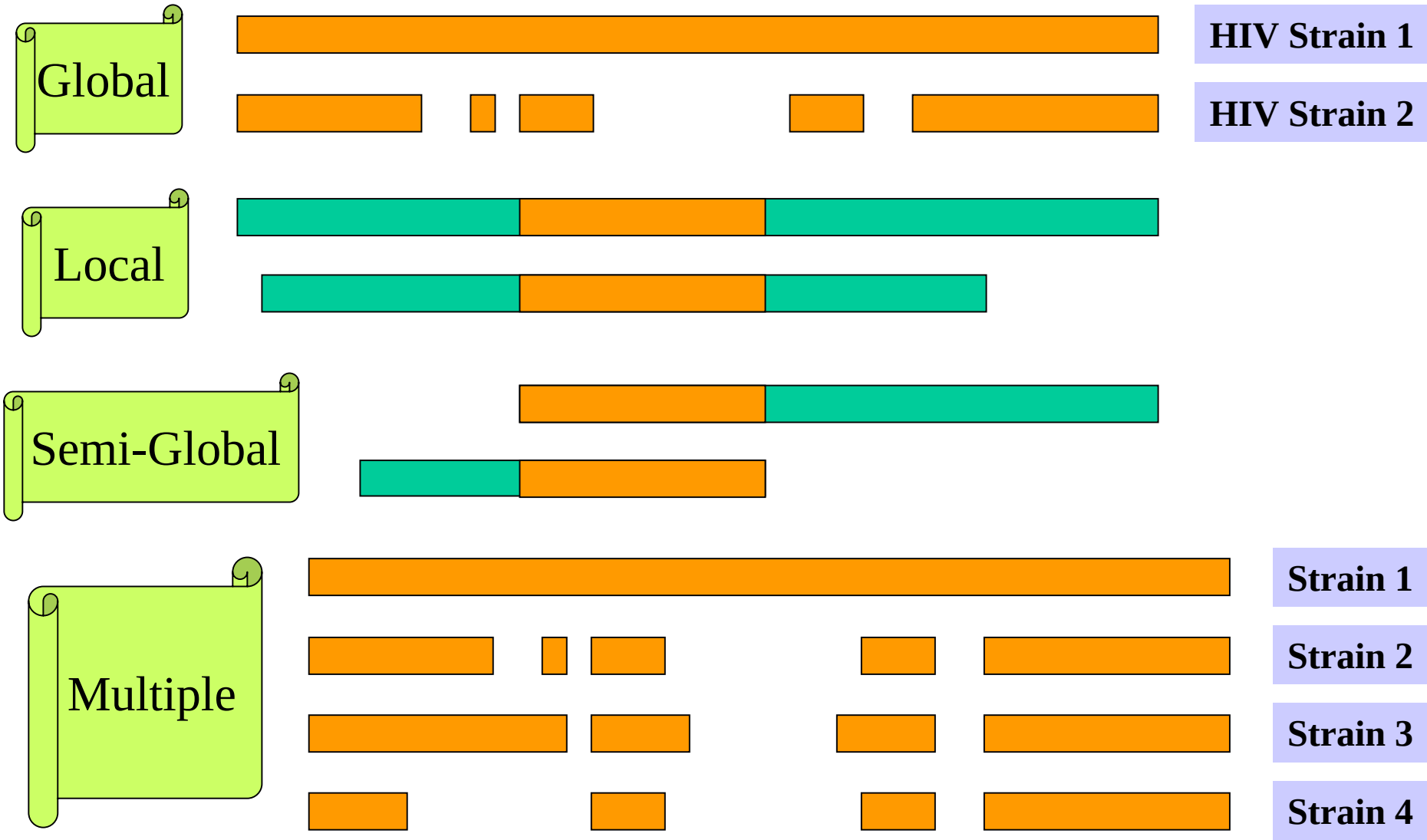
Types of Sequence Alignments



Global Alignment: An example

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

		G	A	A	T	T	C	A	G	T	T	A
	0	0	0	0	0	0	0	0	0	0	0	0
G	0											
G	0											
A	0											
T	0											
C	0											
G	0											
A	0											

Given

$\delta[I, J]$ = Score of Matching
the I^{th} character of sequence V &
the J^{th} character of sequence W

Compute

$S[I, J]$ = Score of Matching
First I characters of sequence V &
First J characters of sequence W

