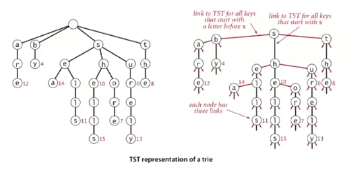
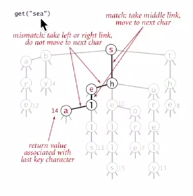
Ternary Search Tries

* Store characters and values in nodes (not keys)
* Each node has 3 children:
  + Smaller (left)
  + Equal (middle)
  + Larger (right)

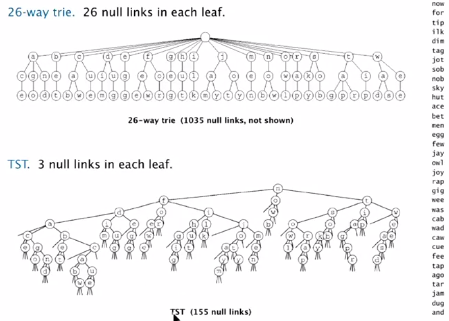


Search hit: Node where search ends has a non-null value

Search miss: Reach a null link or node where search ends has null value



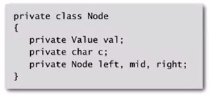
26-way trie vs TST



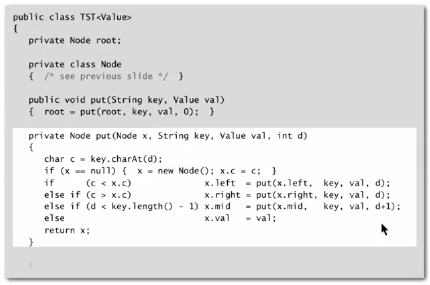
TST representation in Java

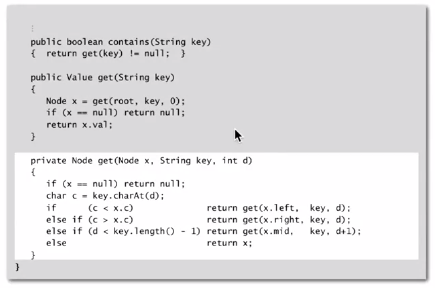
A TST node in Java has five fields:

* A value
* A character c
* A reference to a left TST
* A reference to a middle TST
* A reference to a right TST

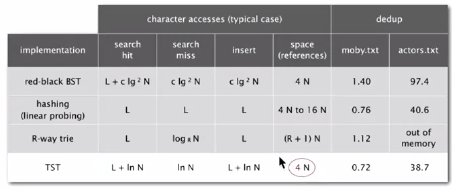


TST Java implementation





Performance

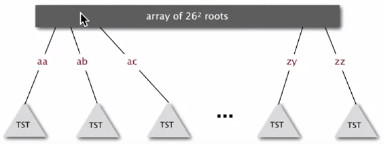


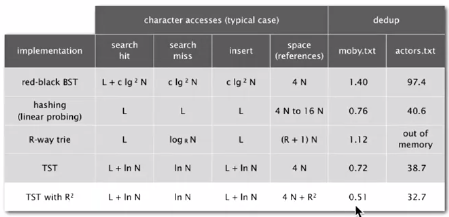
Can build balanced TSTs via rotations to achieve L + log N worst-case guarantees.

**TST is as fast as hashing (for string keys) and is quite space efficient**

Hybrid of R-way trie and TST:

* R2-way branching at root
* Each of R2 root nodes points to a TST





TST vs hashing

Hashing

* Need to examine each 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 (plus others)

Bottom line. TSTs are:

* Faster than hashing (especially for search misses)
* More flexibility than red-black BSTs