Suffix Arrays

Suffix Arrays

- A **suffix array** for a string *T* is an array of the suffixes of *T*$, stored in sorted order.
- By convention, $ precedes all other characters.

8	$
7	e$
4	ense$
0	nonsense$
5	nse$
2	nsense$
1	onsense$
6	se$
3	sense$

Representing Suffix Arrays

- Suffix arrays are typically stored as an array of the start positions of the suffixes.
- Space required: $\Theta(m)$.
- More precisely, space for $T\$$, plus one extra word for each character.

8
7
4
0
5
2
1
6
3

nonsense$

Searching a Suffix Array

- **Recall:** P is a substring of T iff it's a prefix of a suffix of T.
- All matches of P in T have a common prefix, so they'll be stored consecutively.
- Can find all matches of P in T by doing a binary search over the suffix array.

8	$
7	e$
4	ense$
0	nonsense$
5	nse$
2	nsense$
1	onsense$
6	se$
3	sense$

nse

Analyzing the Runtime

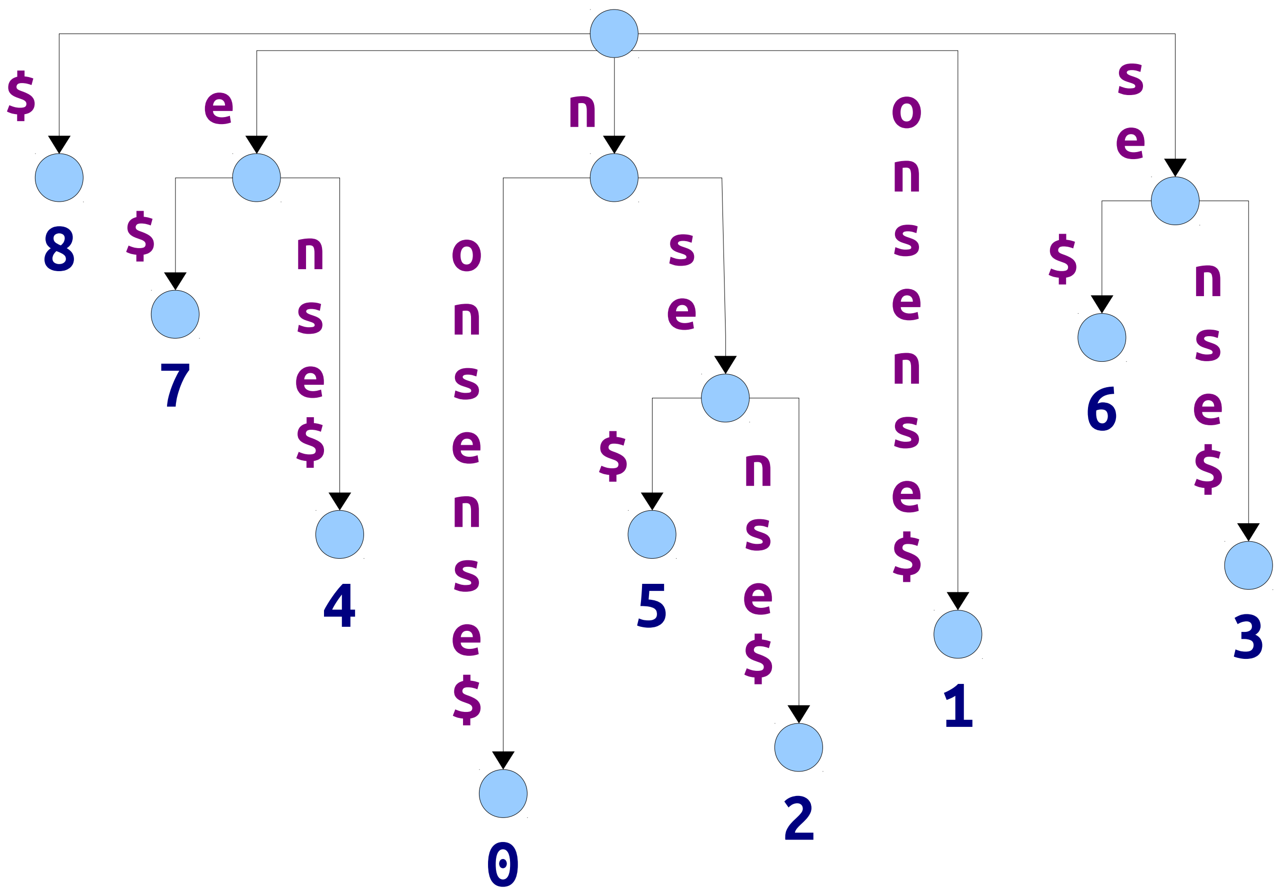
- The binary search will require O(log m) probes into the suffix array.
- Each comparison takes time O(n): have to compare P against the current suffix.
- Time for binary searching: O(n log m).
- Time to report all matches after that point: O(z).
- Total time: **O(n log m + z)**.

