditions (given for example, in set-builder notation).
- Recursive set definitions have ingredients:
  (1) Base case(s), which specify the simplest elements
  (2) Recursion rules, which specify how new elements are built from simpler ones
  (3) Exclusion rule(s), specifying which elements are excluded.

Example: The set S is prime numbers.
Base case: $2 \in S$.

Chapter 8.8: Recursive definitions of sets

- Complicated sets cannot be easily specified by direct conditions (given for example, in set-builder notation).
- Recursive set definitions have ingredients:
  (1) Base case(s), which specify the simplest elements
  (2) Recursion rules, which specify how new elements are built from simpler ones
  (3) Exclusion rule(s), specifying which elements are excluded.

Example: The set S is prime numbers.
Base case: $2 \in S$.
Recursion rule: $n \in S$ if $n \in \mathbb{Z}^+$ and n is not divisible by elements in S smaller than n.

Chapter 8.8: Recursive definitions, strings

Recall $B = \{0, 1\}$ and B^n is a string of length n.

Chapter 8.8: Recursive definitions, strings

Recall $B = \{0, 1\}$ and B^n is a string of length n.
Suppose we want strings of all lengths. We could write:

$$B^* = \{\lambda\} \cup B^1 \cup B^2 \ldots$$

Chapter 8.8: Recursive definitions, strings

Recall $B = \{0, 1\}$ and B^n is a string of length n.
Suppose we want strings of all lengths. We could write:

$$B^* = \{\lambda\} \cup B^1 \cup B^2 \ldots$$

Alternatively, B^* can be defined recursively:
Base: $\lambda \in B^*$
Recursion rule: If $x \in B^*$ then $x0 \in B^*$ and $x1 \in B^*$.

Chapter 8.8: Recursive definitions, strings

Recall $B = \{0, 1\}$ and B^n is a string of length n.
Suppose we want strings of all lengths. We could write:

$$B^* = \{\lambda\} \cup B^1 \cup B^2 \ldots$$

Alternatively, B^* can be defined recursively:
Base: $\lambda \in B^*$
Recursion rule: If $x \in B^*$ then $x0 \in B^*$ and $x1 \in B^*$.

Example: Set S of binary strings with an even number of ones.

Chapter 8.8: Recursive definitions, strings

Recall $B = \{0, 1\}$ and B^n is a string of length n.
Suppose we want strings of all lengths. We could write:

$$B^* = \{\lambda\} \cup B^1 \cup B^2 \ldots$$

Alternatively, B^* can be defined recursively:
Base: $\lambda \in B^*$
Recursion rule: If $x \in B^*$ then $x0 \in B^*$ and $x1 \in B^*$.

Example: Set S of binary strings with an even number of ones.
Base: $\lambda \in S$

Chapter 8.8: Recursive definitions, strings

Recall $B = \{0, 1\}$ and B^n is a string of length n.
Suppose we want strings of all lengths. We could write:

$$B^* = \{\lambda\} \cup B^1 \cup B^2 \ldots$$

Alternatively, B^* can be defined recursively:
Base: $\lambda \in B^*$
Recursion rule: If $x \in B^*$ then $x0 \in B^*$ and $x1 \in B^*$.

Example: Set S of binary strings with an even number of ones.
Base: $\lambda \in S$
Recursion rule: If $x, y \in S$, then so are $x0$, $0x$,$1x1$, xy.

Suppose we want to check if 101101001 is in S. Apply recursion rules backwards: $101101001 \in S$ if $0110100 \in S$,

Chapter 8.8: Recursive definitions, strings

Recall $B = \{0, 1\}$ and B^n is a string of length n.
Suppose we want strings of all lengths. We could write:

$$B^* = \{\lambda\} \cup B^1 \cup B^2 \ldots$$

Alternatively, B^* can be defined recursively:
Base: $\lambda \in B^*$
Recursion rule: If $x \in B^*$ then $x0 \in B^*$ and $x1 \in B^*$.

