Please sign guestbook (www.langmead-lab.org/teaching-materials) to tell me briefly how you are using the slides. For original Keynote files, email me (ben.langmead@gmail.com).
Suffix tree: building

Method 1: build suffix trie
Suffix tree: building

**Method 1:** build suffix trie, coalesce non-branching paths
Suffix tree: building

Method 1: build suffix trie, coalesce non-branching paths, label

$O(m^2)$ time, $O(m^2)$ space
Suffix tree: building

Instead of starting with a big trie and making it smaller, we can start from scratch and "grow" the tree.

At no point do we have something larger than the final tree, so it must be \(O(m)\) at all points.
Suffix tree: building

Build single-edge tree with longest suffix

Add 2\textsuperscript{nd}-longest

Add 3\textsuperscript{rd}-longest

\( (0, 7) \)

\( (0, 7) \) \( (1, 6) \)

\( (0, 1) \)

\( (3, 4) \) \( (1, 6) \)

\( (1, 6) \)

\( (1, 6) \)

\( (0, 7) \)

\( (0, 1) \)

\( (0, 1) \)

\( (1, 6) \)

\( (1, 6) \)

\( (1, 6) \)

\( (0, 7) \)

\(...\)

\( (0, 7) \)

\( (0, 7) \) \( (1, 6) \)

\( (0, 1) \)

\( (3, 4) \) \( (1, 6) \)

\( (1, 6) \)

\( (1, 6) \)

\( (0, 7) \)

\(...\)

\( (0, 7) \)

\( (0, 7) \) \( (1, 6) \)

\( (0, 1) \)

\( (3, 4) \) \( (1, 6) \)

\( (1, 6) \)

\( (1, 6) \)

\( (0, 7) \)

\(...\)

\( (0, 7) \)

\( (0, 7) \) \( (1, 6) \)

\( (0, 1) \)

\( (3, 4) \) \( (1, 6) \)

\( (1, 6) \)

\( (1, 6) \)

\( (0, 7) \)

\(...\)

\( (0, 7) \)

\( (0, 7) \) \( (1, 6) \)

\( (0, 1) \)

\( (3, 4) \) \( (1, 6) \)

\( (1, 6) \)

\( (1, 6) \)

\( (0, 7) \)

\(...\)

\( (0, 7) \)

\( (0, 7) \) \( (1, 6) \)

\( (0, 1) \)

\( (3, 4) \) \( (1, 6) \)

\( (1, 6) \)

\( (1, 6) \)

\( (0, 7) \)

\(...\)
Suffix tree: building

Few steps of “method 2”

$T = \text{abaaba}\$

| Just longest | Longest + 2$^{\text{nd}}$ | + 3$^{\text{rd}}$ |
Suffix tree: building

Few steps of “method 2”

Just longest

Longest + 2\textsuperscript{nd}

T = abaaba$

+ 3\textsuperscript{rd}
Suffix tree: building

Few steps of “method 2”

T = abaaba$

Just longest

Longest + 2\text{nd}

+ 3\text{rd}
Suffix tree: building

Few steps of “method 2”

\( T = \text{abaaba}\$ \)

Just longest

Longest + 2\(^{\text{nd}}\)

+ 3\(^{\text{rd}}\)
Suffix tree: building

Few steps of “method 2”

Just longest

Longest + 2\textsuperscript{nd}

Longest + 3\textsuperscript{rd}

$T = \text{abaaba}\$

Each step adds 1 or 2 new nodes

Possibly 1 internal

1 leaf
Suffix tree: building

Though tree only grows by 1 or 2 nodes in each step, work is dominated by the need to "walk down" for each suffix

$O(m^2)$ time overall
Suffix tree: building

Method 2: build single-edge tree representing longest suffix, augment to include the 2\textsuperscript{nd}-longest, augment to include 3\textsuperscript{rd}-longest, etc

$O(m^2)$ time, $O(m)$ space
Ukkonen's algorithm

On-Line Construction of Suffix Trees

E. Ukkonen

Abstract. An on-line algorithm is presented for constructing the suffix tree for a given string in time linear in the length of the string. The new algorithm has the desirable property of processing the string symbol by symbol from left to right. It always has the suffix tree for the scanned part of the string ready. The method is developed as a linear-time version of a very simple algorithm for (quadratic size) suffix tries. Regardless of its quadratic worst case this latter algorithm can be a good practical method when the string is not too long. Another variation of this method is shown to give, in a natural way, the well-known algorithms for constructing suffix automata (DAWGs).

Key Words. Linear-time algorithm, Suffix tree, Suffix trie, Suffix automaton, DAWG.

Ukkonen's algorithm

Too complex to detail here, but you should know:

- $O(m)$ time and space

It is the most widely used, though not the only algorithm with $O(m)$ time & space

Also starts from scratch and builds up to full tree

Uses and outputs "suffix links," to be discussed later

Building suffix trees: summary

Good methods exist that build the suffix tree incrementally from scratch, eventually reaching the $O(m)$-size tree.

Our "Method 2" does this quite simply, but needs $O(m^2)$ time.

Ukkonen's algorithm is the most widely used.