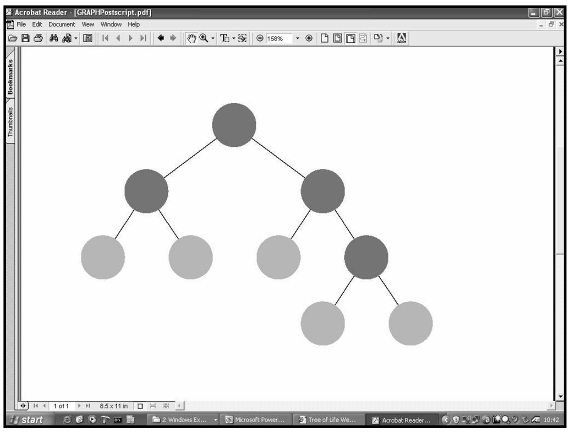
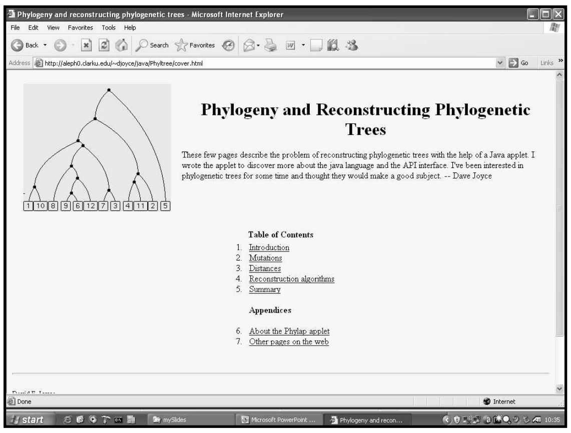
Trees
Rosen Chapter 9
(page 631 onwards)




Phylogeny and reconstructing phylogenetic trees - Microsoft Internet Explorer

http://aleph0.clarku.edu/~djoyce/java/Phyltree/cover.html

Phylogeny and Reconstructing Phylogenetic Trees

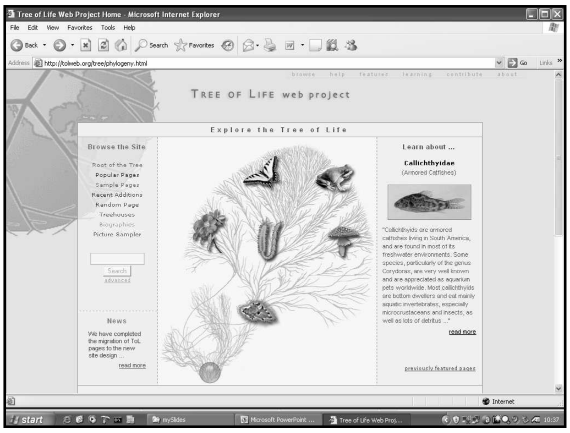
These few pages describe the problem of reconstructing phylogenetic trees with the help of a Java applet. I wrote the applet to discover more about the java language and the API interface. I've been interested in phylogenetic trees for some time and thought they would make a good subject. -- Dave Joyce

Table of Contents

1. Introduction
2. Mutations
3. Distances
4. Reconstruction algorithms
5. Summary

Appendices

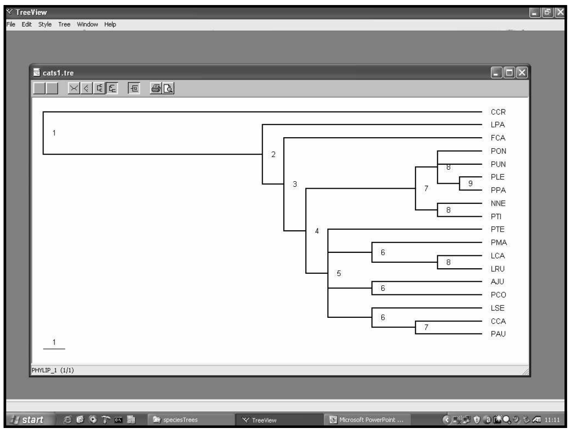
6. About the Phylap applet
7. Other pages on the web

