Data Mining
資料探勘

Suffix Tree

Hung-Yu Kao

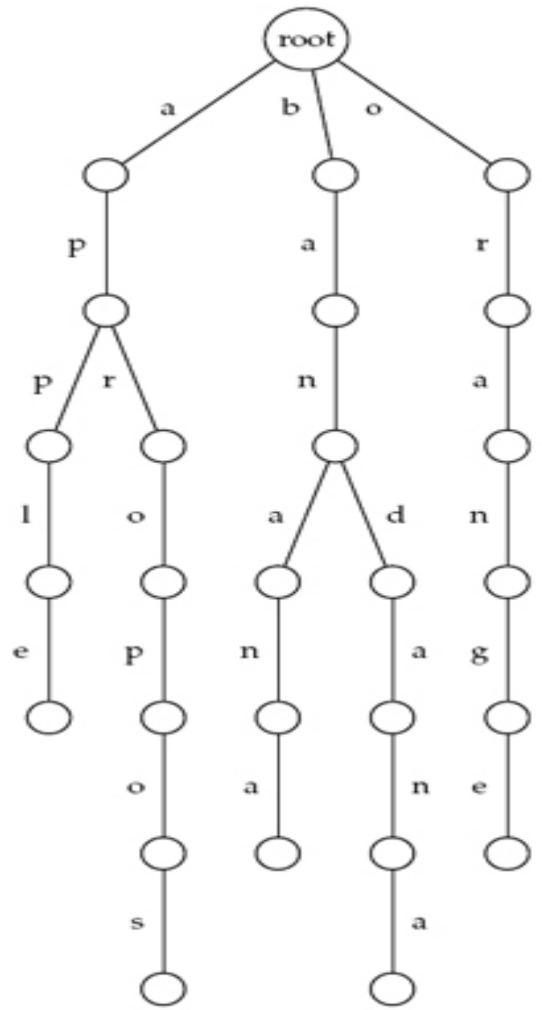
Keyword Trees

Properties of keyword trees:

- Stores a set of keywords in a rooted labeled tree
- Each edge labeled with a letter from an alphabet
- Any two edges coming out of the same vertex have distinct labels
- Every keyword stored can be spelled on **a path from root to some leaf**