Example: Set S of binary strings with an even number of ones.
Base: $\lambda \in S$
Recursion rule: If $x, y \in S$, then so are $x0$, $0x$,$1x1$, xy.

Suppose we want to check if 101101001 is in S. Apply recursion rules backwards: $101101001 \in S$ if $0110100 \in S$,
if $011010 \in S$, if $01101 \in S$, if $1101 \in S$,

Chapter 8.8: Recursive definitions, strings

Recall $B = \{0, 1\}$ and B^n is a string of length n.
Suppose we want strings of all lengths. We could write:

$$B^* = \{\lambda\} \cup B^1 \cup B^2 \ldots$$

Alternatively, B^* can be defined recursively:
Base: $\lambda \in B^*$
Recursion rule: If $x \in B^*$ then $x0 \in B^*$ and $x1 \in B^*$.

Example: Set S of binary strings with an even number of ones.
Base: $\lambda \in S$
Recursion rule: If $x, y \in S$, then so are $x0$, $0x$,$1x1$, xy.

Suppose we want to check if 101101001 is in S. Apply recursion rules backwards: $101101001 \in S$ if $0110100 \in S$,
if $011010 \in S$, if $01101 \in S$, if $1101 \in S$,
if $10 \in S$, if $1 \in S$. (NO!)

Chapter 8.8: Recursive definitions, example

Set S of allowed algebraic expressions (strings) with variable x:

Chapter 8.8: Recursive definitions, example

Set S of allowed algebraic expressions (strings) with variable x:
Base cases: the string "x" is in S, and strings representing a real number are in S.

Chapter 8.8: Recursive definitions, example

Set S of allowed algebraic expressions (strings) with variable x:
Base cases: the string "x" is in S, and strings representing a real number are in S.
Recursive rules: if strings A and B are in S, then so are $A + B$, $A - B$, $(A) * (B)$ and $(A)/(B)$.

Chapter 8.8: Recursive definitions, example

Set S of allowed algebraic expressions (strings) with variable x:
Base cases: the string "x" is in S, and strings representing a real number are in S.
Recursive rules: if strings A and B are in S, then so are $A + B$, $A - B$, $(A) * (B)$ and $(A)/(B)$.

Is $(2) * (x) + (x)/(3)$ in S?

Chapter 8.8: Recursive definitions, example

Set S of allowed algebraic expressions (strings) with variable x:
Base cases: the string "x" is in S, and strings representing a real number are in S.
Recursive rules: if strings A and B are in S, then so are $A + B$, $A - B$, $(A) * (B)$ and $(A)/(B)$.

Is $(2) * (x) + (x)/(3)$ in S? Yes, but not $2 * x + x/3$
Is $x + +$ in S?

Chapter 8.8: Recursive definitions, example

Set S of allowed algebraic expressions (strings) with variable x:
Base cases: the string "x" is in S, and strings representing a real number are in S.
Recursive rules: if strings A and B are in S, then so are $A + B$, $A - B$, $(A) * (B)$ and $(A)/(B)$.