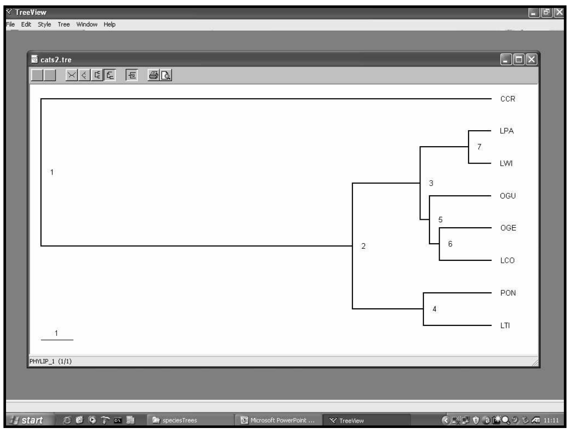


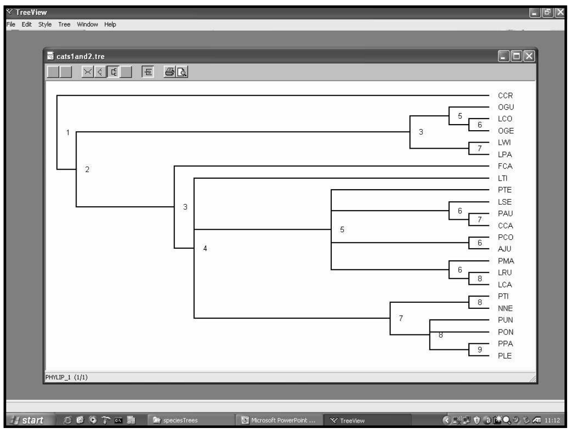
**Callichthyidae**
(Armored Catfishes)

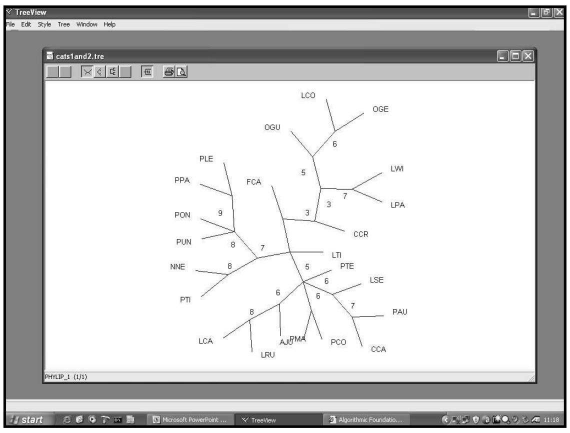
"Callichthyids are armored catfishes living in South America, and are found in most of its freshwater environments. Some species, particularly of the genus Corydoras, are very well known and are appreciated as aquarium pets worldwide. Most callichthyids are bottom dwellers and eat mainly aquatic invertebrates, especially microcrustaceans and insects, as well as lots of detritus ..."

We have completed the migration of ToL pages to the new site design ...









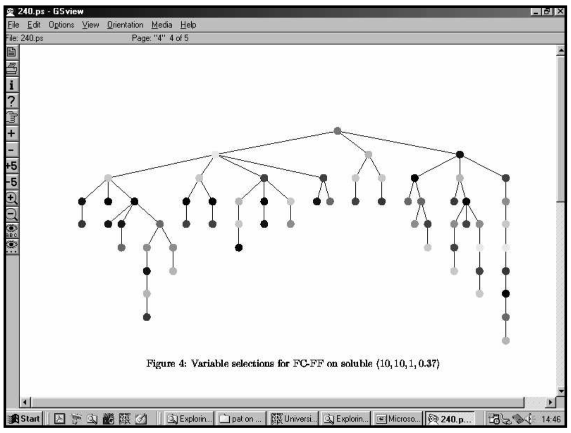
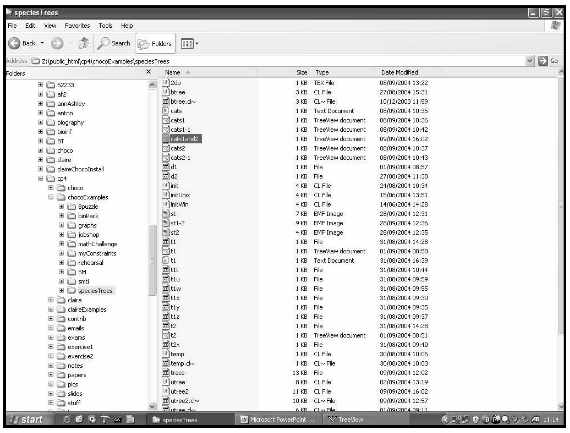