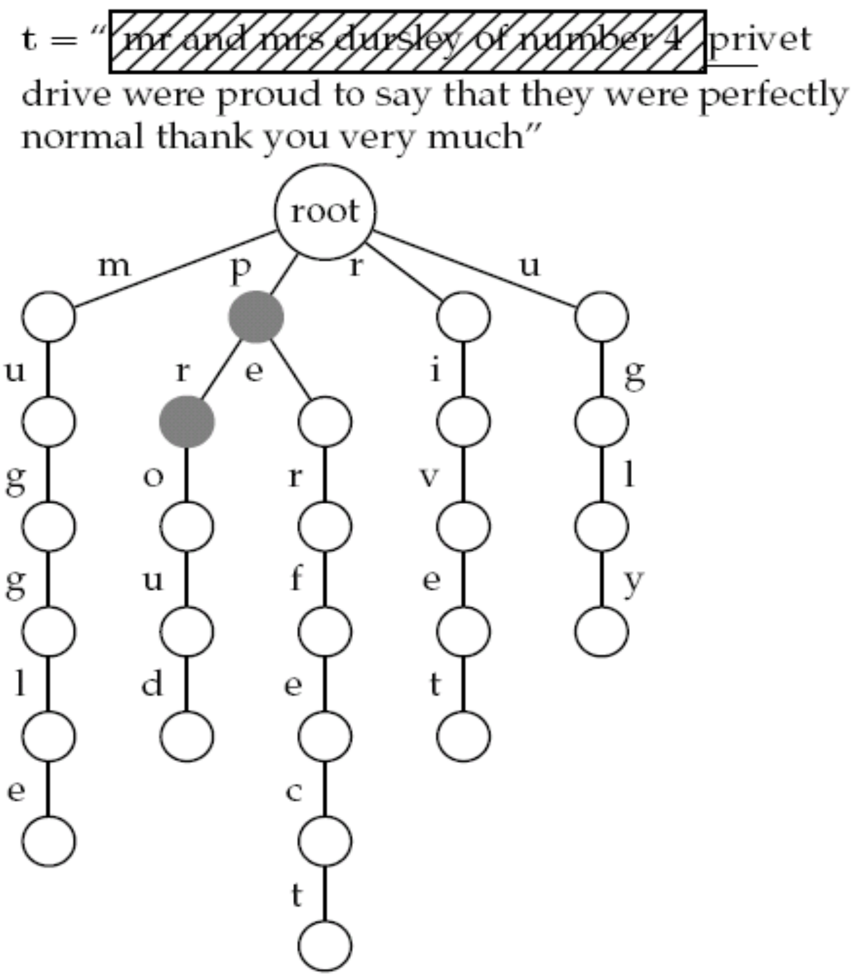
Keyword Trees: Example

- ***Keyword tree*:**
  - Apple
  - Apropos
  - Banana
  - Bandana
  - Orange



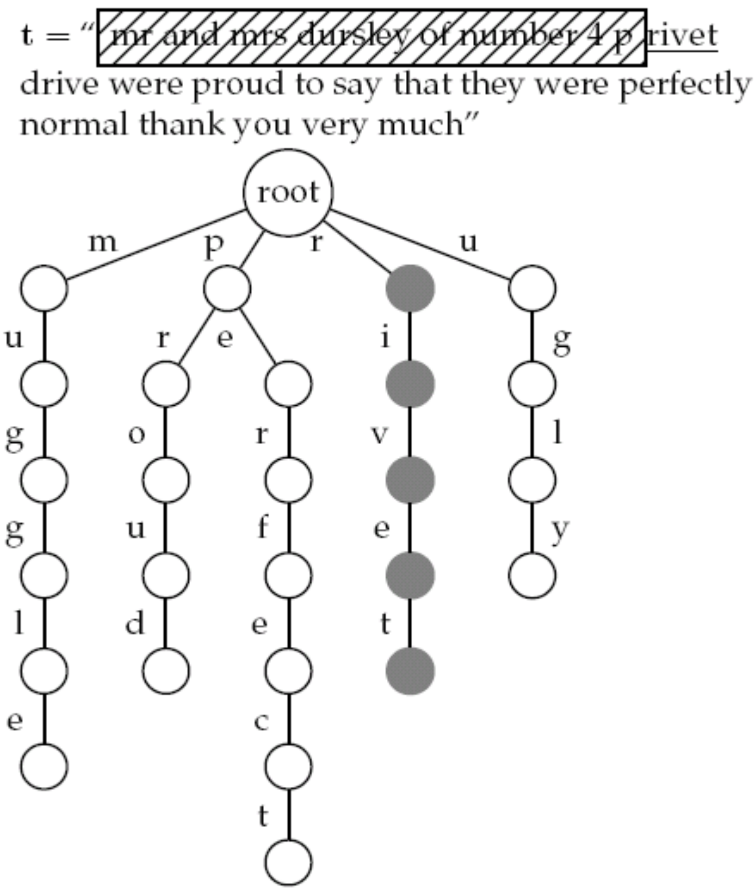
Keyword Trees: Threading

- To match patterns in a text using a keyword tree:
  - Build keyword tree of patterns
  - “**Thread**” the text through the keyword tree



Keyword Trees: Threading (cont.)

- Threading is "complete" when we **reach a leaf** in the keyword tree
- When threading is "complete," we've found a pattern in the text



Use of Keyword Trees

- Search for **multiple patterns** in a text at once
- Build keyword tree in O(N) time; N is total length of all patterns $N = k*n$
- Search in a document with length m
  - With naive threading: O(N + nm)
  - Aho-Corasick algorithm: O(N + m)

Suffix Tree

- Suffix Tree
  - A trie data structure built over the suffixes of the text

```
This is a text. A text has many words. Words are made from letters.
```

Text

```
text. A text has many words. Words are made from letters.
text has many words. Words are made from letters.
many words. Words are made from letters.
words. Words are made from letters.
Words are made from letters.
made from letters.
letters.
```

