COMP261 Lecture 5

Tries

Victoria
UNIVERSITY OF WELLINGTON
Te Whare Wānanga o te Ūpoko o te Ika a Māui
CAPITAL CITY UNIVERSITY

Index Structures

- Indexing Roads and Intersections by ID
- Indexing Roads by prefix of name (name and city!!)
- Indexing Intersections by position
- *What structures do we use to make it fast?*

- Tries are Commonly Used For:
  - Storing large dictionaries in minimal space, but fast access time
  - Doesn't need to store common prefixes multiple times.
  - Allows auto completion, spelling correction
  - Can store numbers (use the bits as the index elements) → binary trie

Tries

- Need to store names in data structure
- Need to access them by prefix
  - given prefix, quickly find all names with that prefix without looking at any other names.
- If it is a map
  - need to store associated value with full name

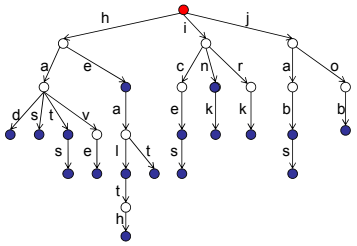
What structure?

:
acton pl avondale
ada st remuera
adair pl weymouth
adam st greenlane
adam sunde pl glen eden
adams pl kamo
adams rd awarua
adams rd kaukapakapa
adams rd manurewa
adams rd thornton bay
adams rd whangapoua
adams rd whareora
adamson rd ormiston
addington ave manurewa
addis pl shelly park
addison dr glendene
addison rd pataua
:

Tries

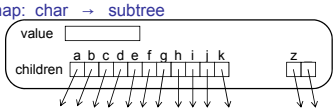
- Tree: set of strings/keys (if map string → value)
  - children indexed by elements of the key
  - nodes corresponding to complete key are marked (contain a value)
  - tree under a node = set of all keys with the prefix so far.

had	ice
has	iced
hat	in
hats	ink
have	irk
he	iron
heal	jab
health	jabs
heat	job



Tries

- Each node contains
  - marker (if set) or associated value (if map)
  - a map: char → subtree



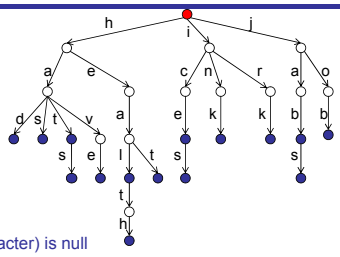
- Algorithm for contains(word) / get(key)

```
node ← root of trie
for each character in word:
    node ← node.child.get(character)
return node.marked   /   node.value
```

Adding to a Trie

add("he")

add("inch")



add (word) / put(word, val)

```
node ← root
for each character in word:
    if node.children.get(character) is null
        node.children.get(character) ← new node
    node ← node.children.get(character)
node.marked ← true   /   node.value ← val
```