Figure 4: Variable selections for FC-FF on soluble (10, 10, 1, 0.37)

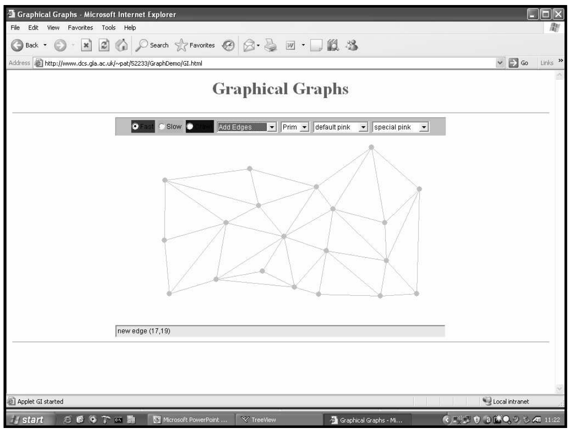


speciesTrees

Address: Z:\public_html\cp4\chocoExamples\speciesTrees

```
Folders
  52233
  af2
  annAshley
  anton
  biography
  bioinf
  BT
  choco
  claire
  claireChocoInstall
  cp4
    choco
    chocoExamples
      8puzzle
      binPack
      graphs
      jobshop
      mathChallenge
      myConstraints
      rehearsal
      SM
      smti
      speciesTrees
    claire
    claireExamples
    contrib
    emails
    exams
    exercise1
    exercise2
    notes
    papers
    pics
    slides
    stuff
```

| Name | Size | Type | Date Modified |
|---|---|---|---|
| 2do | 1 KB | TEX File | 08/09/2004 13:22 |
| btree | 3 KB | CL File | 27/08/2004 15:31 |
| btree.cl~ | 3 KB | CL~ File | 10/12/2003 11:59 |
| cats | 1 KB | Text Document | 08/09/2004 10:35 |
| cats1 | 1 KB | TreeView document | 08/09/2004 10:36 |
| cats1-1 | 1 KB | TreeView document | 08/09/2004 10:42 |
| cats1and2 | 1 KB | TreeView document | 09/09/2004 16:02 |
| cats2 | 1 KB | TreeView document | 08/09/2004 10:37 |
| cats2-1 | 1 KB | TreeView document | 08/09/2004 10:43 |
| d1 | 1 KB | File | 01/09/2004 08:57 |
| d2 | 1 KB | File | 27/08/2004 11:30 |
| init | 4 KB | CL File | 24/08/2004 10:34 |
| initUnix | 4 KB | CL File | 15/06/2004 13:51 |
| initWin | 4 KB | CL File | 14/06/2004 14:28 |
| st | 7 KB | EMF Image | 28/09/2004 12:31 |
| st1-2 | 9 KB | EMF Image | 28/09/2004 12:36 |
| st2 | 4 KB | EMF Image | 28/09/2004 12:35 |
| t1 | 1 KB | File | 31/08/2004 14:28 |
| t1 | 1 KB | TreeView document | 01/09/2004 08:50 |
| t1 | 1 KB | Text Document | 31/08/2004 16:39 |
| t1t | 1 KB | File | 31/08/2004 10:44 |
| t1u | 1 KB | File | 31/08/2004 09:59 |
| t1w | 1 KB | File | 31/08/2004 09:55 |
| t1x | 1 KB | File | 31/08/2004 09:30 |
| t1y | 1 KB | File | 31/08/2004 09:35 |
| t1z | 1 KB | File | 31/08/2004 09:37 |
| t2 | 1 KB | File | 31/08/2004 14:28 |
| t2 | 1 KB | TreeView document | 01/09/2004 08:51 |
| t2x | 1 KB | File | 31/08/2004 09:40 |
| temp | 1 KB | CL File | 30/08/2004 10:05 |
| temp.cl~ | 1 KB | CL~ File | 30/08/2004 10:03 |
| trace | 13 KB | File | 09/09/2004 12:02 |
| utree | 8 KB | CL File | 02/09/2004 13:19 |
| utree2 | 11 KB | CL File | 09/09/2004 16:02 |
| utree2.cl~ | 10 KB | CL~ File | 09/09/2004 12:57 |



