CSSE 230 Day 19

Tree Variations:
EBTs & Tries

Reminders/Announcements

- Use the new Eclipse project for EditorTrees Milestone 2.
- **Milestone 2 grading:** You must pass *all* Milestone 1 tests in order to get any credit for Milestone 2.
- **Exam 2:** Tuesday May 8, 7:00 PM
- **Final Exam:** Tuesday May 22: 8:00 AM
- Scrabble project team preference survey: by Wednesday 5 PM
- **Consider applying to be a CSSE TA for the Fall term** (look for an email in the next week or two)
  - In the fall, we mainly hire work-study or work-opportunity students.
- Reminder: EditorTrees Milestone 2 is much more complex than Milestone 1.
  - If your team is not yet debugging "delete", you probably need to pick up the pace a bit.

What questions do you have?

Editor Trees
Anything else

Agenda

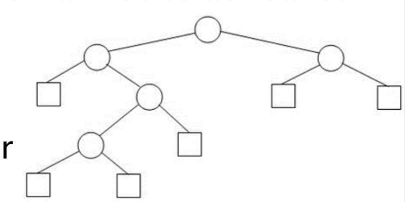
- Extended Binary Trees (EBT), including a proof by induction.
- Digital search trees (tries)
- Directed Acyclic Graphs (DAG)
- EditorTrees work time

Tree Variations

- Extended Binary trees
- Digital Search Trees
- Directed Acyclic Graphs

Extended Binary Tree (EBT) is just a different way to view binary trees: *null* external nodes as leaves

1–2



- An Extended Binary Tree is either
  - an ***external node***, or
  - an (**internal**) root node and two EBTs T_L and T_R.
- We draw internal nodes as circles and external nodes as squares
  - Generic picture and detailed picture
- EBT: An alternative way of viewing binary trees, in which the external nodes represent different "places" where an unsuccessful search can end or an element can be inserted
- Internal nodes are used (later) in calculating average time for successful search
- External nodes in calculating average time for unsuccessful search.

3-10

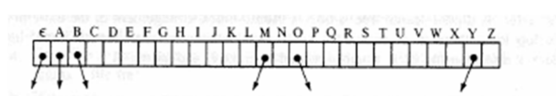
A property of EBTs

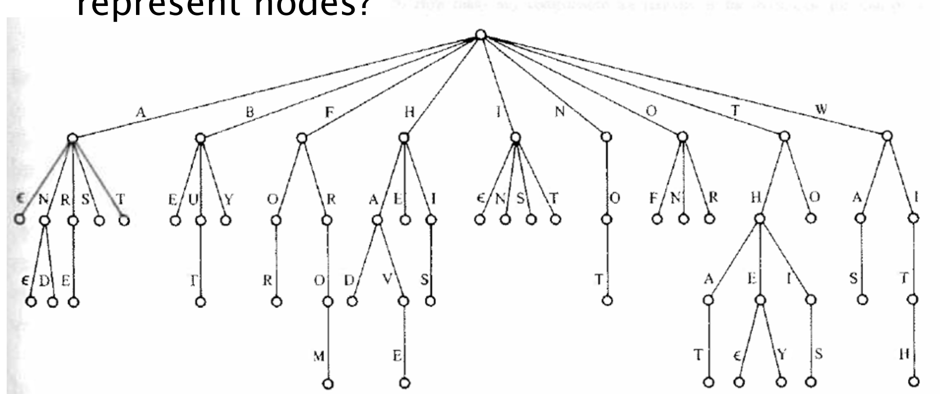
- Property P(N): For any N>=0, any EBT with N internal nodes has _______ external nodes.
- Proof by strong induction, based on the recursive definition.
  - A notation for this problem: IN(T), EN(T)
  - Note that, like a lot of other simple examples, this one can be done without induction.
  - But one purpose of this exercise is practice with strong induction, especially on binary trees.
- What is the crux of any induction proof?
  - Finding a way to relate the properties for larger values (in this case larger trees) to the property for smaller values (smaller trees). Do the proof now.

11

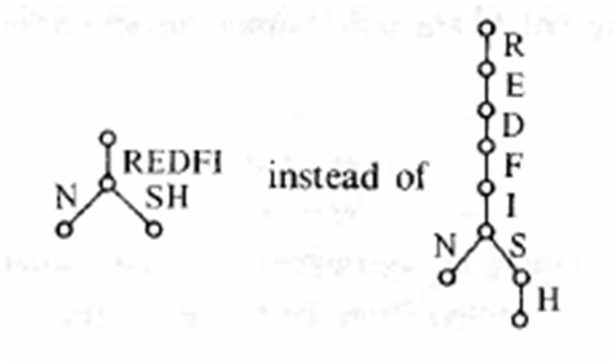
Another approach to search trees

- Digital search tree (trie).
- We store the data digit-by-digit (or letter by letter).
- How to actually represent nodes?



