Infix-querying Sequences Efficiently with Suffix Trees

Rasmus Resen Amossen
SOLUTION ARCHITECT

rasmus.resen.org

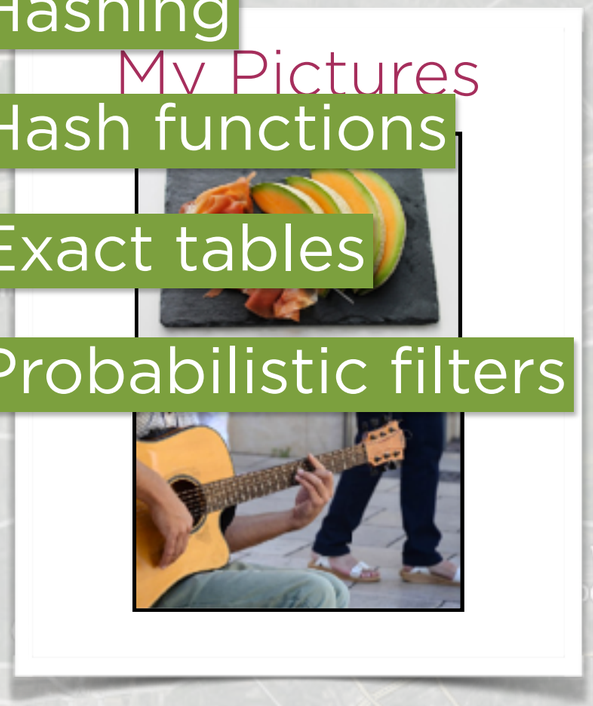
The Match Finder App

Hashing

Hash functions

Exact tables

Probabilistic filters



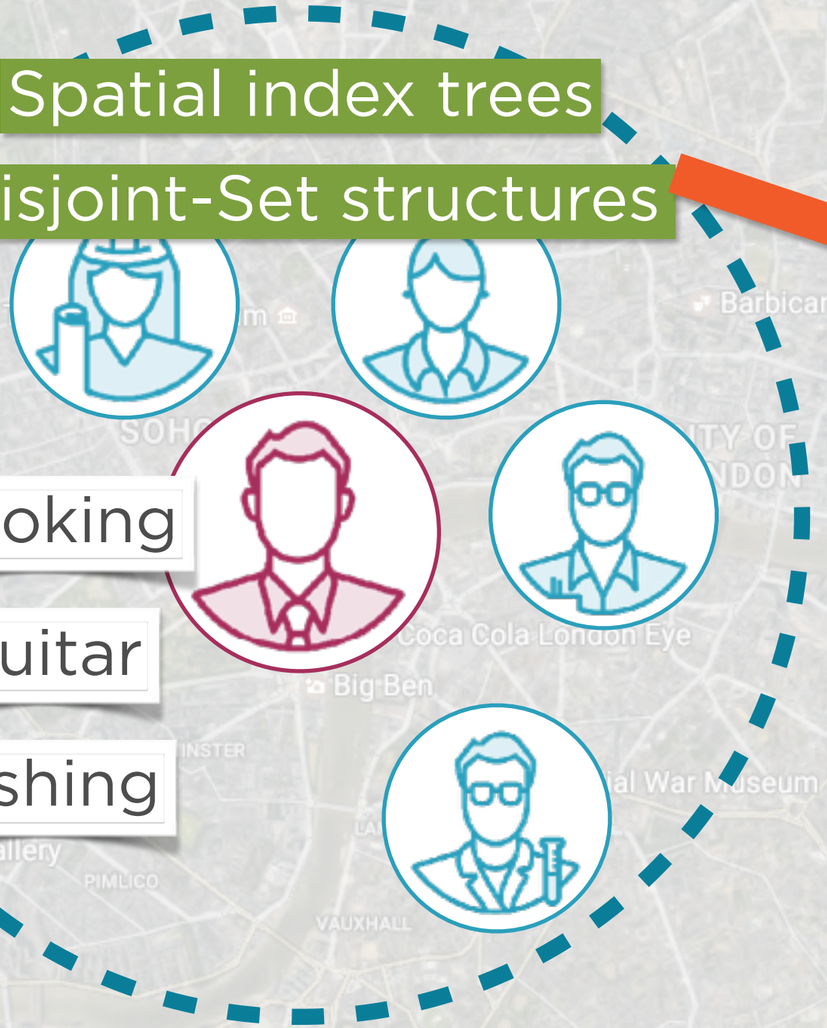
Spatial index trees

Disjoint-Set structures

Cooking

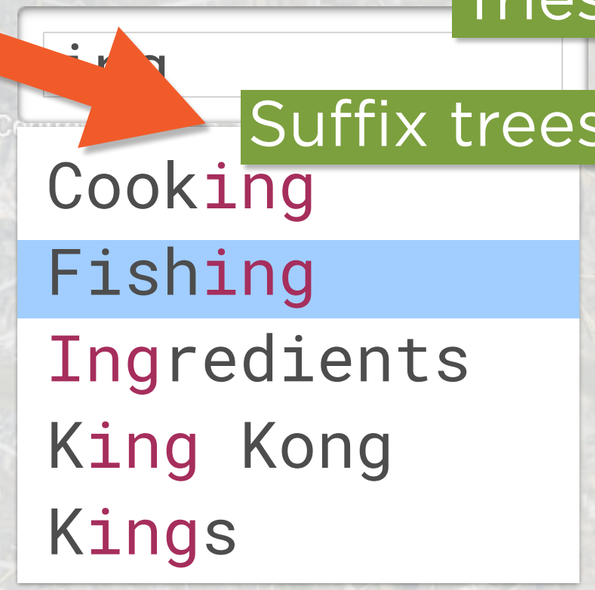
Guitar

Fishing



Tries

Suffix trees



Tag Suggestions

From Stack Overflow

Tags

atal

magento × 36635

Magento is an e-commerce platform written in PHP atop the Zend framework, available under both open-source and commercial licenses.

also:magento-catalog

fatal-error × 1602

An error that causes a program to abort, regardless of the programming language.

datalist × 316

An ASP.NET control for rendering data in a list.

adobe-analytics × 587

