Induction and recursion

Chapter 5

ICSI 210 Discrete Structures

Qi Wang
Department of Computer Science
University at Albany
State University of New York

Topics

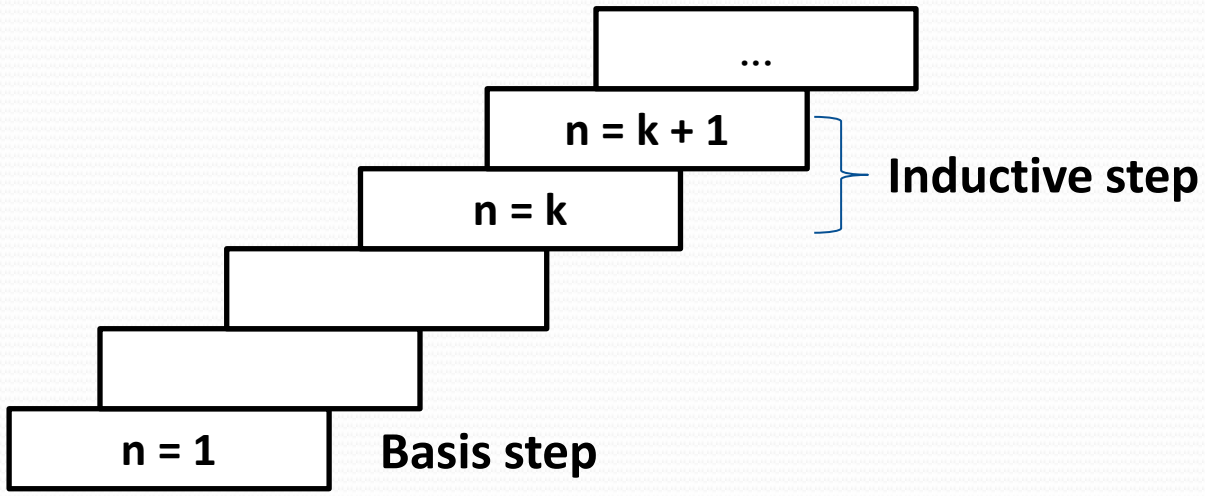
- Mathematical Induction
- Strong Induction
- Recursive Definitions
- Recursive Algorithms

Mathematical Induction

Section 5.1

Mathematical Induction

- Suppose we have an infinite ladder:
  1. We can reach the first rung of the ladder (**Basis step**).
  2. If we can reach a particular rung of the ladder, then we can reach the next rung (**Inductive step**).
- From the basis step, by applying the inductive step any number of times , we can reach any particular rung.



Principle of Mathematical Induction

- **Principle of Mathematical Induction**: To prove that $P(n)$ is true for all positive integers n, we complete these steps:
  - **Basis Step**: Show that $P(1)$ is true.
    - Proofs do not always start at the integer 1.
  - **Inductive Step**: Show that $P(k) \rightarrow P(k + 1)$ is true for all positive integers k. Show that if $P(k)$ is true, then $P(k + 1)$ must also be true.
    - To complete the inductive step, assuming the *inductive hypothesis* that $P(k)$ holds for an arbitrary integer k, show that $P(k + 1)$ must be true.
- $(P(1) \wedge \forall k\,(P(k) \rightarrow P(k + 1))) \rightarrow \forall n\,P(n)$

Mathematical Induction

- Mathematical induction can be used to prove a conjecture once it has been made (and is true).
- It cannot be used to find new theorems.
- Many theorems can be proved in many ways, including by mathematical induction.

Proofs by Mathematical Induction

- **Example**: Show that: $\sum_{i=1}^{n} = \frac{n(n+1)}{2}$

  Solution: Let *P*(*n*) be the proposition that $\sum_{i=1}^{n} = \frac{n(n+1)}{2}$.

  - BASIS STEP: *P*(1) is true since 1(1 + 1)/2 = 1.
  - INDUCTIVE STEP: Assume true for *P*(*k*). The inductive hypothesis is $\sum_{i=1}^{k} = \frac{k(k+1)}{2}$ for an arbitrary positive integer *k.*

P(k + 1) =

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

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

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

Hence, P(k) → P(k + 1) . Therefore, the sum of the first n positive integers is $\sum_{i=1}^{n} = \frac{n(n+1)}{2}$.

