String Matching Problem

(Approximate) String Matching

Input: Text **T** , Pattern **P**

Question(s):

Does **P** occur in **T**?

Find one occurrence of **P** in **T**.

Find all occurrences of **P** in **T**.

Count # of occurrences of **P** in **T**.

Find longest substring of **P** in **T**.

Find closest substring of **P** in **T**.

Locate direct repeats of **P** in **T**.

Many More variants

Applications:

Is **P** already in the database **T**?

Locate **P** in **T**.

Can **P** be used as a primer for **T**?

Is **P** homologous to anything in **T**?

Has **P** been contaminated by **T**?

Is *prefix*(**P**) = *suffix*(**T**)?

Locate tandem repeats of **P** in **T**.

Input:	Text **T**; Pattern **P**
Output:	All occurrences of **P** in **T**.

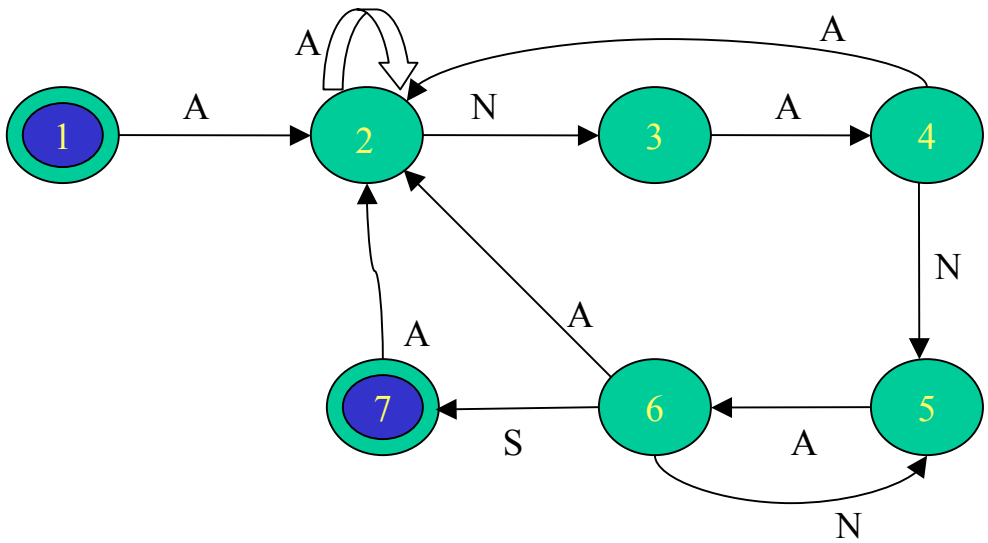
Methods:

- Naïve Method
- Rabin-Karp Method
- FSA-based method
- Knuth-Morris-Pratt algorithm
- Boyer-Moore
- Suffix Tree method
- Shift-And method

Naive Strategy

Finite State Automaton

ANANAS



Finite
State
Automaton

ATAQAANANASPVANAGVERANANESISITALVDANANANANAS

State Transition Diagram

		A	N	S	*
-	0	1	0	0	0
A	1	1	2	0	0
AN	2	3	0	0	0
ANA	3	1	4	0	0
ANAN	4	5	0	0	0
ANANA	5	1	4	6	0
ANANAS	6	1	0	0	0

Input:	Text **T**; Pattern **P**
Output:	All occurrences of **P** in **T**.