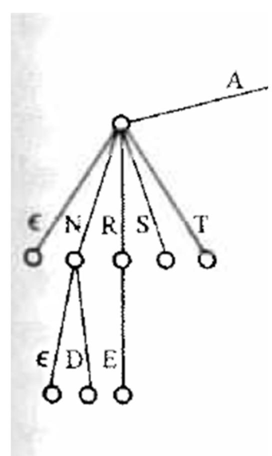


We can collapse single-branch paths to save space

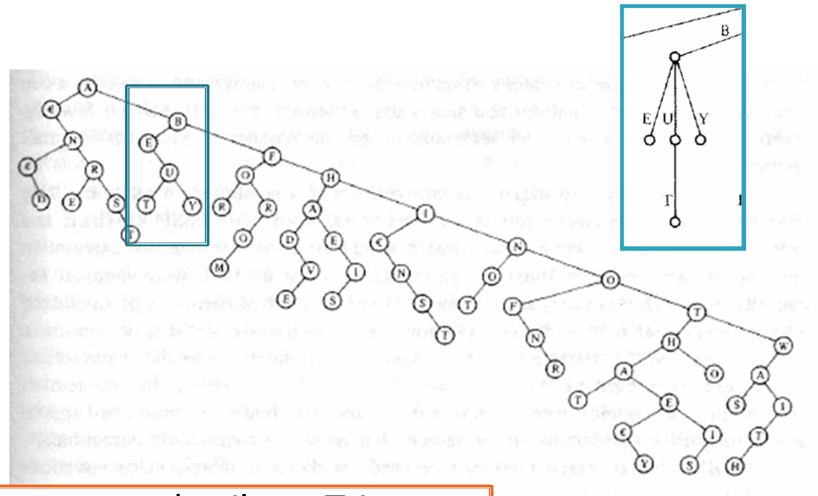


We can share a single static "ϵ-node“ to save space

- The epsilon nodes aren't null; they just show the end of a word.
- There can still be null pointers at each level where there are missing letters



Representing a Trie as a binary tree saves even more space



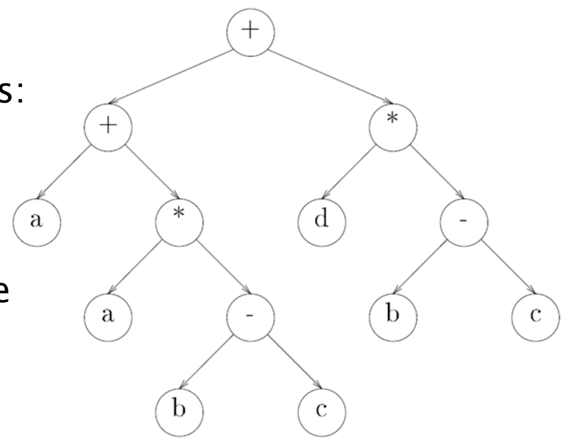
For many more details on Tries, see
http://en.wikipedia.org/wiki/Trie

You can trie to create an interesting trie using this applet

- http://blog.ivank.net/trie-in-as3.html

Expression Tree Variation

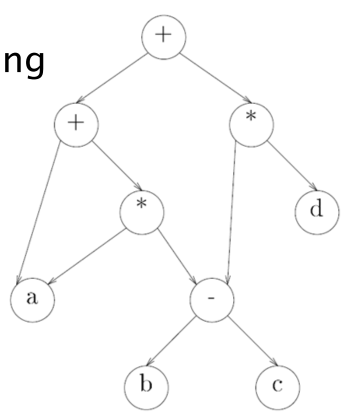
- Consider a tree that represents this expression: a + a * (b – c) + (b – c) * d
- Notice the common sub-expressions: a and (b – c)
- The value of each of those only needs to be computed once



12-15

Directed Acyclic Graph (DAG)

- A useful representation for common sub-expressions: : a + a * (b – c) + (b – c) * d
- A DAG is like a tree with sharing
  - Directed graph
  - No cycles
  - A distinguished root
  - Looks like a tree when doing a
  - traversal, but saves space.




Editor Trees
Work Time