Proofs by Mathematical Induction

- **Example**: Conjecture and prove a formula for the sum of the first n positive odd integers.

  Solution:

  $1 = 1,\ 1 + 3 = 4,\ 1 + 3 + 5 = 9,$

  $1 + 3 + 5 + 7 = 16,\ 1 + 3 + 5 + 7 + 9 = 25.$

  Conjecture that the sum of the first n positive odd integers is n^2.

  $$1 + 3 + 5 + 7 + \cdots + (2n - 1) = n^2.$$

Proofs by Mathematical Induction

Solution cont. :

BASIS STEP: $P(1)$ is true since $1^2 = 1$.

INDUCTIVE STEP: Assume true for $P(k)$. The inductive hypothesis is $P(k)$. $1 + 3 + 5 + \cdots + (2k - 1) = k^2$.
Show $P(k) \rightarrow P(k + 1)$ for every positive integer k.

For $P(k + 1)$,

$$1 + 3 + 5 + \cdots + (2k - 1) + (2k + 1) = P(k) + (2k + 1)$$
$$= k^2 + (2k + 1)$$
$$= (k + 1)^2.$$

Hence, $P(k) \rightarrow P(k + 1)$. Therefore, the sum of the first n positive odd integers is n^2.

Proving Inequalities

- **Example**: Use mathematical induction to prove that $n < 2^n$ for all positive integers n.

Solution: Let $P(n)$ be the proposition that $n < 2^n$.

BASIS STEP: $P(1)$ is true since $1 < 2^1 = 2$.

INDUCTIVE STEP: Assume true for $P(k)$. The inductive hypothesis is $P(k)$, i.e., $k < 2^k$ for an arbitrary positive integer k. Must show that $P(k) \rightarrow P(k + 1)$.
By the inductive hypothesis, $k < 2^k$, it follows that:

$$k + 1 < 2^k + 1 \quad \text{(since } 1 < 2^k\text{)}$$
$$\leq 2^k + 2^k = 2 \cdot 2^k = 2^{k+1}$$

Hence, $P(k) \rightarrow P(k + 1)$. Therefore $n < 2^n$ holds for all positive integers n.

Proving Inequalities

- **Example**: Use mathematical induction to prove that $2^n < n!$, for every integer $n \geq 4$.

Solution: Let $P(n)$ be the proposition that $2^n < n!$.

BASIS STEP: $P(4)$ is true since $2^4 = 16 < 4! = 24$.

INDUCTIVE STEP: Assume true for $P(k)$. The inductive hypothesis is $P(k)$, i.e., $2^k < k!$ for an arbitrary positive integer $k \geq 4$. Must show $P(k) \rightarrow P(k + 1)$.

For $P(k)$, $2^{k+1} = 2 \cdot 2^k$

$< 2 \cdot k!$ (By the inductive hypothesis)

$< (k + 1)k! = (k + 1)!$

Hence, $P(k) \rightarrow P(k + 1)$. Therefore, $2^n < n!$ holds, for every integer $n \geq 4$.

Proving Divisibility Results

- **Example**: Use mathematical induction to prove that $n^3 - n$ is divisible by 3, for every positive integer n.

  Solution: Let $P(n)$ be the proposition that $n^3 - n$ is divisible by 3.

  BASIS STEP: $P(1)$ is true since $1^3 - 1 = 0$, which is divisible by 3.

  INDUCTIVE STEP: Assume true for $P(k)$. The inductive hypothesis is $P(k)$, i.e., $k^3 - k$ is divisible by 3, for an arbitrary positive integer k. $P(k) \rightarrow P(k + 1)$.

Proving Divisibility Results

Solution cont. :

For $P(k+1)$, $(k+1)^3 - (k+1)$

$$= (k^3 + 3k^2 + 3k + 1) - (k+1)$$

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

$(k^3 - k)$ is divisible by 3 b the inductive hypothesis; $3(k^2 + k)$ is divisible by 3. Therefore, the sum $(k^3 - k) + 3(k^2 + k)$ is divisible by 3. $P(k+1)$ is divisible by 3.