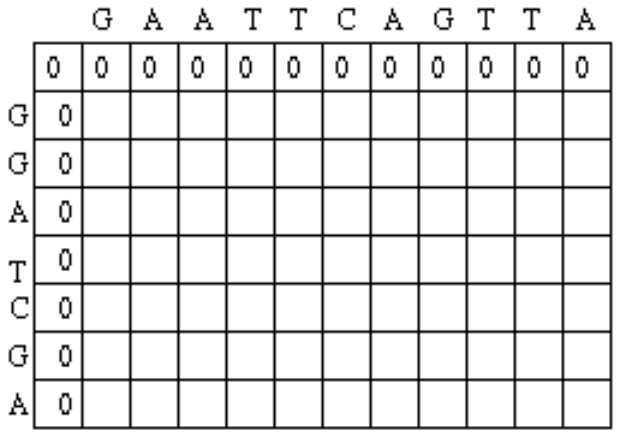
Recurrence Relation

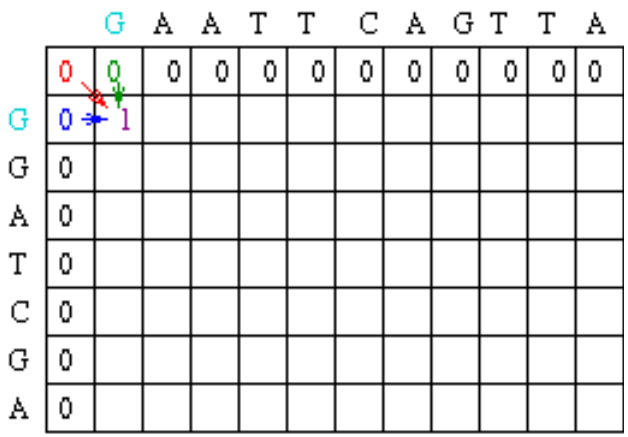
$S[I, J]$ = MAXIMUM {
$S[I-1, J-1] + \delta(V[I], W[J])$,
$S[I-1, J] + \delta(V[I], —)$,
$S[I, J-1] + \delta(— , W[J])$ }

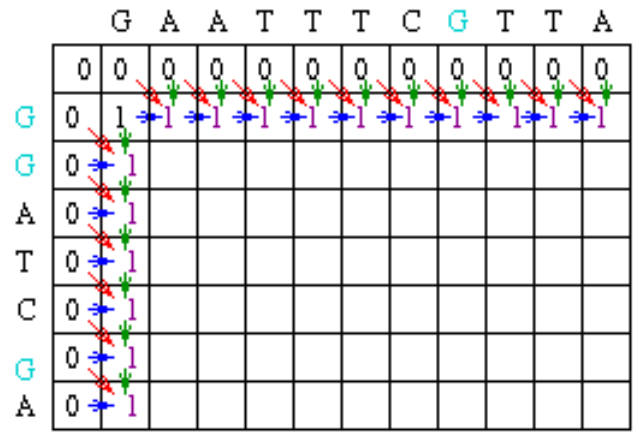
Global Alignment: An example

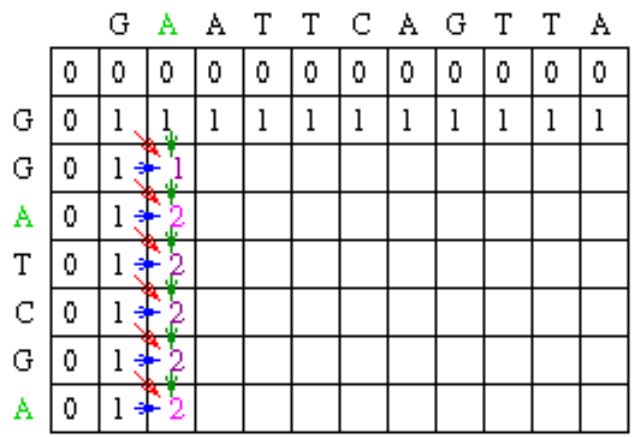
$S[I, J] = MAXIMUM \{$
$S[I-1, J-1] + \delta(V[I], W[J]),$
$S[I-1, J] + \delta(V[I], —),$
$S[I, J-1] + \delta(— , W[J]) \}$