Is $(2) * (x) + (x)/(3)$ in S? Yes, but not $2 * x + x/3$
Is $x + +$ in S? No

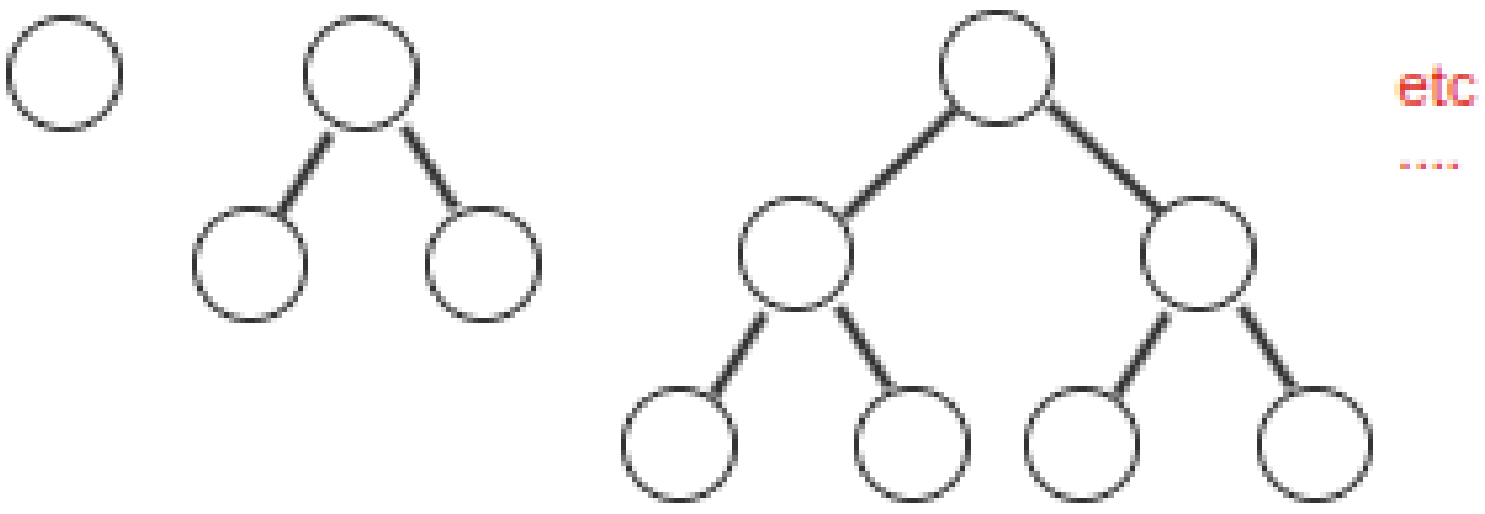
Check $(x - 3)/((x) * (x) + (4) * (x))$ recursively.

Chapter 8.8: Recursive definitions, binary trees

Perfect binary trees are defined recursively:
Base case: A single vertex is a perfect binary tree. The one vertex is called the root.

Chapter 8.8: Recursive definitions, binary trees

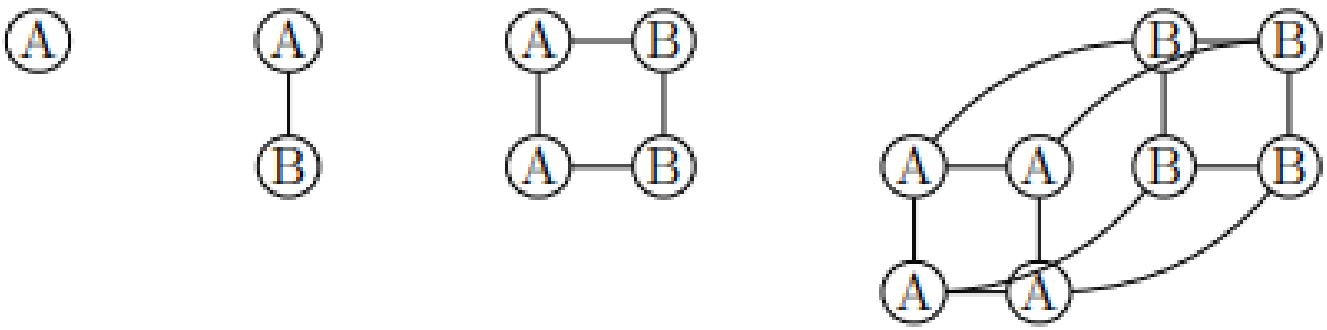
Perfect binary trees are defined recursively:
Base case: A single vertex is a perfect binary tree. The one vertex is called the root.
Recursive rule: If T is a perfect binary tree, a new perfect binary tree can be constructed by taking two copies of T, and adding a new vertex connecting the roots of each copy of T. The new vertex becomes the root.



Chapter 8.8: Recursive definitions, hypercubes

Hypercubes are defined as:

Base case: a single vertex is the hypercube Q_0.
Recursive rule: If Q_{n-1} is a hypercube, the hypercube Q_n consists of two copies of Q_{n-1} with edges joining corresponding vertices.