Hence, $P(k) \rightarrow P(k+1)$. Therefore, $n^3 - n$ is divisible by 3, for every integer positive integer n.

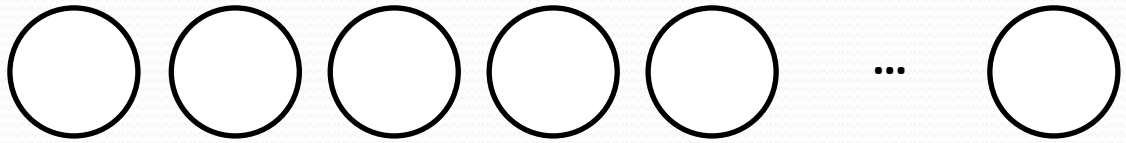
Number of Subsets of a Finite Set

- **Example**: Use mathematical induction to show that if S is a finite set with n elements, where n is a nonnegative integer, then S has 2^n subsets.

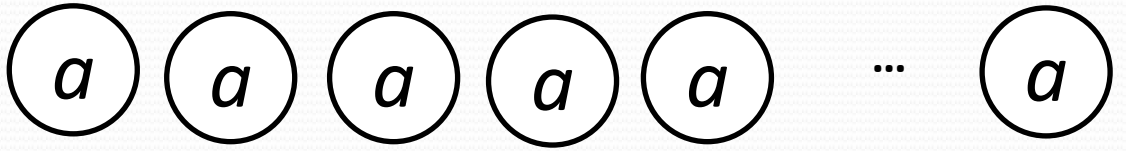
  Solution: Let $P(n)$ be the proposition that a set with n elements has 2^n subsets.
  Basis Step: $P(0)$ is true, because the empty set has only itself as a subset and $2^0 = 1$.
  Inductive Step: Assume $P(k)$ is true for an arbitrary nonnegative integer k. The inductive hypothesis is that every set with k elements has 2^k subsets. Must show $P(k) \rightarrow P(k + 1)$.

Number of Subsets of a Finite Set

Solution cont.: By the inductive hypothesis, there are 2^k subsets for every set with k elements .



For a set with $k + 1$ element, assume a is the $k+1^{th}$ element. By adding a to each of the 2^k subsets, another 2^k subsets are formed.



The total number of subsets of a set with $k + 1$ element is $2 \cdot 2^k = 2^{k+1}$.

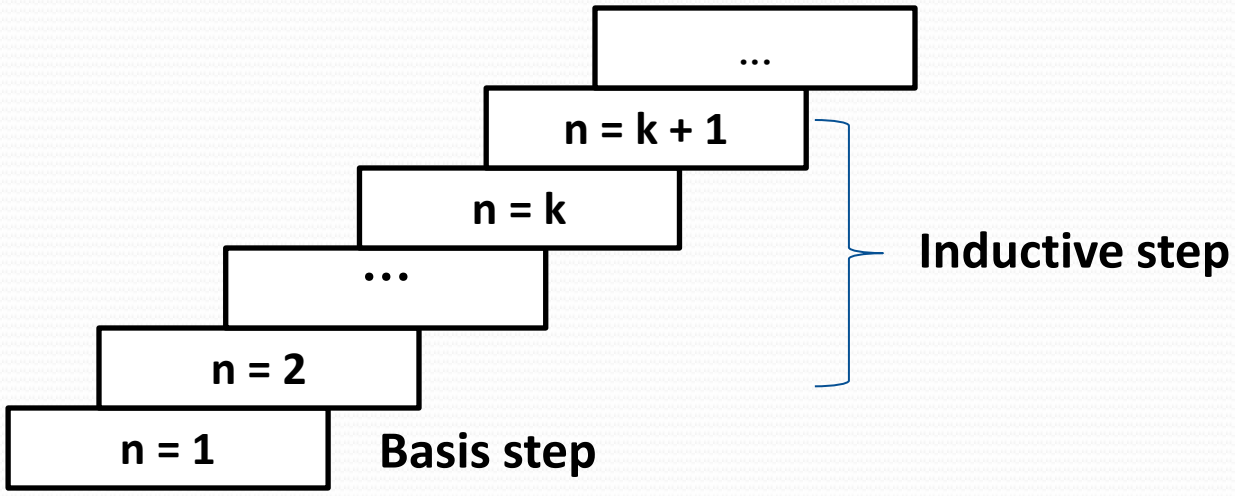
Hence $P(k) \rightarrow P(k + 1)$. Therefore, a set with n elements has 2^n subsets.

