Week 10 - String Search

AD 325 - 2022

Contents

Reading & Videos

- https://www.coursera.org/learn/algorithms-part2/home/week/4

Reference

- https://algs4.cs.princeton.edu/52trie/
- https://algs4.cs.princeton.edu/53substring/
- https://www.geeksforgeeks.org/advantages-trie-data-structure/

Learning Outcomes

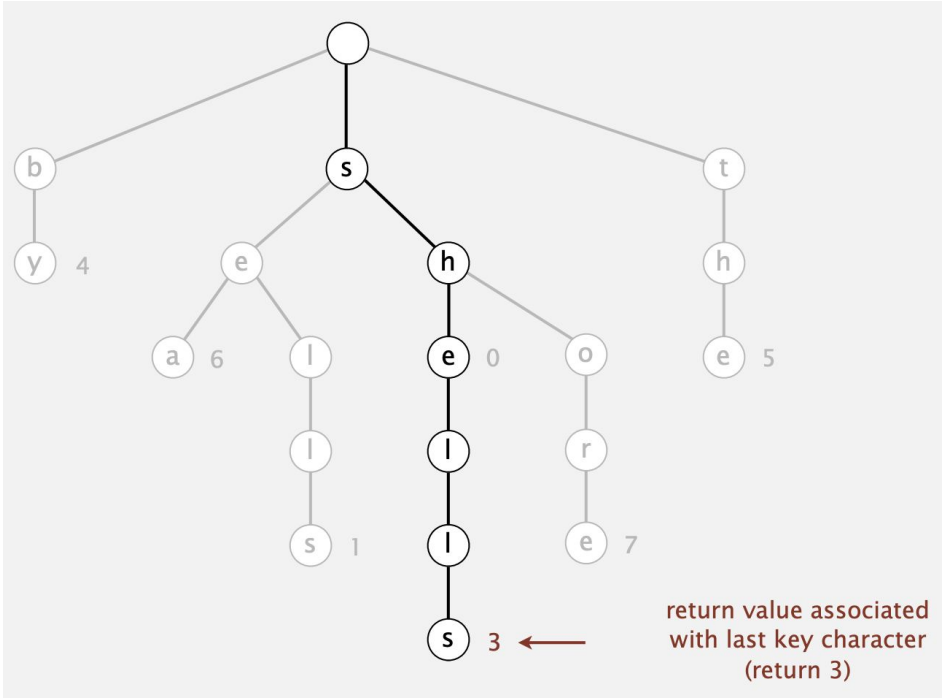
- Trie data structure
- Character-based operations
- Substring search

Trie Data Structure

The Trie (pronounced ‘try’) is a data structure optimized for string keys. As fast as hashing, but more flexible than BST and able to support sorting operations.

R-Way Trie

- Store characters (not keys) & values in nodes
- Each node has R children, one for each possible character (e.g. 256 characters for Latin alphabet)
- Children are represented as an array of nodes accessed by char value
- Nodes can link to a ‘next’ nodes to form words

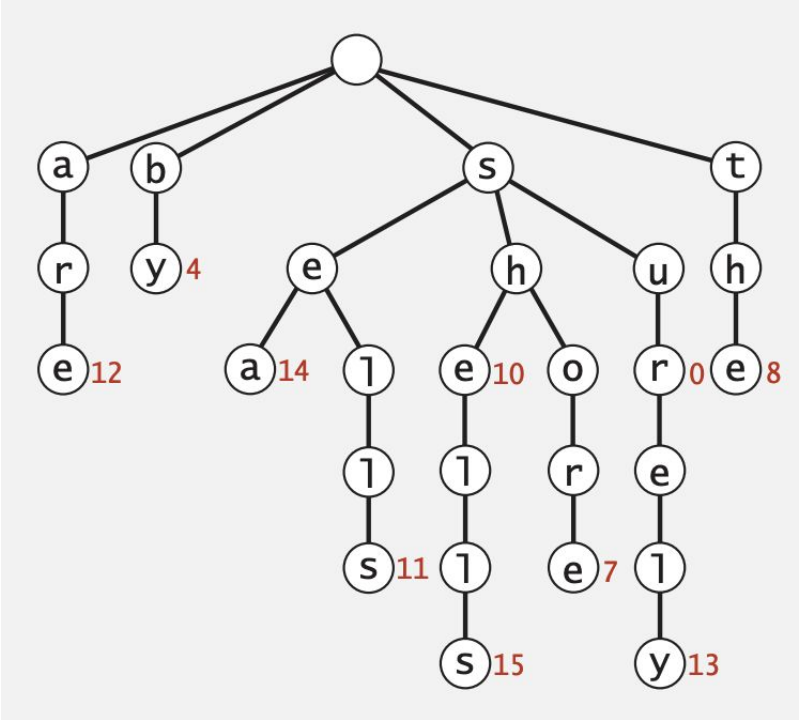


R-Way Trie Performance

- Fast search hit and even faster search miss, but wastes space for null links
- Method of choice for small R
- Too much memory for large R
-

Ternary Search Tries

- Stores characters & values in nodes
- Each node has at most 3 children - smaller (left), equal (middle), larger (right)
- As fast as hashing (for string keys), space efficient



Search in Ternary Search Trie

Follow links corresponding to each character in the key.

- If less, take left link
- if greater, take right link
- If equal, take the middle link and move to the next key character

String Symbol Table Performance

	Search hit	Search miss	Insert	Space
red-black BST	$L + c \lg^2 N$	$c \lg^2 N$	$c \lg^2 N$	4N
hashing	L	L	L	4N to 16N
R-way trie	L	$\log_R N$	L	(R + 1) N
TST	L + ln N	ln N	L + ln N	4N

- N = number of strings
- L = length of string
- R = radix

TST vs. Hashing

Hashing

- Need to examine entire key.
- Search hits and misses cost about the same.
- Performance relies on hash function.
- Does not support ordered symbol table operations.

TSTs

- Works only for strings (or digital keys).
- Only examines just enough key characters.
- Search miss may involve only a few characters.
- Supports ordered symbol table operations (and more).

Bottom line. TSTs are:

- Faster than hashing (especially for search misses).
- More flexible than red-black BSTs.