Adobe Analytics provides real-time analytics for users across many marketing channels. Use this tag for all questions related to the user interface, interpretation of reports, and implementation coding (including DTM).

also:site-catalyst

catalyst × 487

Catalyst is a Perl web application framework, similar to Ruby on Rails, Spring (Java), and Maypole.

azure-data-lake × 476

Azure Data Lake Analytics is a suite of three big data services in Microsoft Azure: HDInsight, Data Lake Store, and Data Lake Analytics. These fully managed services make it easy to get started and easy to scale big data jobs written in Hive, Pig, Spark, Storm, and U-SQL. To learn more, check out: https://azure.microsoft.com/en-us/solutions/data-lake/

Demo

Beginning Keyword Suggestions

Important Observation

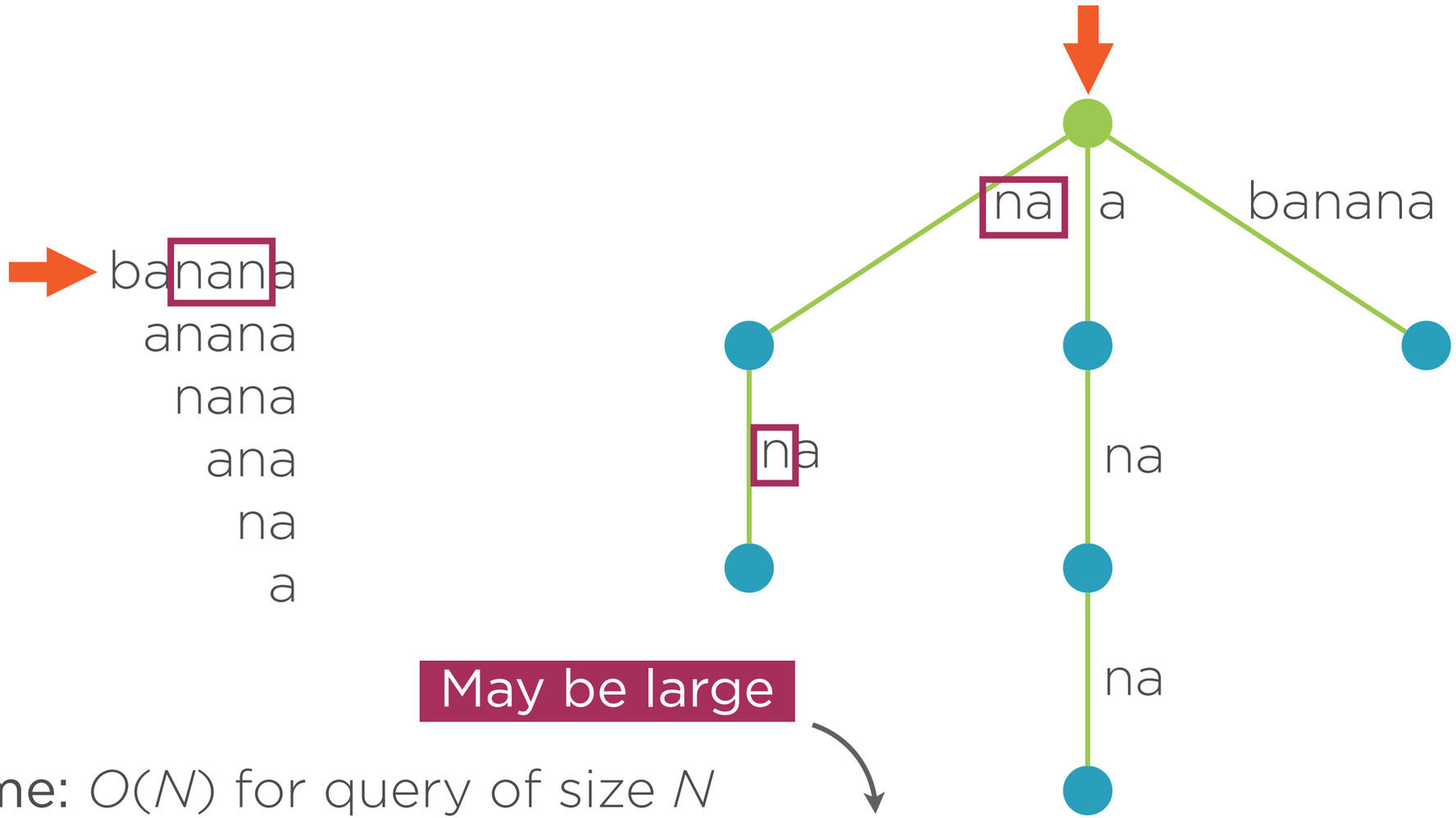
Pluralsight
luralsight
uralsight
ralsight
alsight
lsight
sight
ight
ght
ht
t