Chapter 8.8: Recursive definitions, problems

(1) Give a recursive definition for the set of integers divisible by 7.

(2) Suppose that a string (excluding text and numbers) has parenthesis () and square brackets []. To be properly nested, left parenthesis (or brackets) must match to a right one, and a right bracket cannot precede a left parenthesis, and vice versa.
So for example, ()[] and ([]([])) are allowed but
([)] and [()[]) are not.
Give a recursive definition for such strings.

Chapter 8.9: Structural induction

To verify that a recursively defined set has a certain property, the method of (strong) induction can be used.

Chapter 8.9: Structural induction

To verify that a recursively defined set has a certain property, the method of (strong) induction can be used.

Example: Let S be the set defined by
Base case: 2 and 3 are in S
Recursive rules: if $x \in S$, then $3x$ and $2x$ are in S.

Chapter 8.9: Structural induction

To verify that a recursively defined set has a certain property, the method of (strong) induction can be used.

Example: Let S be the set defined by
Base case: 2 and 3 are in S
Recursive rules: if $x \in S$, then $3x$ and $2x$ are in S.

Prove: $S = \{x : x = 2^l 3^m, l, m \in \mathbb{N}\}$.

Chapter 8.9: Structural induction

To verify that a recursively defined set has a certain property, the method of (strong) induction can be used.

Example: Let S be the set defined by
Base case: 2 and 3 are in S
Recursive rules: if $x \in S$, then $3x$ and $2x$ are in S.

Prove: $S = \{x : x = 2^l 3^m, l, m \in \mathbb{N}\}$.

Proof: By induction on n, where n is the index to a sorted list of integers in S.

Chapter 8.9: Structural induction

To verify that a recursively defined set has a certain property, the method of (strong) induction can be used.

Example: Let S be the set defined by
Base case: 2 and 3 are in S
Recursive rules: if $x \in S$, then $3x$ and $2x$ are in S.

Prove: $S = \{x : x = 2^l 3^m, l, m \in \mathbb{N}\}$.

Proof: By induction on n, where n is the index to a sorted list of integers in S.
Base case: For $n = 1$ and $n = 2$, the elements are 2 and 3.

Chapter 8.9: Structural induction

To verify that a recursively defined set has a certain property, the method of (strong) induction can be used.

Example: Let S be the set defined by
Base case: 2 and 3 are in S
Recursive rules: if $x \in S$, then $3x$ and $2x$ are in S.

Prove: $S = \{x : x = 2^l 3^m, l, m \in \mathbb{N}\}$.

Proof: By induction on n, where n is the index to a sorted list of integers in S.
Base case: For $n = 1$ and $n = 2$, the elements are 2 and 3.
Inductive hypothesis: Suppose for all $k \leq n$, x_k is the product of 2's and 3's.

Chapter 8.9: Structural induction

To verify that a recursively defined set has a certain property, the method of (strong) induction can be used.

Example: Let S be the set defined by
Base case: 2 and 3 are in S
Recursive rules: if $x \in S$, then $3x$ and $2x$ are in S.

Prove: $S = \{x : x = 2^l 3^m, l, m \in \mathbb{N}\}$.

Proof: By induction on n, where n is the index to a sorted list of integers in S.
Base case: For $n = 1$ and $n = 2$, the elements are 2 and 3.
Inductive hypothesis: Suppose for all $k \leq n$, x_k is the product of 2's and 3's.
By construction $x_{n+1} = 2x_k$ or $x_{n+1} = 3x_k$, where $k \leq n$. By the inductive hypothesis, it follows that x_{n+1} can also be written as the product of 2's and 3's.

Chapter 8.9: Structural induction, example

Suppose a set of binary strings S is defined by (base case) $\lambda \in S$ and (recursive rules) if $x, y, \in S$, then $x0$, $0x$ $1x1$ and xy are in S.

