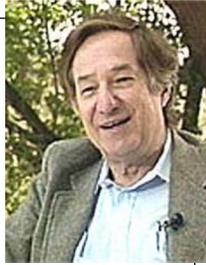


## Topic 25

### Tries

“In 1959, (Edward) Fredkin recommended that BBN (Bolt, Beranek and Newman, now BBN Technologies) purchase the very first PDP-1 to support research projects at BBN. *The PDP-1 came with no software whatsoever.*



Fredkin wrote a PDP-1 assembler called FRAP (Free of Rules Assembly Program);”

Tries were first described by René de la Briandais in *File searching using variable length keys.*

## Clicker 1

- ▶ How would you pronounce “Trie”
- A. “tree”
- B. “tri – ee”
- C. “try”
- D. “tiara”
- E. something else

CS314

Tries

2

## Tries aka Prefix Trees

- ▶ Pronunciation:
- ▶ From retrieval
- ▶ Name coined by Computer Scientist Edward Fredkin
- ▶ **Retrieval** so “tree”
- ▶ ... but that is very confusing so most people pronounce it “try”

CS314

Tries

3

## Predictive Text and AutoComplete

- ▶ Search engines and texting applications guess what you want after typing only a few characters

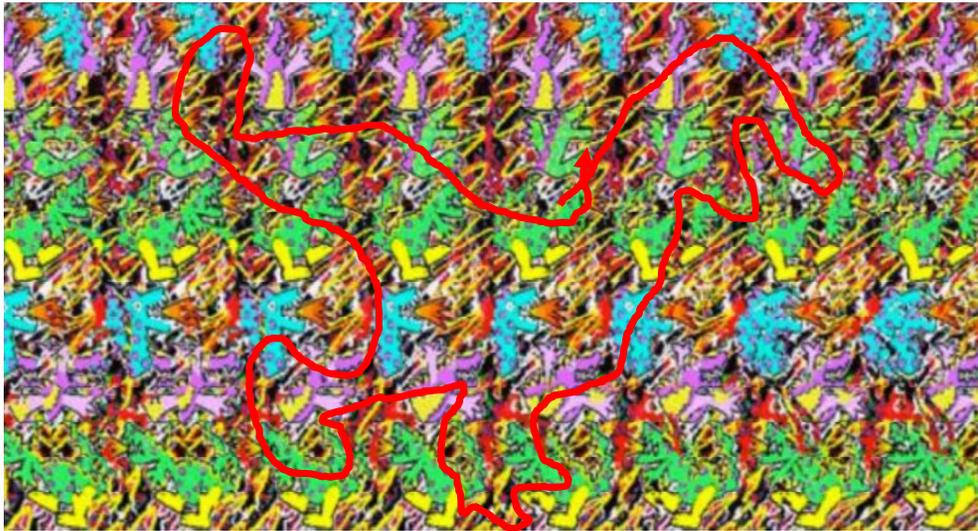
Hel

---

hello  
hellboy  
hello fresh  
helen keller  
helena christensen  
hello may  
hell or high water  
hello neighbor  
helzberg  
help synonym

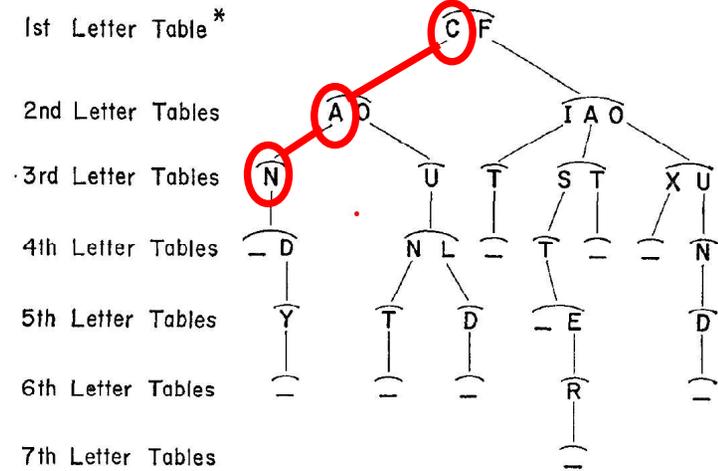


?????



