Announcement

• Extra credit opportunity:
  – Sorting dance
    • [https://www.youtube.com/watch?v=CjozYYZppqM](https://www.youtube.com/watch?v=CjozYYZppqM)
    • (Any sorting algorithm)
    • Will replace your worst project score
    • You need to have 10+ students
      – Multiple workshop teams together?!
    • You have to upload the video to YouTube and provide us the link by Dec 08 (include the list of students participated)
  • Bonus points for the best dance!
Extremely useful data structure
Special cases include
- Huffman tree
- Expression tree
- Decision tree (in machine learning)
- Heap data structure (later lecture)
Binary Trees

Depth 2

Height 3

Height 1

5, 4, 2, 9 are internal nodes

3, 7, 1, 9 are leaves
Ancestors and Descendants

1, 0, 4, 5 are ancestors of 1

0, 8, 1, 7 are descendants of 0
Expression Trees

\[ 4 \times (3+2) - (6-3) \times \frac{5}{3} \]
How to construct Expression Trees?

Infix

4*(3+2) – (6-3)*5/3

Postfix

4 3 2 + * 6 3 – 5 * 3 / –

Shunting Yard Algorithm
How to construct Expression Trees?

4 3 2 + * 6 3 - 5 * 3 / -
How to construct Expression Trees?

4 3 2 + * 6 3 - 5 * 3 / -
How to construct Expression Trees?

$\frac{6 \times 3}{5 - 3} \div (4 + 2)$

$\text{Stack}$
How to construct Expression Trees?

Stack
How to construct Expression Trees?

```
- 5 * 3 / -
```

```
* 6 3
```

```
4 +
```

```
3 2
```

Stack
How to construct Expression Trees?

Stack
How to construct Expression Trees?

Stack
How to construct Expression Trees?

\[
\frac{4}{3 + \frac{6 - 3}{5}}
\]
How to construct Expression Trees?
How to construct Expression Trees?

```
4 + 3
/ 5
- 2
/
3
```

```
* 6
/
3
```

```
* 3
/
3
```

```
- 2
/
5
```

```
+ 3
/
3
```

```
- 2
/
5
```

```
* 3
/
3
```

```
- 2
/
5
```

```
* 6
/
3
```

```
- 2
/
5
```

```
* 3
/
3
```

```
- 2
/
5
```

```
* 6
/
3
```

```
- 2
/
5
```

```
* 3
/
3
```

```
- 2
/
5
```
Finally!

\[ 4 \times (3+2) - (6-3) \times \frac{5}{3} \]
Character Encoding

- **UTF-8 encoding:**
  - Each character occupies 8 bits
  - For example, ‘A’ = 0x41

- A text document with $10^9$ characters is $10^9$ bytes long

- But characters were not born equal
English Character Frequencies
Variable-Length Encoding: Idea

- Encode letter E with fewer bits, say $b_E$ bits
- Letter J with many more bits, say $b_J$ bits
- We gain space if

$$b_E \cdot f_E + b_J \cdot f_J < 8f_E + 8f_J$$

where $f$ is the frequency vector

- Problem: how to decode?
One Solution: Prefix-Free Codes

1 0 1 1 1 0 1 0 0
c e b a

CSC172, Fall 2017
WHY ONLY BINARY TREE?
Any Tree can be “Encoded” as a Binary Tree
LMC-RS Representation

• In this representation, every node has two pointers:
  – LMC (Left-most-child)
  – RS (Right Sibling)
public class Node {
    public int key;
    public Node lmc, rs;

    public Node(int item)
    {
        key = item;
        lmc = rs = null;
    }
}
There are many ways to traverse a binary tree
- (reverse) In order
- (reverse) Post order
- (reverse) Pre order
- Level order = breadth first
FULL VS. COMPLETE BINARY TREE
Each node is either
(1) an internal node with exactly two non-empty children or
(2) a leaf
Complete Binary Tree
Complete Binary Tree

- has a restricted shape obtained by starting at the root and filling the tree by levels from left to right.

- In the complete binary tree of height $d$, all levels except possibly level $d-1$ are completely full.

- The bottom level has its nodes filled in from the left side.
Let’s see the examples again
Full vs. Complete Binary Tree

[Diagram showing two binary trees, one labeled "Full but not complete" and the other labeled "Complete but not full"]
TREE USING JAVA
public class Node {
    public int key;
    public Node left, right;

    public Node(int item) {
        key = item;
        left = right = null;
    }
}
Inorder Traversal

Inorder-Traverse(BTNode root)

- Inorder-Traverse(root.left)
- Visit(root)
- Inorder-Traverse(root.right)

Also called the (left, node, right) order
void inorder_print(BTNode root)
{
    if (root != null) {
        inorder_print(root.left);
        printNode(root);
        inorder_print(root.right);
    }
}
Run Time

- Suppose “visit” takes $O(1)$-time, say $c$ sec
  - $n_l = \# \text{ of nodes on the left sub-tree}$
  - $n_r = \# \text{ of nodes on the right sub-tree}$
  - Note: $n - 1 = n_l + n_r$
- $T(n) = T(n_l) + T(n_r) + c$
- Induction: $T(n) \leq cn$, i.e. $T(n) = O(n)$
- $T(n) \leq cn_l + cn_r + c$
  - $= c(n-1) + c$
  - $= cn$
Reverse Inorder Traversal

- \textbf{RevInorder-Traverse}(root.right)
- \textbf{Visit}(root)
- \textbf{RevInorder-Traverse}(root.left)

The (right, node, left) order
The other 4 traversal orders

- Preorder: \((\text{node, left, right})\)
- Reverse preorder: \((\text{node, right, left})\)
- Postorder: \((\text{left, right, node})\)
- Reverse postorder: \((\text{right, left, node})\)

We’ll talk about level-order later
What is the preorder output for this tree?

5
4
3
8
0
7
1
9
2
8
7
129
What is the postorder output for this tree?
void inorder_print(BTNode root) {
    if (root != NULL) {
        inorder_print(root.left);
        printNode(root);
        inorder_print(root.right);
    }
}

Write the above routine without the recursive calls?

Use a stack

Don’t use a stack
Reconstruct the tree from inorder+postorder

Inorder: 3 4 8 7 0 1 5 9 2
Preorder: 5 4 3 0 8 7 1 2 9
Questions to Ponder

• Can you reconstruct the tree given its postorder and preorder sequences?

• How about inorder and reverse postorder?

• How about other pairs of orders?

• How many trees are there which have the same in/post/pre-order sequence? (suppose keys are distinct)
Number of trees with a given inorder sequence

\[ C_n = \sum_{i=1}^{n} C_{i-1} C_{n-i} \]

\[ C_0 = 1 \]

\[ C_n = \frac{1}{n+1} \binom{2n}{n} \approx \frac{4^n}{n^{3/2} \sqrt{\pi}} \]

https://en.wikipedia.org/wiki/Catalan_number
What is a traversal order good for?

- Many things

- E.g., Evaluate(root) of an expression tree
  - If root is an operand, return the operand
  - Else
    - A = Evaluate(root.left)
    - B = Evaluate(root.right)
    - Return A  root.key  B
      - root.key is one of the operator

- What traversal order is the above?
How to do level-order traversal?

A (FIFO) Queue

CSC172, Fall 2017
void levelorder_print(BTNode root) {

    // Implement

}
Acknowledgement

- A lot of the slides are taken from various sources including the Internet, Dr. Hung Ngo’s slides, and various other sources.