Suffixes

Suffix Tree

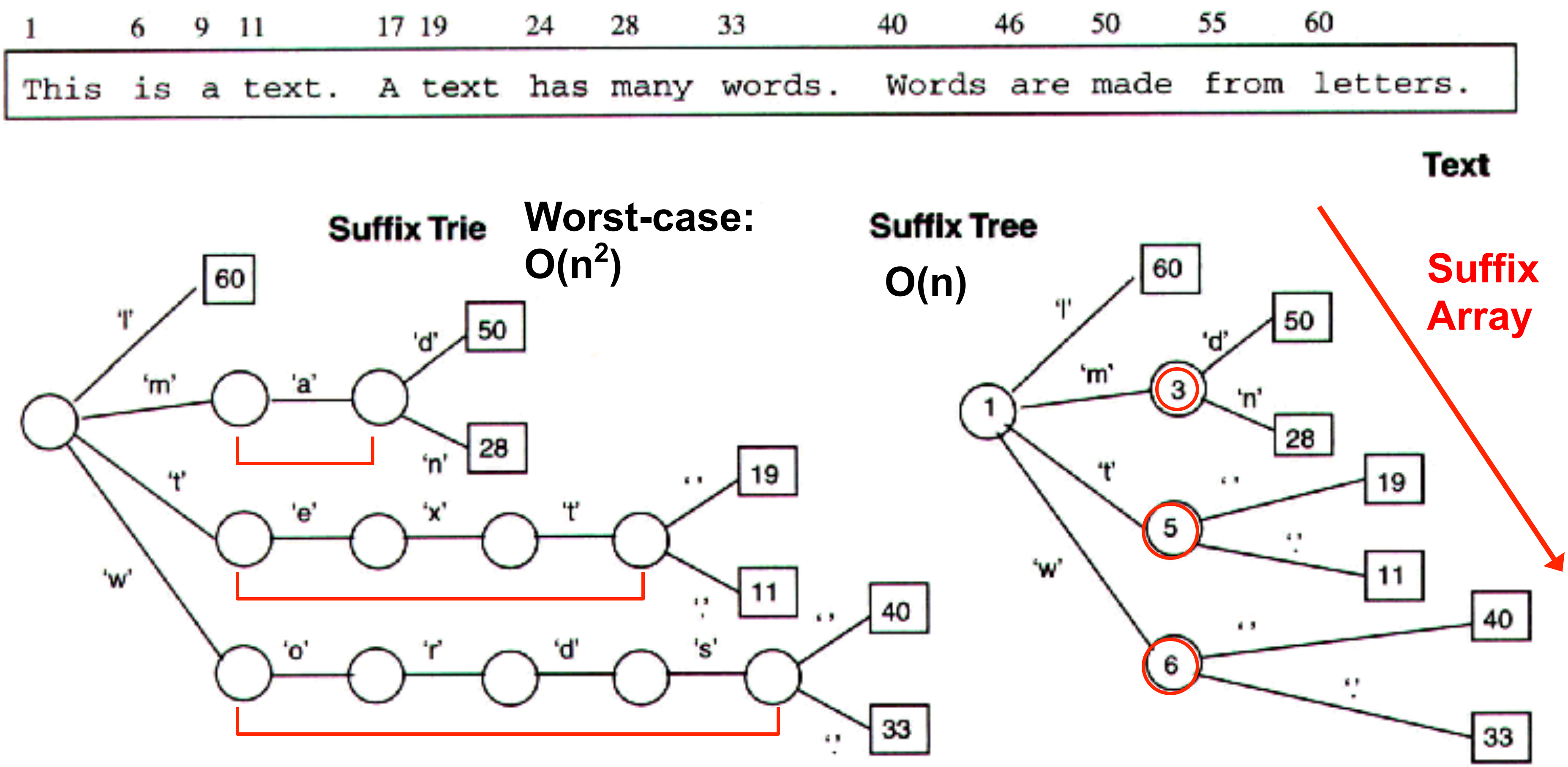
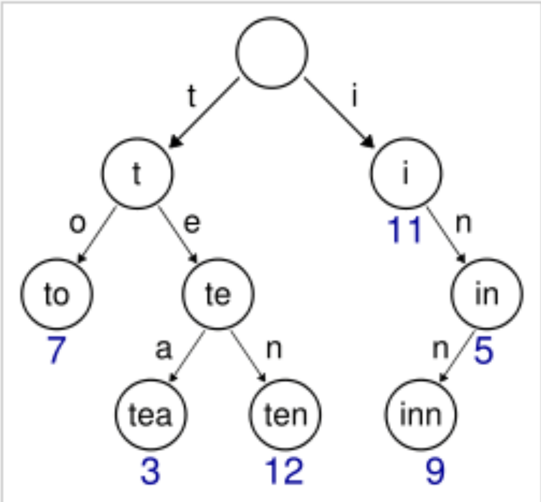


Figure 8.6 The suffix trie and suffix tree for the sample text.