Picture of a Dinosaur

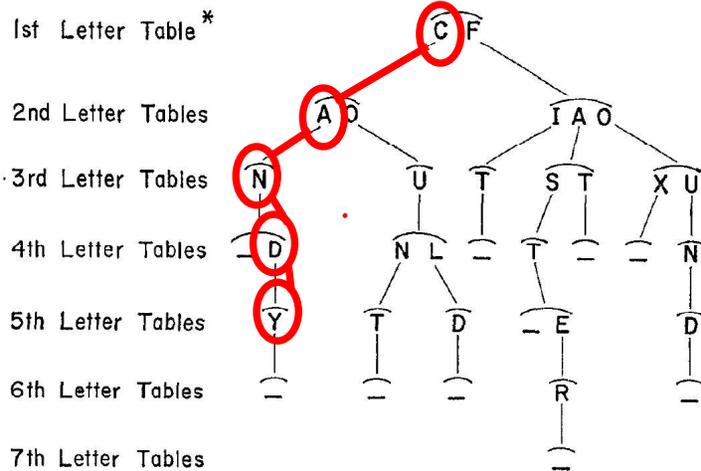
Can



\*All entries of any one table are covered by a single arc (—).

Fig. 1—Formation of a set of tables.

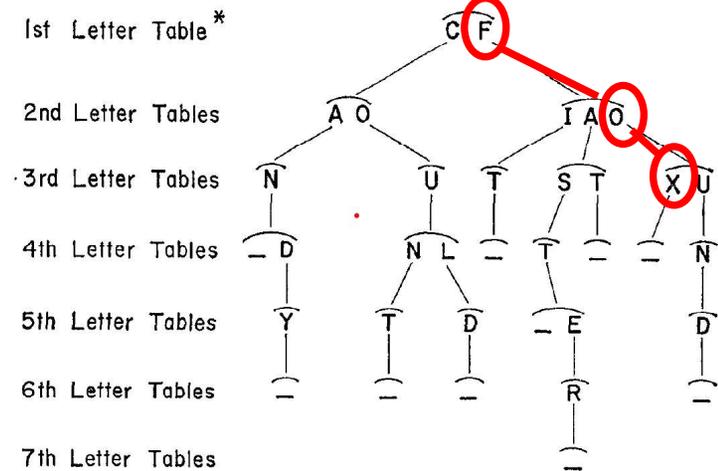
Candy



\*All entries of any one table are covered by a single arc (—).

Fig. 1—Formation of a set of tables.

Fox



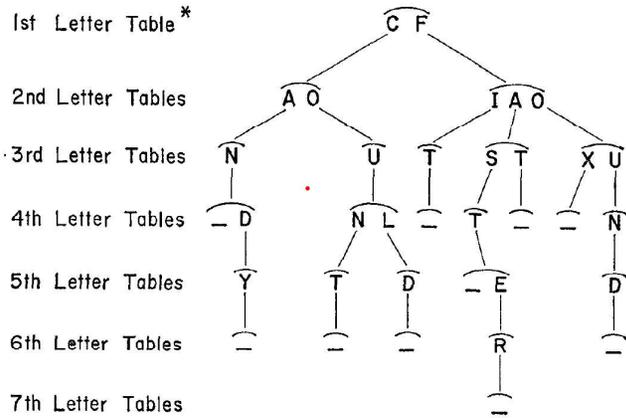
\*All entries of any one table are covered by a single arc (—).

Fig. 1—Formation of a set of tables.

## Clicker 2

► Is “fast” in the dictionary represented by this Trie?

- A. No
- B. Yes
- C. It depends



\*All entries of any one table are covered by a single arc (—).

Fig. 1—Formation of a set of tables.