Any **infix**
is a **prefix**
of some **suffix**

Conceptually:

1. Put all suffixes in a trie
2. Search the trie
   for prefix matches

Suffixes in a Compressed Trie



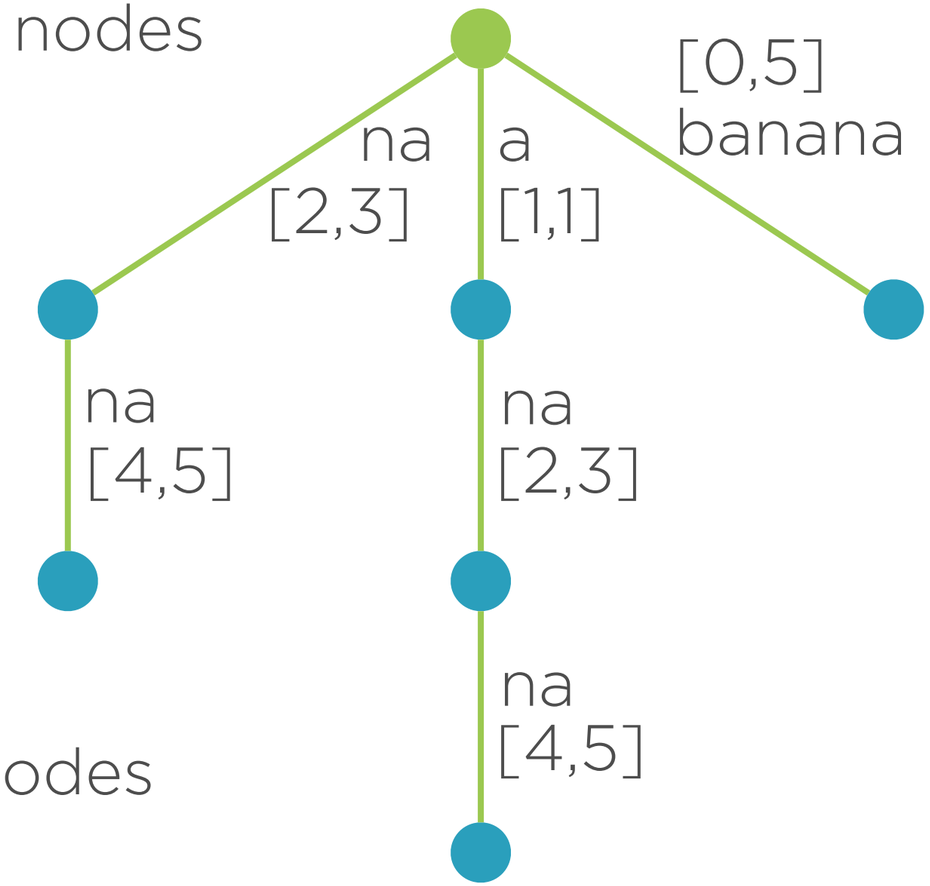
Query time: $O(N)$ for query of size N
Query time with subtree fetch: $O(N + |\text{subtree}|)$
Space: A word with L letters occupies $O(L^2)$ space