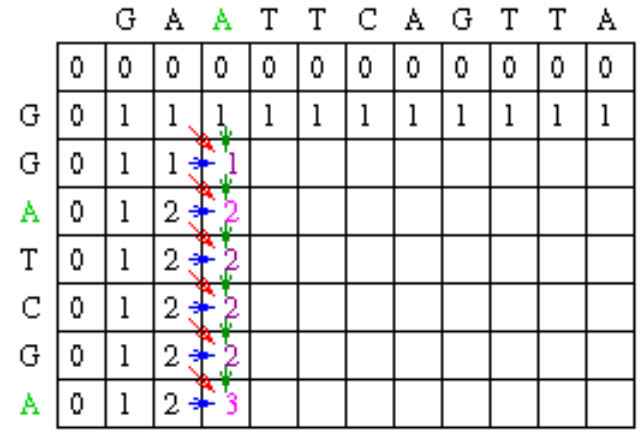
```
V: G A A T T C A G T T A
W: G G A T C G A
```

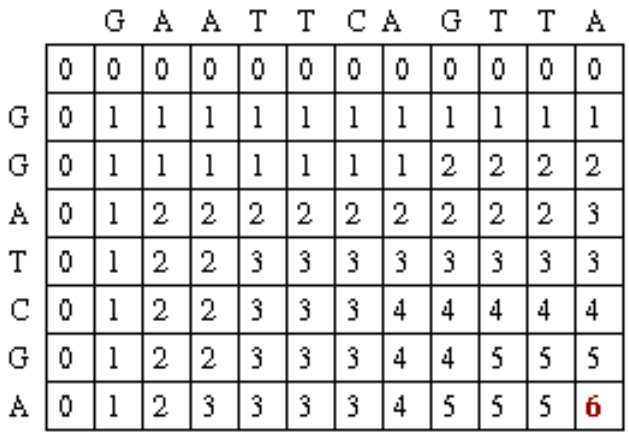




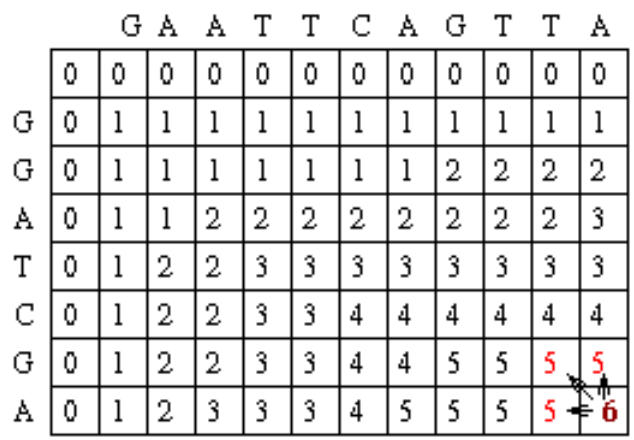


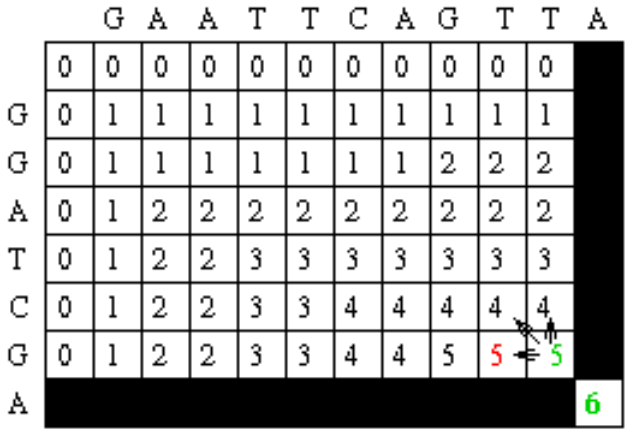


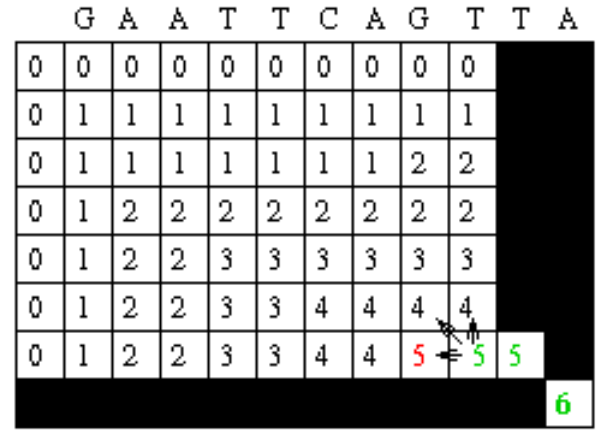


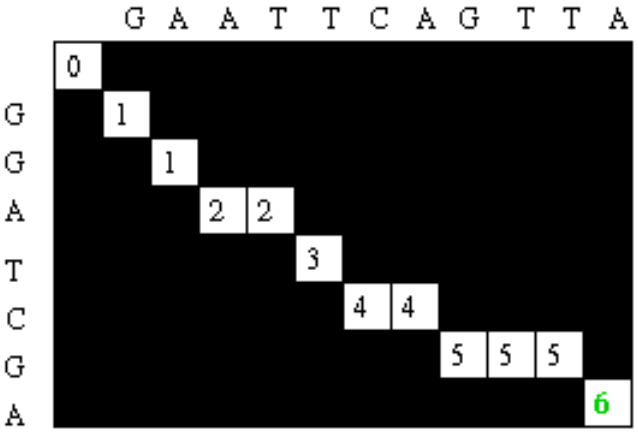