Sliding Window Strategy:

Initialize window on T;

While (window within T) do

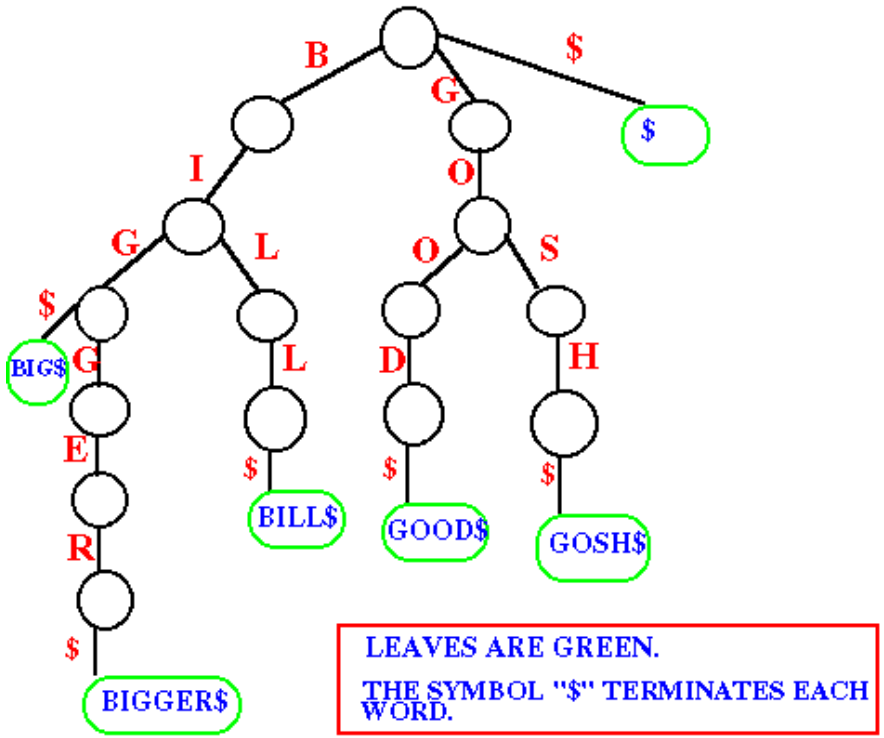
    Scan: if (window = P) then report it;

    Shift: shift window to right (by ?? positions)

endwhile;

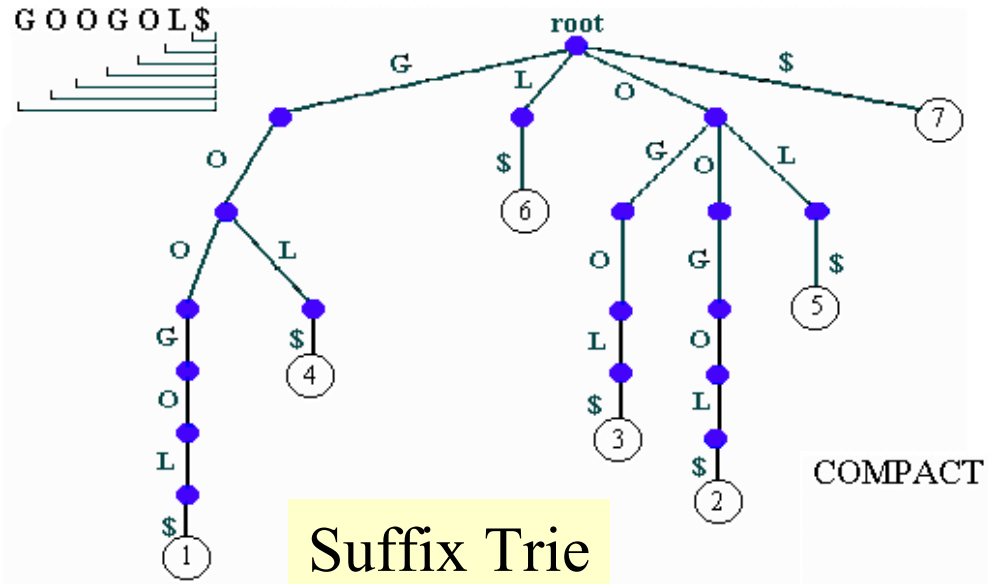
Tries

Storing:
BIG
BIGGER
BILL
GOOD
GOSH



In this figure, the strings either start with B or G. Therefore, the root of the trie is connected to 3 edges called B, G and $.