Saving Space

Final words are *internal* nodes

Implicit suffix tree

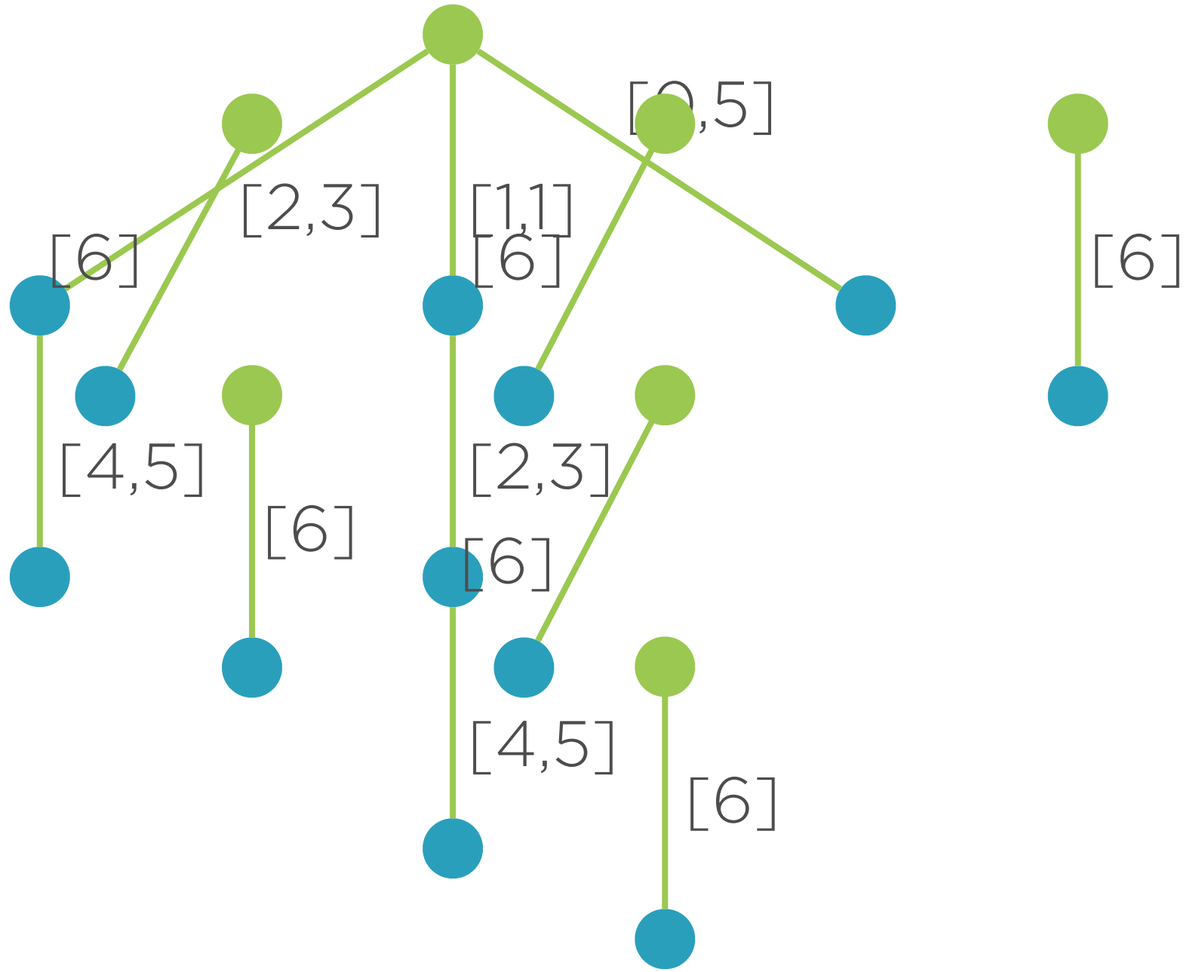
012345
banana



A word with L letters have L final nodes
At most $O(L)$ non-final nodes
Total space: $O(L)$