Strong Induction

Section 5.2

Strong Induction

- Strong Induction: To prove that $P(n)$ is true for all positive integers n, where $P(n)$ is a propositional function, complete two steps:
  - Basis Step: Verify that the proposition $P(1)$ is true.
  - Inductive Step: Show the conditional statement $[P(1) \wedge P(2) \wedge \cdots \wedge P(k)] \rightarrow P(k+1)$ holds for all positive integers k.



Strong Induction

- Strong Induction is sometimes called *the second principle of mathematical induction* or *complete induction*.
- The principles of mathematical induction and strong induction are all equivalent.
- Can always use strong induction instead of mathematical induction.
  - Use whichever is simpler.

Proofs by Strong Induction

- **Example**: Show that if n is an integer greater than 1, then n can be written as the product of primes.

  Solution:

  Let $P(n)$ be the proposition that n can be written as a product of primes.

  BASIS STEP: $P(2)$ is true since 2 itself is prime.

  INDUCTIVE STEP: The inductive hypothesis is $P(j)$ is true for all integers j with $2 \leq j \leq k$. Must show $[P(2) \wedge P(2) \wedge \cdots \wedge P(k)] \rightarrow P(k+1)$ is true for all $n \geq 2$.

Proofs by Strong Induction

Solution cont.:

To show that $P(k + 1)$ must be true under this assumption, two cases need to be considered:

- If $k + 1$ is prime, then $P(k + 1)$ is true.
- Otherwise, $k + 1$ is composite and can be written as the product of two positive integers a and b with $\mathbf{2 \leq a \leq b < k + 1}$. By the inductive hypothesis a and b can be written as the product of primes and therefore $k + 1$ can also be written as the product of those primes.

Hence, $[P(2) \wedge P(2) \wedge \cdots \wedge P(k)] \rightarrow P(k + 1)$ is true for all $n \geq 2$. Therefore, if n is an integer greater than 1, then n can be written as the product of primes.

Proofs by Strong Induction

- **Example**: Prove that every amount of postage of 12 cents or more can be formed using just 4-cent and 5-cent stamps.

  Solution: Let $P(n)$ be the proposition that postage of n cents can be formed using 4-cent and 5-cent stamps.
  BASIS STEP: $P(12)$, $P(13)$, $P(14)$, and $P(15)$ hold.

  $P(12)$ uses three 4-cent stamps.

  $P(13)$ uses two 4-cent stamps and one 5-cent stamp.

  $P(14)$ uses one 4-cent stamp and two 5-cent stamps.

  $P(15)$ uses three 5-cent stamps.

Proofs by Strong Induction

Solution cont.:

P(16) can be formed by adding one 4-cent stamp to P(12) or replacing the three 5-cent stamps by four 4-cent stamps in *P*(15). For any amount larger than 16, they can be formed by adding one 4-cent stamp to a previous amount. For example,

P(16) = *P*(12) + 4-cent stamp

P(17) = *P*(13) + 4-cent stamp

P(18) = *P*(14) + 4-cent stamp

P(19) = *P*(15) + 4-cent stamp

…

Proofs by Strong Induction

Solution cont.:

INDUCTIVE STEP: The inductive hypothesis is $P(j)$ holds for $12 \leq j \leq k$, where $k \geq 15$. Must show $[P(2) \wedge P(12) \wedge \cdots \wedge P(k)] \rightarrow P(k + 1)$ is true for all $n \geq 12$.

For $k + 1$,

$k + 1 = (k - 3) + 4$.

$P(k - 3)$ holds since $k - 3 \geq 12$.

To form postage of $k + 1$ cents, add a 4-cent stamp to the postage for $k - 3$ cents.

