

Blowing a gasket

Classic recursive examples



CS112 Scientific Computation

Department of Computer Science
Wellesley College

Recursion

Recursion solves a problem by solving a smaller instance of the same problem

Think *divide, conquer, and glue*, where all of the subproblems have the same "shape" as the original problem



Factorial

$$\begin{aligned}4! &= 4 \times (3 \times 2 \times 1) \\ &= 4 \times 3!\end{aligned}$$

$$\begin{aligned}3! &= 3 \times (2 \times 1) \\ &= 3 \times 2!\end{aligned}$$

$$\begin{aligned}2! &= 2 \times (1) \\ &= 2 \times 1!\end{aligned}$$

$$1! = 1$$

recursive cases

```
function result = fact (n)
if (n <= 1)           % base case
    result = 1;
else                 % recursive step
    result = n * fact(n-1);
end
```

base case

3

The recursive leap of faith

To solve a problem recursively:

If it is simple enough
then solve it immediately

Otherwise
express solution in terms of
smaller, similar problem(s)

Initially this will take some
faith on your part



4

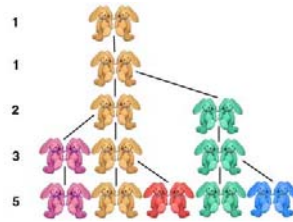
Fibonacci numbers, or multiplying rabbits?

Fibonacci numbers first appear around 500 B.C. in writings of a Sanscrit grammarian named Pingala who studied rhythm in language



Leonardo Fibonacci studied these numbers around 1200 in the context of multiplying rabbits

1 1 2 3 5 8 13 21 34 ...

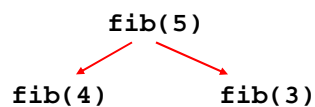


- In the first month, there's one newborn pair
- Newborn pairs become fertile in their second month
- Each month, every fertile pair begets a new pair
- Rabbits never die

5

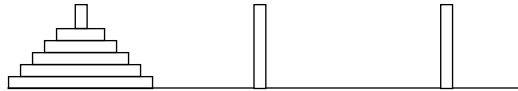
Fibonacci function

```
function result = fibonacci (n)
if (n <= 2)      % base case
    result = 1;
else            % recursive step
    result = fibonacci(n-1) + fibonacci(n-2);
end
```



6

Towers of Hanoi



7

A natural recursive solution

```
function towersOfHanoi (n, source, destination, spare)
if (n == 1)           % base case
    disp(['move disk 1 from ' source ' to ' destination]);
else                 % recursive step
    towersOfHanoi(n-1, source, spare, destination);
    disp(['move disk ' num2str(n) ' from ' source ' to ' destination]);
    towersOfHanoi(n-1, spare, destination, source);
end
```



Move N-1 smallest disks to pole B.



Move largest disk to pole C.



Move N-1 smallest disks to pole C.



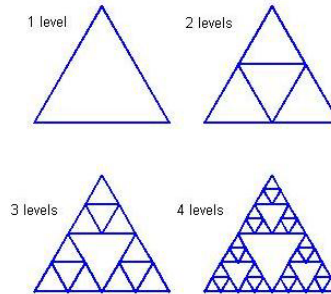
8

Sierpinski Gasket



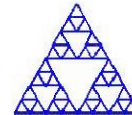
The **Sierpinski Gasket** was devised by the Polish mathematician, Waclaw Sierpinski, in the early 1900's

It is a self-similar figure composed of triangles whose construction can be described recursively



9

Sierpinski Gasket



To construct a Sierpinski Gasket of **level n** and **side length len** :

if **level $n = 1$**

then draw an equilateral triangle of size **len**

otherwise

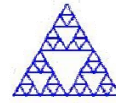
draw a Sierpinski Gasket of **level $(n-1)$** and **side length $len/2$**
in lower left corner

draw a Sierpinski Gasket of **level $(n-1)$** and **side length $len/2$**
in lower right corner

draw a Sierpinski Gasket of **level $(n-1)$** and **side length $len/2$**
in the top corner

