CS 5: Trees Revisited!
True Story...
Binary search trees (BSTs)

Set ADT:
- insert(x)
- find(x)
- delete(x)

<table>
<thead>
<tr>
<th></th>
<th>insert</th>
<th>delete</th>
<th>find</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dictionary</td>
<td>O(n)</td>
<td>O(n)</td>
<td>O(n)</td>
</tr>
<tr>
<td>Array</td>
<td>O(n)</td>
<td>O(n)</td>
<td>O(n)</td>
</tr>
<tr>
<td>Sorted array</td>
<td>O(n)</td>
<td>O(n)</td>
<td>O(log n)</td>
</tr>
<tr>
<td>Linked list</td>
<td>O(1)</td>
<td>O(n)</td>
<td>O(n)</td>
</tr>
<tr>
<td>Sorted linked list</td>
<td>O(n)</td>
<td>O(n)</td>
<td>O(n)</td>
</tr>
<tr>
<td>BST</td>
<td>“good”</td>
<td>“good”</td>
<td>“good”</td>
</tr>
</tbody>
</table>

“good” seems pretty imprecise!
Assume for now that there are no duplicate elements!

What’s “good” about this?
>>> tree = None
>>> tree = insert(42, tree)
>>> tree
(42, None, None)
>>> tree = insert(47, tree)
>>> tree
(42, None, (47, None, None))
>>> find(27, tree)
False
>>> find(47, tree)
True
>>> inorder(tree)
[42, 47]
inorder(tree)

def inorder(tree):
    if tree == None: return []
    else:
def inorder(tree):
    if tree == None:
        return []
    else:
        root, left, right = tree
        return inorder(left) + [root] + inorder(right)
def find(x, tree):
    if tree == None: return False
    else:
        # Insert code here
def find(x, tree):
    if tree == None: return False
    else:
        root, left, right = tree
        if x == root: return True
        elif x < root: return find(x, left)
        else: return find(x, right)
def insert(x, tree):
    if tree == None: return (x, None, None)
    else:
        root, left, right = tree
        if x == root:  # Already in tree, no need to insert
            return tree
        return (x, left, right)
insert(num, tree)

def insert(x, tree):
    if tree == None: return (x, None, None)
    else:
        root, left, right = tree
        if x == root: # Already in tree, no need to insert
            return tree
        elif x < root:
            return (root, insert(x, left), right)
        else:
            return (root, left, insert(x, right))

This will be useful when we save the tree to a file.
Binary search trees (BSTs)

**Set ADT:**
- `insert(x)`
- `find(x)`
- `delete(x)`

<table>
<thead>
<tr>
<th></th>
<th><code>insert</code></th>
<th><code>delete</code></th>
<th><code>find</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Dictionary</td>
<td>O(n)</td>
<td>O(n)</td>
<td>O(n)</td>
</tr>
<tr>
<td>Array</td>
<td>O(n)</td>
<td>O(n)</td>
<td>O(n)</td>
</tr>
<tr>
<td>Sorted array</td>
<td>O(n)</td>
<td>O(n)</td>
<td>O(log n)</td>
</tr>
<tr>
<td>Linked list</td>
<td>O(1)</td>
<td>O(n)</td>
<td>O(n)</td>
</tr>
<tr>
<td>Sorted linked list</td>
<td>O(n)</td>
<td>O(n)</td>
<td>O(n)</td>
</tr>
<tr>
<td>BST</td>
<td>“good”</td>
<td>“good”</td>
<td>“good”</td>
</tr>
</tbody>
</table>

Demo self-balancing trees!
Twenty Questions Game!

• Demo!
• Playing the game gives back a new tree!
• How do we save the tree?
• How do we restore the tree? (Optional, but recommended!)

(“Is it bigger than a breadbox?”,
 (“an elephant”, None, None),
 (“a mouse”, None, None)
)
def play(tree):
    if leaf(tree):
        return
    playLeaf(tree)
    else:
        root, yesChild, noChild = tree
        answer = input(root + " ")
        if yes(answer):

        else:

        tree = ...
while(...):
    tree = play(tree)
def play(tree):
    if leaf(tree):
        return
    playLeaf(tree)
    else:
        root, yesChild, noChild = tree
        answer = input(root + " ")
        if yes(answer):
            return (root, play(yesChild), noChild)
        else:
            return (root, yesChild, play(noChild))
Saving a tree to a file!