Tries

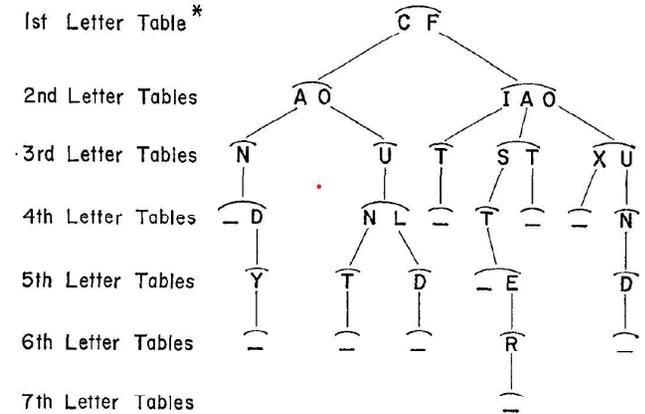
13

CS314

## Clicker 3

► Is “fist” in the dictionary represented by this Trie?

- A. No
- B. Yes
- C. It depends



\*All entries of any one table are covered by a single arc (—).

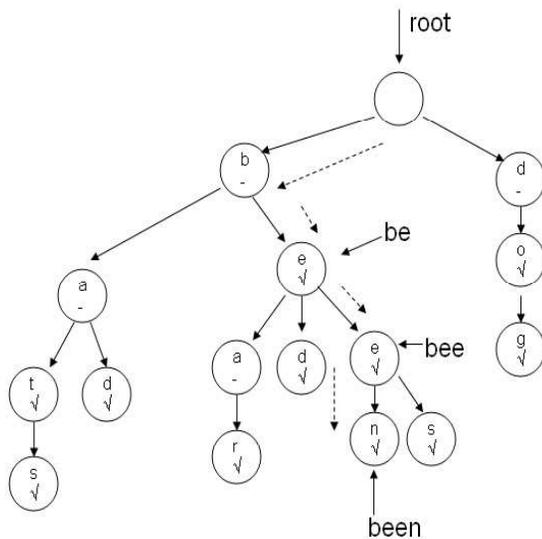
Fig. 1—Formation of a set of tables.

14

CS314

## Tries

- Another example of a Trie
- Each node stores:
  - A char
  - A boolean indicating if the string ending at that node is a word
  - A list of children



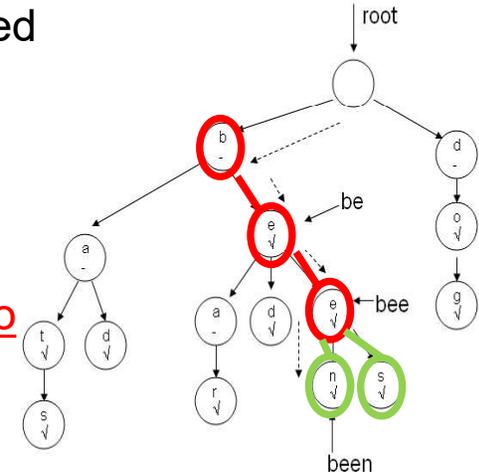
Tries

15

CS314

## Predictive Text and AutoComplete

- As characters are entered we descend the Trie
- ... and from the current node ...
- ... we can descend to terminators and leaves to see all possible words based on current prefix
- b, e, e -> bee, been, bees



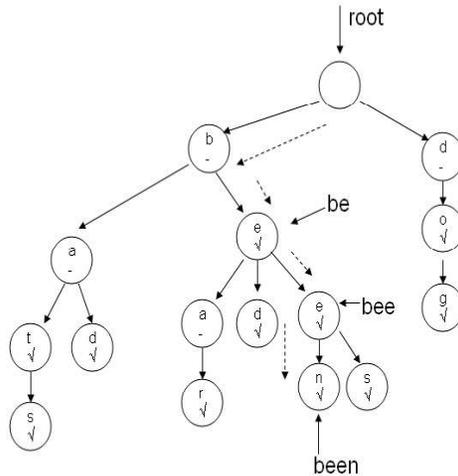
Tries

16

CS314

## Tries

- ▶ Stores words and phrases.
  - other values possible, but typically Strings
- ▶ The whole word or phrase is not actually stored in a single node.
- ▶ ... rather the path in the tree represents the word.



