Tries

Ben Langmead

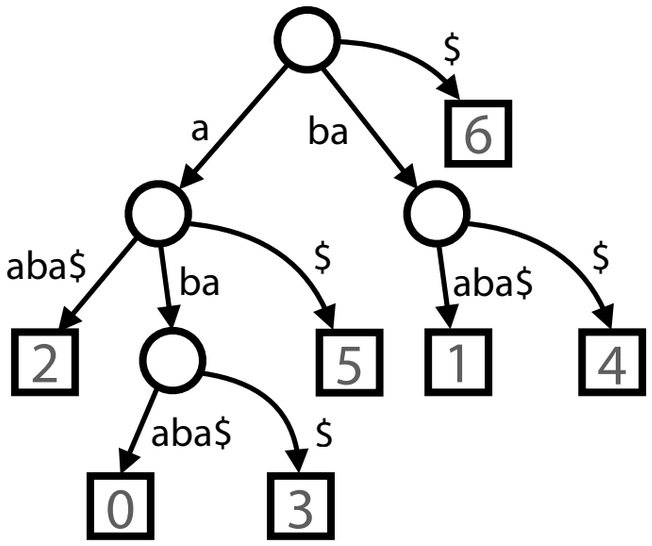
Indexing

We've seen indexes built over substrings of *T*

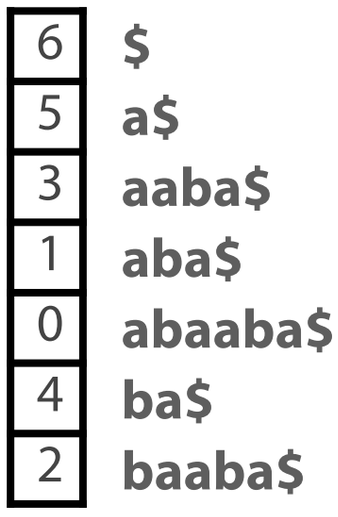
```
CGTGC:  0,4
GCGTG:  3
GTGCC:  1
GTGCT:  5
TGCCT:  2
TGCTT:  6
```

Indexing with suffixes

Different approach is to index ***suffixes*** of *T*. This yields surprisingly **economical** & **practical** data structures:



Suffix Tree



Suffix Array

FM Index

See other playlist

Tries

Trie ("try"): a tree representing a collection of strings (keys)

The smallest tree such that

Each edge is labeled with a character $c \in \Sigma$

For given node, at most one child edge has label c

Each key is "spelled out" along a path starting at root

Short version: each key is "spelled out" along a path from the root and **common prefixes** are collapsed as much as possible

Helpful for implementing a *set* or *map* when the keys are strings

Tries

Keys: instant, internal, internet

Key	Value
instant	1
internal	2
internet	3

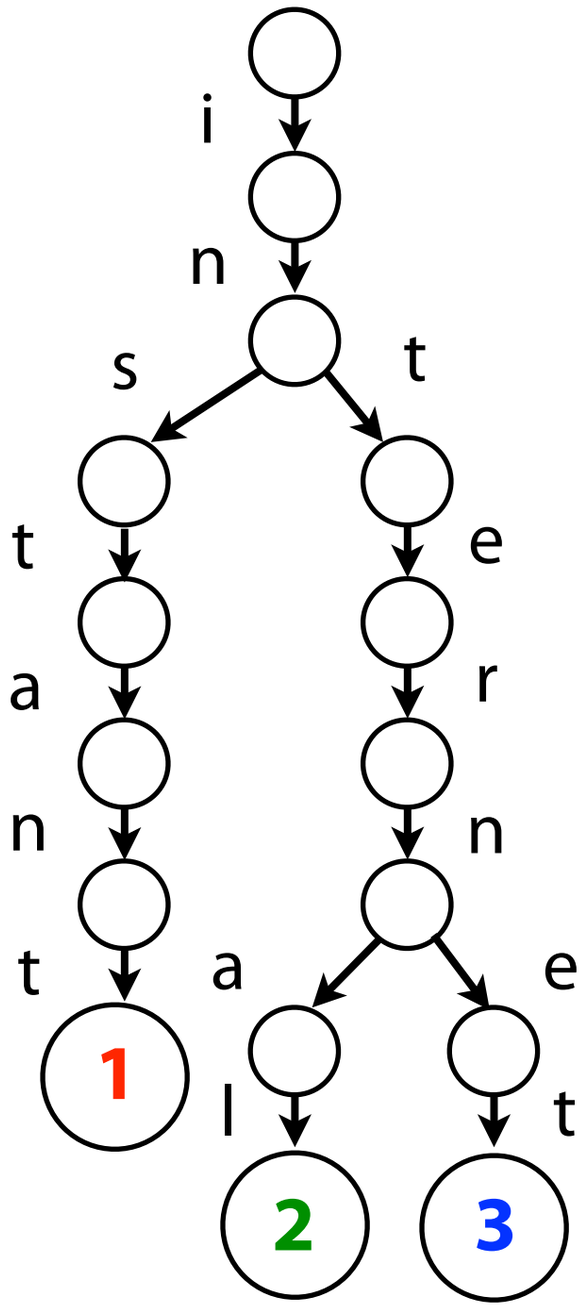
Smallest tree such that:

- Each edge is labeled with a char $c \in \Sigma$
- For given node, at most one child edge has label c
- Each key is "spelled out" along a path starting at root

Tries

Keys: instant, internal, internet

Key	Value
instant	1
internal	2
internet	3



Smallest tree such that:

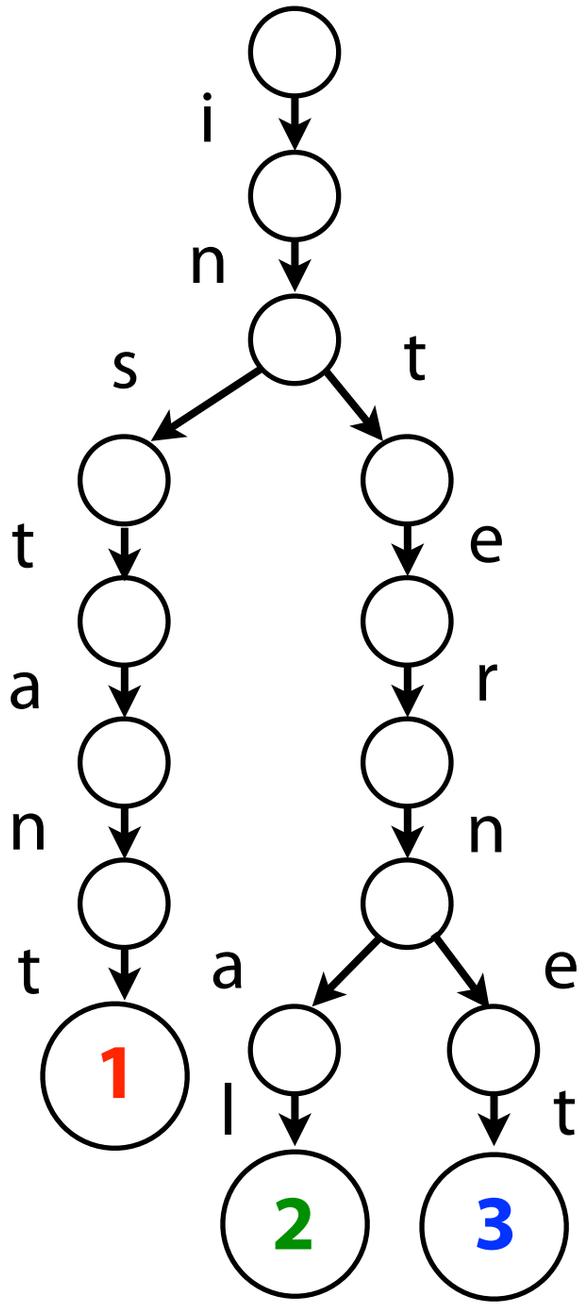
- Each edge is labeled with a char $c \in \Sigma$
- For given node, at most one child edge has label c
- Each key is "spelled out" along a path starting at root

Tries

How do we check whether “infer” is in the trie?