• What does the file look like?

(Is it bigger than a breadbox?,
(“an elephant”, None, None),
(“a mouse”, None, None))
inputList = [A, Internal node, B, Internal node, C, Leaf, D, Leaf, E, Leaf]

A Is it bigger than a breadbox?
B Does it wear a plastic pocket protector?
C professor
D elephant
E mouse

buildTree

Root= A

buildTree

Root= B

buildTree

Root= C

buildTree

Root= D

buildTree

Root= E
inputList = [A, Internal node, B, Internal node, C, Leaf, D, Leaf, E, Leaf]

buildTree

[A Is it bigger than a breadbox? B Does it wear a plastic pocket protector? C professor D elephant E mouse]

Root= A

buildTree

[B, Internal node, C, Leaf, D, Leaf, E, Leaf]

buildTree

[Root= B

C]

buildTree

[D, Leaf, E, Leaf]

buildTree

[E, Leaf]
inputList = [A, Internal node, B, Internal node, C, Leaf, D, Leaf, E, Leaf]
**A** Is it bigger than a breadbox?  
**B** Does it wear a plastic pocket protector?  
**C** professor  
**D** elephant  
**E** mouse

inputList = [A, Internal node,  
B, Internal node,  
C, Leaf,  
D, Leaf,  
E, Leaf]

```
buildTree
```

```
A

buildTree
```

```
B

buildTree
```

```
C

buildTree
```

```
D

buildTree
```

```
E

buildTree
```

```
Root= A

buildTree
```

```
[ B, Internal node,  
C, Leaf,  
D, Leaf,  
E, Leaf]

buildTree
```

```
[ C, Leaf,  
D, Leaf,  
E, Leaf]

buildTree
```

```
[ E, Leaf]

buildTree
```

```
[ ]
```

A Is it bigger than a breadbox?  
B Does it wear a plastic pocket protector?  
C professor  
D elephant  
E mouse  
Leaf
inputList = [A, Internal node, B, Internal node, C, Leaf, D, Leaf, E, Leaf]

buildTree
A

root = A

buildTree

[B, Internal node, C, Leaf, D, Leaf, E, Leaf]

buildTree

[C, Leaf, D, Leaf, E, Leaf]

buildTree

[E, Leaf]

buildTree

[E, Leaf]
inputList = [A, Internal node, B, Internal node, C, Leaf, D, Leaf, E, Leaf]

buildTree

[B, Internal node, C, Leaf, D, Leaf, E, Leaf]

buildTree

[C, Leaf, D, Leaf, E, Leaf]

buildTree

The empty residue list is the mark of success!
What you’ll submit...

• Your code, as usual, on Gradescope
• A .txt file with a game tree that you share with the class at...

https://tinyurl.com/y9nx68na
How many species are known?

- Number of known species
  - 6K mammals
  - 10K birds
  - 10K reptiles
  - 7K amphibians
  - 30K fishes
  - 1.3M invertebrates

J.B.S. Haldane
1892-1964

“An inordinate fondness for...”
400,000 Named Beetle Species!

http://evolution.berkeley.edu/evosite/evo101/VIIB1bBeetles.shtml
"Inordinate Fondness"
Explained: Why Are There So Many Beetles?
Brian D. Farrell

The phylogeny of the Phytophaga, the largest and oldest radiation of herbivorous beetles, was reconstructed from 115 complete DNA sequences for the 18S nuclear ribosomal subunit and from 212 morphological characters. The results of these analyses were used to interpret the role of angiosperms in beetle diversification. Jurassic fossils represent basal lineages that are still associated with conifers and cycads. Repeated origins of angiosperm-feeding beetle lineages are associated with enhanced rates of beetle diversification, indicating a series of adaptive radiations. Collectively, these radiations represent nearly half of the species in the order Coleoptera and a similar proportion of herbivorous insect species.

When the British biologist J. B. S. Haldane was asked by a group of theologians what one could conclude as to the nature of the Creator from a study of His creation, Haldane is said to have answered, “An inordinate fondness for beetles” (1). Haldane’s remark reflects the numerical domination of described species by the insect order Coleoptera (2), the diversity of which exceeds that of any other known animal or plant group. Because over half of all beetles are herbivorous and because the diversity of the remainder is comparable to that of other large, young, and nonherbivorous insect orders (3), a reconstruction of the phylogenesis of beetle herbivory would contribute substantially to an understanding of...
Coevolution

“I can understand how a flower and a bee might slowly become, either simultaneously or one after the other, modified and adapted in the most perfect manner to each other, by the continued preservation of individuals presenting mutual and slightly favourable deviations of structure.”

