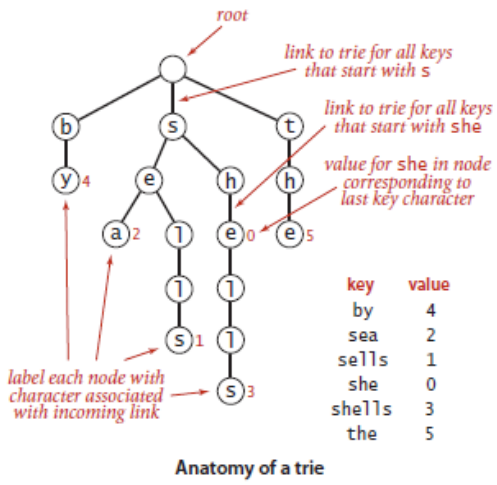
Tries

APIs we are trying to develop

public class StringST<Value>	
StringST()	*create a symbol table*
void put(String key, Value val)	*put key-value pair into the table (remove* key *if value is* null*)*
Value get(String key)	*value paired with* key *(*null *if* key *is absent)*
void delete(String key)	*remove* key *(and its value)*
boolean contains(String key)	*is there a value paired with* key*?*
boolean isEmpty()	*is the table empty?*
String longestPrefixOf(String s)	*the longest key that is a prefix of* s
Iterable<String> keysWithPrefix(String s)	*all the keys having* s *as a prefix*
Iterable<String> keysThatMatch(String s)	*all the keys that match* s *(where . matches any character)*
int size()	*number of key-value pairs*
Iterable<String> keys()	*all the keys in the table*

API for a symbol table with string keys

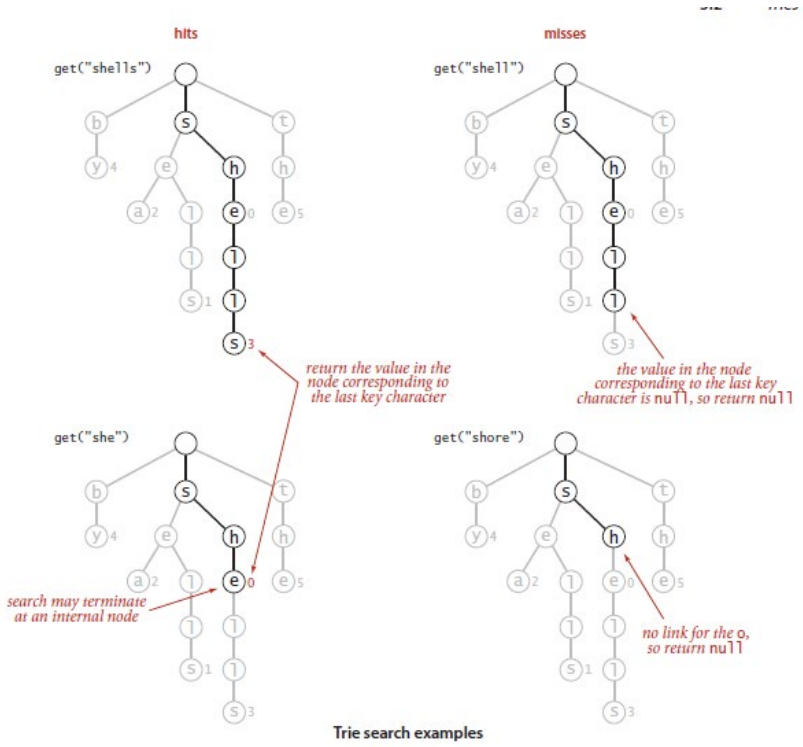
Basic Properties



Anatomy of a trie

Each node has R links, where R is the alphabet size. We omit null links for brevity. We store the value associated with the key in the last character node of the key.

Search in a trie



Trie search examples

Algorithm

:start

start at the root

while we don’t reach last character key or null link

end while

if the value of the last character is not null, match

if the value of the last character is null, not match

if the search terminates with a null link, not match.

:end

The base idea: Go from node to another node.

Insertion into a trie

:start

start at the root

while you reach the last character or a null link

end while

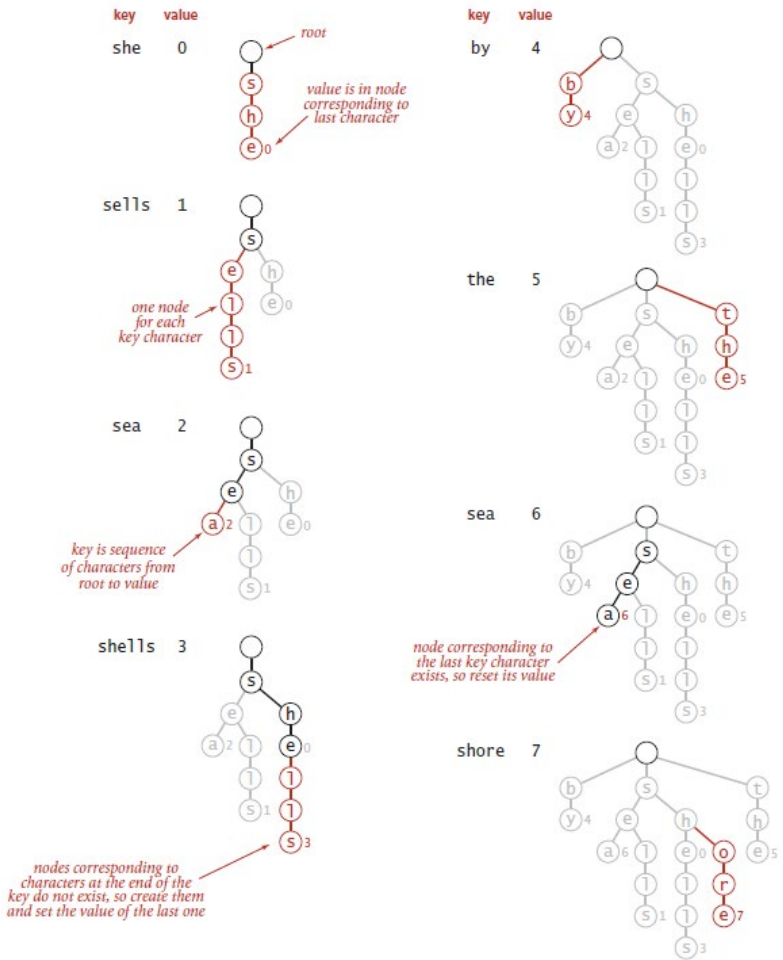
if encounter the null link before reaching the end of the key

