

# Trees

Recursive data structures



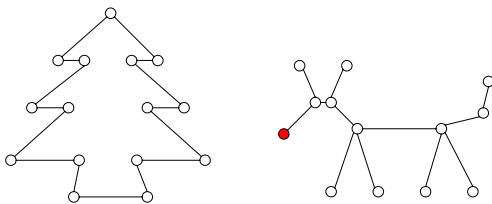
**CS112 Scientific Computation**  
Department of Computer Science  
Wellesley College

## Trees are everywhere



Trees 21-2

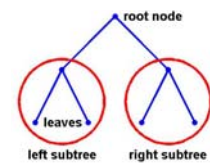
## One of these is not a tree



Trees 21-3

## Binary trees

- o A **binary tree** has at most two branches at each node
- o top node of a binary tree is the **root node**
- o nodes with no branches are **leaves**
- o subtrees below a node are the **left** and **right subtrees**
- o nodes store **values**
- o We will represent a binary tree using a cell array of 3 elements:



`{value {left subtree} {right subtree}}`

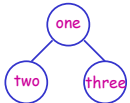
Trees 21-4

## A binary forest

```
>> seed = {'one' []};
```



```
>> twig = {'one' ['two' []] ['three' []]};
```



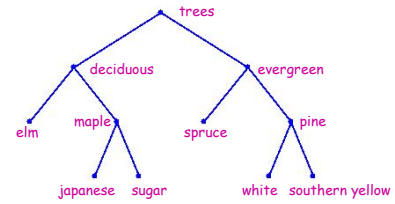
```
>> elm = {'one' ['two' [] ['four' []]]
         ['three' ['five' []]]};
```



Trees 21-5

## Phylogenetic tree of trees

```
phyl = {'trees' ['deciduous' ['elm' []] ...
                {'maple' ['japanese' []] ...
                  {'sugar' []}}] ...
        ['evergreen' ['spruce' []] ...
                  {'pine' ['white' []] ...
                    {'southern yellow' []}}]};
```



Trees 21-6

## Processing trees

Trees are **defined recursively** so  
we will **process them recursively**

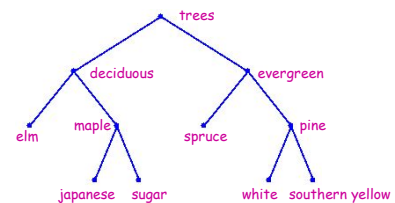
To solve a problem recursively:  
If it is simple enough  
then solve it  
Otherwise  
express solution in terms  
of smaller, similar problem(s)



Trees 21-7

## Print the contents of a tree

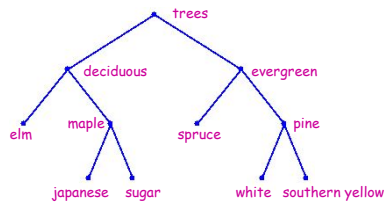
```
phyl = {'trees' ['deciduous' ['elm' []] ...
                {'maple' ['japanese' []] ...
                  {'sugar' []}}] ...
        ['evergreen' ['spruce' []] ...
                  {'pine' ['white' []] ...
                    {'southern yellow' []}}]};
```



Trees 21-8

## 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



Trees 21-9

## Let's write the code

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

```
function printTree (tree)
% prints all of the strings contained in the input tree
```

Trees 21-10

## 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) % if tree is not empty...
    disp(tree{1}); % print string in root node
    printTree(tree{2}); % print left subtree
    printTree(tree{3}); % print right subtree
end
```

Trees 21-11

## So is it empty?

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



Trees 21-12

## Count the nodes in a tree

If the tree is empty

return zero

Otherwise

count nodes in left subtree

count nodes in right subtree

return sum of counts plus one



Trees 21-13

## Count the nodes in a tree

If the tree is empty

return zero

Otherwise

count nodes in left subtree

count nodes in right subtree

return sum of counts plus one

```
function nodes = countNodes(tree)
```

```
if isEmpty(tree)
```

```
nodes = 0;
```

```
else
```

```
nodes = 1 + countNodes(tree{2}) + countNodes(tree{3});
```

```
end
```



Exercise: Modify `countNodes` to compute the height of a tree

Trees 21-14

## Search for a string in a tree

```
>> found = searchTree(phlyo, 'pine')
```

```
pine
```

```
white
```

```
southern yellow
```

```
found =
```

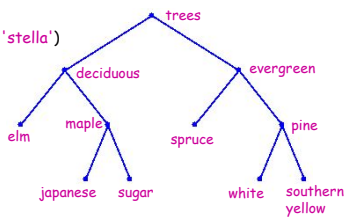
```
1
```

```
>> found = searchTree(phlyo, 'stella')
```

```
found =
```

```
0
```

```
>>
```



Write `searchTree`

Trees 21-15

## Let's play twenty questions

```
>> twentyQuests(twenty)
```

```
four legs? yes
```

```
hooves? no
```

```
stripes? yes
```

```
tiger? yes
```

```
Wow! I won!
```

```
Play again? yes
```

```
four legs? no
```

```
fly? yes
```

```
feathers? no
```

```
butterfly? no
```

```
Oh well, better luck next time
```

```
Play again? no
```

```
>>
```



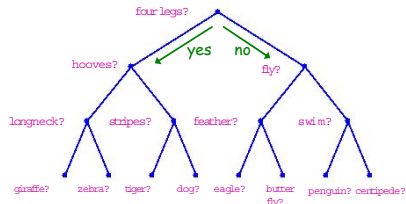
Trees 21-16

## Game tree for twenty questions

```

twenty = {'four legs?'
         {'hooves?' 'long neck?' {'giraffe?' {} {} } ...
         {'stripes?' 'tiger?' {} {} } ...
         {'fly?' 'feathers?' 'eagle?' {} {} } ...
         {'swim?' 'penguin?' {} {} } ...
         {'centipede?' {} {} {} } };

```



Trees 21-17

## TwentyQuests

```

function twentyQ (questTree)
% recursive function that implements the twenty questions game
reply = input(questTree{1}, 's'); % print question in root node and
if isLeaf(questTree) % input user's reply (yes or no)
    if strcmp(reply, 'yes') % leaf node is reached, so
        disp('Wow! I won!'); % determine outcome of game
    else
        disp('Oh well, better luck next time');
    end
else % leaf node not yet reached
    if strcmp(reply, 'yes') % if user replied yes, follow left subtree
        twentyQ(questTree(2))
    else % otherwise follow right subtree
        twentyQ(questTree(3))
    end
end

```

Trees 21-18

## TwentyQuests

```

function twentyQuests (questTree)
% plays the twenty questions game, using a binary tree that
% contains questions in the nodes
playAgain = 1;
while playAgain % keeps going as long as user answers yes to
    twentyQ(questTree) % Play again? question
    playAgain = strcmp(input('Play again? ', 's'), 'yes');
end

```

Trees 21-19