Recursive Definition and Recurrence Relations

Recursive Definition

- Sequences
- Sets
- Functions
- Algorithms
- Recurrence Relations
- Solution of Recurrence Relations
 - Proving Closed Form with Induction
 - Substitution Method

Imdad ullah Khan

Inductive or Recursive Definition

An inductive or recursive definition is just defining things in terms of simpler/smaller version(s) of itself

Explicitly define base case(s) and build upon that

- Recursively Defined Sequences
- Recursively Defined Sets
- Recursively Defined Functions
- Recursive Algorithms

Recursively Defined Sequences

 $\{f_n\} = 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, \dots$

Fibonacci Numbers

 $f_0 = 0$ $f_1 = 1$ $f_n = f_{n-1} + f_{n-2} \quad (n > 1)$

$$f_n = \begin{cases} 0 & n = 0 \\ 1 & n = 1 \\ f_{n-1} + f_{n-2} & n > 1 \end{cases}$$

 $\{t_n\} = 0, 1, 3, 6, 10, 15, 21, 28, 36, 45, 55, 66, 78, 91, 105, 120, 136$

$$t_n = \begin{cases} 0 & \text{if } n = 0 \\ t_{n-1} + n & \text{if } n \ge 1 \end{cases}$$

Triangular Numbers

Recursive Definition: Sequence

Closed form of recurrence relation and almost every statement about recursively defined structures are usually proved using induction

 ${f_n} = 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, \dots$

Fibonacci Numbers

$$\begin{array}{l} f_0 \ = \ 0 \\ f_1 \ = \ 1 \\ f_n \ = \ f_{n-1} + f_{n-2} \\ \end{array} \left(\begin{array}{l} n > 1 \end{array} \right) \\ \end{array} \right) \\ f_n \ = \ \begin{cases} 0 \\ 1 \\ f_{n-1} + f_{n-2} \\ n > 1 \\ \end{cases} \\ \begin{array}{l} f_n \ = \ \\ f_{n-1} + f_{n-2} \\ n > 1 \\ \end{array} \right) \\ \end{array}$$

Every third number is even

Recursive Definition: Sequence



Basis stepn = 0: $f_{3n} = f_0 = 0$ is evenInductive Hypothesis:Suppose $f_{3(n-1)}$ is evenInductive Step:Using IH, show that f_{3n} is even

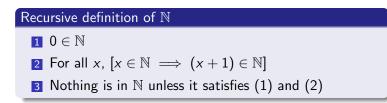
$$f_{3n} = f_{3n-1} + f_{3n-2}$$

= $f_{3n-2} + f_{3n-3} + f_{3n-2}$
= $2f_{3n-2} + f_{3(n-1)}$

Hence f_{3n} is even

Recursive Definition: Sets

The set of natural numbers, $\mathbb{N} = \{0,1,2,\ldots\}$



Why is condition (3) necessary?

otherwise $\{0,.7,1,1.7,2,2.7,3,3.7,\ldots\}$ could qualify to be called $\mathbb N$

Recursively Defined Functions

f

Recursive definition of factorial function n! = n(n-1)(n-2)...(3)(2)(1)

$$f(n) = n! \qquad \bullet f(0) = 0! = 1 \\ \bullet f(n+1) = (n+1)f(n) = (n+1)n! = (n+1)!$$

$$(n+1) = (n+1)f(n)$$

= $(n+1)(n)f(n-1)$
= $(n+1)(n)(n-1)f(n-2)$
= $(n+1)(n)(n-1)(n-2)f(n-3)$
: :
= $(n+1)(n)(n-1)(n-2)(n-3)\dots(3)(2)(1)f(0)$
= $(n+1)(n)(n-1)(n-2)(n-3)\dots(3)(2)(1)1$
= $(n+1)!$

Recursively Defined Functions

Definition of exponentiation function

$$a^n = \underbrace{a \times a \times \ldots \times a}_{n \text{ times}}$$

A recursive definition of exponentiation

$$a^{n} = \begin{cases} a * a^{n-1} & \text{if } n > 1 \\ a & \text{if } n = 1 \\ 1 & \text{if } n = 0 \end{cases}$$