Start at root and try to match successive characters of “infer” to edges in trie

"Walking down"

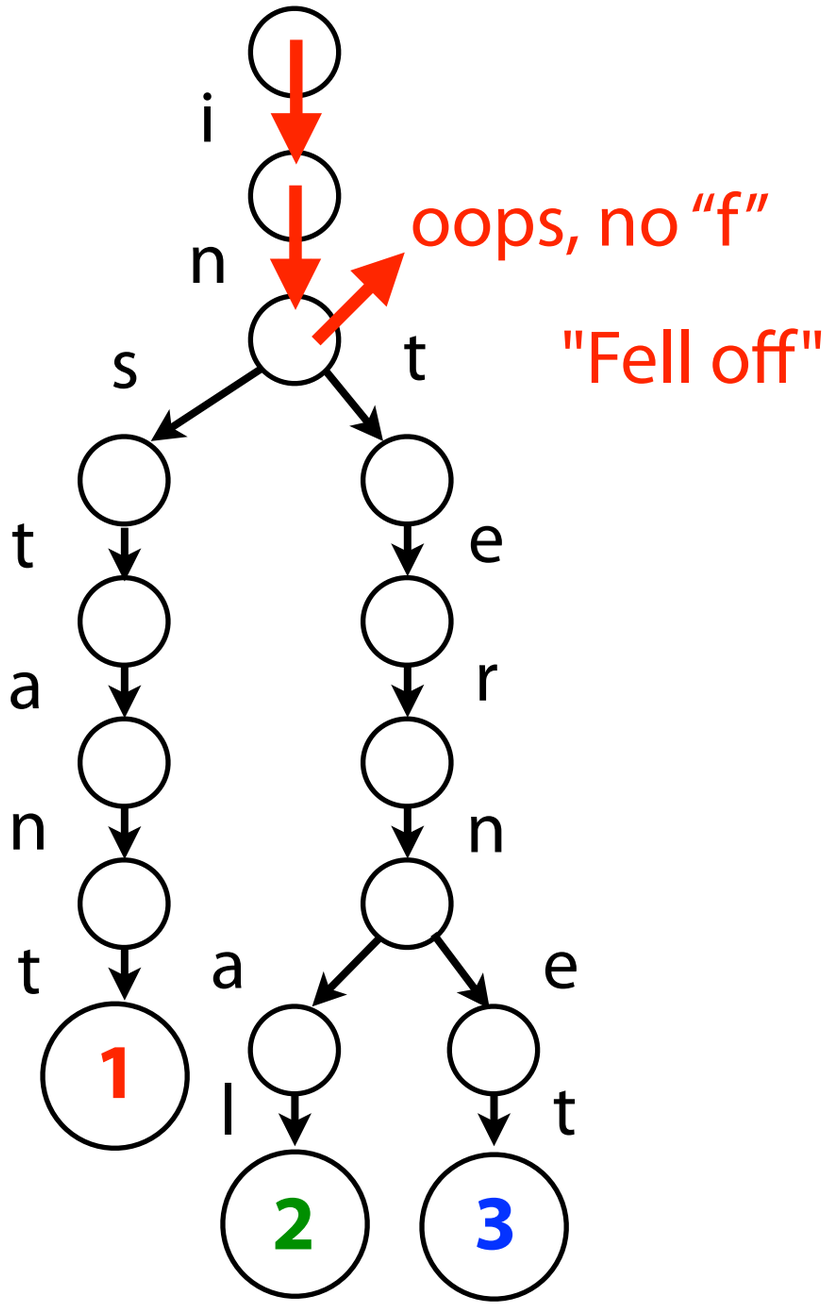


Tries

How do we check whether “infer” is in the trie?