A Useful Observation

A Loss of Structure

- Many algorithms on suffix trees involve looking for internal nodes with various properties:
  - Longest repeated substring: internal node with largest string depth.
  - Longest common extension: lowest common ancestor of two nodes.
- Because suffix arrays do not store the tree structure, we lose access to this information.

Suffix Trees and Suffix Arrays

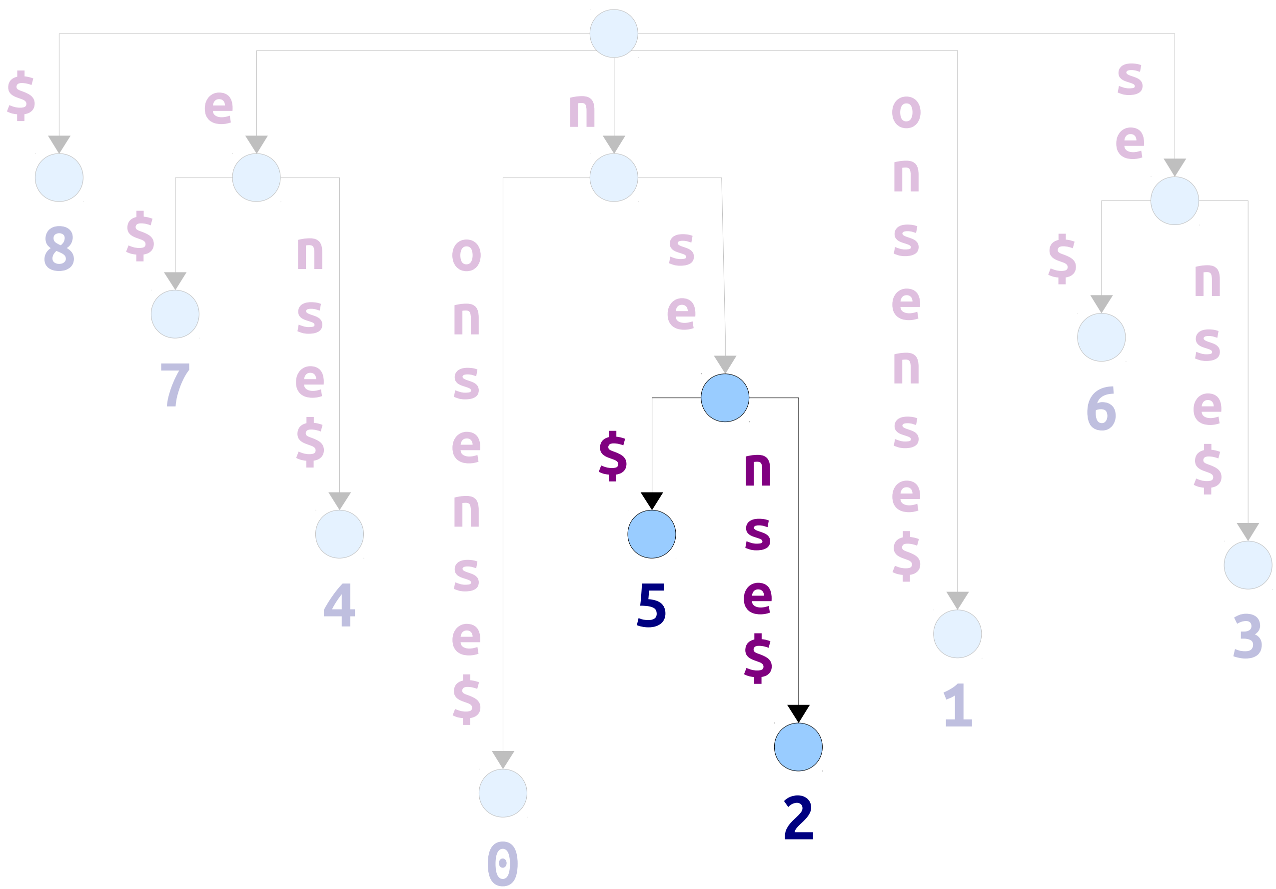


8	$
7	e$
4	ense$
0	nonsense$
5	nse$
2	nsense$
1	onsense$
6	se$
3	sense$

nonsense$
012345678

Nifty Fact: The suffix array can be constructed from an ordered DFS over a suffix tree!