Trie

- The term trie comes from "retrieval, from computer science
- **Prefix tree**, an ordered tree data structure
- used to store an associative array where the keys are strings

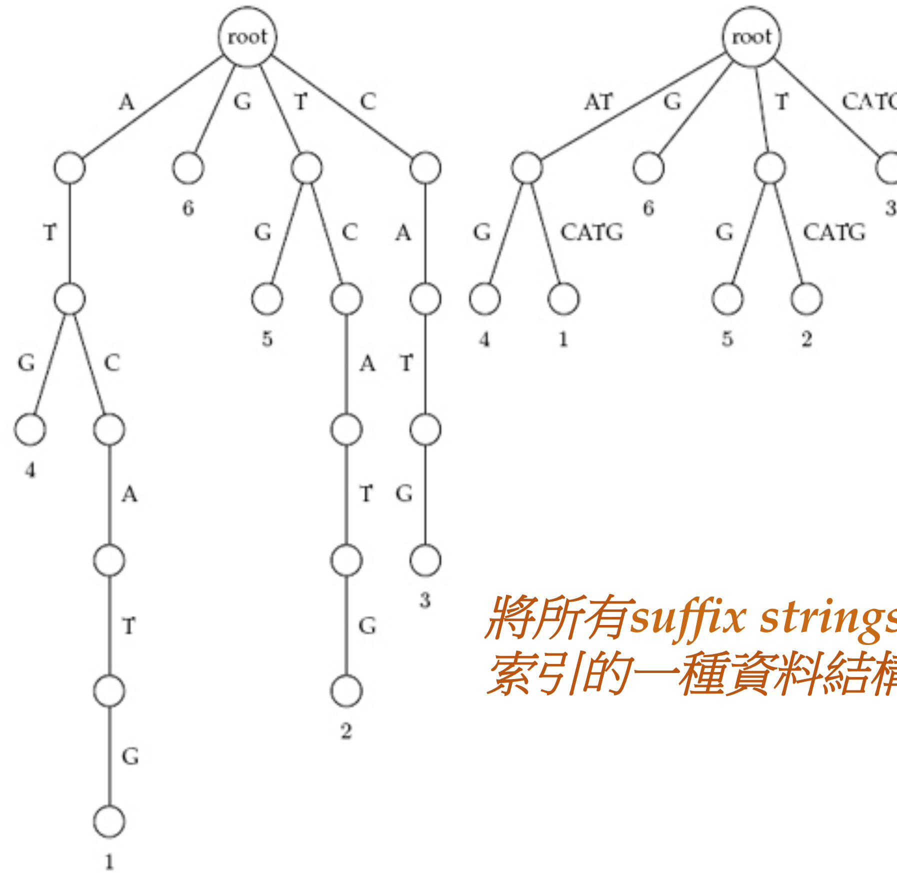


A **trie** for keys "to", "tea", "ten", "i", "in", and "inn".

http://en.wikipedia.org/wiki/Trie

Suffix Trees: Properties

- Similar to keyword trees, except **edges are condensed**
  - All internal edges have at least **two outgoing edges**
  - Leaves labeled with **suffix start position**



(a) Keyword tree

(b) Suffix tree

將所有suffix strings做索引的一種資料結構

Use of Suffix Trees

- Suffix trees hold all suffixes of a text
  - i.e., ATCGC: ATCGC, TCGC, CGC, GC, C
  - Builds in **O(*m*)** time for text of length ***m***
- To find any pattern of length ***n*** in a text:
  - Build suffix tree for text
  - Thread the pattern through the suffix tree
  - Can find pattern in text in **O(*n*)** time!
    - Compare with **O(m*n*) by a direct sequential search**
    - Knuth-Morris-Pratt algorithm also O(n)
- O(*n* + *m*) time for "Pattern Matching Problem"
  - Build suffix tree and lookup pattern