Suffix Trees

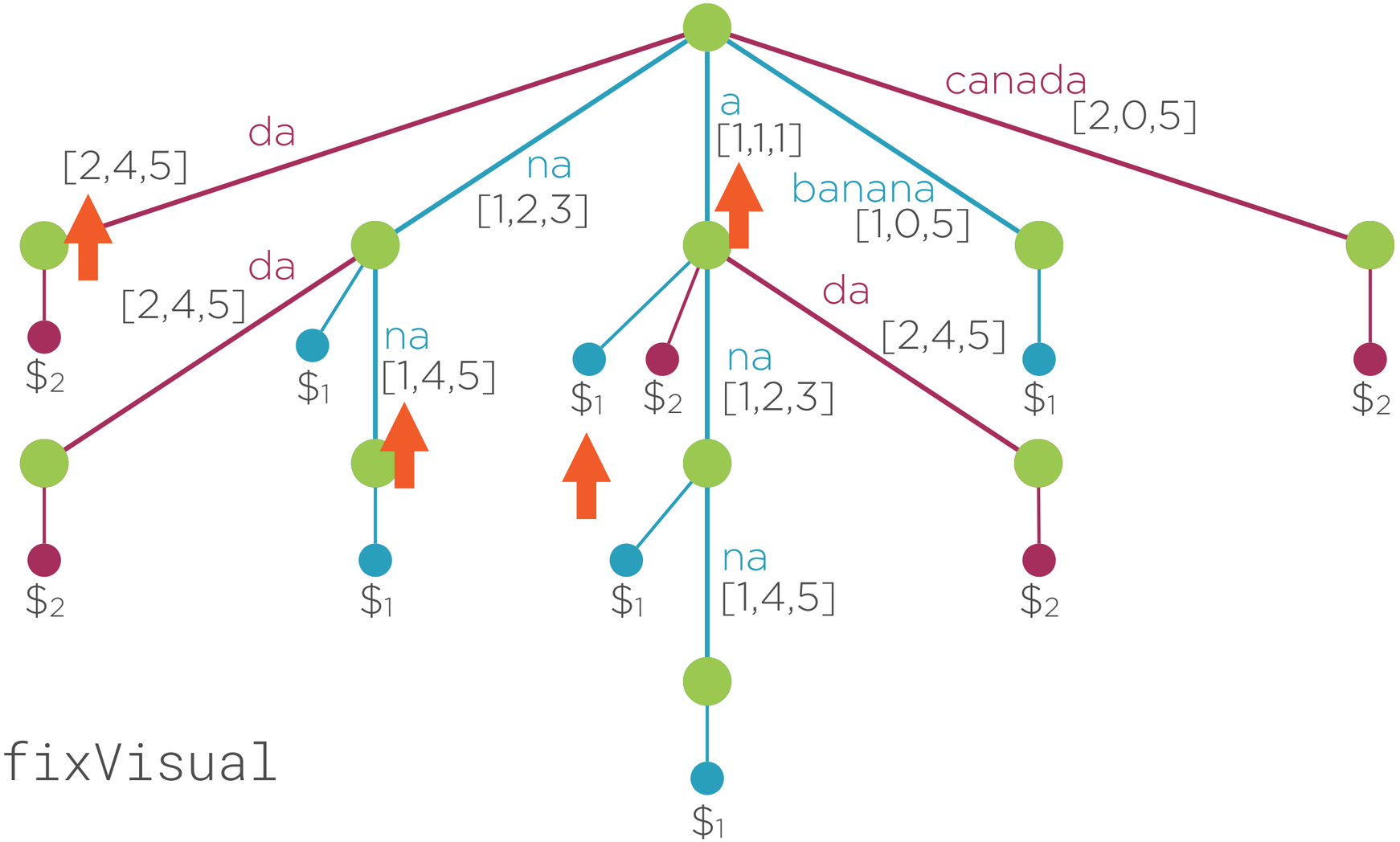
0123456
banana$



Generalized Suffix Tree

1: 012345
banana

2: 012345
canada



tinyurl.com/suffixVisual

Building a Suffix Tree

For a total of N characters...

1. Build trie, then compress it
   $O(N^2)$ characters in total
   Complexity: $O(N^2)$

Building a Suffix Tree

For a total of *N* characters...

2

Esko Ukkonen's algorithm from 1995

Complexity: *O*(*N*) for constant sized alphabets

Algorithm: `tinyurl.com/ukkonenDuke`

Original paper: `tinyurl.com/suffixUkkonen`

See it: `tinyurl.com/ukkonenVisual`

Building a Suffix Tree

For a total of *N* characters...

