Character-Based Operations

String symbol table API supports several useful character-based operations

**Prefix match**: Keys with prefix “sh:” “she,” “shell,” and “shore”

**Wildcard match**: Keys that match “.he:” “she” and “the”

**Longest prefix**: Key that has the longest prefix of “shellsort:” “shells”

In addition to standard StringST API (directly below), also add the following:  
*(StringST(), void put(String key, Value val), Value get(String key) & void delete(String key))*

Public class String ST<Value>  
:

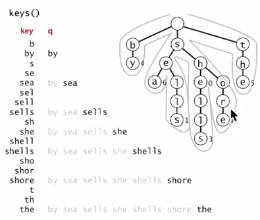
Iterable<String> keys() : all keys  
Iterable<String> keysWithPrefix(String s) : keys having s as a prefix  
Iterable<String> keysThatMatch(String s) : keys that match s (where . is a wildcard)  
String longestPrefixOf(String s) : longest key that is a prefix of s

Note: can also add other ordered ST methods, e.g. floor() and rank()

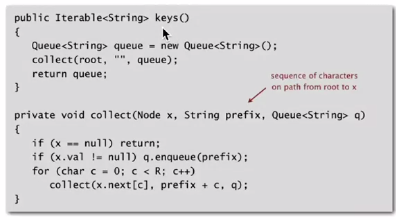
Ordered iteration

To iterate through all keys in sorted order:

* Do inorder traversal of trie; add keys encountered to a queue
* Maintain sequence of characters on path from root to node



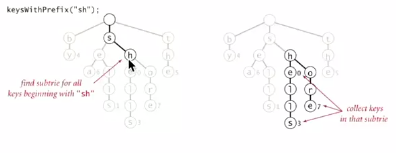
Implementation of keys()



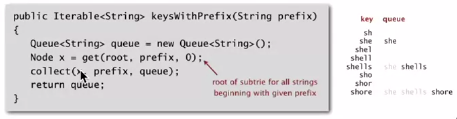
Prefix matches are useful for features such as autocomplete.

**Prefix matches in an R-way trie**

Find all keys in a symbol table with a given prefix



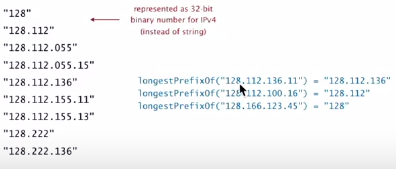
Java implementation



Longest prefix

Find longest key in symbol table that is a prefix of a query string

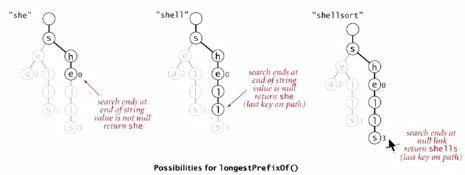
E.g. To send packet toward destination IP address, router chooses IP address in routing table that is longest prefix match:



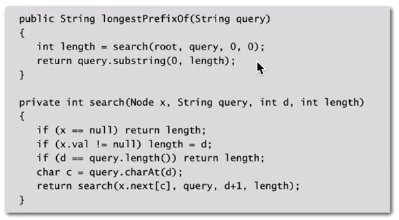
Not the same as floor… floor(“128.112.100.16”) = “128.112.055.15”

Find longest key in symbol table that is a prefix of a query string:

* Search for a query string
* Keep track of longest key encountered



Longest prefix search



Application: T9 texting (on phones without querty and only numpad)

Goal: type text messages on a phone keypad

**Multi-tap input**: enter a letter by repeatedly pressing a key until the desired letter appears

T9 text input:

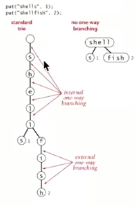
* Find all words that correspond to given sequence of numbers
* Press 0 to see all completion options

E.g. hello

* Multi-tap: 4 4 3 3 5 5 5 5 5 5 6 6 6
* T9: 4 3 5 5 6

Patricia trie

* Remove one-way branching
* Each node represents a sequence of characters
* Implementation: one step beyond this course



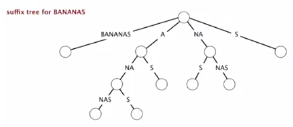
Applications

* Database search
* P2P network search
* IP routing table (longest prefix match)

AKA: crit-bit tree, radix tree

Suffix tree

* Patricia tree of suffixes of a string
* Linear-time construction: beyond this course



Applications:

* Linear-time: LRS, LCS
* Computational biology databases