Ternary Search Trees

a.k.a.

Bentley-Saxe-Sedgewick Tries

Andrew W. Appel

Princeton
University

Problem: ordered lookup tables on strings

Operations

insert: string × value $\rightarrow^{!}$ unit

lookup: string → value

next: string → string

! means "with side effect"

get next key in sorted order;
or "iterator over keys"

Example strings:

```
http://www/cs.princeton.edu/courses/archive/spring19/cos510/sf/vfa/toc.html
http://www/cs.princeton.edu/courses/archive/spring19/cos510/sf/vfa/SearchTree.html#lab61
http://www/cs.princeton.edu/courses/archive/spring19/cos510/sf/vfa/Trie.html
http://www/cs.princeton.edu/index.php
http://www/cs.princeton.edu/courses/archive/spring19/cos510/sf/vfa/Trie.html#lab127
```

Obvious solution

Balanced binary search trees with string keys

- insert: logN comparisons
- lookup: logN comparisons
- next: logN pointer operations, no comparisons

Problem: each comparison can take a long time

http://www/cs.princeton.edu/courses/archive/spring19/cos510/sf/vfa/toc.html
http://www/cs.princeton.edu/courses/archive/spring19/cos510/sf/vfa/SearchTree.html#lab61
http://www/cs.princeton.edu/courses/archive/spring19/cos510/sf/vfa/Trie.html
http://www/cs.princeton.edu/index.php
http://www/cs.princeton.edu/courses/archive/spring19/cos510/sf/vfa/Trie.html#lab127

(if we didn't need "next"): Obvious solution #2

Hash tables

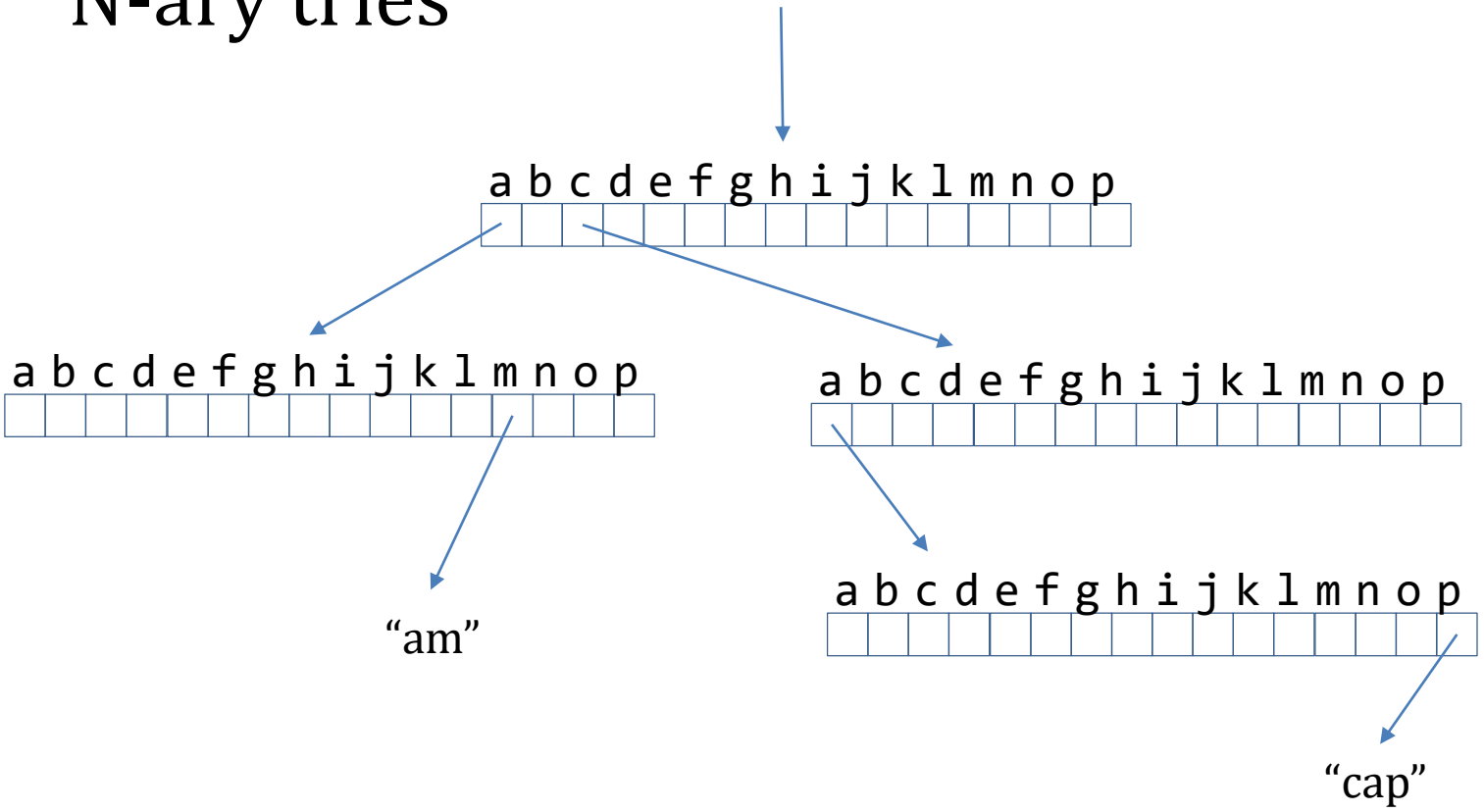
- insert: constant number of comparisons
- lookup: constant number of comparisons