3

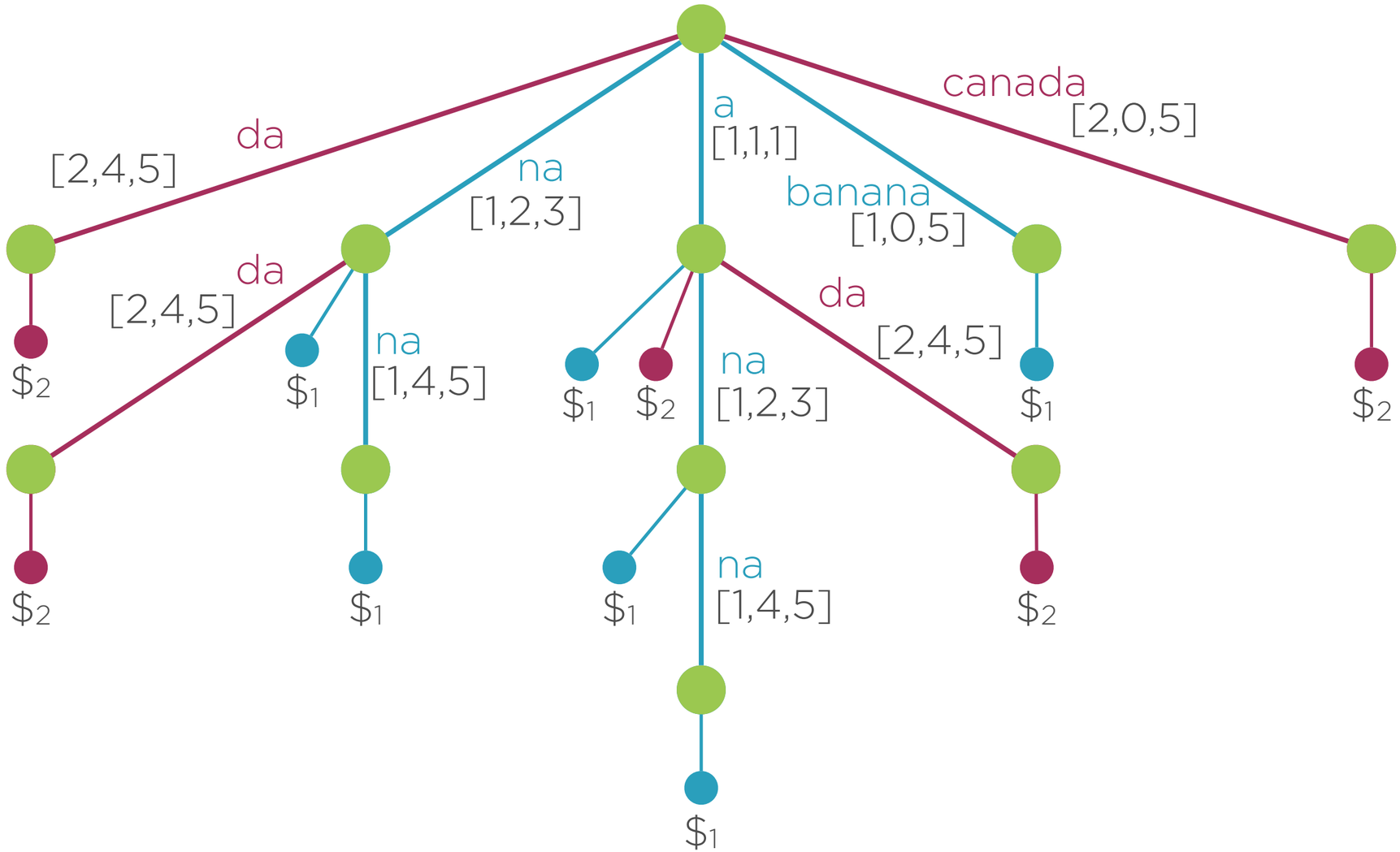
Martin Farach's algorithm from 1997

Complexity: *O*(*N*) for all alphabets

Original paper: tinyurl.com/suffixFarach

Demo

Implementing Keyword Suggestions


da
[2,4,5]
$2
na
[1,2,3]
da
[2,4,5]
$2
$1
na
[1,4,5]
$1
a
[1,1,1]
$1
$2
na
[1,2,3]
$1
na
[1,4,5]
$1
banana
[1,0,5]
da
[2,4,5]
$1
$2
canada
[2,0,5]
$2

2 examples...

Allowing Mismatches

`There are many courses at the **Pluralsight** web site`

Edit distance
(or Levenshtein distance)

Insert

Delete

Substitute

`Pluralsiht`

Edit distance = 2
(to “PQuralsght”)

Non-trivial problem!

