KS141203 MATEMATIKA DISKRIT
(*DISCRETE MATHEMATICS*)

Recursive Definition

Ahmad Muklason, Ph.D.

Outline

1. Recursive definition
2. Recursive algorithm
3. Iteration

Recursive Definitions

Def. The process of defining an object in terms of itself is called recursion.

e.g. We specify the terms of a sequence using

(1) an explicit formula:

$$a_n = 2^n, \quad n = 0, 1, 2, \ldots$$

(2) a recursive form:

$$a_0 = 1,$$

$$a_{n+1} = 2a_n \,, n = 0, 1, 2, \ldots$$

Example 1. Suppose that f is defined recursively by

$$f(0) = 3 \,, f(n+1) = 2f(n) + 3$$

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

$f(1) = 2f(0) + 3 = 9$ $\quad$ $f(2) = 2f(1) + 3 = 21$

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

Recursive Definitions (cont'd)

Example 2. Give a recursive definition of the factorial function $F(n) = n!$.

Solution:

initial value : $F(0) = 1$

recursive form : $F(n) = (n)! = n.\ (n-1)\ !$

$= n\ .\ F\ (n-1)$

For example $F(5) = 5\ .\ F(4) = 5\ .\ 4\ .\ F(3) = 5\ .\ 4\ .\ 3\ .\ F(2)$

$= 5\ .\ 4\ .\ 3\ .\ 2\ .\ F(1) = 5\ .\ 4\ .\ 3.\ 2\ .\ 1\ .\ F(0)$

$= 5.\ 4.\ 3.\ 2.\ 1.\ 1 = 120$

Recursive Definitions (cont'd)

Example 3. The Fibonacci numbers $f_0, f_1, f_2 \ldots$, are defined by :

$$f_0 = 0\ ,$$
$$f_1 = 1\ ,$$
$$f_n = f_{n-1} + f_{n-2}\ , \text{ for } n = 2, 3, 4, \ldots$$

what is f_4 ?

Solution:

$$f_4 = f_3 + f_2 = (f_2 + f_1) + (f_1 + f_0) = f_2 + 2$$
$$= (f_1 + f_0) + 2 = 3$$

Recursively defined sets

Same two parts:

1. Base case (or basis step)
2. Recursive step

Examples:

The set of positive integers

Basis step: $1 \in S$

Recursive step: if $x \in S$, then $x+1 \in S$

- The set of odd positive integers
- Basis step: $1 \in S$
- Recursive step: If $x \in S$, then $x+2 \in S$

- The set of positive integer powers of 3
- Basis step: $3 \in S$
- Recursive step: If $x \in S$, then $3*x \in S$

Recursive Algorithms

Sometimes we can reduce the solution to a problem with a particular set of input to the solution of the same problem with smaller input values.

Example:

gcd(a, b) = gcd(b mod a, a), (when $a < b$; $a > 0$)

gcd(5, 8) = gcd(8 mod 5, 5) = gcd(3, 5)

gcd(3, 5) = gcd(5 mod 3, 3) = gcd(2, 3)

gcd(2, 3) = gcd(3 mod 2, 2) = gcd(1, 2)

gcd(1, 2) = gcd(2 mod 1, 1) = gcd(0, 1)

gcd(0, 1) = 1 → the basis step/initial value; gcd (0, b) = b

Recursive Algorithms (cont'd)

Def 1. An algorithm is called **recursive** if it solves a problem by **reducing** it to an instance of the same problem with **smaller input**.

Example 1: Recursive algorithm for computing gcd

Algorithm 1

Procedure gcd (a, b : non-negative integer with $a < b$)

if $a = 0$ **then** gcd(a, b) := b

else gcd(a, b) := gcd(b mod a, a).

Recursive Algorithms (cont'd)

Example 2: Give a recursive algorithm for computing a^n, where $a \in \mathbf{R}$ and $a \neq 0$, $n \in \mathbf{N}$.

Solution:

recursive definition of a^n:

initial value : $a^0 = 1$

recursive def : $a^1 = a \cdot a^{n-1}$

Algorithm 2.

Procedure power (a : nonzero real number,
n : nonnegative integer)
if $n = 0$ **then** power(a, n) := 1
else power(a, n) := a * power(a, n-1).

Recursive Algorithms (cont'd)

Example 3. Give the value of $n!$, $n \in \mathbf{Z}^+$

Solution: Note : $n! = n \times (n\text{-}1)!$

Algorithm 3

procedure factorial (n: positive integer)

if $n = 0$ **then** factorial (n) := 1

else factorial (n) := n x factorial (n-1)

Recursion and Iteration

Recursive definition: expresses the value of a function at a positive integers in terms of values of the smaller integers.

Iterative definition: starts with the value of the function at 1 or more integers (the base cases) and successively apply the recursive definition to find the value of the function at successive larger integers.

Example: Find Fibonacci numbers (**Note :** $f_0 = 0, f_1 = 1, f_n = f_{n-1} + f_{n-2}$ for $n \geq 2$)

Algorithm 4 (Recursive Fibonacci)

procedure Fibonacci (*n* : nonnegative integer)
if *n* = 0 **then** Fibonacci (0) := 0
else if *n* = 1 **then** Fibonacci (1) := 1
else Fibonacci (*n*) := Fibonacci (*n-1*)+Fibonacci (*n-2*)

Recursion and Iteration (cont'd)

Algorithm 5. (Iterative Fibonacci)

procedure iterative_fibonacci (n: nonnegative integer)
if $n = 0$ **then** $y := 0$ // $y = f_0$
else begin
    $x := 0$
    $y := 1$ // $y = f_1$
    for $i := 1$ **to** $n-1$
    begin
        $z := x + y$
        $x := y$
        $y := z$
    end
  end
{y is f_n}

	$i = 1$	$i = 2$	$i = 3$
z	f_2	f_3	f_4
x	f_1	f_2	f_3
y	f_2	f_3	f_4