Tries (aka Digital Trees)
Trie Overview

- From the word retrieval
  - Pronounced \textit{tree} or \textit{try}

- Alternative to BSTs for implementing the Set ADT and Map ADT

- Tree data structure that uses \textit{position} rather than \textit{comparisons}

- Assumes keys are a sequence of \textit{symbols/characters/digits} from some \textit{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\}

- \textit{Request: ponder parallels to the comparison sort vs radix sort trade-offs here}
BST vs Tries for Set ADT

- 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