Another recursive definition of exponentiation called repeated squaring

$$a^{n} = \begin{cases} a^{n/2} \cdot a^{n/2} & \text{if } n > 1 \text{ is even} \\ a \cdot a^{(n-1)/2} \cdot a^{(n-1)/2} & \text{if } n \text{ is odd} \\ 1 & \text{if } n = 0 \end{cases}$$

Input: a and $n \ge 0$ Output: a^n $a^n = \begin{cases} a * a^{n-1} & \text{if } n > 1\\ a & \text{if } n = 1\\ 1 & \text{if } n = 0 \end{cases}$

Algorithm Computing a^n using the recursive definition

```
function REC-EXP(a,n)
```

if n = 0 then
 return 1
else if n = 1 then
 return a
else

return a * REC-EXP(a, n-1)

Recursive Algorithm: Repeated Squaring

Input: a and
$$n \ge 0$$

Output: a^n

$$a^n = \begin{cases} a^{n/2} \cdot a^{n/2} & \text{if } n > 1 \text{ is even} \\ a \cdot a^{(n-1)/2} \cdot a^{(n-1)/2} & \text{if } n \text{ is odd} \\ 1 & \text{if } n = 0 \end{cases}$$

Algorithm Computing *aⁿ* using repeated squaring

```
function REP-SQ-EXP(a, n)

if n = 0 then

return 1

else if n > 0 AND n is even then

z \leftarrow \text{REP-SQ-EXP}(a, n/2)

return z * z

else

z \leftarrow \text{REP-SQ-EXP}(a, (n-1)/2)
```

return a * z * z

Recursive Algorithms: Searching in Sorted Array

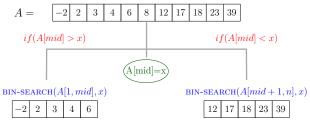
Input: Sorted array *A* of *n* numbers and a number *x* **Output:** Index of *x* in *A* if $x \in A$ or -1 if $x \notin A$

▷ Notice the input array is sorted

- 1 Compare *A*[*mid*] with *x*
- 2 If not equal, eliminate the half where x cannot lie
- **3** Search x in the remaining half

▷ same problem but smaller





Recursive Algorithms: Binary Search

Input: Sorted array A of n numbers and a number x Output: Index of x in A if $x \in A$ or -1 if $x \notin A$

Algorithm Binary Search for x in sorted array A[st,..., end]

```
function BIN-SEARCH(A, st, end, x)
  if end < st then
     return -1
  else
    mid \leftarrow rac{(end + st)}{2}
    if A[mid] = x then
       return mid
                                                         If found return index
    else if A[mid] > x then
       return BIN-SEARCH(A, st, mid - 1, x)
    else
       return BIN-SEARCH(A, mid + 1, end, x)
```

Defining a structure in terms of itself!

Google	recursion				
	Web	Images	Maps	Shopping	More -
	About 4,350,000 results (0.24 seconds)				
	Did you mean: recursion				

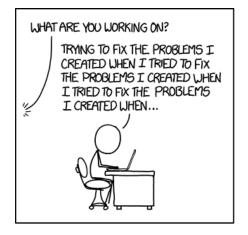
Figure: Google gets the joke!

Defining a structure in terms of itself!

252	indentation, paragraph, see paragraph index, 30, 74-76, 150-154, 211-212 \index, 74, 212 curly brace in argument, 75 idx file entry writen by 208 in command argument, 153 space around, 169	interrow space in array or tabali 169,007 in squarray enviro in tabortatack.11 interword space.14.
	special character in argument, 75 too may on page, 143 Vindexentry, 75, 212 on idx file, 208 Vindexapace, 75 Indicator, error, 29, 134 Vinf (inf), 44 subscript of, 190 infinite loop, 252	before or after \hs, in math mode, 51 produced by invisit too much with \al. without line break, \intextsep, 200 intraword dash, 14, 17 invalid character error invisible
	infinite work stretchable length, 100, 102, 215 information, moving, 65–77, 207–209 \infty (∞), 43 ink, rectangular blob of, 106 inproceedings bibliography entry type, 161 input character, 32 file, 12	character, 12, 32 command, 169 delimiter, 47 environment, 169 term made with \m text, 82, 97 \iota (<i>i</i>), 41 \isodd, 196 inim the shape, 16, 1 in math mode, 51, 1 used for emphasis, 1 \item, 24-25, 34, 184

Figure: Learn infinite loops from this book!





Is there a base case for recursive stress?