10

Sierpinski Gasket: The Code



```
function sierpinski (levels, x, y, length)
% draws Sierpinski's Gasket with input number of levels and
% input side length, and lower left corner located at (x,y)
if (levels == 1)      % base case: draw a triangle
    triangle(x, y, length);
else                  % recursive step
    % draw smaller Sierpinski Gasket in lower left corner
    sierpinski(levels-1, x, y, length/2);
    % draw smaller Sierpinski Gasket in lower right corner
    sierpinski(levels-1, x+length/2, y, length/2);
    % draw smaller Sierpinski Gasket in upper corner
    sierpinski(levels-1, x+length/4, y+sqrt(3)*length/4, length/2);
end

function triangle (x, y, size)
% draws equilateral triangle of input size with lower left corner at (x,y)
plot([x x+size x+size/2 x], [y y y+sqrt(3)*size/2 y]);
```

11

Madam I'm Adam

Suppose we want to test whether a string is a palindrome

How can we express the solution to this problem using recursion?



12

Think recursively!

If the tree is empty

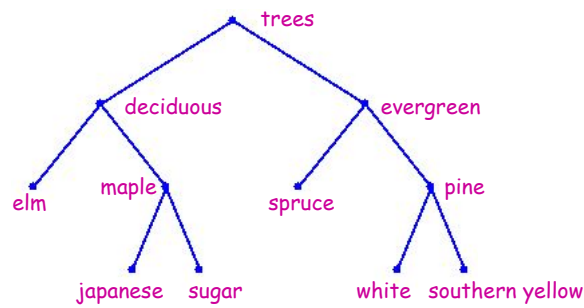
do nothing

Otherwise

print the string in the root node

print the contents of the left subtree

print the contents of the right subtree



15

Let's write the code

If the tree is not empty

print the string in the root node

print the contents of the left subtree

print the contents of the right subtree

```
function printTree (tree)
```

```
% prints all of the strings contained in the input tree
```

```
if ~isEmpty(tree)
```

```
    disp(tree{1});
```

```
    printTree(tree{2});
```

```
    printTree(tree{3});
```

```
end
```

```
% if tree is not empty...
```

```
% print string in root node
```

```
% print left subtree
```

```
% print right subtree
```



16

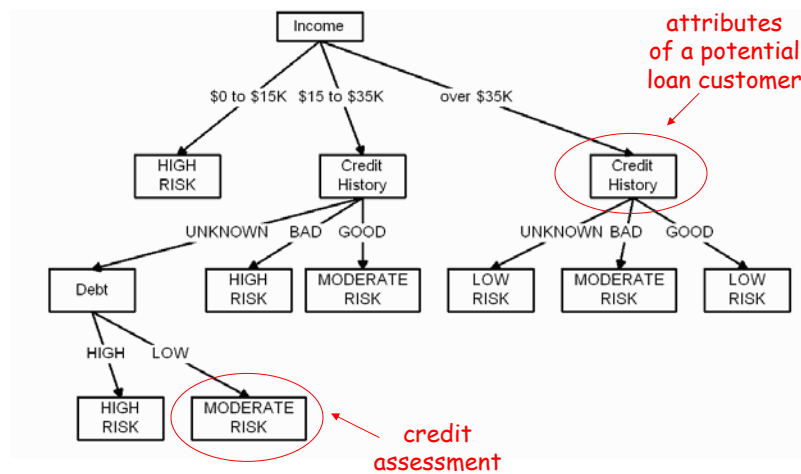
So is it empty?

```
function answer = isEmpty (tree)
% returns true if the input tree is an empty cell array
answer = (length(tree) == 0);
```



17

Decisions, decisions



18

Data Mining with decision trees

Medicine: What health factors result in increased risk for certain diseases or other health problems?

Economics: What social, political, or financial factors contribute most directly to certain economic indicators?

Geology: What physical factors best predict catastrophic events such as earthquakes and volcanoes?

