Algorithmic Methods
Tricks of the Trade
5A

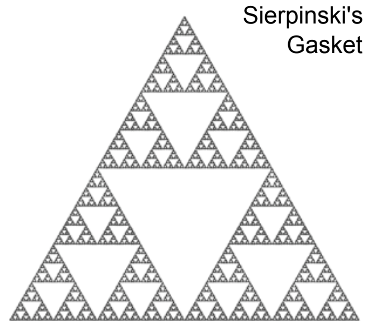
Recursion

Recursion

- A recursive operation is an operation that is defined in terms of itself.

http://fusionanomaly.net/recursion.jpg

Sierpinski's Gasket



Recursion

- Every recursive definition includes two parts:
  - Base case (non-recursive)
    A simple case that can be done without solving the same problem again.
  - Recursive case(s)
    One or more cases that are "simpler" versions of the original problem.
    - By "simpler", we sometimes mean "smaller" or "shorter" or "closer to the base case".

Factorial

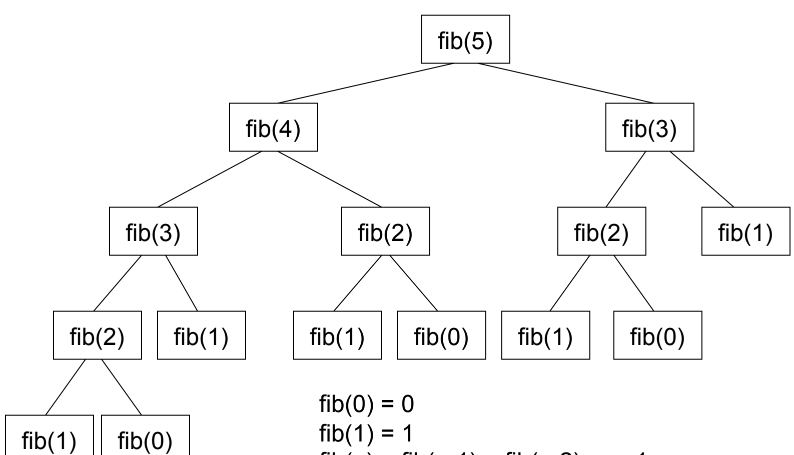
- Definition: n! = n(n-1)(n-2)…(2)(1)
- Since (n-1)(n-2)…(2)(1) = (n-1)!
  - n! = n(n-1)!, for n > 0
  - n! = 1 for n = 0 (base case)
- Example:

4! = 4(3!) = 4(6) = 24
3! = 3(2!) = 3(2) = 6
2! = 2(1!) = 2(1) = 2
1! = 1(0!) = 1(1) = 1

Fibonacci Numbers

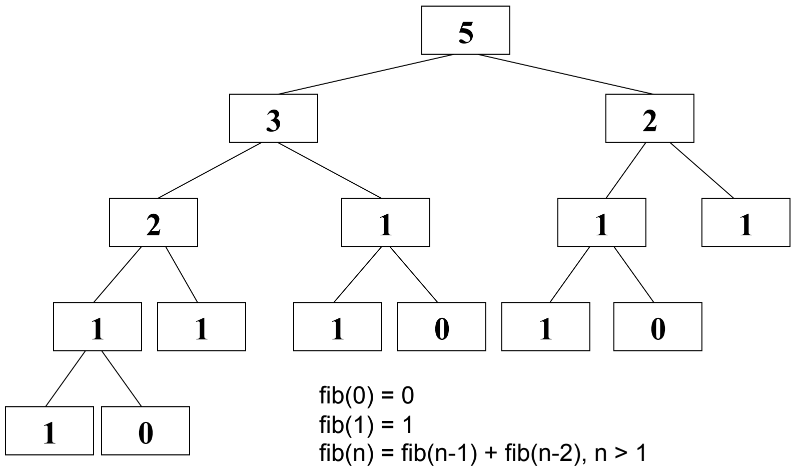
- A sequence of numbers each number is the sum of the previous two numbers in the sequence, starting the sequence with 0 and 1.
- 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, etc.
- Let fib(n) = the n^{th} Fibonacci number, n ≥ 0
  - fib(0) = 0 (base case)
  - fib(1) = 1 (base case)
  - fib(n) = fib(n-1) + fib(n-2), n > 1