Hence, $[P(2) \wedge P(12) \wedge \cdots \wedge P(k)] \rightarrow P(k + 1)$ is true for all n ≥ 12. Therefore, $P(n)$ holds for all $n \geq 12$.

Proofs by Mathematical Induction

Alternative Solution (Proof by mathematical induction):

Let $P(n)$ be the proposition that postage of n cents can be formed using 4-cent and 5-cent stamps.

BASIS STEP: Postage of 12 cents can be formed using three 4-cent stamps.

INDUCTIVE STEP: The inductive hypothesis $P(k)$ for any positive integer k is that postage of k cents can be formed using 4-cent and 5-cent stamps. Must show $P(k) \rightarrow P(k + 1)$ is true for all $n \geq 12$.

Proofs by Mathematical Induction

Solution cont. :

To show $P(k + 1)$ where $k \geq 12$, consider two cases:

- If at least one 4-cent stamp has been used, then a 4-cent stamp can be replaced with a 5-cent stamp to yield a total of $k + 1$ cents.
- No 4-cent stamp have been used and at least three 5-cent stamps were used. Three 5-cent stamps can be replaced by four 4-cent stamps to yield a total of $k + 1$ cents.

Hence, $P(k) \rightarrow P(k + 1)$ is true for all $n \geq 12$. Therefore, $P(n)$ holds for all $n \geq 12$.

Recursive Definitions

Section 5.3

Recursively Defined Functions

- **Definition**: A *recursive* or *inductive definition* of a function consists of two steps.

  BASIS STEP: Specify the value of the function at zero.
  RECURSIVE STEP: Give a rule for finding its value at an integer from its values at smaller integers.

- A function $f(n)$ is the same as a sequence $a_0, a_1, \ldots, a_i$, where $f(i) = a_i$.

Recursively Defined Functions

- **Example**: Suppose f is defined by

  BASIS STEP: $f(0) = 3$

  RECURSIVE STEP: $f(n + 1) = 2f(n) + 3$

  Find $f(1)$, $f(2)$, $f(3)$ and $f(4)$.

  Solution:

  $f(1) = 2f(0) + 3 = 2 \cdot 3 + 3 = 9$

  $f(2) = 2f(1) + 3 = 2 \cdot 9 + 3 = 21$

  $f(3) = 2f(2) + 3 = 2 \cdot 21 + 3 = 45$

  $f(4) = 2f(3) + 3 = 2 \cdot 45 + 3 = 93$

Recursively Defined Functions

- **Example:** Give a recursive definition of the factorial function $n!$.

  Solution:

  BASIS STEP: $f(0) = 1$

  RECURSIVE STEP: $f(n + 1) = (n + 1) \cdot f(n)$

Recursively Defined Functions

- **Example**: Give a recursive definition of the sequence $\sum_{k=0}^{n} a_k$.

Solution:

BASIS STEP: $\sum_{k=0}^{0} a_k = a_0$

RECURSIVE STEP: $\sum_{k=0}^{n+1} a_k = (\sum_{k=0}^{n} a_k) + a_{n+1}$

Fibonacci Numbers

- **Example** : The Fibonacci numbers are defined as follows:

  BASIS STEP: $f_0 = 0$

  $f_1 = 1$

  RECURSIVE STEP: $f_n = f_{n-1} + f_{n-2}$

  Find f_2, f_3, f_4 and f_5.

  Solution:

  $f_2 = f_1 + f_0 = 1 + 0 = 1$

  $f_3 = f_2 + f_1 = 1 + 1 = 2$

  $f_4 = f_3 + f_2 = 2 + 1 = 3$

  $f_5 = f_4 + f_3 = 3 + 2 = 5$

Recursively Defined Sets

- Recursive definitions of sets have two parts:

  BASIS STEP: specifies an initial collection of elements.

  RECURSIVE STEP: gives the rules for forming new elements in the set from those already known to be in the set.

- Assume that the set contains only the elements specified in the basis step and generated by applications of the rules in the recursive step.

Recursively Defined Sets

- **Example** : Subset of Integers S:

  BASIS STEP: $3 \in S$.