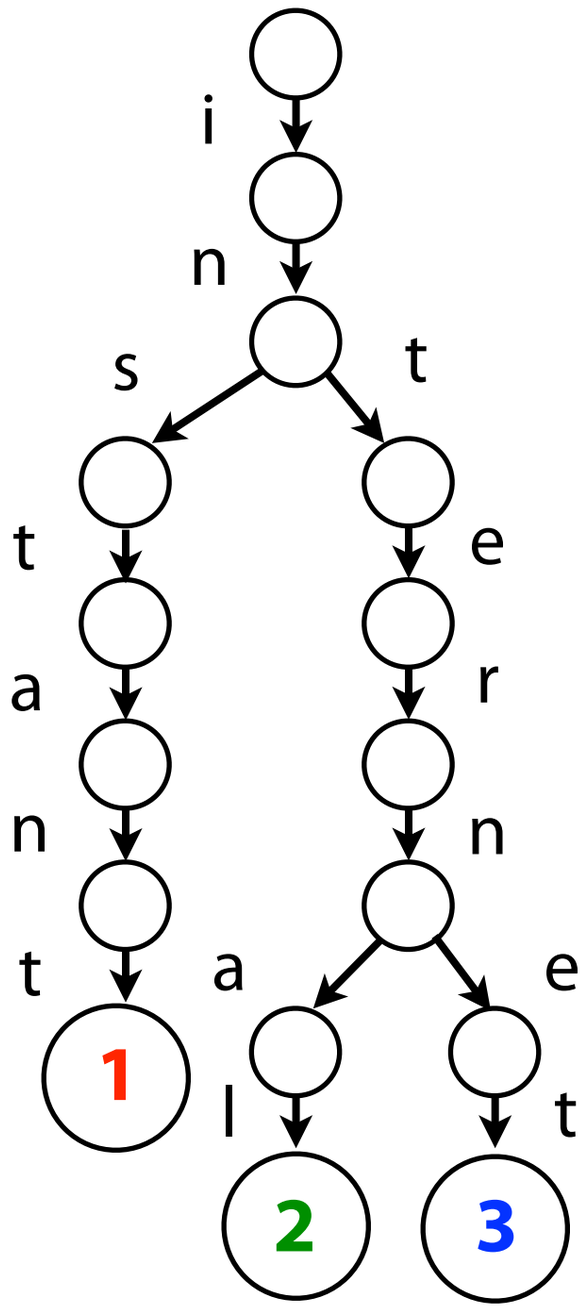
Start at root and try to match successive characters of “infer” to edges in trie

"Walking down"