Traceback



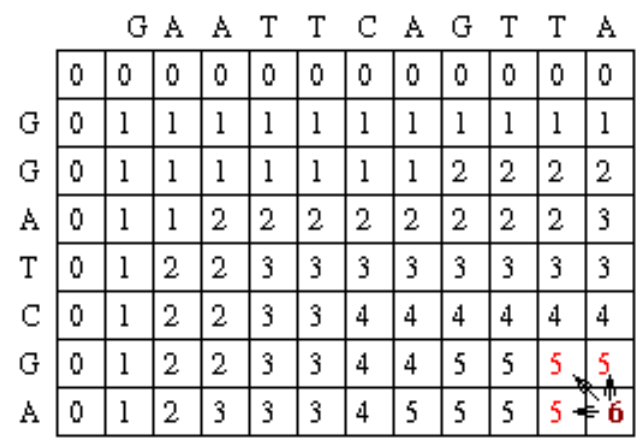


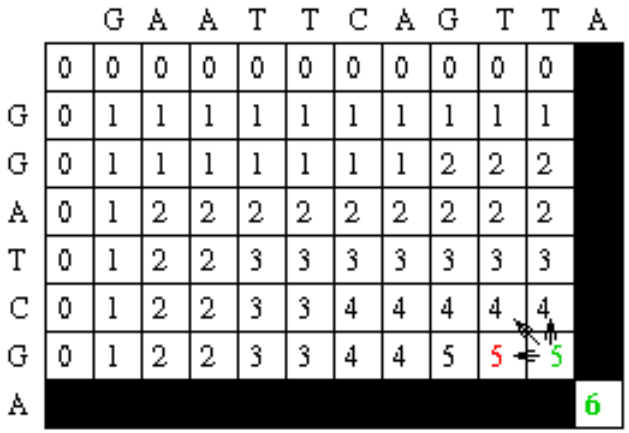


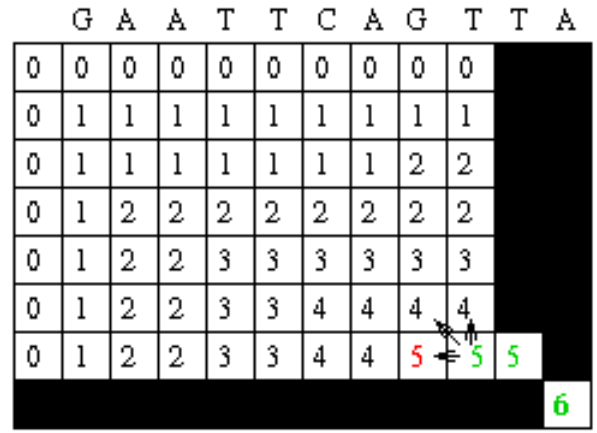


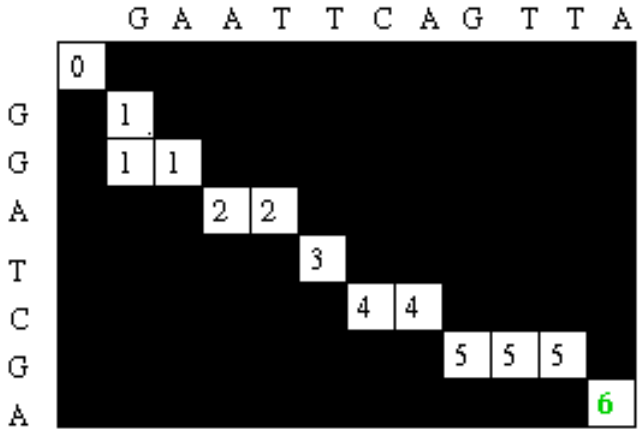

```
V: G A A T T C A G T T A
   |   |   | |   |     |
W: G G A - T C - G - - A
```

