Set 06: Trees

CS240: Data Structures and Data Management

Jérémy Barbay

Outline

Tree ADT

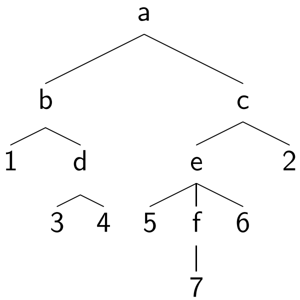
- Operations
  - `root()`, `size()`
  - `isInternal( node )`, `children( node )`, `parent( node )`
  - `attachSubtree( node, tree )`, `detachSubtree( node )`
- Use trees to implement other ADTs

Definitions

- Recursive: A finite collection of nodes (at least one) that is
  1. A single distinguished node called the root or
  2. Partitioned into $k+1$ subcollections: a designated root node connected together with k trees, $T_1 \dots T_k$, by an edge
- Graph which is
- List of nodes and oriented edges s.t.

Terminology

- parent, child, sibling, subtree
- ancestor, descendent
  (**note**: a node is its own ancestor and descendent)
- external node (leaf)
- internal node



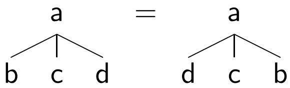
Applications

Depth/Height

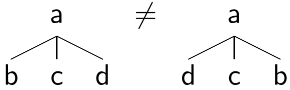
- Node Depth – The number of edges between the node, and the root of the entire tree
- Node Height – The maximum number of edges between the node and any of its descendants
- Note:
  - $\textsc{Depth}(\mathit{root}) = 0$
  - $\textsc{Height}(\mathit{leaf}) = 0$
- **Exercise**: How do we compute each of these for a given node?

Tree variants

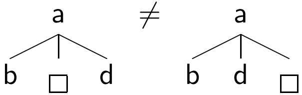
- Unordered – like a graph



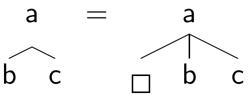
- Ordered – linear ordering on the children (first, second, . . .)



- Cardinal – children identified by their absolute position.



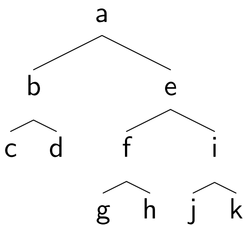
- Ordinal – children identified by their rank.



Tree variants (cont')

- Binary – each node has at most 2 children (cardinal tree)
- Proper Binary – each node has 0 or 2 children
- Full Binary – proper binary tree, all leaves at the same level

Example



This tree is:

- binary
- Proper Binary
- Full Binary

We will study more binary trees with

- Priority Queue ADT (Heaps)
- Ordered Dictionaries ADT (AVL Trees)

Binary Trees

Binary Tree Data Structures

- Linked Structure
  - Tree Node with 4 fields

<table>
<tr><td colspan="2">parent</td></tr>
<tr><td colspan="2">data</td></tr>
<tr><td>left</td><td>right</td></tr>
</table>

  - Parent is optional
- Array
  - An array of size 2^{h+1}, from 1 to 2^{h+1}.
  - children of cell i at positions $2i$ and $2i+1$.
  - Special value indicates no node.
  - More on this with `heaps`.
- There are more sophisticated ones...

Properties of Binary Trees

Theorem

Let $|E|$ and $|I|$ represent the number of external and internal nodes respectively in a proper binary tree. Then

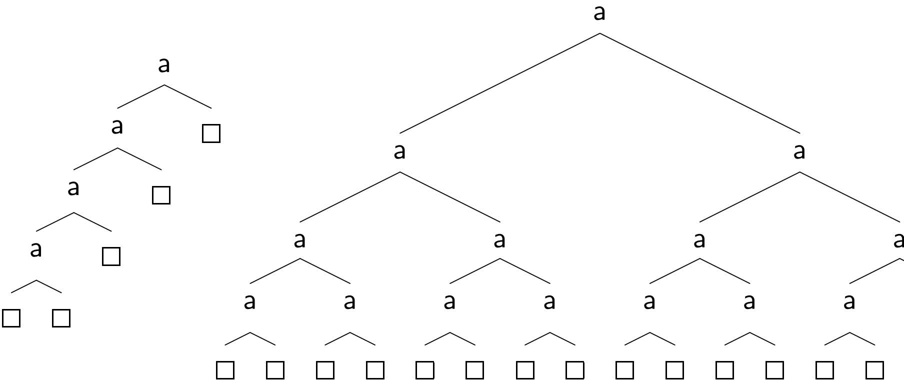
$$|E| = |I| + 1$$

Proof: By Induction
Base Case(s):
Inductive Cases:

1. Root node has one internal child
2. Root node has two internal children

More Properties of proper binary trees

- $h + 1 \leq |E| \leq 2^h$

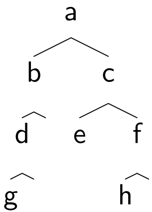


- As $n = |E| + |I|$, this gives bounds on $|I|$, n and h.

Recursive General Traversal

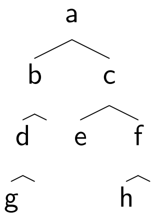
General Traverse(*node*)

Visit *node*
if *node* has left child **then**
   TRAVERSE(*node.left*)
end if
Visit *node*
if *node* has right child **then**
   TRAVERSE(*node.right*)
end if
Visit *node*



This algorithm is easily modified for other depth first traversals, or for trees of higher degree.