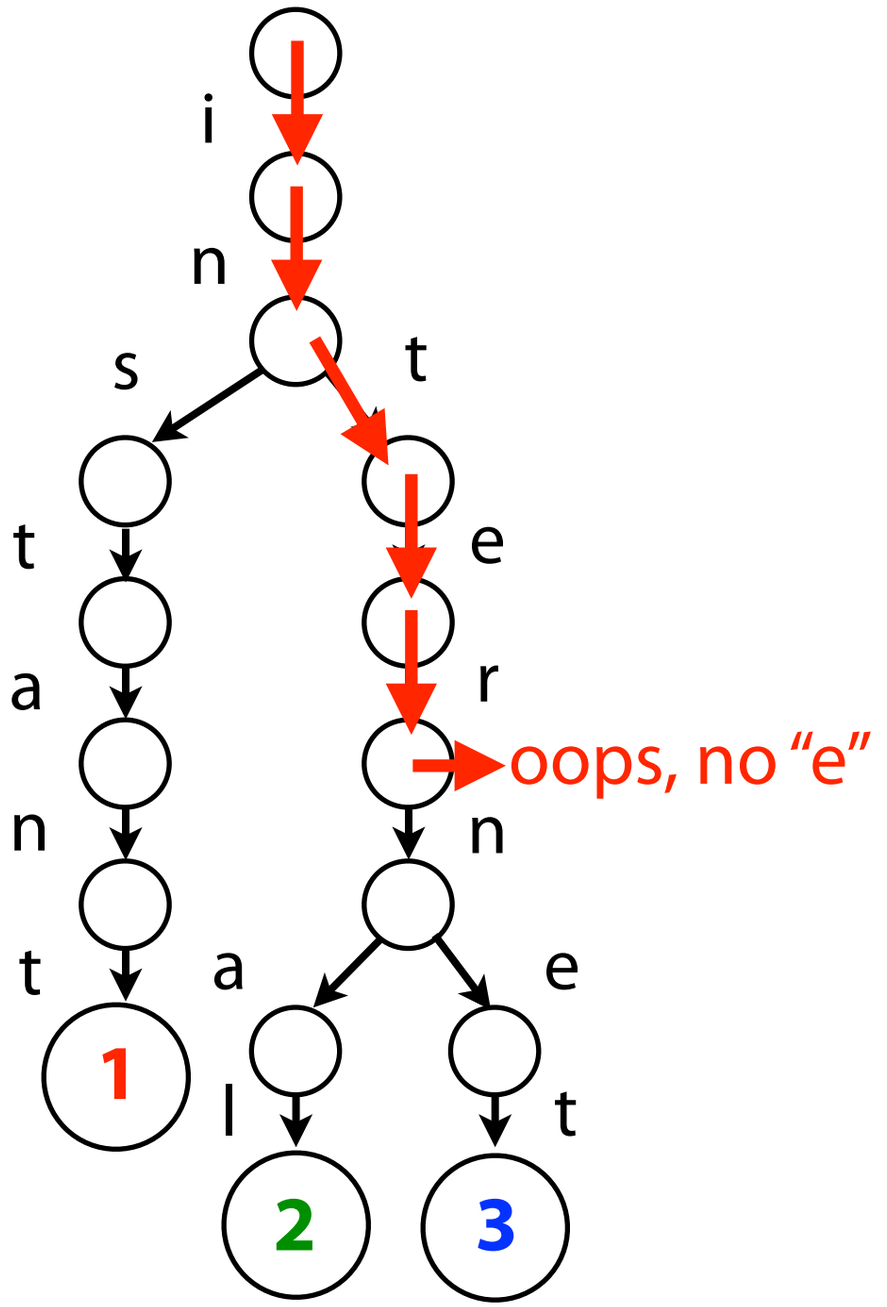
Tries

Query: “interesting”



Tries

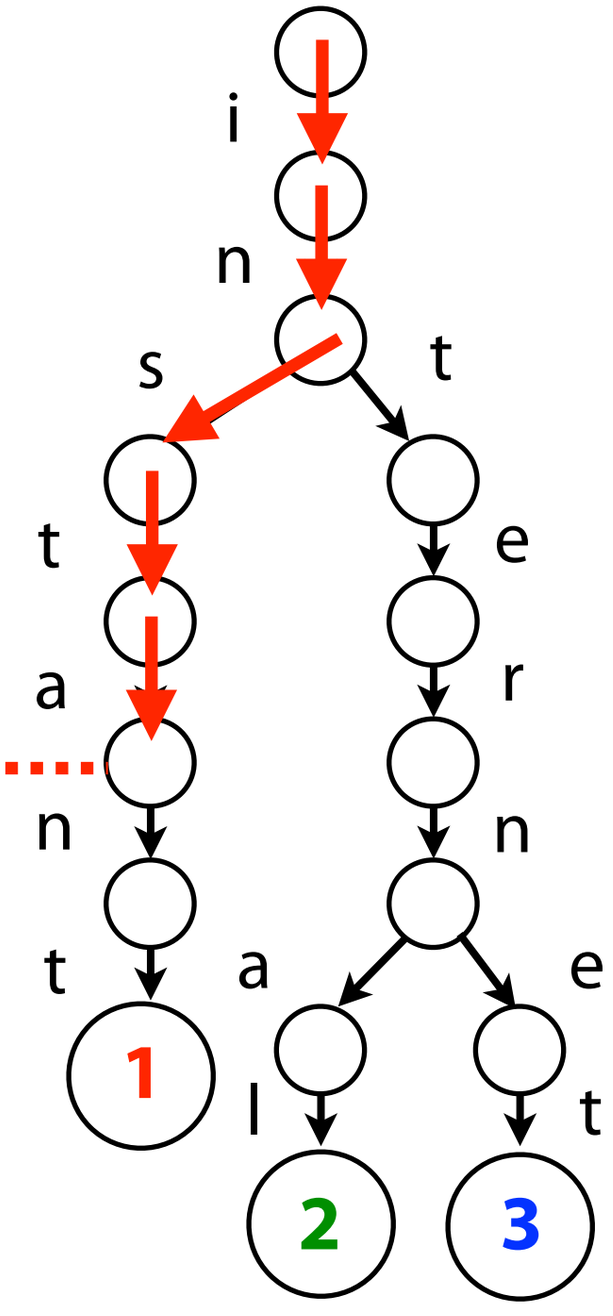
Query: “interesting”