## Implementing a Trie

```
public class Trie {

    private TNode root;
    private int size;
    private int numNodes;

    public Trie() {
        root = new TNode();
        numNodes = 1;
    }
}
```

## TNode Class

```
private static class TNode {
    private boolean word;
    private char ch;
    private LinkedList<TNode> children;
}
```

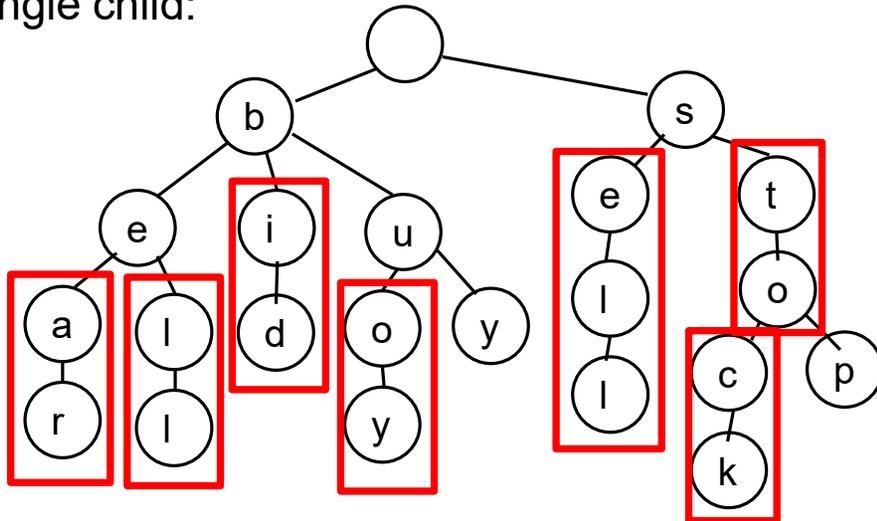
- ▶ Basic implementation uses a LinkedList of TNode objects for children
- ▶ Other options?
  - ArrayList?
  - Something more exotic?

## Basic Operations

- ▶ Adding a word to the Trie
- ▶ Getting all words with given prefix
- ▶ Demo in IDE

## Compressed Tries

- Some words, especially long ones, lead to a chain of nodes with single child, followed by single child:



## Compressed Trie

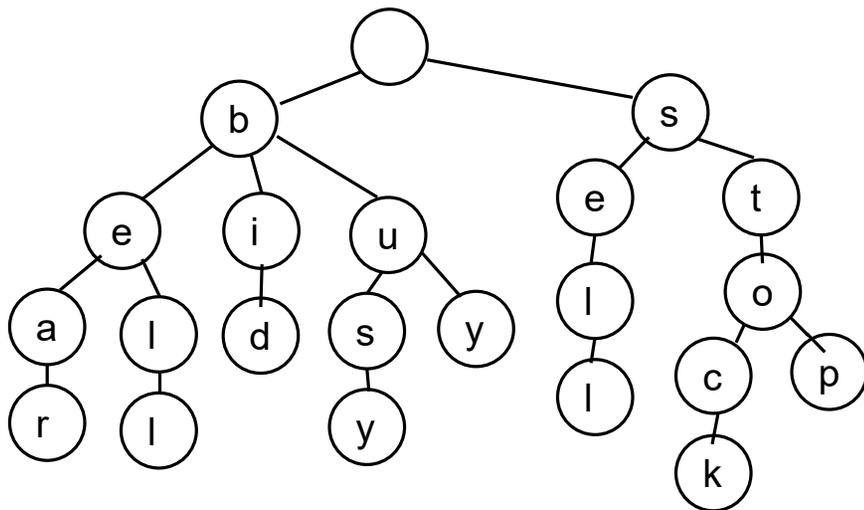
- Reduce number of nodes, by having nodes store Strings
- A chain of single child followed by single child (followed by single child ... ) is compressed to a single node with that String
- Does not have to be a chain that terminates in a leaf node
  - Can be an internal chain of nodes