More specific Traversals



- Depth-First Traversal
  - General Traversal:
  - Pre-Order:
  - In-Order:
  - Post-Order:
- Breadth-First Traversal:
  - Level-Order:

Representation of a binary tree

Which traversal permit to identify a binary tree by the trace?

1. general:
2. pre-order:
3. in-order:
4. post-order:
5. breadth-first order:
6. level-order:

Summary

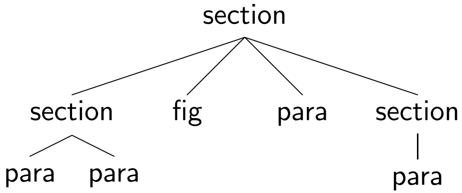
- The Tree ADT define
  - operators for navigation and construction;
  - terms: Height, Depth, ...
  - properties
  - with many variants: Cardinal/Ordinal, ...
- The Binary Tree is a particular cardinal variant, which will be studied more in details later.

Outline

Tree Encodings

Documents structured as a tree

Some trees represent static documents, which must be stored.

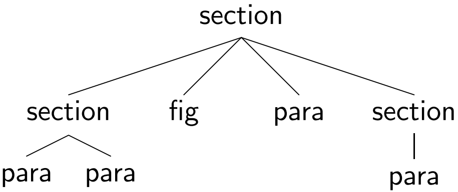


How do we encode a tree?

XML notation

How to exchange trees between applications.

```
<section>
  <section>
    <para> (...) </para>
    <para> (...) </para>
  </section>
  <fig> (...) </fig>
  <para> (...) </para>
  <section>
    <para> (...) </para>
  </section>
</section>
```



Totally specifies an ordinal tree?

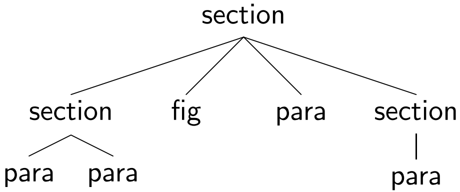
Applications?

\Tree notation

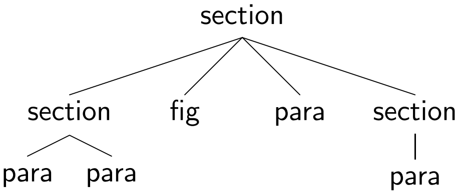
```
\Tree
[ .{section}
  [ .{section}
    {para}
    {para}
  ]
  {fig}
  {para}
  [ .{section}
    {para}
  ]
]
```

Totally specifies an ordinal tree?

Applications?

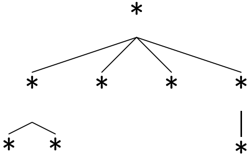


Separating structure from content



Encode separately

- the structure



- from the content *ssppfpsp*

How much space do we need to encode each part?

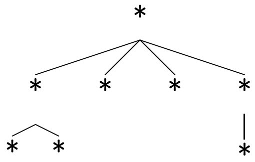
Structural Encoding of Ordinal Trees

Theorem

An ordinal tree of n nodes can be encoded in $2n$ bits.

`Transpose`(x)

print "("
for each child c of x **do**
  `Transpose`(c)
end for
print ")"



Exercises

1. How do we build the tree from the string?
2. Can we do the same for binary (cardinal) trees?

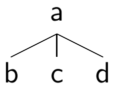
Outline

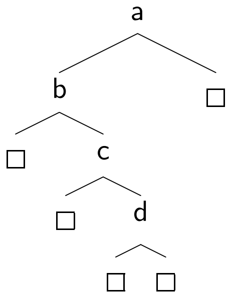
Binary Tree Representation of Ordinal Trees

Theorem

An ordinal tree T can be represented by a (cardinal) binary tree T'.

- For each internal node $v \in T$, an internal node $v' \in T'$
- If v has an immediate sibling w, then w' is the right child of v'
- If v has first child w, then w' is the left child of v'
- Fill all other spots with empty external nodes (i.e. leaves).



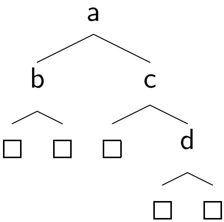


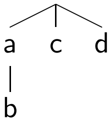
Ordinal Tree Representation of Binary Trees

Theorem

A (cardinal) binary tree T can be represented by forest of ordinal trees.

- Two-by-two correspondance between internal nodes.
- The right child of v is the sibling of v'.
- The left child of v is the first child of v'.
- Ignore empty subtrees,



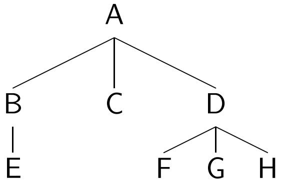


Exercise

Ordinal to Binary

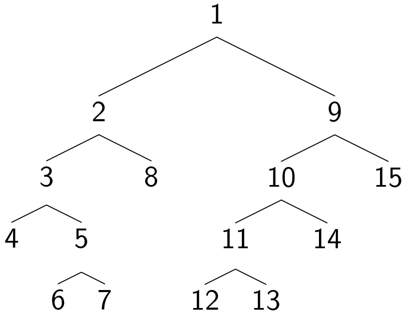
Binary-Tree Representation T':

Original Tree T:



Exercise

Binary to Ordinal



Operations on this representation

Given an ordinal tree representated in a binary tree:

1. How to compute the height?
2. How to compute the maximum degree?

Exercise:

Given a binary tree representated as a forest of ordinal trees, how to compute the height?