Alternative Traceback










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

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

Previous

Improved Traceback

		G	A	A	T	T	C	A	G	T	T	A
	0	0	0	0	0	0	0	0	0	0	0	0
G	0	×1	←1	← 1	← 1	← 1	← 1	← 1	×1	← 1	← 1	← 1
G	0	×1	↑1	↑1	↑1	↑1	↑1	↑1	×2	← 2	← 2	← 2
A	0	↑1	↑1	×2	← 2	← 2	← 2	×2	↑2	↑2	↑2	×3
T	0	↑1	← 2	↑2	×3	×3	← 3	← 3	← 3	×3	×3	↑3
C	0	↑1	↑2	↑2	↑3	↑3	×4	← 4	← 4	← 4	← 4	← 4
G	0	↑1	↑2	↑2	↑3	↑3	↑4	↑4	×5	← 5	← 5	← 5
A	0	↑1	↑2	×3	↑3	↑3	↑4	×5	↑5	↑5	↑5	×6

Improved Traceback

		G	A	A	T	T	C	A	G	T	T	A
	0	0	0	0	0	0	0	0	0	0	0	0
G	0	×1	←1	← 1	← 1	← 1	← 1	← 1	×1	← 1	← 1	← 1
G	0	×1	↑1	↑1	↑1	↑1	↑1	↑1	×2	← 2	← 2	← 2
A	0	↑1	↑1	×2	← 2	← 2	← 2	×2	↑2	↑2	↑2	×3
T	0	↑1	← 2	↑2	×3	×3	← 3	← 3	← 3	×3	×3	↑3
C	0	↑1	↑2	↑2	↑3	↑3	×4	← 4	← 4	← 4	← 4	← 4
G	0	↑1	↑2	↑2	↑3	↑3	↑4	↑4	×5	← 5	← 5	← 5
A	0	↑1	↑2	×3	↑3	↑3	↑4	×5	↑5	↑5	↑5	×6

Improved Traceback

		G	A	A	T	T	C	A	G	T	T	A
	0	0	0	0	0	0	0	0	0	0	0	0
G	0	×1	←1	← 1	← 1	← 1	← 1	← 1	×1	← 1	← 1	← 1
G	0	×1	↑1	↑1	↑1	↑1	↑1	↑1	×2	← 2	← 2	← 2
A	0	↑1	↑1	×2	← 2	← 2	← 2	×2	↑2	↑2	↑2	×3
T	0	↑1	← 2	↑2	×3	×3	← 3	← 3	← 3	×3	×3	↑3
C	0	↑1	↑2	↑2	↑3	↑3	×4	← 4	← 4	← 4	← 4	← 4
G	0	↑1	↑2	↑2	↑3	↑3	↑4	↑4	×5	← 5	← 5	← 5
A	0	↑1	↑2	×3	↑3	↑3	↑4	×5	↑5	↑5	↑5	×6

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

Subproblems

- Optimally align V[1..I] and W[1..J] for every possible values of I and J.
- Having optimally aligned
  - V[1..I-1] and W[1..J-1]
  - V[1..I] and W[1..J-1]
  - V[1..I-1] and W[1, J]

  it is possible to optimally align V[1..I] and W[1..J]

- $O(mn)$,
  where m = length of V,
  and n = length of W.

Generalizations of Similarity Function

- Mismatch Penalty = α
- Spaces (Insertions/Deletions, InDels) = β
- Affine Gap Penalties:
  (Gap open, Gap extension) = (γ,δ)
- Weighted Mismatch = $\Phi(a,b)$
- Weighted Matches = $\Omega(a)$

Alternative Scoring Schemes

		G	A	A	T	T	C	A	G	T	T	A
	0	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11	-12
G	-2	× 1	← -1	← -2	← -3	← -4	← -5	← -6	← -7	← -8	← -9	← -10
G	-3	↑-1	× -1	← -3	← -4	← -5	← -6	← -7	× -5	← -7	← -8	← -9
A	-4	↑-2	× 0	× 0	← -2	← -3	← -4	← -5	← -6	← -7	← -8	× -7
T	-5	↑-3	↑ -2	↑-2	× 1	← -1	← -2	← -3	← -4	← -5	← -6	← -7
C	-6	↑-4	↑ -3	↑-3	↑-1	× -1	× 0	← -2	← -3	← -4	← -5	← -6
G	-7	↑-5	↑-4	↑-4	↑-2	↑-3	↑-2	× -2	× -1	← -3	← -4	← -5
A	-8	↑-6	↑-5	↑-5	↑-3	↑-4	↑-3	× -1	↑-3	× -3	× -5	× -3