Charles Darwin, The Origin of Species
Coevolution

- Mutualism
  - Flower + Bee +
- Parasitism
  - Host + Parasite -

- Two species exert selective pressure on each other and
- Evolve in response to each other
Phylogenetic Trees

Input: DNA or protein sequences for each species

Output: Species tree

**Spongiforma squarepantsii**, a new species of gasteroid bolete from Borneo

INTRODUCTION

An unusual sponge-shaped (cf. *Spongia*, Porifera), terrestrial fungus was encountered by Peay et al. (2010) during a recent study of ectomycorrhizal (ECM) community structure in the dipterocarp-dominated forest of the Lambir Hills in Sarawak, Malaysia. The form of the sporocarp was unusual enough that before microscopic examination the collectors were uncertain whether the fungus was a member of the Ascomycota or the Basidiomycota. However, on laboratory examination it was recognized as a species of the new genus *Spongiforma* Desjardin, Manf. Binder, Roekring & Flegel
Fig-Wasp Mutualism

From A. P. Jackson, “Cophylogeny of the Ficus Microcosm,” Biological Reviews, 79,
Gopher Louse Parasitism

Indigobirds Parasitize Firefinches

- High level of host specificity (e.g. mouth markings)
Two Reconstructions for Gopher/Lice

Figure 13: The optimal reconstructions found by Page (a) and Ronquist (b).
This problem is hard!

• How hard? NP-hard! (Harvey Mudd group 2009)
The Biological Challenge
Results
Evidence for Coevolution!

Randomly Generated Problem Instances

Original Problem Instance

Solve for optimal cost

Number of samples

Cost of Ficus / Agaonidae mapping

Cost of sample

370 380 390 400 410 420 430 440 450 460 470 480 490 500 510 520 530 540 550 560 570 580 590

0 5 10 15 20 25 30 35 40 45
A hitchhikers guide to the Galápagos: co-phylogeography of Galápagos mockingbirds and their parasites

Jan Štefka¹,²*, Paquita EA Hoeck³, Lukas F Keller³ and Vincent S Smith¹

Abstract

Background: Parasites are evolutionary hitchhikers whose phylogenies often track the evolutionary history of their hosts. Incongruence in the evolutionary history of closely associated lineages can be explained through a variety of...
Jane 3 is Popular!

About 977,000,000 results (0.25 seconds)
The “great?” xscape

Costscape: Ficus-CeratosolenCosts.pdf

- <11, 3, 1, 13> Count = 2
- <11, 0, 4, 6> Count = 2
- <10, 0, 5, 3> Count = 10
- <9, 0, 6, 1> Count = 8
- <8, 0, 7, 0> Count = 16
- <11, 1, 3, 7> Count = 2
- <11, 2, 2, 9> Count = 2
- <11, 4, 0, 17> Count = 1
Recent work by your classmates...

Welcome to DTL-RnB

What is DTL-RnB?

DTL-RnB (Reconciliation Browser) is a tool for browsing the potentially large landscape of maximum parsimony reconciliations (i.e., gene trees or host and parasite trees). This tool is based on algorithms and techniques described in the paper “DTL-RnB: A Flexible Browser of DTL Reconciliations” by Weiyun Ma, Dmitriy Smirnov, Juliet Forman, Annalise Schweickart, Carter Slocum, Srinidhi Srinivasan.
DTL-RnB: Algorithms and Tools for Summarizing the Space of DTL Reconciliations

W. Ma* D. Smirnov† J. Forman* A. Schweickart* C. Slocum‡ S. Srinivasan* R. Libeskind-Hadas*
*Harvey Mudd College, Claremont, California, USA †Pomona College, Claremont, California, USA
‡California Polytechnic University, Pomona, California, USA

Abstract—Phylogenetic tree reconciliation is an important technique for reconstructing the evolutionary histories of species and genes and other dependent entities. Reconciliation is typically performed in a maximum parsimony framework and the number of optimal reconciliations can grow exponentially with the size of the trees, making it difficult to understand the solution space. This paper demonstrates how a small number of reconciliations can be found that collectively contain the most highly supported events in the solution space. While we show that the
Questions, Comments...