Chapter 8.9: Structural induction, example

Suppose a set of binary strings S is defined by (base case) $\lambda \in S$ and (recursive rules) if $x, y, \in S$, then $x0$, $0x$ $1x1$ and xy are in S.
Prove: Any string in S has an even number of zeros.
Proof: By induction on $n =$ string of length n (notice there is no explicit index here, so we must create one).
Base case: For $n = 0$, λ is the only string, which has no ones.

Chapter 8.9: Structural induction, example

Suppose a set of binary strings S is defined by (base case) $\lambda \in S$ and (recursive rules) if $x, y, \in S$, then $x0$, $0x$ $1x1$ and xy are in S.
Prove: Any string in S has an even number of zeros.
Proof: By induction on $n =$ string of length n (notice there is no explicit index here, so we must create one).
Base case: For $n = 0$, λ is the only string, which has no ones.
Inductive hypothesis: Suppose all strings of length $\leq n$ have an even number of 1's.

Chapter 8.9: Structural induction, example

Suppose a set of binary strings S is defined by (base case) $\lambda \in S$ and (recursive rules) if $x, y, \in S$, then $x0$, $0x$ $1x1$ and xy are in S.
Prove: Any string in S has an even number of zeros.
Proof: By induction on $n =$ string of length n (notice there is no explicit index here, so we must create one).
Base case: For $n = 0$, λ is the only string, which has no ones.
Inductive hypothesis: Suppose all strings of length $\leq n$ have an even number of 1's.
Let $s \in S$ be a string of length $n + 1$. There are four cases:

Chapter 8.9: Structural induction, example

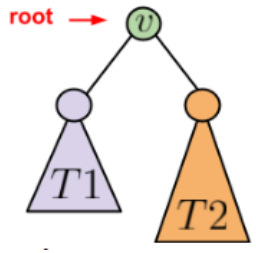
Suppose a set of binary strings S is defined by (base case) $\lambda \in S$ and (recursive rules) if $x, y, \in S$, then $x0$, $0x$ $1x1$ and xy are in S.
Prove: Any string in S has an even number of zeros.
Proof: By induction on $n =$ string of length n (notice there is no explicit index here, so we must create one).
Base case: For $n = 0$, λ is the only string, which has no ones.
Inductive hypothesis: Suppose all strings of length $\leq n$ have an even number of 1's.
Let $s \in S$ be a string of length $n + 1$. There are four cases:
(1) $s = x0$, where $x \in S$ and x has length n. Then s and x both have an even number of ones by the inductive hypothesis.

Chapter 8.9: Structural induction, example

Suppose a set of binary strings S is defined by (base case) $\lambda \in S$ and (recursive rules) if $x, y, \in S$, then $x0$, $0x$ $1x1$ and xy are in S.
Prove: Any string in S has an even number of zeros.
Proof: By induction on $n =$ string of length n (notice there is no explicit index here, so we must create one).
Base case: For $n = 0$, λ is the only string, which has no ones.
Inductive hypothesis: Suppose all strings of length $\leq n$ have an even number of 1's.
Let $s \in S$ be a string of length $n + 1$. There are four cases:
(1) $s = x0$, where $x \in S$ and x has length n. Then s and x both have an even number of ones by the inductive hypothesis.
(2) $s = 0x$, where $x \in S$. The proof is the same as case 1.
(3) $s = 1x1$, where $x \in S$. By the inductive hypothesis, x has an even number of 1's, and therefore so does s.
(4) $s = xy$, where $x, y \in S$, and the lengths of x, y are $\leq n$. By the inductive hypothesis, both x and y have an even number of 1's, and therefore so does s.

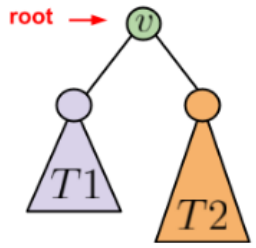
Chapter 8.9: Full binary trees

A full binary tree is defined recursively as:
Base case: A single vertex is a full binary tree, and is its own root.
Recursive rule: If T_1, T_2 are full binary trees, then a new binary tree can be formed by attaching the roots of these trees to a new vertex, which becomes the root.