Fibonacci Numbers



fib(0) = 0
fib(1) = 1
fib(n) = fib(n-1) + fib(n-2), n > 1

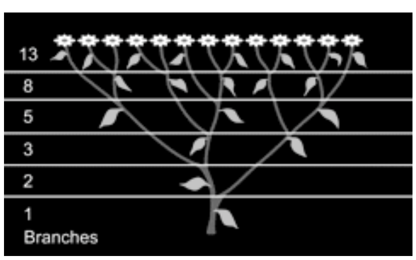
Fibonacci Numbers

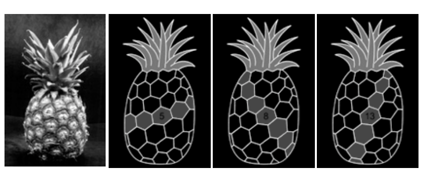


fib(0) = 0
fib(1) = 1
fib(n) = fib(n-1) + fib(n-2), n > 1

Fibonacci Numbers in Nature

http://britton.disted.camosun.bc.ca/fibslide/jbfibslide.htm
http://www.geom.uiuc.edu/~demo5337/s97b/art.htm





Recursive Sum (in Scheme)

- Computing the sum of a list of numbers.
  Use: **(sum (list 30 28 45 12))**
- Recursive Definition:

```
(define (sum numlist)
  (if (null? numlist)        <-- is numlist empty?
   0                         <-- if yes, result is 0
   (+ (first numlist)          } if no, result is the first number
      (sum (rest numlist)))    } + the sum of the rest of the numbers
  )
)
```

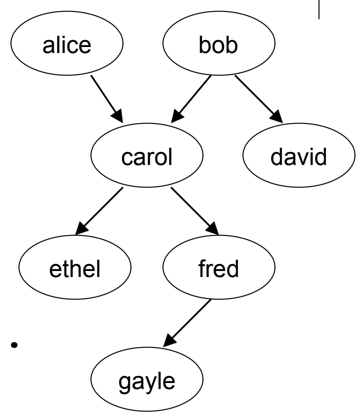

Recursive Sum (in Scheme)

```
(sum (list 30 28 45 12))
(+ 30 (sum (list 28 45 12)))
(+ 30 (+ 28 (sum (list 45 12))))
(+ 30 (+ 28 (+ 45 (sum (list 12)))))
(+ 30 (+ 28 (+ 45 (+ 12 (sum (list ))))))   <-- empty list
(+ 30 (+ 28 (+ 45 (+ 12 0 ))))
(+ 30 (+ 28 (+ 45 12 )))
(+ 30 (+ 28 57 ))
(+ 30 85 )
115
```

```
(if (null? numlist)
  0
  (+ (first numlist)
     (sum (rest numlist)))
)
```

Recursion in Prolog

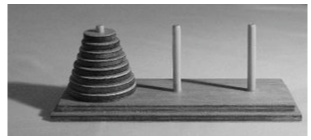
```
ancestor(X, Z) :-
   parent(X, Z).
ancestor(X, Z) :-
   parent(X, Y),
   ancestor(Y, Z).


?- ancestor(alice, gayle).
```
yes



Recursion in Prolog

```
ancestor(alice, gayle) :-
   parent(alice, Y), ancestor(Y, gayle).
Try Y = carol.
   parent(alice, carol).                        YES
   ancestor(carol, gayle) :-
     parent(carol, gayle).                    NO
   ancestor(carol, gayle) :-
      parent(carol, Y'), ancestor(Y', gayle).
   Try Y' = fred.
       parent(carol, fred).             YES
       ancestor(fred, gayle) :-
               parent(fred, gayle).       YES
```

Towers of Hanoi



Towers of Hanoi with 8 discs.

- A puzzle invented by French mathematician Edouard Lucas in 1883.
- At a temple far away, priests were led to a courtyard with three pegs and 64 discs stacked on one peg in size order.
  - Priests are only allowed to move one disc at a time from one peg to another.
  - Priests may not put a larger disc on top of a smaller disc at any time.
- The goal of the priests was to move all 64 discs from the leftmost peg to the rightmost peg.
- According to the legend, the world would end when the priests finished their work.

Towers of Hanoi

Move N discs from peg X to peg Y
(Let Z represent the other peg.)

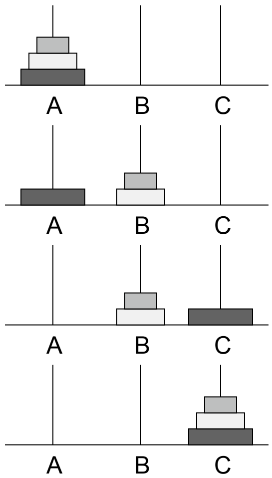
a. Move N-1 discs from peg X to peg Z (if N>1).

b. Move 1 disc from peg X to peg Y.

c. Move N-1 discs from peg Z to peg Y (if N>1).

extra peg Z

Towers of Hanoi (N=3)



Move 3 discs from peg A to peg C.
(extra peg is B)

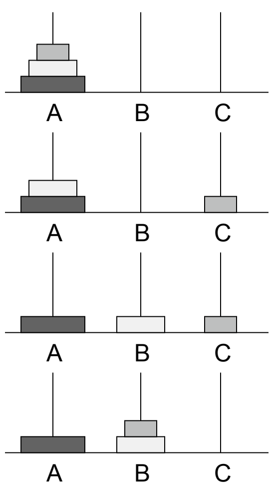
a. Move 2 discs from peg A to peg B.
(RECURSIVE... see next slide)

b. Move 1 disc from peg A to peg C.