create nodes of for the characters of key not discovered

if encountered the last character of the key before reaching a null link

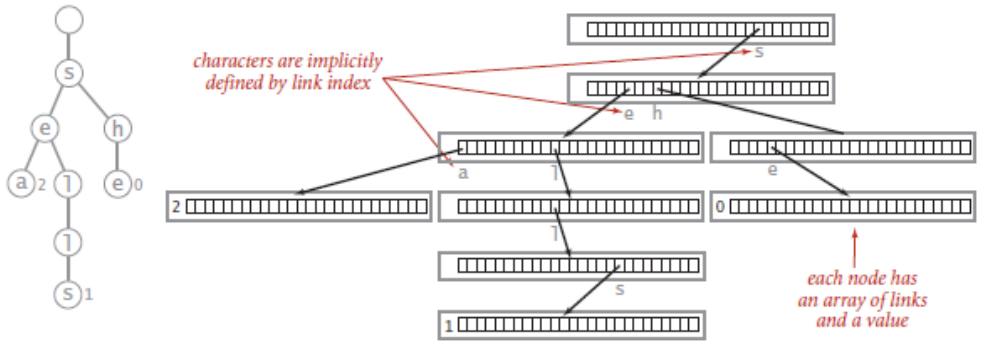
set the node's value

:end



Trie construction trace for standard indexing client

Node representation.

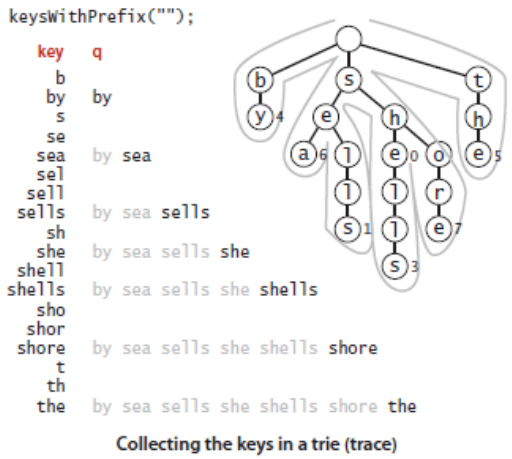


Trie representation ($R = 26$)

->Every node has R links, one for each possible character.

->Characters and keys are implicitly stored in the data structure.

Collecting keys



Collecting the keys in a trie (trace)

Algorithm for finding keys with prefix

:start

:input subtree containing the prefix, prefix, queue

if subtree is null, stop

if substree's value is not null, add prefix to queue

for each character in alphabet

call the algorithm recursively <- subtree.chilren [char], pre + char, q

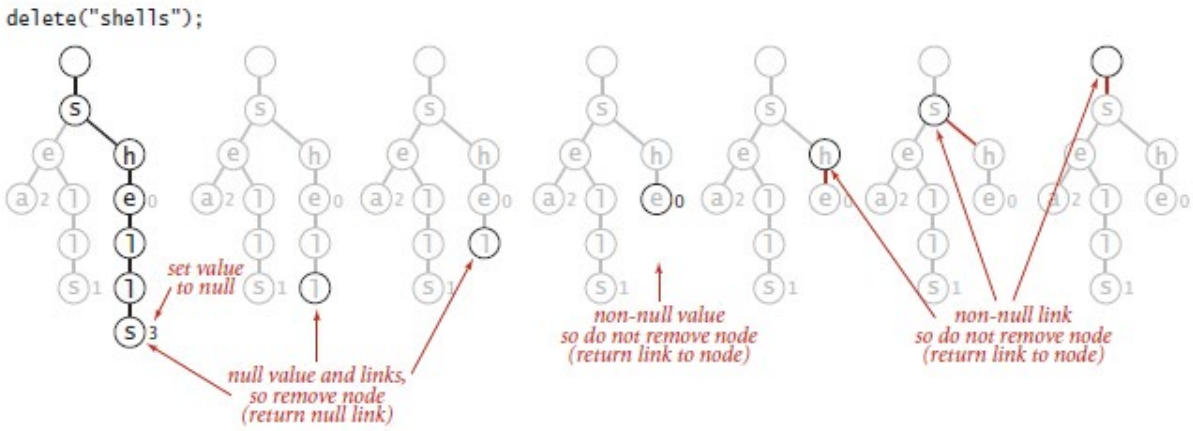
:end

Deletion

User normal search to located the node that is marked -> curNode

if curNode has null only null links /children

remove the curNode and above recursively



Deleting a key (and its associated value) from a trie