Tries

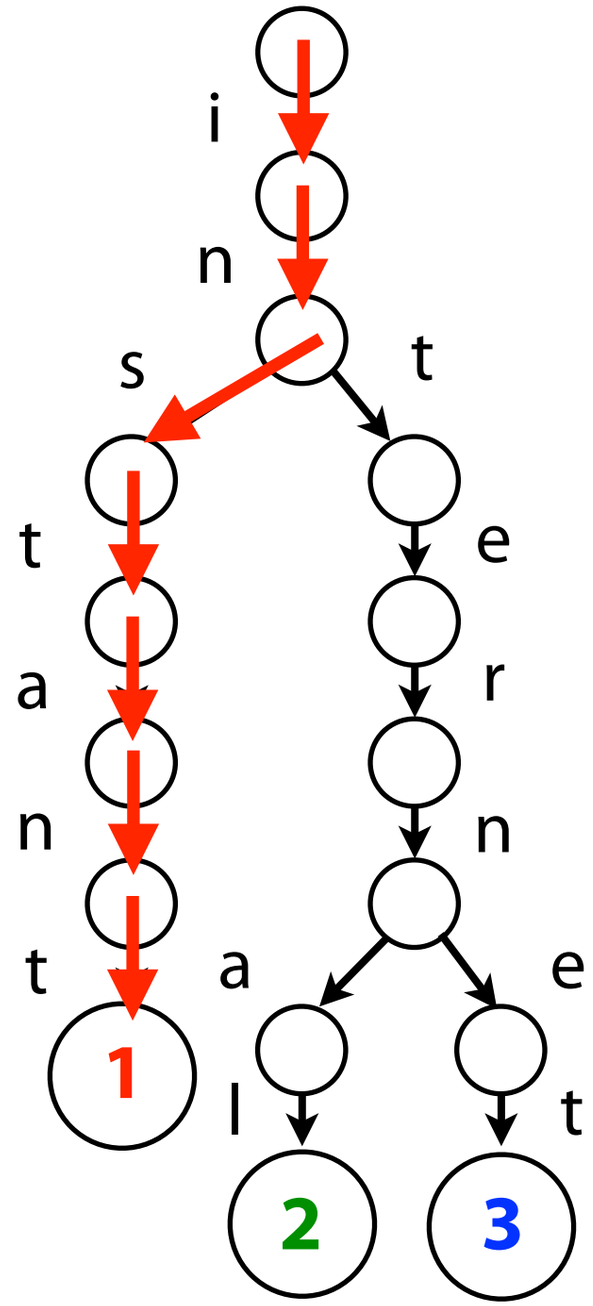
Query: “insta”

No value associated with node, so “insta” wasn’t a key



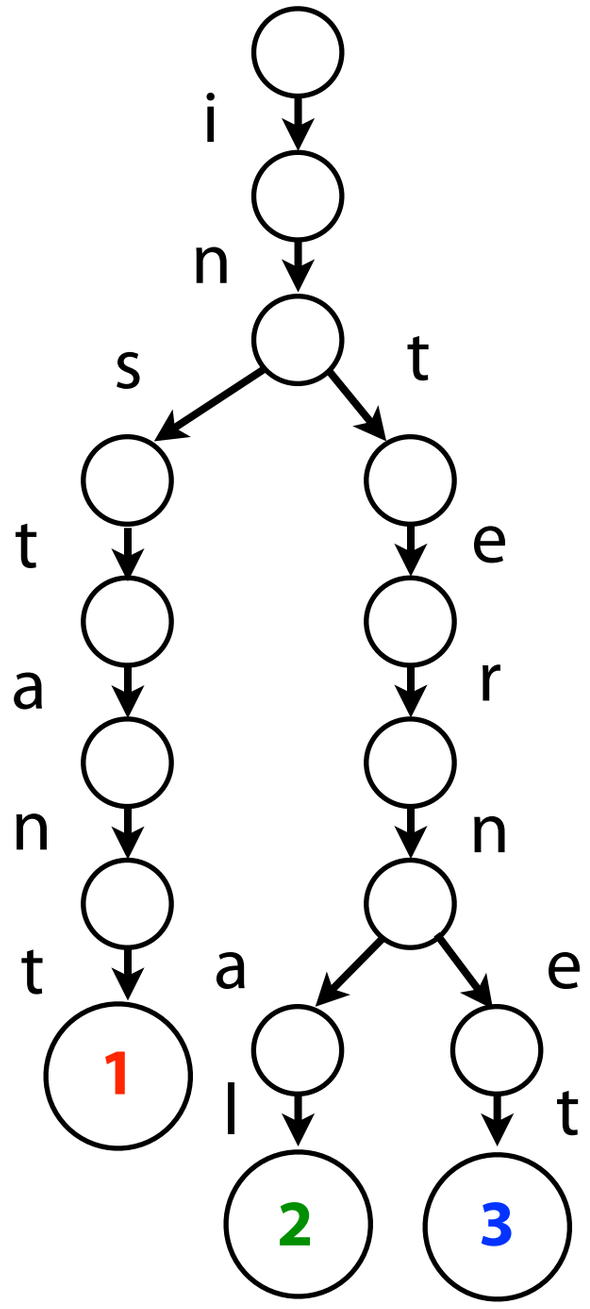
Tries

Query: “instant”



It’s a key!