Graphical Graphs - Microsoft Internet Explorer

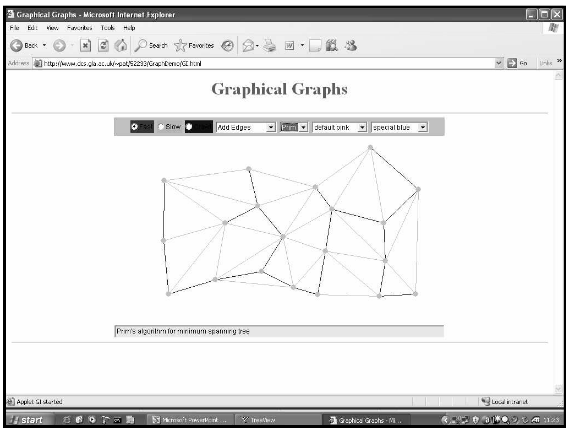
Address: http://www.dcs.gla.ac.uk/~pat/52233/GraphDemo/GI.html

Graphical Graphs

Slow | Add Edges | Prim | default pink | special pink

new edge (17,19)

Applet GI started



Graphical Graphs - Microsoft Internet Explorer

Address: http://www.dcs.gla.ac.uk/~pat/52233/GraphDemo/GI.html

Graphical Graphs

Fast | Slow | Add Edges | Prim | default pink | special blue

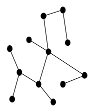
Prim's algorithm for minimum spanning tree

Applet GI started

Where else might we see trees?

1. Table of contents of a book (chapters, sections, subsections, …)
2. Organisational charts (boss is at the top)
3. Decision procedures (Hayne's manual for repairing a car)
4. The local sewage system
5. As a data structure (for storing information)
6. As an ephemeral structure, as in combinatorial search (backtracking)
7. Your family tree.
8. …

## So, what is a tree?



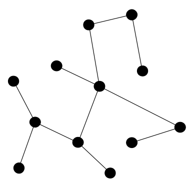
A tree is a connected undirected graph with no simple circuits

A tree is a connected graph with n-1 edges

A tree is a graph such that there is a unique simple path between any pair of vertices

All of the above say the same thing!

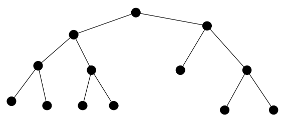
## An unrooted tree



Kinds of nodes/vertices in a tree

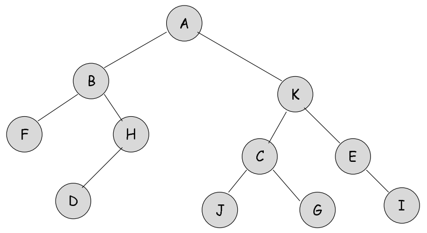
• the root (this tree doesn't have one)
• leaf nodes (there are 6 here)
• interior nodes (there are 5 here)

## A Rooted tree



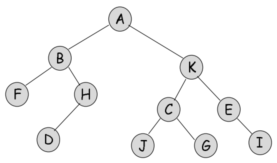
A rooted tree has one vertex designated as the root and every other edge is directed away from the root

The above tree is a binary tree
We put the root at the top by convention



The parent of H is B
The sibling of G is J
The ancestors of I are E, K, and A
C is a child of K
The descendants of B are F, H, and D
A is the root, and has no ancestors
The leaf nodes have no children
Again, the tree is a binary tree

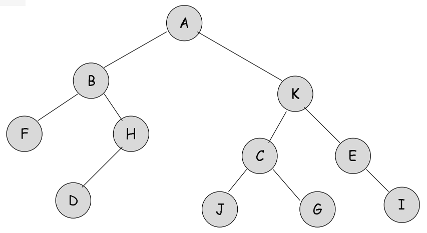
## The height of a binary tree



- The height of a leaf node is 0
- The height of an empty tree is 0
- The height of a node x is 1 + max(height(left(x)),height(right(x)))

Note: I've assumed that we have functions left, right, isNode, and isLeaf

## Traversals

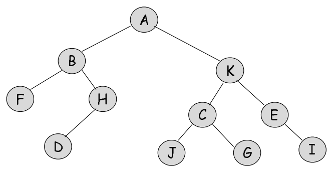


If you've got some structure one of the 1st things you want to be able to do is to traverse it!

