Tries (aka Digital Trees)

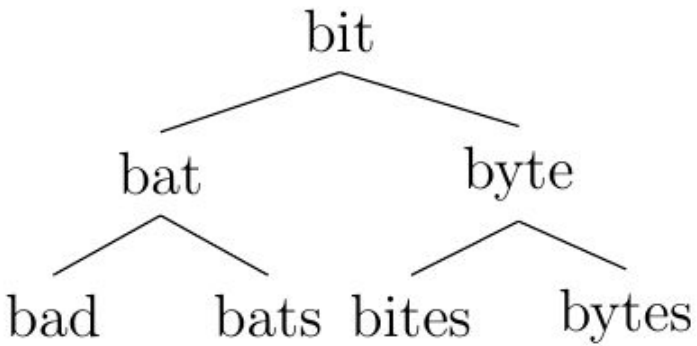
CSCI 241
Brian Hutchinson

Trie Overview

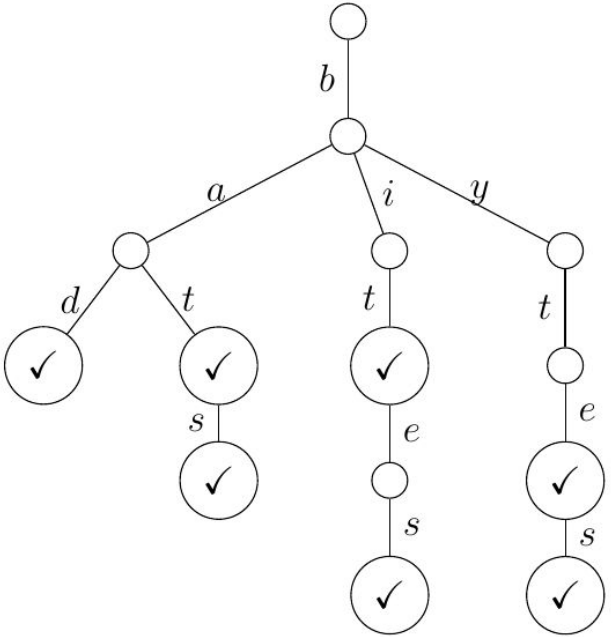
- From the word re*trie*val
  - Pronounced *tree* or *try*
- Alternative to BSTs for implementing the Set ADT and Map ADT
- Tree data structure that uses *position* rather than *comparisons*
- Assumes keys are a sequence of *symbols/characters/digits* from some *alphabet*
  - E.g., keys are W-numbers, alphabet = {0,1,2,...,9,W}
  - E.g., keys are English words without punctuation, alphabet = {a,b,...z,A,B,...Z}
- *Request: ponder parallels to the comparison sort vs radix sort trade-offs here*

BST vs Tries for Set ADT

Binary Search Tree



Trie



- Two tree data structures storing the same seven keys.
- Keys are sequences of lower case English characters.

Example TrieNode Definition

```
class TrieNode<S,V> {
        HashMap<S,TrieNode> children;  // S is symbol type; e.g., Character
        boolean terminates;            // for implementing Set ADT (checkmark)
        V value;                       // for implementing Map ADT
}
```

Example for this node:

- this.terminates == false
- this.value == null
- children maps
  - ‘a’ to
  - ‘i’ to
  - ‘y’ to