Suffix Trees: Example

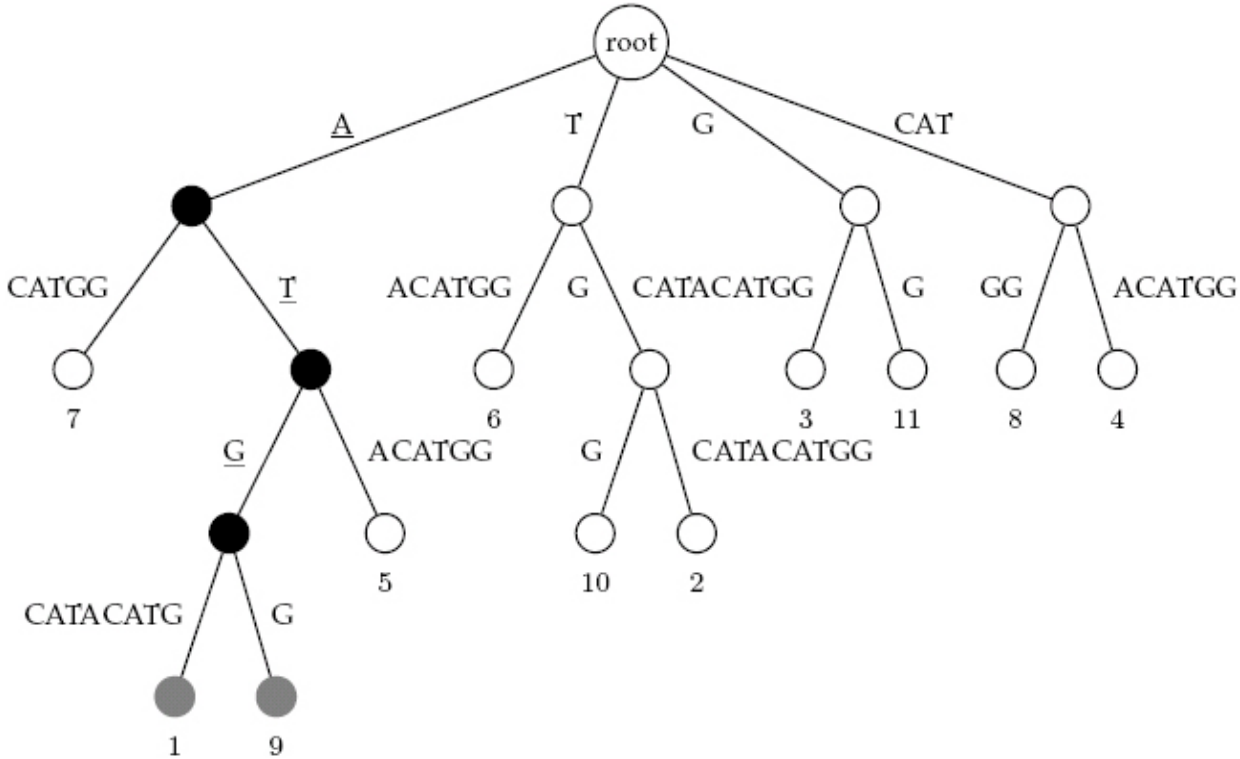


Figure 9.6 Threading the pattern ATG through the suffix tree for the text ATGCATACATGG. The suffixes ATGCATACATGG and ATGG both match, as noted by the gray vertices in the tree (the p-matching leaves). Each p-matching leaf corresponds to a position in the text where p occurs.

Suffix Tree – Space, Time

- Suffix Tree
  - Each node of the trie takes 12 to 24 bytes
    - A funny research topic
  - Only word beginnings are indexed:120% to 240% over the text size
  - Other information
    - The position
    - DocID: consider more than two documents
- Prefix Tree is still in the same case ?

Suffix Search

- ***Range Search***
  - Given two patterns “P1” and “P2”, find words between P1 and P2
    - Suffix array: perform binary search for P1 and P2, pointers between both results are the answer
    - Suffix tree: pointers between two subtrees
- Phrase Search: the length of word beginnings
- ***Proximity Search***