Again, we'll stick to rooted binary trees

We have 3 traversals
1. preorder
2. inorder
3. postorder

## Traversals



```
preorder(x)
 if isNode(x)
 then print(x),
        preorder(left(x)),
        preorder(right(x))
```

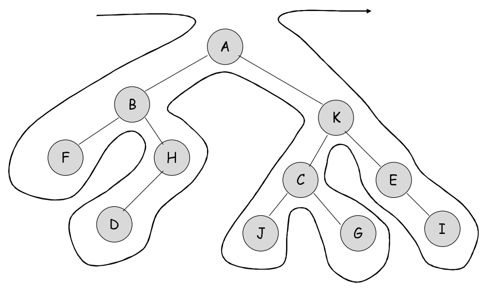
A,B,F,H,D,K,C,J,G,E,I

```
inorder(x)
 if isNode(x)
 then inorder(left(x)),
        print(x),
        inorder(right(x))
```

F,B,D,H,A,J,C,G,K,E,I

```
postorder(x)
 if isNode(x)
 then print(x),
        postorder(left(x)),
        postorder(right(x))
```

F,D,H,B,J,G,C,I,E,K,A



A walk round the tree
- if we "print" on 1st visit we get preorder
- if we "print" on 2nd visit we get inorder
- if we "print" on last visit we get postorder

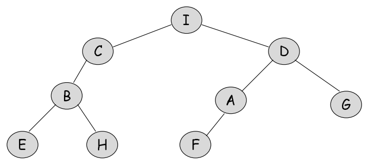
Determine the tree from its traversals

1. Preorder: ICBEHDAFG
2. Inorder: EBHCIFADG
3. Postorder: EHBCFAGDI

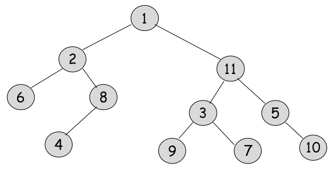
- (a) I is the root (from 1)
- (b) E, B, H, and C are to the left of I (from (a) and 2)
- (c) F, A, D, and G are to the right of I (from (a) 2)
- (d) C is the first left node of I (from (c) and 1)
- (e) D is the first right node of I (from (c) and 1)
- (f) possibly we have
  - B to the left of C,
  - E to the left of B,
  - H to the right of B … as this satisfies 1 and 2
- (g) F and A are left of D, and G is right of D (from 2)
- (h) F must be left of A (from 1)
- (j) the tree is now fully defined

Determine the tree from its traversals

1. Preorder: ICBEHDAFG
2. Inorder: EBHCIFADG
3. Postorder: EHBCFAGDI

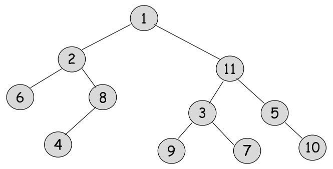


How would you represent a tree in a computer?



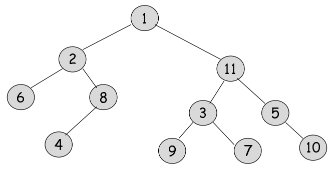
Might have a btree data structure with attributes

- data
  - the actual information in a node
- left
  - the btree to the left, or nil
- right
  - the btree to the right, or nil



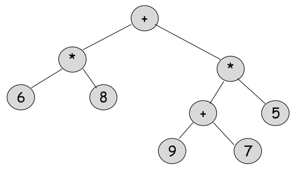
|  | *data* | *left* | *right* |
|---|---|---|---|
| 1 | 1 | 2 | 11 |
| 2 | 2 | 6 | 8 |
| 3 | 3 | 9 | 7 |
| 4 | 4 | −1 | −1 |
| 5 | 5 | −1 | 10 |
| 6 | 6 | −1 | −1 |
| 7 | 7 | −1 | −1 |
| 8 | 8 | 4 | −1 |
| 9 | 9 | −1 | −1 |
| 10 | 10 | −1 | −1 |
| 11 | 11 | 3 | 5 |

Might use a 2d array



Might use a 1d array , giving parent of a node

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|
| −1 | 1 | 11 | 8 | 11 | 2 | 3 | 2 | 3 | 5 | 1 |



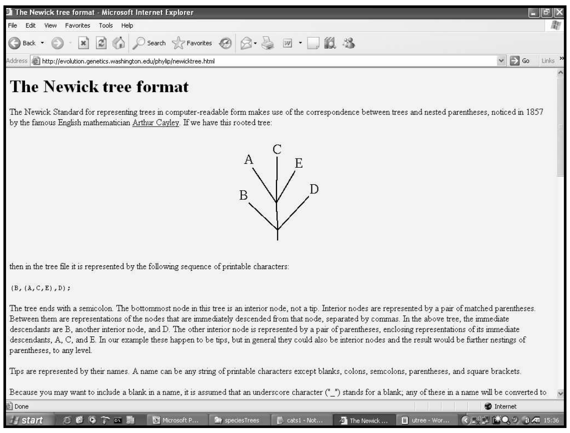
An expression

(6 * 8) + ((9 + 7) * 5)

What would a preorder, inorder and postorder traversal of this tree looklike?