Match +1
Mismatch –2
Gap (-2, -1)

```
V: G A A T T C A G T T A
   |   | |   |   |     |
W: G G A T - C – G - - A
```

Local Sequence Alignment

- Example: comparing long stretches of anonymous DNA; aligning proteins that share only some motifs or domains.
- Smith-Waterman Algorithm

Recurrence Relations (Global vs Local Alignments)

- S[I, J] = MAXIMUM {
  S[I-1, J-1] + δ(V[I], W[J]),
  S[I-1, J] + δ(V[I], —),
  S[I, J-1] + δ(— , W[J]) }

Global Alignment

- S[I, J] = MAXIMUM { 0,
  S[I-1, J-1] + δ(V[I], W[J]),
  S[I-1, J] + δ(V[I], —),
  S[I, J-1] + δ(— , W[J]) }

Local Alignment

Local Alignment: Example

		G	A	A	T	T	C	A	G	T	T	A
	0	0	0	0	0	0	0	0	0	0	0	0
G	0	× 1	0	0	0	0	0	0	0	0	0	0
G	0	× 1	← 0	0	0	0	0	0	× 1	0	0	0
A	0	0	× 2	× 1	0	0	0	× 1	0	0	0	× 1
T	0	0	↑ 0	× 1	× 2	← 1	0	0	0	× 1	× 1	0
C	0	0	0	0	↑ 0	× 0	× 2	0	0	0	0	0
G	0	0	0	0	0	0	0	0	× 1	0	0	0
A	0	0	× 1	× 1	0	0	0	× 1	0	0	0	× 1

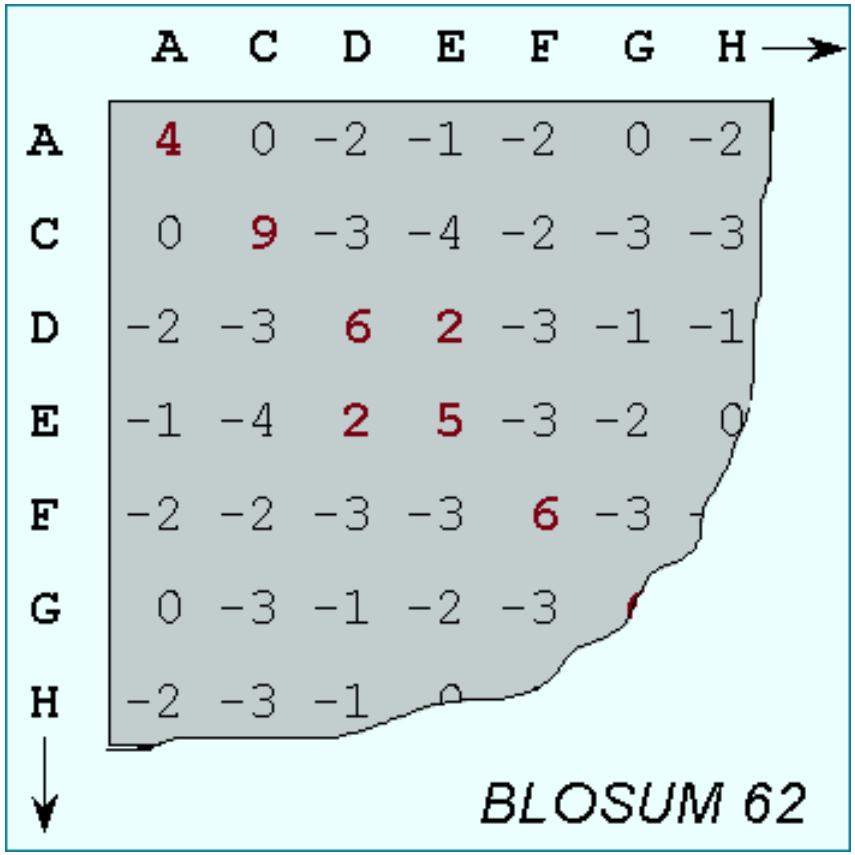
Match +1
Mismatch –1
Gap (-1, -1)

```
V: - G A A T T C A G T T A
     |   | |   |
W: G G - A T - C – G - - A
```

Properties of Smith-Waterman Algorithm

- How to find all regions of "high similarity"?
  - Find all entries above a threshold score and traceback.
- What if: Matches = 1 & Mismatches/spaces = 0?
  - Longest Common Subsequence Problem
- What if: Matches = 1 & Mismatches/spaces = $-\infty$?
  - Longest Common Substring Problem
- What if the average entry is positive?
  - Global Alignment

How to score mismatches?