  RECURSIVE STEP: If $x \in S$ and $y \in S$, then $x + y$ is in S.

  Initially 3 is in S, then 3 + 3 = 6, then 3 + 6 = 9, etc.

  Is S the set of all positive multiples of 3?

Recursively Defined Sets

- **Example**: The natural numbers **N:**

  BASIS STEP: 0 ∈ **N.**

  RECURSIVE STEP: If *n* is in **N**, then *n* + 1 is in **N**.

  Initially 0 is in N, then 0 + 1 = 1, then 1 + 1 = 2, etc.

Strings

- **Definition**: The set Σ^* of *strings* over the alphabet Σ:

  BASIS STEP: $\lambda \in \Sigma^*$ (λ is the empty string).

  RECURSIVE STEP: If w is in Σ^* and x is in Σ, then $wx \in \Sigma^*$.

Strings

- **Example**: If Σ = {0,1}, the strings in Σ* are the set of all bit strings, λ, 0, 1, 00, 01, 10, 11, etc.

  Solution:

  λ ∈ Σ*.

  Since λ ∈ Σ* and 0 ∈ Σ, λ0 ∈ Σ*, λ0 = 0, **0 ∈ Σ*.**

  Since λ ∈ Σ* and 1 ∈ Σ, λ1 ∈ Σ*, λ1 = 1, **1∈ Σ*.**

  Since 0 ∈ Σ* and 0 ∈ Σ, **00 ∈ Σ*.**

  Since 0 ∈ Σ* and 1 ∈ Σ, **01 ∈ Σ*.**

  Since 1 ∈ Σ* and 0 ∈ Σ, **10 ∈ Σ*.**

  Since 1 ∈ Σ* and 1 ∈ Σ, **1** 1 **∈ Σ*.**

  …

  Hence, Σ* = {λ, 0, 1, 00, 01, 10, 11, …}

Strings

- **Example**: If $\Sigma = \{a, b\}$, show that aab is in Σ^*.

  Solution:

  Since $\lambda \in \Sigma^*$ and $a \in \Sigma$, $\lambda a \in \Sigma^*$, $\lambda a = a$, $a \in \Sigma^*$.

  Since $a \in \Sigma^*$ and $a \in \Sigma$, $aa \in \Sigma^*$.

  Since $aa \in \Sigma^*$ and $b \in \Sigma$, $aab \in \Sigma^*$.

  Hence, aab is in Σ^*.

String Concatenation

- **Definition**: Two strings can be *concatenated*. Let Σ be a set of symbols and Σ* be the set of strings formed from the symbols in Σ. The concatenation of two strings, denoted by ·, is defined recursively as follows.

  BASIS STEP: If $w \in \Sigma^*$, then $w \cdot \lambda = w$.

  RECURSIVE STEP: If $w_1 \in \Sigma^*$ and $w_2 \in \Sigma^*$ and $x \in \Sigma$, then $w_1 \cdot (w_2 x) = (w_1 \cdot w_2)x$.

- Often $w_1 \cdot w_2$ is written as $w_1 w_2$.
  - If w_1 = *abra* and w_2 = *cadabra*, the concatenation $w_1 w_2$ = ***abracadabra***.

Length of a String

- **Example**: Give a recursive definition of $l(w)$, the length of the string w.

  Solution: The length of a string can be recursively defined by:

  BASIS STEP: $l(\lambda) = 0$;

  RECURSIVE STEP: $l(wx) = l(w) + 1$ if $w \in \Sigma^*$ and $x \in \Sigma$.

  Σ : The alphabet

Balanced Parentheses

- **Example**: Give a recursive definition of the set of balanced parentheses P.

  Solution:

  BASIS STEP: $() \in P$

  RECURSIVE STEP: If $w \in P$, then $()w \in P$, $(w) \in P$ and $w() \in P$.

Balanced Parentheses

- **Example**: Show that (()()) is in *P,* the set of balanced parentheses.

  Solution:

  Since () ∈ *P*, ()() ∈ *P*

  Since ()() ∈ *P*, (()()) ∈ *P*

  Hence, (()()) is in *P*.