Suffix Trees and Suffix Arrays



8	$
7	e$
4	ense$
0	nonsense$
5	nse$
2	nsense$
1	onsense$
6	se$
3	sense$

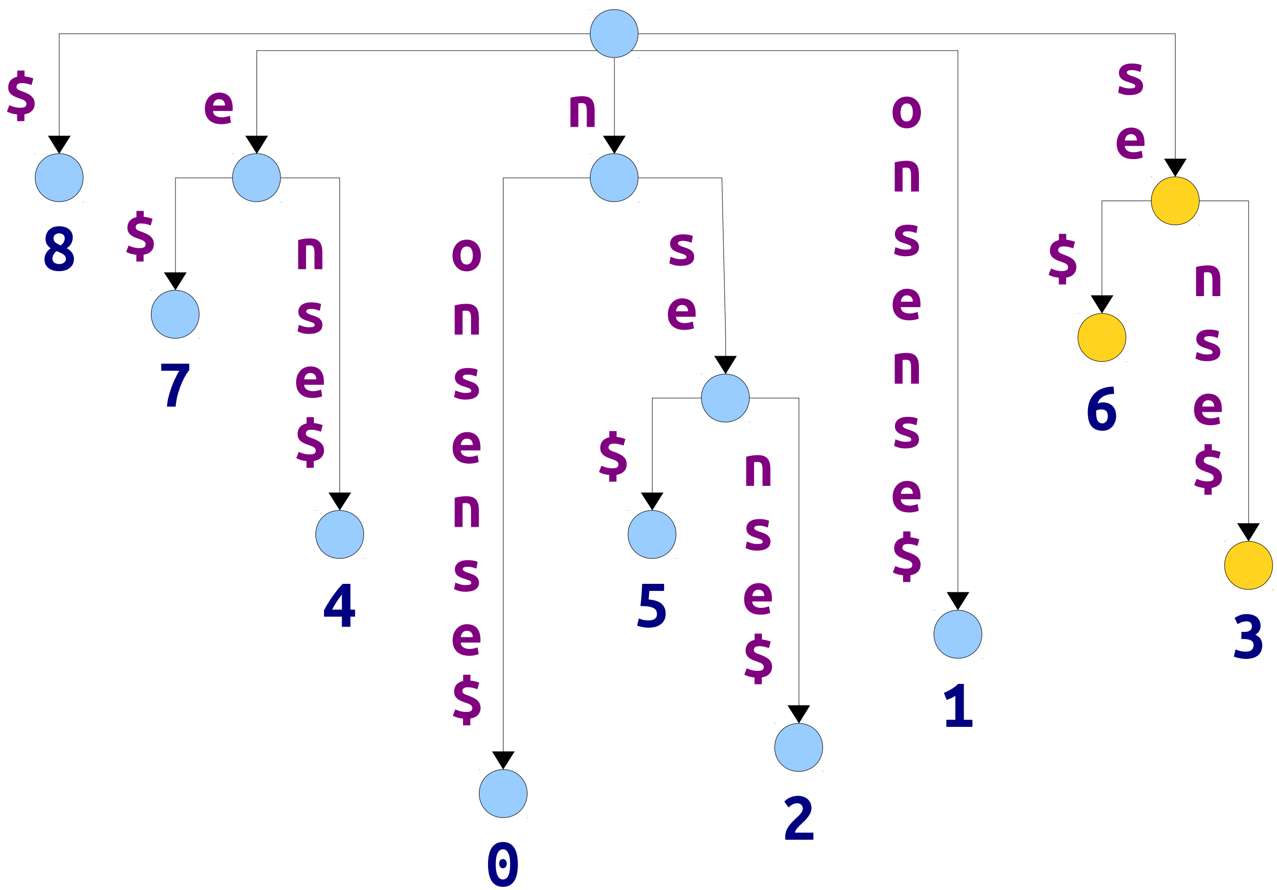
nonsense$
012345678

Nifty Fact: Adjacent strings with a common prefix correspond to subtrees in the suffix tree.

Longest Common Prefixes

- Given two strings x and y, the **longest common prefix** or (**LCP**) of x and y is the longest prefix of x that is also a prefix of y.
- The LCP of x and y is denoted $\mathrm{lcp}(x, y)$.
- LCP information is a fundamental link between suffix trees and suffix arrays.

Suffix Trees and Suffix Arrays



8	$
7	e$
4	ense$
0	nonsense$
5	nse$
2	nsense$
1	onsense$
6	se$
3	sense$

nonsense$
012345678

Nifty Fact: The lowest common ancestor of suffixes *x* and *y* has string label given by lcp(*x*, *y*).

Pairwise LCP

- **Fact:** There is an algorithm (due to Kasai et al.) that constructs, in time O(m), an array of the LCPs of adjacent suffix array entries.
- Check the paper for details; note that there's a typo in their pseudocode; "`j + h`" should be "`k + h`."

LCP	SA	Suffix
	8	$
0	7	e$
1	4	ense$
0	0	nonsense$
1	5	nse$
3	2	nsense$
0	1	onsense$
0	6	se$
2	3	sense$