BLOSUM n Substitution Matrices

- For each amino acid pair a, b
  - For each BLOCK
    - Align all proteins in the BLOCK
    - Eliminate proteins that are more than n% identical
    - Count F(a), F(b), F(a,b)
    - Compute Log-odds Ratio

$$\log\left(\frac{F(a,b)}{F(a)F(b)}\right)$$

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**.

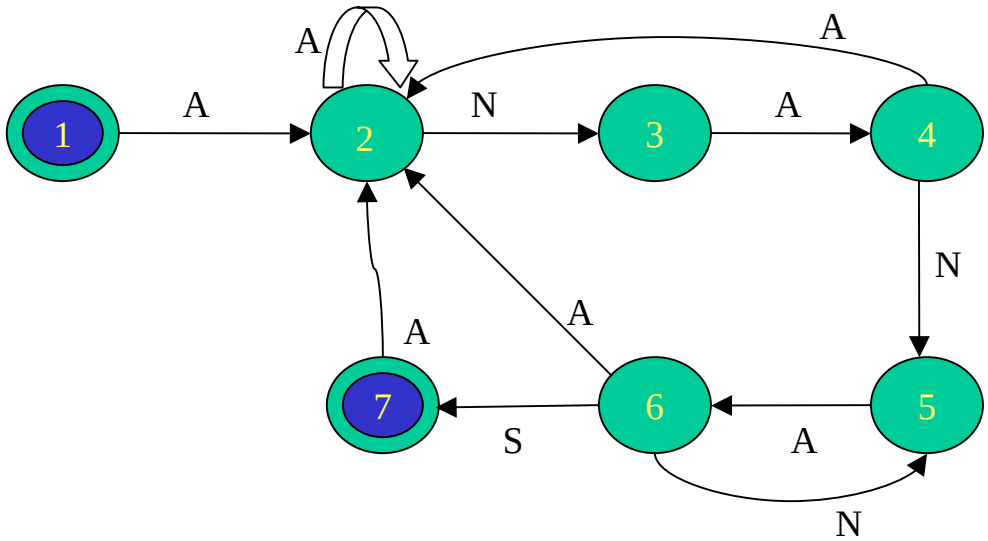
Methods:

- 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

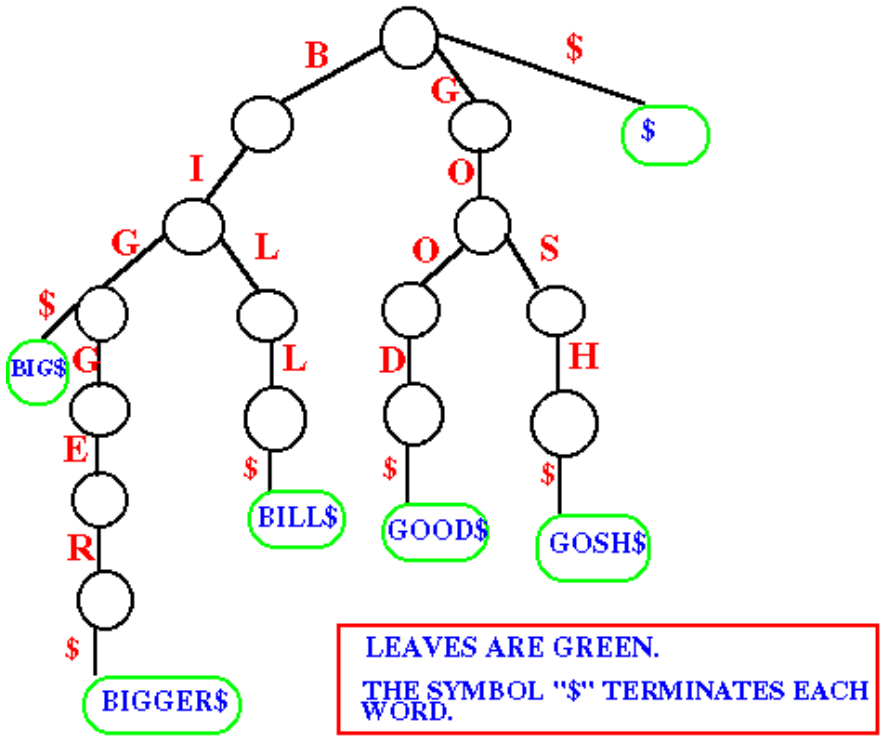
Scan: if (window = P) then report it;