extra
peg

c. Move 2 discs from peg B to peg C.
(RECURSIVE... see two slides ahead)

Towers of Hanoi (N=2)



Move 2 discs from peg A to peg B.
(extra peg is C)

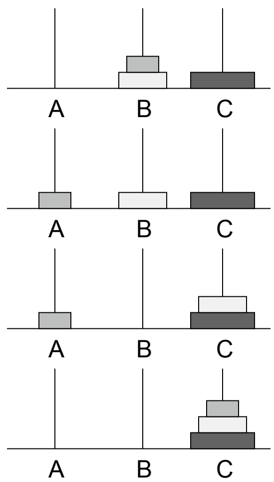
a. Move 1 disc from peg A to peg C.

b. Move 1 disc from peg A to peg B.

extra
peg

c. Move 1 disc from peg C to peg B.

Towers of Hanoi (N=2)



Move 2 discs from peg B to peg C. (extra peg is A)

a. Move 1 disc from peg B to peg A.

b. Move 1 disc from peg B to peg C.

c. Move 1 disc from peg A to peg C.

extra peg

Towers of Hanoi

Discs	Moves
1	______
2	______
3	______
4	______
5	______
...	
n	__________________

What is the fewest number of disc moves needed for a problem with n discs?

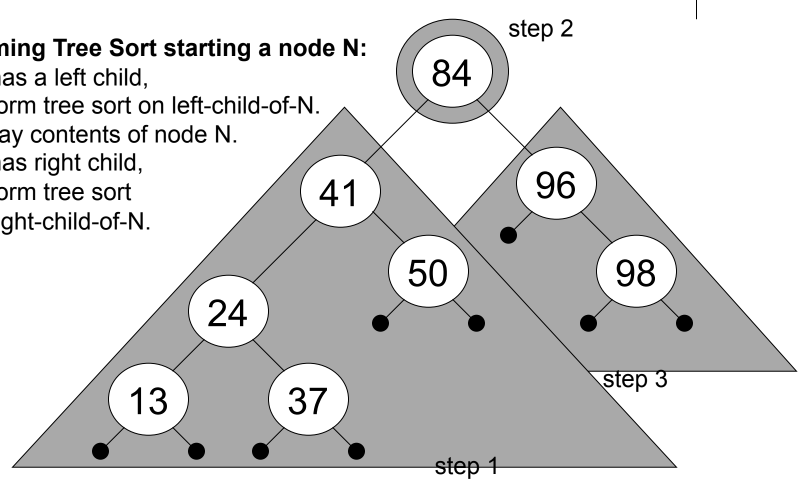
Towers of Hanoi

- If the priests moved a disc at a rate of 1 per second using the fewest number of disc moves, it would take the priests roughly 585 billion years to complete this puzzle!
  - The universe is currently about 13.7 billion years old.

Tree Sort Algorithm revisited

Performing Tree Sort starting a node N:

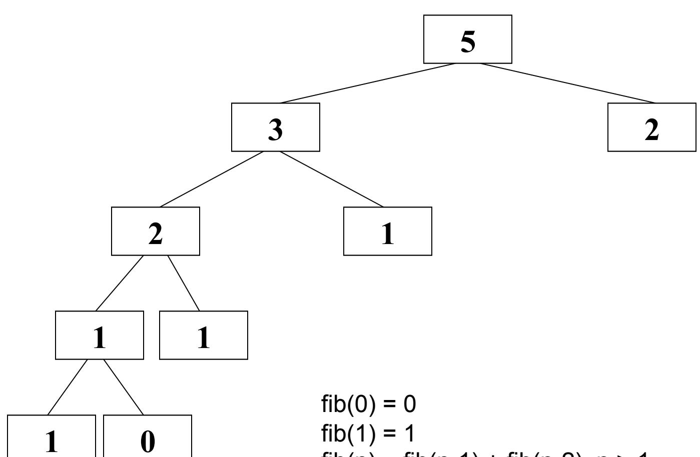
1. If N has a left child,
   perform tree sort on left-child-of-N.
2. Display contents of node N.
3. If N has right child,
   perform tree sort
   on right-child-of-N.



Dynamic Programming

- A technique that…
  - The problem is broken into sub-problems, and these sub-problems are solved and the solutions remembered, in case they need to be solved again.
  - All sub-problems that might be needed are solved in advance and then used to build up solutions to larger problems.

Fibonacci Numbers revisited



fib(0) = 0
fib(1) = 1
fib(n) = fib(n-1) + fib(n-2), n > 1

n	fib(n)
0	0
1	1
2	1
3	2
4	3
5	5

memoization