Chapter 8.9: Full binary trees

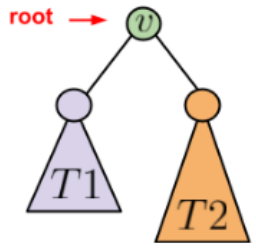
A full binary tree is defined recursively as:
Base case: A single vertex is a full binary tree, and is its own root.
Recursive rule: If T_1, T_2 are full binary trees, then a new binary tree can be formed by attaching the roots of these trees to a new vertex, which becomes the root.



How can a tree be verified to be a full binary tree? Apply rules backwards (example).

Chapter 8.9: Full binary trees

A full binary tree is defined recursively as:
Base case: A single vertex is a full binary tree, and is its own root.
Recursive rule: If T_1, T_2 are full binary trees, then a new binary tree can be formed by attaching the roots of these trees to a new vertex, which becomes the root.



How can a tree be verified to be a full binary tree? Apply rules backwards (example).

Chapter 8.9: Structural induction, full binary trees

Define $v(T)$ as the number of vertices in a binary tree.

Chapter 8.9: Structural induction, full binary trees

Define $v(T)$ as the number of vertices in a binary tree.

Prove: All full binary trees have an odd number of vertices.

Chapter 8.9: Structural induction, full binary trees

Define $v(T)$ as the number of vertices in a binary tree.

Prove: All full binary trees have an odd number of vertices.
Proof: By induction on the number n of vertices.

Chapter 8.9: Structural induction, full binary trees

Define $v(T)$ as the number of vertices in a binary tree.

Prove: All full binary trees have an odd number of vertices.
Proof: By induction on the number n of vertices.
(Base) A full binary tree with a single vertex has an odd number of vertices.
(Inductive hypothesis) Suppose that all full binary trees with n or fewer vertices have an odd number of vertices.

Chapter 8.9: Structural induction, full binary trees

Define $v(T)$ as the number of vertices in a binary tree.

Prove: All full binary trees have an odd number of vertices.
Proof: By induction on the number n of vertices.
(Base) A full binary tree with a single vertex has an odd number of vertices.
(Inductive hypothesis) Suppose that all full binary trees with n or fewer vertices have an odd number of vertices. Consider a tree T' which has $n+1$ vertices. T' must be formed from two trees T_1 and T_2, each of which has n or fewer vertices.

Chapter 8.9: Structural induction, full binary trees

Define $v(T)$ as the number of vertices in a binary tree.

Prove: All full binary trees have an odd number of vertices.
Proof: By induction on the number n of vertices.
(Base) A full binary tree with a single vertex has an odd number of vertices.
(Inductive hypothesis) Suppose that all full binary trees with n or fewer vertices have an odd number of vertices. Consider a tree T' which has $n+1$ vertices. T' must be formed from two trees T_1 and T_2, each of which has n or fewer vertices. By the inductive hypothesis, both $v(T_1)$ and $v(T_2)$ are odd, therefore T' has $v(T_1)+v(T_2)+1$ vertices, which is also odd.

Chapter 8.9: Structural induction, full binary trees

The **height** $h(T)$ of a binary tree is the maximum number of links from the root to the vertices.

Chapter 8.9: Structural induction, full binary trees

The **height** $h(T)$ of a binary tree is the maximum number of links from the root to the vertices.

Prove: If T is a full binary tree, then $v(T) \geq 2h(T) + 1$.

Chapter 8.9: Structural induction, full binary trees

The **height** $h(T)$ of a binary tree is the maximum number of links from the root to the vertices.

Prove: If T is a full binary tree, then $v(T) \geq 2h(T) + 1$.
Proof: By induction on the number n of vertices.

Chapter 8.9: Structural induction, full binary trees

The **height** $h(T)$ of a binary tree is the maximum number of links from the root to the vertices.

Prove: If T is a full binary tree, then $v(T) \geq 2h(T) + 1$.
Proof: By induction on the number n of vertices.
(Base) A full binary tree with a single vertex has $h(T) = 0$ and $v(T) = 1$, which satisfies the inequality.