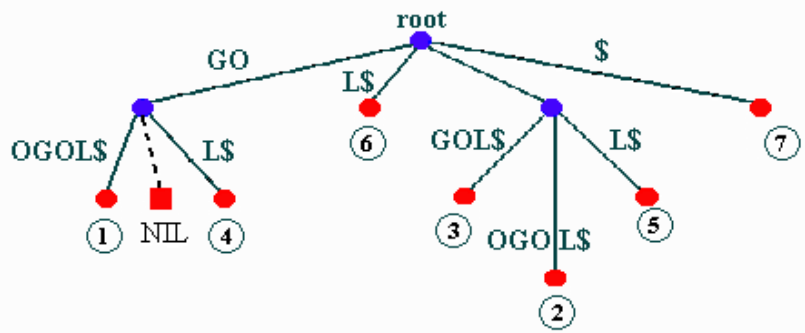
Suffix Tries & Compact Suffix Tries



Suffix Trie

Store all suffixes of GOOGOL$

COMPACT TRIE OF SUFFIXES OF THE TEXT: *GOOGOL$*



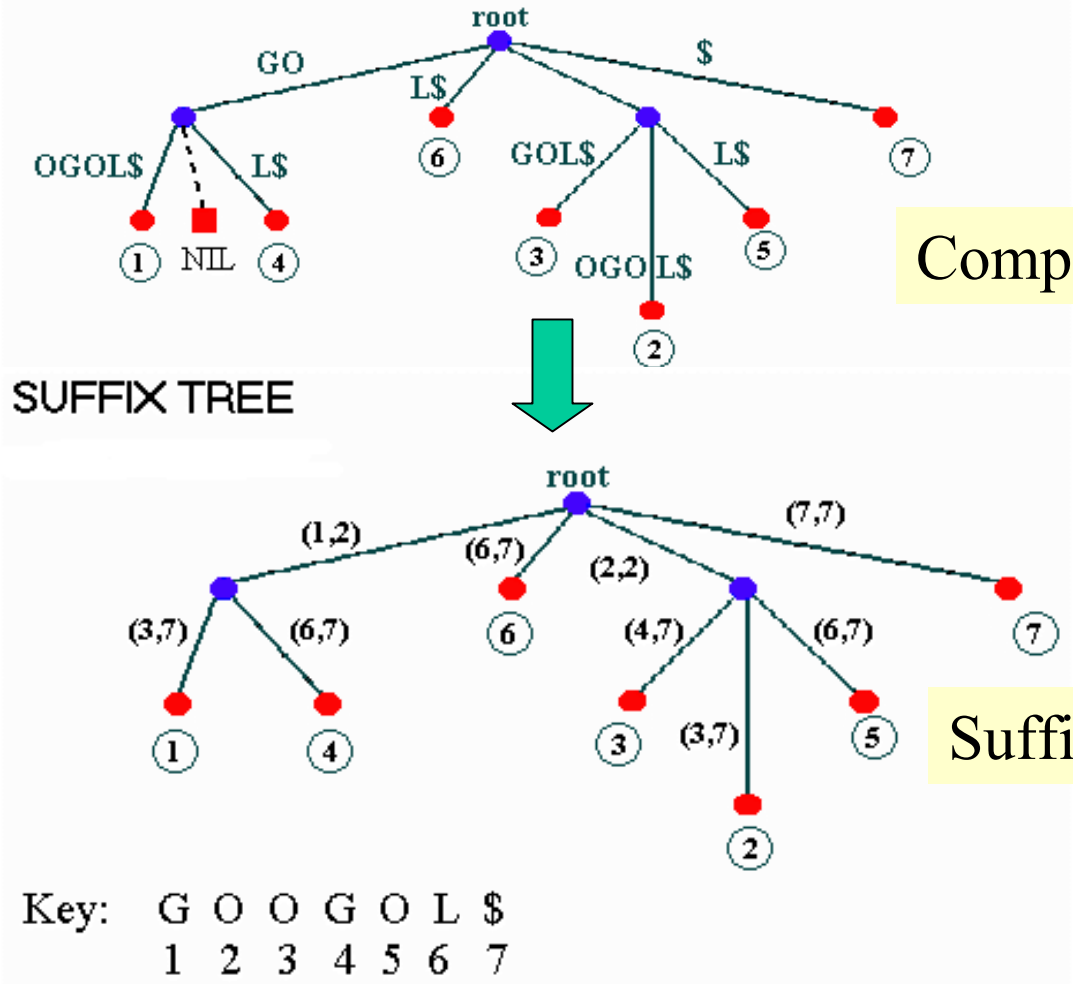
Active node, correspond to a suffix of the text

