Strings: Tries and Suffix Trees

Papamakarios Theodoros

January 29, 2015

String matching: Given text $T \in \Sigma^*$ and pattern $P \in \Sigma^*$, find some/all occurences of P in T as a substring.

- One shot: $O(T)$ time (Knuth, Morris, Pratt; Boyer & Moore; Karp & Rabin).
- Static Data Structures: Preprocess T, query $= P$.
  Goal:
  - $O(P)$ time per query.
  - $O(T)$ space.

Given k strings $T_1, \ldots, T_k$ and query P, determine where P fits among the k strings in lexicographical order.

Trie: Rooted tree with child branches labeled with letters in Σ.

Given k strings $T_1, \ldots, T_k$ and query P, determine where P fits among the k strings in lexicographical order.

> **Trie**: Rooted tree with child branches labeled with letters in Σ.

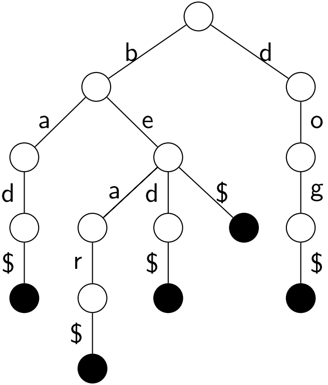
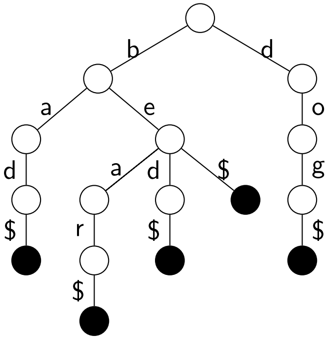


Figure : Trie representation of {bear, bed, be, bad, dog}.



Trie representation. Node stores children as: (T $=$ # nodes in trie $\leq \sum_{i=1}^{k} |T_i|$)

	query	size
array	$O(P)$	$O(T\Sigma)$
linked lists	$O(P\Sigma)$	$O(T)$
BST	$O(P \log \Sigma)$	$O(T)$
suffix trays	$O(P + \log \Sigma)$	$O(T)$

Application: Sorting strings $T_1, \dots, T_k$

Repeatedly insert into trie/tray

$\Rightarrow O(\sum_{i=1}^{k}(T_i + \log \Sigma)) = O(T + k \log \Sigma)$ time

$(<< O(Tk \log k))$ via comparisons)

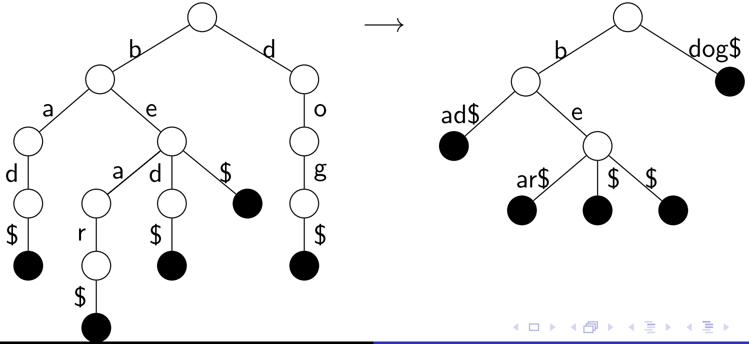
Application: Sorting strings $T_1, \ldots, T_k$

Repeatedly insert into trie/tray

$\Rightarrow O(\sum_{i=1}^{k}(T_i + \log \Sigma)) = O(T + k \log \Sigma)$ time

($<< O(Tk \log k)$) via comparisons)

Compressed Trie: Contract nonbranching paths to single edge (# nodes $= O(k)$).



Suffix Tree: Compressed trie of all $|T|$ suffixes of T (with \$ appended)

- $|T| + 1$ leaves
- edge label $=$ substring $T[i:j]$ of T
    - store as two indices $(i, j) \Rightarrow O(T)$ space

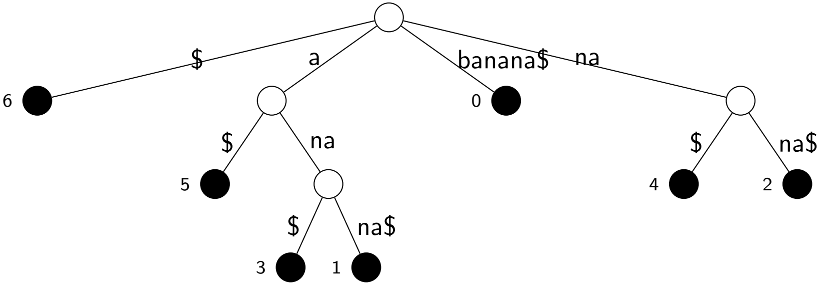
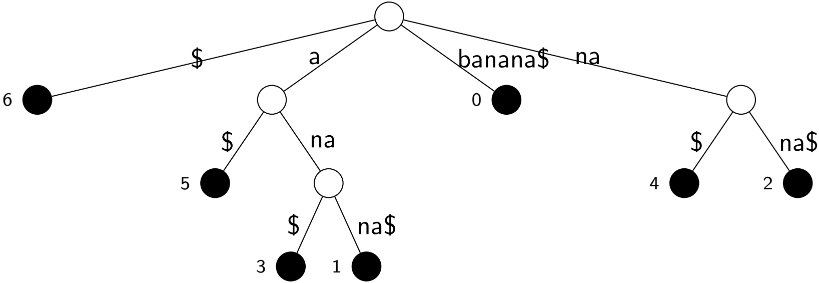


Figure : Suffix tree for $T =$

b	a	n	a	n	a	\$
0	1	2	3	4	5	6



b	a	n	a	n	a	$
0	1	2	3	4	5	6

Applications:

- search for P gives subtree whose leaves correspond to all occurrences of P
  - $O(P)$ time via hashing
  - $O(P + \log \Sigma)$ via trays ($\Rightarrow$ leaves sorted in T)
- list first k occurrences in $O(k)$ more time

Applications:

- Given two indices i and j, find the longest common prefix of $T[i:]$ and $T[j:]$
  - least common ancestor
- multiple documents via multiple \$s: $T = T_1\$_1 \dots T_k\$_k$
- document retrieval
- many more...

Suffix trees can be built in $O(T)$ time

Erik Demaine.
Lecture notes on advanced data structures.
http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-851-advanced-data-structures-spring-2012/calendar-and-notes/, 2012.