## Discrete Mathematics

## 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

$$
\left\{f_{n}\right\}=0,1,1,2,3,5,8,13,21,34,55,89,144, \ldots
$$

## Fibonacci Numbers

$$
\begin{aligned}
& f_{0}=0 \\
& f_{1}=1 \\
& f_{n}=f_{n-1}+f_{n-2} \quad(n>1)
\end{aligned}
$$

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

## Recursive Definition: Sequence

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

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

## Triangular Numbers

## Recursive Definition: Sequence

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

$$
\left\{f_{n}\right\}=0,1,1,2,3,5,8,13,21,34,55,89,144, \ldots
$$

## Fibonacci Numbers

$$
\begin{aligned}
& f_{0}=0 \\
& f_{1}=1 \\
& f_{n}=f_{n-1}+f_{n-2} \quad(n>1)
\end{aligned}
$$

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

Every third number is even

## Recursive Definition: Sequence

## Theorem

## $f_{3 n}$ is even

Basis step $\quad n=0: \quad f_{3 n}=f_{0}=0$ is even
Inductive Hypothesis: Suppose $f_{3(n-1)}$ is even
Inductive Step: Using IH, show that $f_{3 n}$ is even

$$
\begin{aligned}
f_{3 n} & =f_{3 n-1}+f_{3 n-2} \\
& =f_{3 n-2}+f_{3 n-3}+f_{3 n-2} \\
& =2 f_{3 n-2}+f_{3(n-1)}
\end{aligned}
$$

Hence $f_{3 n}$ is even

## Recursive Definition: Sets

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

## Recursive definition of $\mathbb{N}$

$10 \in \mathbb{N}$
2 For all $x,[x \in \mathbb{N} \Longrightarrow(x+1) \in \mathbb{N}]$
3 Nothing is in $\mathbb{N}$ unless it satisfies (1) and (2)

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

Recursive definition of factorial function $n!=n(n-1)(n-2) \ldots(3)(2)(1)$

$$
\begin{array}{ll}
f(n)=n! & f(0)=0!=1 \\
& ■ f(n+1)=(n+1) f(n)=(n+1) n!=(n+1)!
\end{array}
$$

$$
\begin{aligned}
f(n+1) & =(n+1) f(n) \\
& =(n+1)(n) f(n-1) \\
& =(n+1)(n)(n-1) f(n-2) \\
& =(n+1)(n)(n-1)(n-2) f(n-3) \\
& \vdots \quad \vdots \\
& =(n+1)(n)(n-1)(n-2)(n-3) \ldots(3)(2)(1) f(0) \\
& =(n+1)(n)(n-1)(n-2)(n-3) \ldots(3)(2)(1) 1 \\
& =(n+1)!
\end{aligned}
$$

## 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

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

## Recursive Algorithm for Exponentiation

Input: $a$ and $n \geq 0$
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 $a * \operatorname{REC}-\operatorname{EXP}(a, n-1)$

## Recursive Algorithm: Repeated Squaring

Input: $a$ and $n \geq 0$
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 $a^{n}$ using repeated squaring

function REP-SQ-EXP $(a, n)$
if $n=0$ then
return 1
else if $n>0$ AND $n$ is even then
$z \leftarrow \operatorname{REP}-\operatorname{SQ}-\operatorname{EXP}(a, n / 2)$
return $z * z$
else
$z \leftarrow \operatorname{REP}-\operatorname{SQ}-\operatorname{EXP}(a,(n-1) / 2)$
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 $x \notin A$
$\triangleright$ 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
$\triangleright$ 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 $A[s t, \ldots, e n d]$

```
function \(\operatorname{BIN}-\operatorname{SEARCH}(A, s t, e n d, x)\)
    if end <st then
        return -1
    else
        mid \(\leftarrow \frac{(\text { end }+s t)}{2}\)
        if \(A[\mathrm{mid}]=x\) then
        return mid \(\quad \triangleright\) If found return index
    else if \(A[\) mid \(]>x\) then
        return \(\operatorname{BIN}-\operatorname{SEARCH}(A, s t\), mid \(-1, x)\)
    else
        return \(\operatorname{BIN}-\operatorname{SEARCH}(A\), mid +1 , end,\(x)\)
```


## Recursion

## 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!

## Recursion

## Defining a structure in terms of itself!

| indentation, paragraph, see paragraph index, 30, 74-76, 150-154, 211-212 <br> \index, 74, 212 <br> curly brace in argument, 75 <br> idx file entry written by, 208 <br> in command argument, 153 <br> space around, 169 <br> special character in argument, 75 <br> too many on page, 143 <br> \indexentry, 75, 212 <br> on idx file, 208 <br> \indexspace, 75 <br> indicator, error, 29, 134 <br> \inf (inf), 44 <br> subscript of. 190 <br> infinite loop, 252 <br> infinitely stretchable length, 100, 102, 215 <br> information, moving, 65-77, 207-209 <br> \infty ( $\infty$ ), 43 <br> ink, rectangular blob of, 106 <br> inproceedings bibliography entry type, 161 <br> input <br> character, 32 <br> file, 12 | interrow space in array or tabul 169, 207 <br> in eqnarray euviry in lshortatack, 1 : interword space, 14 , before or after $\mathrm{lha}_{3}$ in math mode, 51 produced by invisil too much with \s? without line break, <br> lintextsep, 200 <br> intraword dash, 14, 17 invalid character error invisible <br> character, 12. 32 command, 169 delimiter, 47 environment, 169 term made with leb text, 82,97 <br> liota (t), 41 <br> \isodd, 196 <br> italic type shape, 16,37 <br> in math mode, 51, 15 <br> used for emplasis, 17 <br> liten, 24-25, 34, 184 |
| :---: | :---: |

Figure: Learn infinite loops from this book!

## Recursion: Base Case

The programmer got stuck in the shower because the instructions on the shampoo bottle said,

Lather, Rinse, Repeat.


## Speaking of Programmers...



## Is there a base case for recursive stress?