Standard format for trees

- Cayley Notation
- Newick Notation



# The Newick tree format

The Newick Standard for representing trees in computer-readable form makes use of the correspondence between trees and nested parentheses, noticed in 1857 by the famous English mathematician Arthur Cayley. If we have this rooted tree:

then in the tree file it is represented by the following sequence of printable characters:

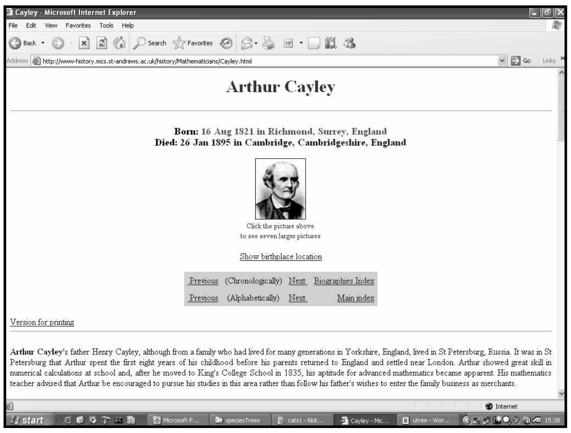
```
(B,(A,C,E),D);
```

The tree ends with a semicolon. The bottommost node in this tree is an interior node, not a tip. Interior nodes are represented by a pair of matched parentheses. Between them are representations of the nodes that are immediately descended from that node, separated by commas. In the above tree, the immediate descendants are B, another interior node, and D. The other interior node is represented by a pair of parentheses, enclosing representations of its immediate descendants, A, C, and E. In our example these happen to be tips, but in general they could also be interior nodes and the result would be further nestings of parentheses, to any level.

Tips are represented by their names. A name can be any string of printable characters except blanks, colons, semcolons, parentheses, and square brackets.

Because you may want to include a blank in a name, it is assumed that an underscore character ("_") stands for a blank; any of these in a name will be converted to

The Newick tree format - Microsoft Internet Explorer
http://evolution.genetics.washington.edu/phylip/newicktree.html
A
B
C
D
E



## Arthur Cayley

**Born: 16 Aug 1821 in Richmond, Surrey, England**
**Died: 26 Jan 1895 in Cambridge, Cambridgeshire, England**

Click the picture above
to see seven larger pictures

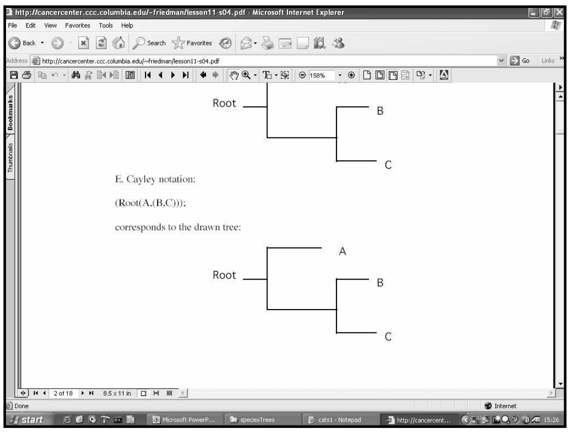
Show birthplace location

Previous (Chronologically) Next Biographies Index
Previous (Alphabetically) Next Main index

Version for printing

**Arthur Cayley**'s father Henry Cayley, although from a family who had lived for many generations in Yorkshire, England, lived in St Petersburg, Russia. It was in St Petersburg that Arthur spent the first eight years of his childhood before his parents returned to England and settled near London. Arthur showed great skill in numerical calculations at school and, after he moved to King's College School in 1835, his aptitude for advanced mathematics became apparent. His mathematics teacher advised that Arthur be encouraged to pursue his studies in this area rather than follow his father's wishes to enter the family business as merchants.