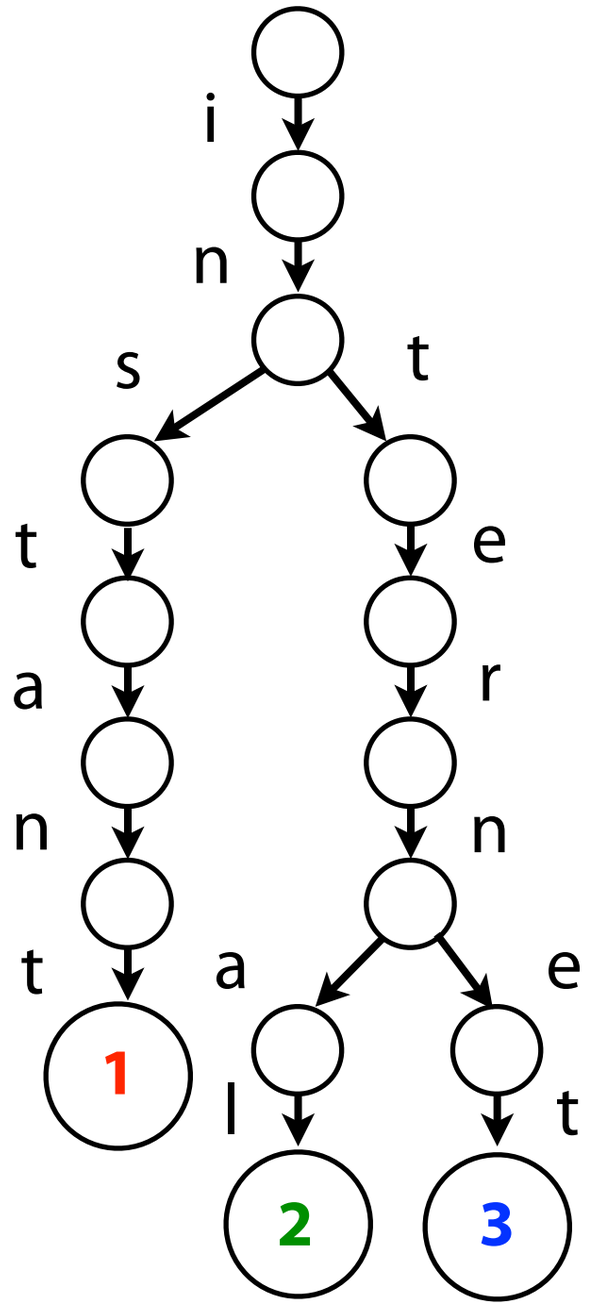
Tries



Checking for presence of key P, where $|P| = n$ traverses $\leq$ $\boldsymbol{n}$ edges

If total length of all keys is N, trie has $\leq$ $\boldsymbol{N}$ edges

Tries



How to represent edges between a node and its children?

Map (from characters to child nodes)