Ukkonen uses suffix trees and dynamic programming

See `tinyurl.com/approxMatch`

Longest Common Substring

Longest Common Substring

Human DNA:
3.3 GB

DNA of suspect

ACAGTTTCGATGCTTAACCCTTGACCAGCTCGACCGAAGAATGCGCCGTGATGGCCTCTC
TGTCAACGACCCACACTTAAAGGAATATTGGGCATACGCATATACGCTTGACGGCTCAGC
ATACGTGCTCGCCGAAGCCCCCACTGACGATTAAGCACTAGCATGAGCGCTTGCGATGTG
AGAGTTATAATCATGATCGAGCGTCTGCGTGGATACCACCGCTGCAGGTGATTCCACCCC
ATTCTGGCCCCAAATATTCTGTTACCCTCACCATACTCGCAGCCTTGTTGTCTATGTGAG
TAAACAGTTTATAGACCCTACGAAAATAACGGCGGGATCAAATTTATAAGTCGTCATCTC
TCTCGCGACCCACATAGATCCCGACTCAGTCAGATCGACGCGCTCAAGGACATTGCCACG
CAGCGCAGATGTCGGCATACCTCACTACATTAACAATTTCATTTGTTTTGTTCTATAAGT
GACGCAGATCGTTTGCGTTCTTCCCTTGCTATTATTAGTTATTTTGGACGGTATAGCCGC
GGGGGAGGTTGAAACAAGGTGCGACGGATGGTCTGAGCATTTGTTTACTCATAACCTGTA
ACGAACCAGCCCAGTGGGAAAGTCTTTTTAGGCAGTGTAGGAGGTCAAATTATCATAATC
ACTCTACCTAATCTCCTGAGGCTCAAGGCGGAACGACATTCGCGGAAGGGGCCCCTATCG
ATTACCGCACCATTACCGTACAGCCCTTGTTCTTTGAACTTGAGGACCAACGACGTAATC
CACTTCAAGAGACGTAAGGGGATTTTTGGAAAACTAATTTGTGCGAGCAGCAGGGGGCAT
CTAAATGGTCCATTTGAGTTAGTAGCCAACACACCGACCTAAATAGTAGATGTCCCCTGA
GGGGAGTTGCCATTTGTGAGAGGATCCGTGCCGAGGATGACAGGGTATCGCCAGAGGGCG
ACAGTTTCGATGCTTAACCCTTGACCAGCTCGACCGAAGAATGCGCCGTGATGGCCTCTC
TGTCAACGACCCACACTTAAAGGAATATTGGGCATACGCATATACGCTTGACGGCTCAGC
ATACGTGCTCGCCGAAGCCCCCACTGACGATTAAGCACTAGCATGAGCGCTTGCGATGTG
AGAGTTATAATCATGATCGAGCGTCTGCGTGGATACCACCGCTGCAGGTGATTCCACCCC
ATTCTGGCCCCAAATATTCTGTTACCCTCACCATACTCGCAGCCTTGTTGTCTATGTGAG
TAAACAGTTTATAGACCCTACGAAAATAACGGCGGGATCAAATTTATAAGTCGTCATCTC
TCTCGCGACCCACATAGATCCCGACTCAGTCAGATCGACGCGCTCAAGGACATTGCCACG
CAGCGCAGATGTCGGCATACCTCACTACATTAACAATTTCATTTGTTTTGTTCTATAAGT