Balanced Parentheses

- **Example**: Show that is))(() is not in *P,* the set of balanced parentheses.

  Solution:

  Although () is in *P*, there is no rules that can be used to form))(, the rest of the))((). Therefore**,**))(() is not in *P*.

Recursive Algorithms

Section 5.4

Algorithms

- **Definition**: An *algorithm* is a finite sequence of precise instructions for performing a computation or for solving a problem.
  - **Input*:*** An algorithm has input values from a specified set.
  - **Output:** From each set of input values, an algorithm produces output values that are the solution to the problem.
- To write an algorithm, we can simply use the English language to describe the sequence of steps used(Input-Process-Output).

MAX Algorithm

- **Example:** Describe an algorithm for finding the largest value in a finite sequence of integers.

  Solution:

  Input: a finite sequence of integers
  Process:

  1. Set the temporary maximum equal to the first integer in the sequence.
  2. Compare it with the next integer in the sequence, and if the next integer is larger, set the temporary maximum equal to this integer.

MAX Algorithm

Solution cont.:

3. Repeat the previous step if there are more integers in the sequence.
4. Stop when there are no integers left in the sequence. The temporary maximum at this point is the largest integer in the sequence.

Output: the largest value

Exponentiation Algorithm

- **Example:** Describe an algorithm for computing a^n, where a is a nonzero real number and n is a nonnegative integer.

Solution:

Input: a: nonzero real number, n:nonnegative integer

Process:

1. If n is equal to 0, the result is 1. (Initial case: $a^0 = 1$.)
2. Otherwise, set the initial result equal to 1. while $n > 0$,
   1) Multiply the result by a,
   2) Decrement n by 1.
3. Return the final result.

Output: a^n

Factorial Algorithm

- **Example:** Give an algorithm for computing $n!$, where n is a nonnegative integer.

Solution:

Input: *n:* nonnegative integer

Process:

1. If n is equal to 0, the result is 1. (Initial case: 0! = 1.)
2. Otherwise, set the initial result equal to 1. While $n > 0$,
    1) Multiply the result by n,
    2) Decrement n by 1.
3. Return the final result.

Output: $n!$

Iterative Algorithms

- **Definition:** Iteration is the repetition of a process. It is a repetition of a mathematical or computational procedure applied to the result of a previous application, typically as a means of obtaining successively closer approximations to the solution of a problem.
- The previous three algorithms are iterative algorithms.

Recursive Algorithms

- **Definition**: An algorithm is called *recursive* if it solves a problem by reducing it to an instance of the same problem with smaller input, the base case.
- For the algorithm to terminate, the instance of the problem must eventually be reduced to some initial case, the base case, for which the solution is known.

Recursive Exponentiation Algorithm

- **Example**: Give a recursive algorithm for computing a^n, where a is a nonzero real number and n is a nonnegative integer. (Use the recursive definition of $a^n = a \times a^{n-1}$.

Recursive Exponentiation Algorithm

Solution:

Input: a: nonzero real number
n:nonnegative integer

Process:

1. If n is equal to 0, the result is 1. (Base Case: $a^0 = 1$.)
2. Otherwise, **the result is a times a^{n-1}.** (Recursive case)
3. Return the final result.

Output: a^n

- The problem a^n is reduced to an instance of the same problem with smaller input a^{n-1} . It is eventually reduced the base case, for which the solution is known.

Recursive Factorial Algorithm

- **Example**: Give a recursive algorithm for computing $n!$, where n is a nonnegative integer. Use the recursive definition of the factorial function.

  $n! = n \times (n - 1)!$

Recursive Factorial Algorithm

Solution:

Input: *n:* nonnegative integer

Process:

1. If *n* is equal to 0, the result is 1. (Base Case: 0! = 1.)
2. Otherwise, **the result is *n* times (*n* -1)!.** (Recursive Case)
3. Return the final result.

Output: *n*!

- The problem *n*! is reduced to an instance of the same problem with smaller input (*n* -1)!. It is eventually reduced the base case, for which the solution is known.

Recursion and Iteration

- An iterative approach:
  - much less computation
  - harder to implement
- A recursive procedure:
  - less efficient,
  - easier to implement