Inactive node, one for each symbol of the alphabet not associated with any string

Internal node, each have at least two children in a compact trie

Compact Suffix Trie

Suffix Tries to Suffix Trees



Compact Suffix Trie

Suffix Tree

Suffix Trees

- Linear-time construction!
- String Matching, Substring matching, substring common to k of n strings
- All-pairs prefix-suffix problem
- Repeats & Tandem repeats
- Approximate string matching

Shift-And Method (Baeza-Yates & Gonnet)

Idea: *Build a bit-matrix M* such that

M[I,J] = 1 ⇔ P[1..I] = T[J-I+1 .. J]

Thus, M[I,J] = 1 ⇔ (M[I-1, J-1] = 1) & (P[I] = T[J])

M	T	A	G	T	A	G	A	A	G	A	A	C
A	0	1	0	0	1	0	1	1	0	1	1	0
G		0	1	0	0	1	0	0	1	0	0	0
A			0	0	0	0	1	0	0	1	0	0
A				0	0	0	0	1	0	0	1	0
C					0	0	0	0	0	0	0	1

Shift-And (Cont'd)

Idea: *Operate on column bit-vectors*

Step 1: Build a bit-matrix U such that for each e $\in \Sigma$

U[I,e] = 1 $\Leftrightarrow$ P[I] = e

U	A	C	G	T
A	1	0	0	0
G	0	0	1	0
A	1	0	0	0
A	1	0	0	0
C	0	1	0	0

Step 2: M[J] = RightShift(M[J-1]) && U[T[J]]

Shift-And (Cont'd)

Step 2: M[J] = RightShift(M[J-1]) && U[T[J]]

M	T	A	G	T	A	G	A	A	G	A	A	C
A	0	1	0	0	1	0	1	1	0	1	1	0
G		0	1	0	0	1	0	0	1	0	0	0
A			0	0	0	0	1	0	0	1	0	0
A				0	0	0	0	1	0	0	1	0
C					0	0	0	0	0	0	0	1

U	A	C	G	T
A	1	0	0	0
G	0	0	1	0
A	1	0	0	0
A	1	0	0	0
C	0	1	0	0

Shift-And (Generalizations)

Generalization 1: *Wild Cards*: match all characters.

U	A	C	G	T
A	1	0	0	0
G	0	0	1	0
A	1	0	0	0
*	1	1	1	1
C	0	1	0	0

Generalization 2: *k Mismatches*: Compute $M_0, M_1, \ldots, M_k$

$M_s[J] = \text{RightShift}(M_{s-1}[J-1]$ **AND** $U[T[J]])$
OR $M_{s-1}[J]$
OR $M_{s-1}[J-1]$

String Matching Methods: Overview

Methods:

- Naïve Method **O(mn)** *time*
- Rabin Karp Method **O(mn)** *time;* Fast on average.
- FSA-based method **O(n+mA)** *time*
- Knuth-Morris-Pratt algorithm **O(n+m)** *time*
- Boyer-Moore **O(mn)** time; Very fast on average.
- Suffix Tree method; **O(m+n)** *time*
- Shift-And method; Fast on average; Bit operations.

Why Sequence Analysis?

- Mutation in DNA is a natural evolutionary process. Thus sequence similarity may indicate common ancestry.
- In biomolecular sequences (DNA, RNA, protein), high sequence similarity implies significant structural and/or functional similarity.
- Errors possible in database.