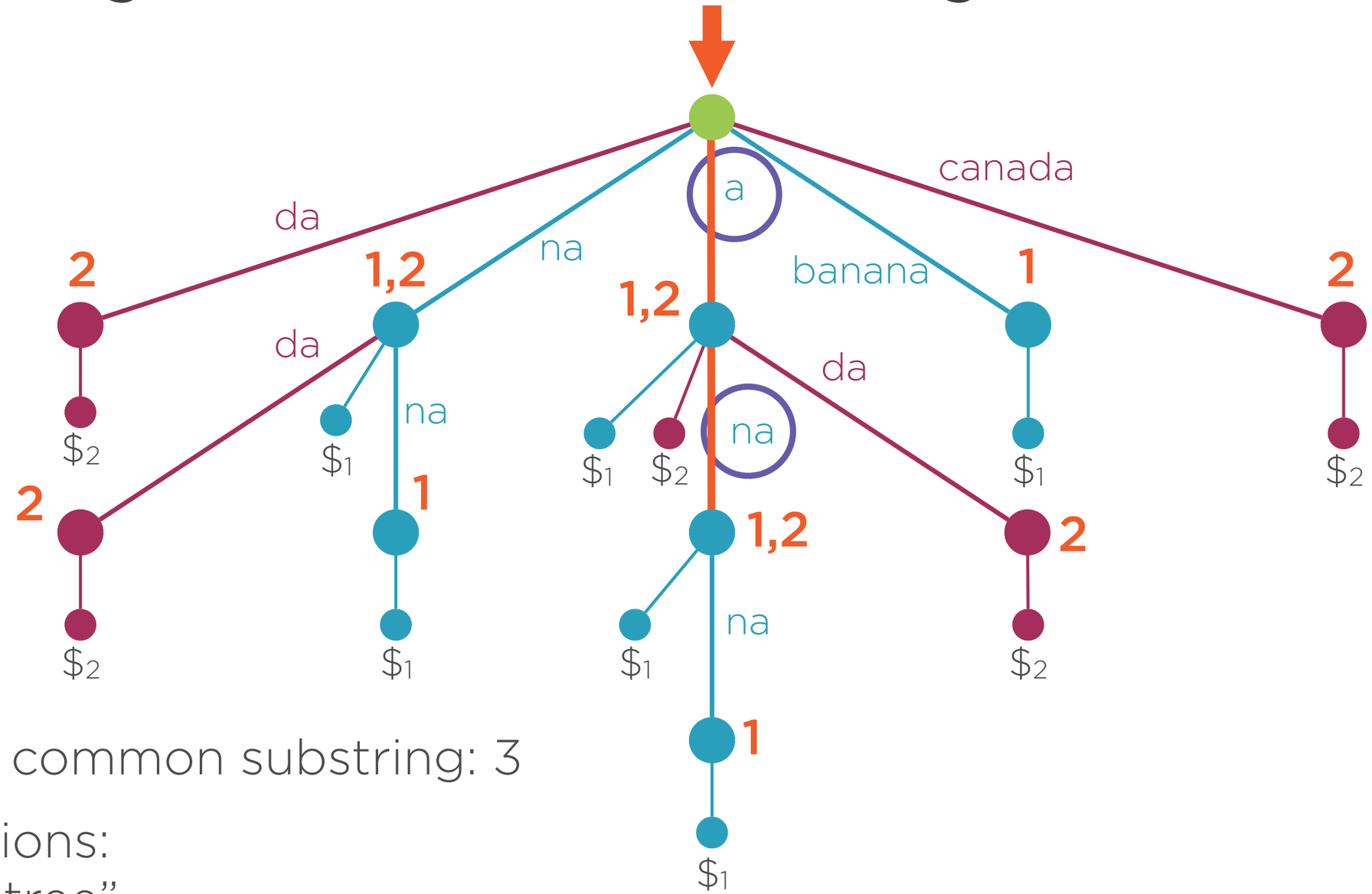
DNA from evidence

GGGGAGTTGCCATTTGTGAGAGGATCCGTGCCGAGGATGACAGGGTATCGCCAGAGGGCG
TAACGCGACCCACATAGATCCCGACTCAGTCAGATCGACGCGCTCAAGGACATTGCCACG
GCAGTCTCGATGCTTAACCCTTGACCAGCTCGACCGAAGAATGCGCCGTGATGGCCTCTC
TGTCAACGACCCACAATTAAAGGAATATTGGGCATACGCATATACGCTTTACGGCTCAGC
AGTCGTGCTCGCCGAAGCCCCCACTGACGATTAAGCACTAGCATGAGCGCTTGCGATGTG
CTAGTTATAATCATGATCGAGCGTCTGCGTGGATACCACCGCTGCAGGTGATTCCACGGA
ATTCTGGCCCCAAATATTCTGTTACCCTCACCATACTCGCAGCCTTGTTGTCTATGTGAG
TAAACAGTTTATAGACCCTACGAAAATAACGGCGGGATCAAATTTATAAGTCGTCATCTC
TCTCGCGACCCACATAGATCCCGACTCAGTCAGATCGACGCGCTCAAGGACATTGCCACG
CAGCGCAGATGTCGGCATACCTCACTACATTAACAATTTCATTTGTTTTGTTCTATAAGT
GACGCAGATCGTTTGCGTTCTTCCCTTGCTATTATTAGTTATTTTGGACGGTATAGCCGC
GGGGGAGGTTGAAACAAGGTGCGACGGATGGTCTGAGCATTTGTTTACTCATAACCTGTA
ACGAACCAGCCCAGTGGGAAAGTCTTTTTAGGCAGTGTAGGAGGTCAAATTATCATAATC
ACTCTACCTAATCTCCTGAGGCTCAAGGCGGAACGACATTCGCGGAAGGGGCCCCTATCG
ATTACCGCACCATTACCGTACAGCCCTTGTTCTTTGAACTTGAGGACCAACGACGTAATC
CACTTCAAGAGACGTAAGGGGATTTTTGGAAAACTAATTTGTGCGAGCAGCAGGGGGCAT
CTAAATGGTCCATTTGAGTTAGTAGCCAACACACCGACCTAAATAGTAGATGTCCCCTGA
GGGGAGTTGCCATTTGTGAGAGGATCCGTGCCGAGGATGACAGGGTATCGCCAGAGGGCG
GGGGGAGGTTGAAACAAGGTGCGACGGATGGTCTGAGCATTTGTTTACTCATAACCTGTA
ACAGTTTCGATGCTTAACCCTTGACCAGCTCGACCGAAGAATGCGCCGTGATGGCCTCTC
TGTCAACGACCCACACTTAAAGGAATATTGGGCATACGCATATACGCTTGACGGCTCAGC
ATACGTGCTCGCCGAAGCCCCCACTGACGATTAAGCACTAGCATGAGCGCTTGCGATGTG
AGAGTTATAATCATGATCGAGCGTCTGCGTGGATACCACCGCTGCAGGTGATTCCACCCC
ATTCTGGCCCCAAATATTCTGTTACCCTCACCATACTCGCAGCCTTGTTGTCTATGTGAG

Longest Common Substring

1: 012345
banana

2: 012345
canada



Length of longest common substring: 3

For other applications:
Wikipedia: “Suffix tree”

Lessons Learned

Useful for keyword suggestion

Supports infix matches

Compressed trie of all suffixes

Uses indexes
instead of substrings

Query time

- proportional with query string
- independent on total amount of text