Chapter 8.9: Structural induction, full binary trees

The **height** $h(T)$ of a binary tree is the maximum number of links from the root to the vertices.

Prove: If T is a full binary tree, then $v(T) \geq 2h(T) + 1$.
Proof: By induction on the number n of vertices.
(Base) A full binary tree with a single vertex has $h(T) = 0$ and $v(T) = 1$, which satisfies the inequality.
(Inductive hypothesis) Suppose that if T is a full binary tree with n or fewer vertices, then $v(T) \geq 2h(T) + 1$.

Chapter 8.9: Structural induction, full binary trees

The **height** $h(T)$ of a binary tree is the maximum number of links from the root to the vertices.

Prove: If T is a full binary tree, then $v(T) \geq 2h(T) + 1$.
Proof: By induction on the number n of vertices.
(Base) A full binary tree with a single vertex has $h(T) = 0$ and $v(T) = 1$, which satisfies the inequality.
(Inductive hypothesis) Suppose that if T is a full binary tree with n or fewer vertices, then $v(T) \geq 2h(T) + 1$.
Consider a tree T' which has $n + 1$ vertices. T' must be formed from two trees T_1 and T_2, and by construction
$h(T') = \max(h(T_1), h(T_2)) + 1$.

Chapter 8.9: Structural induction, full binary trees

The **height** $h(T)$ of a binary tree is the maximum number of links from the root to the vertices.

Prove: If T is a full binary tree, then $v(T) \geq 2h(T) + 1$.
Proof: By induction on the number n of vertices.
(Base) A full binary tree with a single vertex has $h(T) = 0$ and $v(T) = 1$, which satisfies the inequality.
(Inductive hypothesis) Suppose that if T is a full binary tree with n or fewer vertices, then $v(T) \geq 2h(T) + 1$.
Consider a tree T' which has $n + 1$ vertices. T' must be formed from two trees T_1 and T_2, and by construction
$h(T') = \max(h(T_1), h(T_2)) + 1$.

By the inductive hypothesis, $v(T_1) \geq 2h(T_1) + 1$ and $v(T_2) \geq 2h(T_2) + 1$. Therefore,
$v(T') = v(T_1) + v(T_2) + 1 \geq 2(h(T_1) + h(T_2)) + 3$

Chapter 8.9: Structural induction, full binary trees

The **height** $h(T)$ of a binary tree is the maximum number of links from the root to the vertices.

Prove: If T is a full binary tree, then $v(T) \geq 2h(T) + 1$.
Proof: By induction on the number n of vertices.
(Base) A full binary tree with a single vertex has $h(T) = 0$ and $v(T) = 1$, which satisfies the inequality.
(Inductive hypothesis) Suppose that if T is a full binary tree with n or fewer vertices, then $v(T) \geq 2h(T) + 1$.
Consider a tree T' which has $n + 1$ vertices. T' must be formed from two trees T_1 and T_2, and by construction
$h(T') = \max(h(T_1), h(T_2)) + 1$.

By the inductive hypothesis, $v(T_1) \geq 2h(T_1) + 1$ and $v(T_2) \geq 2h(T_2) + 1$. Therefore,
$v(T') = v(T_1) + v(T_2) + 1 \geq 2(h(T_1) + h(T_2)) + 3$
$\geq 2\max(h(T_1), h(T_2)) + 3 = 2(h(T') - 1) + 3 = 2h(T') + 1$.

Chapter 8.9: Structural induction, problems

(1) Consider a set of strings defined recursively as follows:
Base case: $a \in S$
Recursive rules: if $x \in S$, then so are xb, bx, axa, xaa.
Prove that every string in S contains an odd number of a's.

(2) Prove that if T is a perfect binary tree, that
$h(T) = \log_2(v(T) + 1) - 1$.
(Note: if T is formed from two copies of T, then
$h(T') = h(T) + 1$ and $v(T') = 2v(T) + 1$).