CS314

Tries

22

## Original, Uncompressed

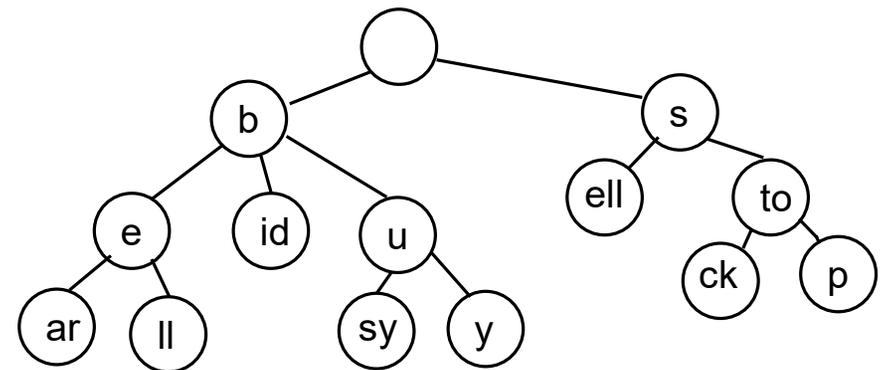


CS314

Tries

23

## Compressed Version



8 fewer nodes compared to uncompressed version  
s – t – o – c – k

CS314

Tries

24

# Data Structures

- ▶ Data structures we have studied
  - arrays, array based lists, linked lists, maps, sets, stacks, queues, trees, binary search trees, graphs, hash tables, red-black trees, priority queues, heaps, tries
- ▶ Most program languages have some built in data structures, native or library
- ▶ Must be familiar with performance of data structures
  - best learned by implementing them yourself

# Data Structures

- ▶ We have **not** covered every data structure

**Abstract data types** [ edit source | edit beta ]

- Container
- Map/Associative array/Dictionary
- Multimap
- List
- Set
- Multiset
- Priority queue
- Queue
- Deque
- Stack
- String
- Tree
- Graph

Some properties of abstract data types:

Structure	Stable	Unique	Cells per Node
Bag (multiset)	no	no	1
Set	no	yes	1
List	yes	no	1
Map	no	yes	2

"Stable" means that input order is retained. Other str

**Arrays** [ edit source | edit beta ]

- Array
- Bidirectional map
- Bit array
- Bit field
- Bitboard
- Bitmap
- Circular buffer
- Control table
- Image
- Dynamic array
- Gap buffer
- Hashed array tree
- Heightmap
- Lookup table
- Matrix
- Parallel array
- Sorted array
- Sparse array
- Sparse matrix
- liffe vector
- Variable-length array

**Lists** [ edit source | edit beta ]

- Doubly linked list
- Linked list
- Self-organizing list
- Skip list
- Unrolled linked list
- VList
- Xor linked list
- Zipper
- Doubly connected edge list
- Difference list

**Heaps** [ edit source | edit ]

- Heap
- Binary heap
- Weak heap
- Binomial heap
- Fibonacci heap
  - AF-heap
- 2-3 heap
- Soft heap
- Pairing heap
- Leftist heap
- Treap
- Beap
- Skew heap
- Ternary heap
- D-ary heap

**Trees** [ edit source | edit ]

In these data structures each

- Tree
- Radix tree
- Suffix tree
- Suffix array
- Compressed suffix array
- FM-index
- Generalised suffix tree
- B-tree
- Judy array
- X-fast tree
- Y-fast tree
- Ctree

**Multitway trees** [ edit source ]

**Graphs** [ edit source | edit beta ]

- Graph
- Adjacency list
- Adjacency matrix
- Graph-structured stack
- Scene graph
- Binary decision diagram
- Zero suppressed decision diagram
- And-inverter graph
- Directed graph
- Directed acyclic graph
- Propositional directed acyclic graph
- Multigraph
- Hypergraph

**Other** [ edit source | edit beta ]

- Lightmap
- Winged edge
- Doubly connected edge list
- Quad-edge
- Routing table
- Symbol table

[http://en.wikipedia.org/wiki/List\\_of\\_data\\_structures](http://en.wikipedia.org/wiki/List_of_data_structures)

# Data Structures

- ▶ deque, b-trees, quad-trees, binary space partition trees, skip list, sparse list, sparse matrix, union-find data structure, Bloom filters, AVL trees, 2-3-4 trees, and more!
- ▶ Must be able to learn new and apply new data structures