Cayley - Microsoft Internet Explorer
http://www-history.mcs.st-andrews.ac.uk/history/Mathematicians/Cayley.html

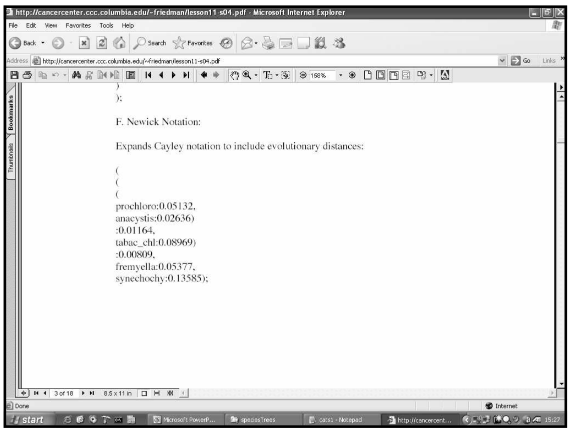


E. Cayley notation:

(Root(A,(B,C)));

corresponds to the drawn tree:

http://cancercenter.ccc.columbia.edu/~friedman/lesson11-s04.pdf - Microsoft Internet Explorer
Root
B
C
A
Root
B
C



```
)
);
```

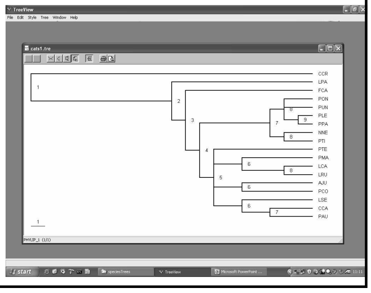
F. Newick Notation:

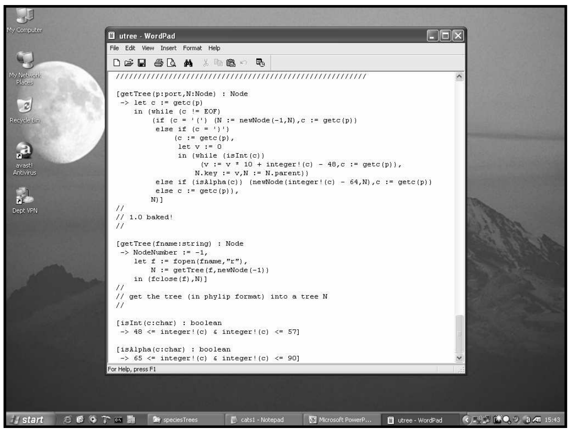
Expands Cayley notation to include evolutionary distances:

```
(
(
(
prochloro:0.05132,
anacystis:0.02636)
:0.01164,
tabac_chl:0.08969)
:0.00809,
fremyella:0.05377,
synechochy:0.13585);
```

http://cancercenter.ccc.columbia.edu/~friedman/lesson11-s04.pdf - Microsoft Internet Explorer

(((((((PLE:1,PPA:1)9:1,PON:2,PUN:2)8:1,(
NNE:2,PTI:2)8:1)7:5,
(((LCA:2,LRU:2)8:3,PMA:5)6:2,PTE:7,(AJ
U:5,PCO:5)6:2,
(LSE:5,(CCA:3,PAU:3)7:2)6:2)5:1)4:1,FC
A:9)3:1,LPA:10)2:10,CCR:20)1;





```
//////////////////////////////////////////////////////////////

[getTree(p:port,N:Node) : Node
 -> let c := getc(p)
    in (while (c != EOF)
        (if (c = '(') (N := newNode(-1,N),c := getc(p))
         else if (c = ')')
              (c := getc(p),
               let v := 0
               in (while (isInt(c))
                     (v := v * 10 + integer!(c) - 48,c := getc(p)),
                   N.key := v,N := N.parent))
         else if (isAlpha(c)) (newNode(integer!(c) - 64,N),c := getc(p))
         else c := getc(p)),
        N)]
//
// 1.0 baked!
//

[getTree(fname:string) : Node
 -> NodeNumber := -1,
    let f := fopen(fname,"r"),
        N := getTree(f,newNode(-1))
    in (fclose(f),N)]
//
// get the tree (in phylip format) into a tree N
//

[isInt(c:char) : boolean
 -> 48 <= integer!(c) & integer!(c) <= 57]

[isAlpha(c:char) : boolean
 -> 65 <= integer!(c) & integer!(c) <= 90]
```

Trees are beautiful recursive structures
There are many tree-based data structures
There are many algorithms for trees
There is a lot to learn about trees
There is a lot of research going on about trees