V-sis Oncogene - Homologies

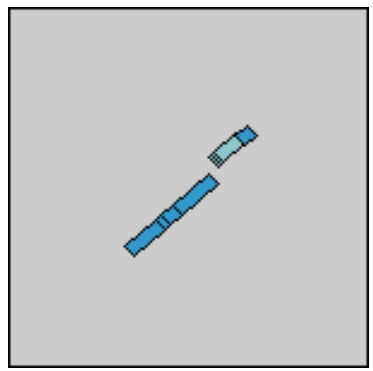
Sequences producing significant alignments:	Score (bits)	E Value
gi|332623|gb|J02396.1|SEG_SSVPCS2 Simian sarcoma virus v-si...	4591	0.0
gi|61774|emb|V01201.1|RESSV1 Simian sarcoma virus proviral ...	4504	0.0
gi|332622|gb|J02395.1|SEG_SSVPCS1 Simian sarcoma virus LTR ...	1283	0.0
gi|885929|gb|U20589.1|GLU20589 Gibbon leukemia virus envelo...	1140	0.0
gi|4505680|ref|NM_002608.1| Homo sapiens platelet-derived g...	954	0.0
gi|20987438|gb|BC029822.1| Homo sapiens, platelet-derived g...	954	0.0
gi|338210|gb|M12783.1|HUMSISPDG Human c-sis/platelet-derive...	954	0.0

Sequence Alignment

```
>gi|4505680|ref|NM_002608.1|  __ Homo sapiens platelet-derived growth
   factor beta polypeptide (simian sarcoma viral (v-sis) oncogene
   homolog) (PDGFB), transcript variant 1, mRNA Length = 3373 Score =
   954 bits (481), Expect = 0.0 Identities = 634/681 (93%), Gaps =
   3/681 (0%) Strand = Plus / Plus
Query: 1015 agggggaccccattcctgaggagctctataagatgctgagtggccactcgattcgctcct 1074
           |||||||||||||||| |||||||| ||| |||||||||||| |||||||| |||||||
Sbjct: 1084 agggggaccccattcccgaggagctttatgagatgctgagtgaccactcgatccgctcct 1143
       > 21 E  G  D  P  I  P  E  E  L  Y  E  M  L  S  D  H  S  I  R  S
Query: 1075 tcgatgacctccagcgcctgctgcagggagactccggaaaagaagatggggctgagctgg 1134
           | ||||| ||||| |||||||||| |||||| ||||| | |||||||||| ||| |||
Sbjct: 1144 ttgatgatctccaacgcctgctgcacggagaccccggagaggaagatggggccgagttgg 1203
       > 61 D  L  N  M  T  R  S  H  S  G  G  E  L  E  S  L  A  R  G  R
```

Sequence Alignment

Sequence 1 gi 332624 Simian sarcoma virus v-sis transforming protein p28 gene, complete cds; and 3' LTR long terminal repeat, complete sequence. **Length** 2984 (1 .. 2984)
Sequence 2 gi 4505680 Homo sapiens platelet-derived growth factor beta polypeptide (simian sarcoma viral (v-sis) oncogene homolog) (PDGFB), transcript variant 1, mRNA **Length** 3373 (1 .. 3373)



Similarity vs. Homology

- Homologous sequences share common ancestry.
- Similar sequences are “near” to each other by some criteria. Similarity can be measured using appropriate criteria.

Types of Sequence Alignments

- Global Alignment: similarity over entire length
- Local Alignment: no overall similarity, but some segment(s) is/are similar
- Semi-global Alignment: end segments may not be similar
- Multiple Alignment: similarity between sets of sequences

Global Sequence Alignment

- Needleman-Wunsch-Sellers algorithm.
- Dynamic Programming (DP) based.
  - Overlapping Subproblems
  - Recurrence Relation
  - Table to store solutions to subproblems
  - Ordering of subproblems to fill table
  - Traceback to find solution

Global Alignment: An example

```
V: G A A T T C A G T T A
W: G G A T C G A
```