Problem: hashing and comparison can take a long time

http://www/cs.princeton.edu/courses/archive/spring19/cos510/sf/vfa/toc.html
http://www/cs.princeton.edu/courses/archive/spring19/cos510/sf/vfa/SearchTree.html#lab61
http://www/cs.princeton.edu/courses/archive/spring19/cos510/sf/vfa/Trie.html
http://www/cs.princeton.edu/index.php
http://www/cs.princeton.edu/courses/archive/spring19/cos510/sf/vfa/Trie.html#lab127

Another solution

N-ary tries



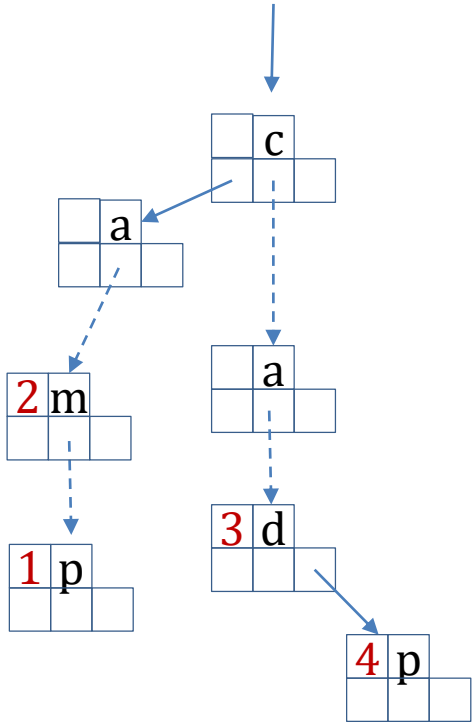
Good: Insert, lookup take linear time in the length of the string: optimal!
Bad: ASCII alphabet size=128, need 128 words per tree node, wastes space & time
Meh: Iterator can take 128 operations to find next key . . .

Ternary Search Trees

Fast algorithms for sorting and searching strings,

by Jon L. Bentley and Robert Sedgewick, in *SODA'97: Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms,* pp. 360-369, 1997.

amp↦1
am ↦2
cad ↦3
cap ↦4



length of string: n
alphabet size: k
averaged used alphabet: u

Balanced TST lookup:
worst case $n \cdot \log(k)$

Unbalanced TST lookup:
worst case $n \cdot k$
typical case: $n \cdot u$ $\quad u \ll k$