Idea 1: Hash table

Idea 2: Sorted lists

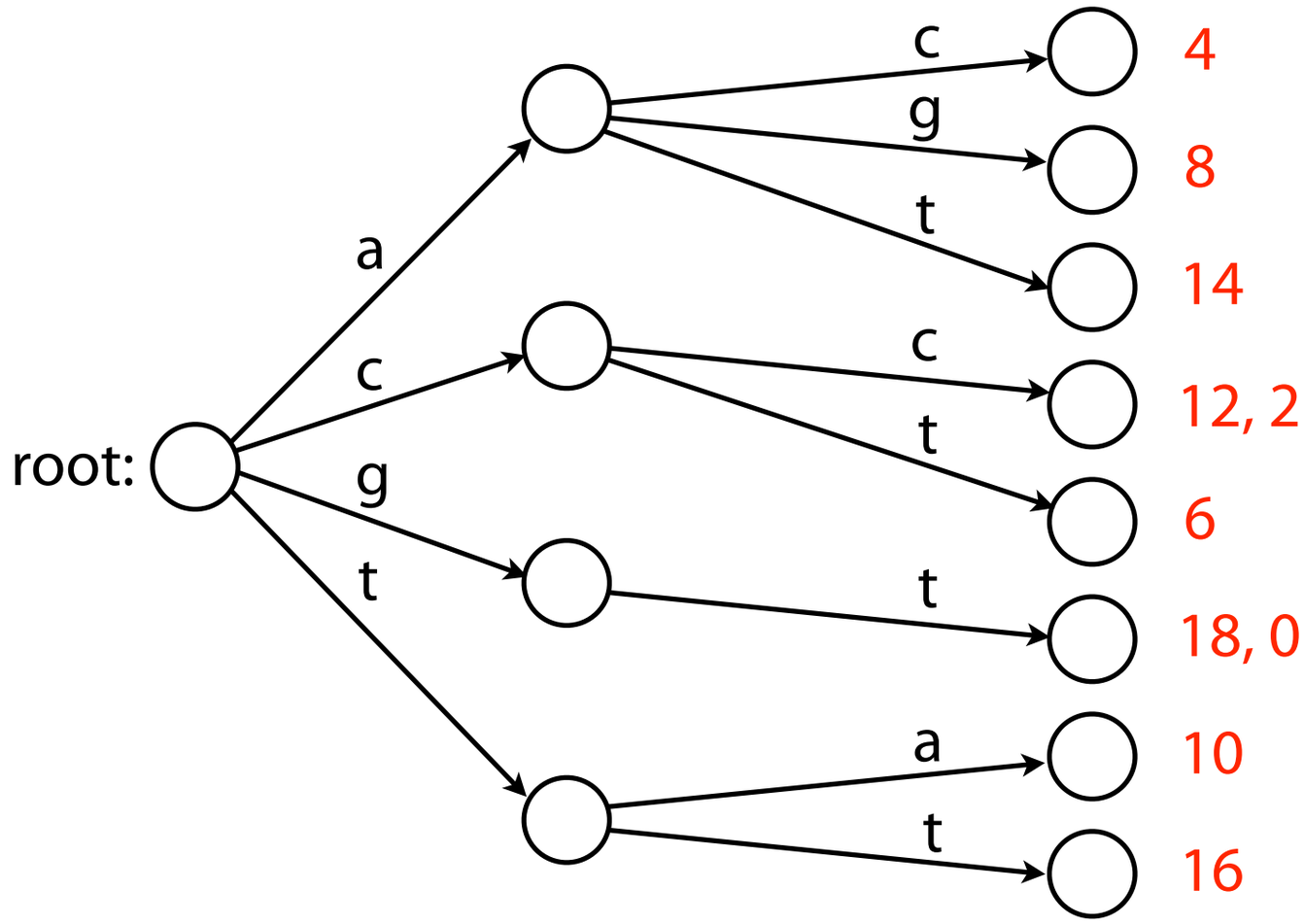
Assuming hash table, it's reasonable to say querying with P, $|P| = n$, is $O(n)$ time

Tries

Could use trie to represent *k*-mer index.
Map *k*-mers to offsets where they occur

ac	4
ag	8
at	14
cc	12
cc	2
ct	6
gt	18
gt	0
ta	10
tt	16

Index



Tries: alternatives

Tries aren't the only way to encode sets or maps over strings using a tree.

E.g. ternary search tree:

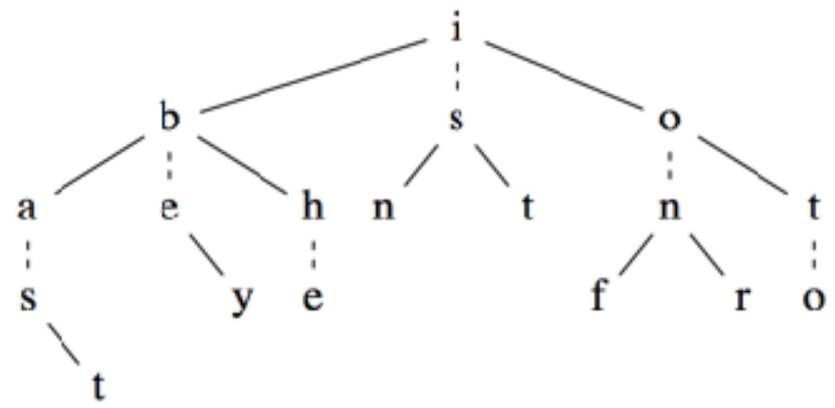


Figure 2. A ternary search tree for 12 two-letter words

Bentley, Jon L., and Robert Sedgewick. "Fast algorithms for sorting and searching strings." *Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 1997