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

endwhile;

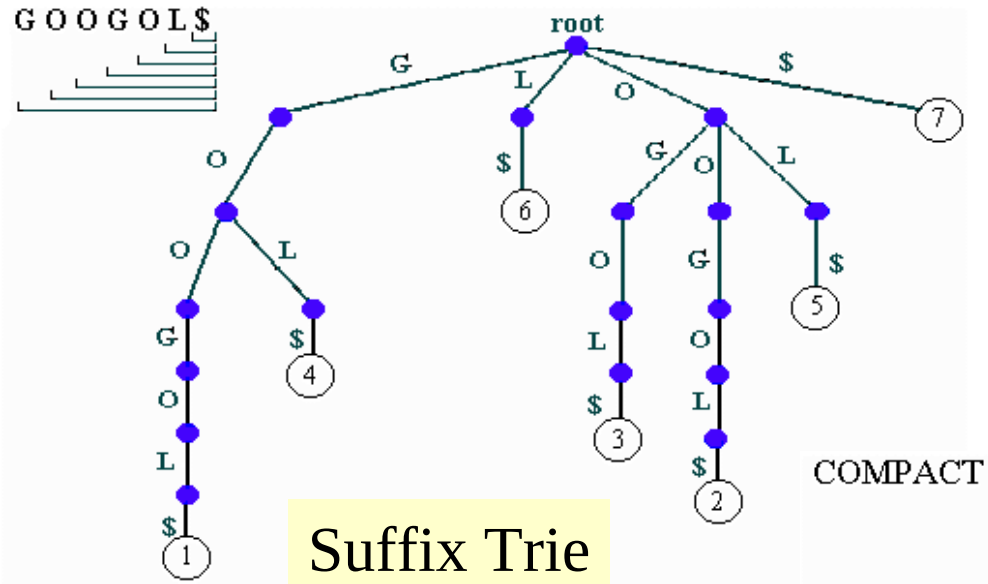
Tries

Storing:
BIG
BIGGER
BILL
GOOD
GOSH



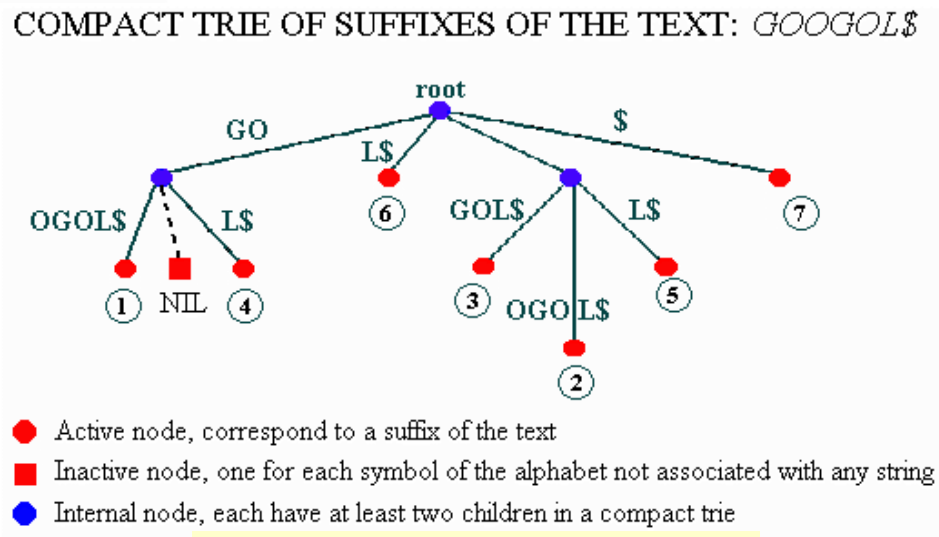
In this figure, the strings either start with B or G. Therefore, the root of the trie is connected to 3 edges called B, G and $.

Suffix Tries & Compact Suffix Tries



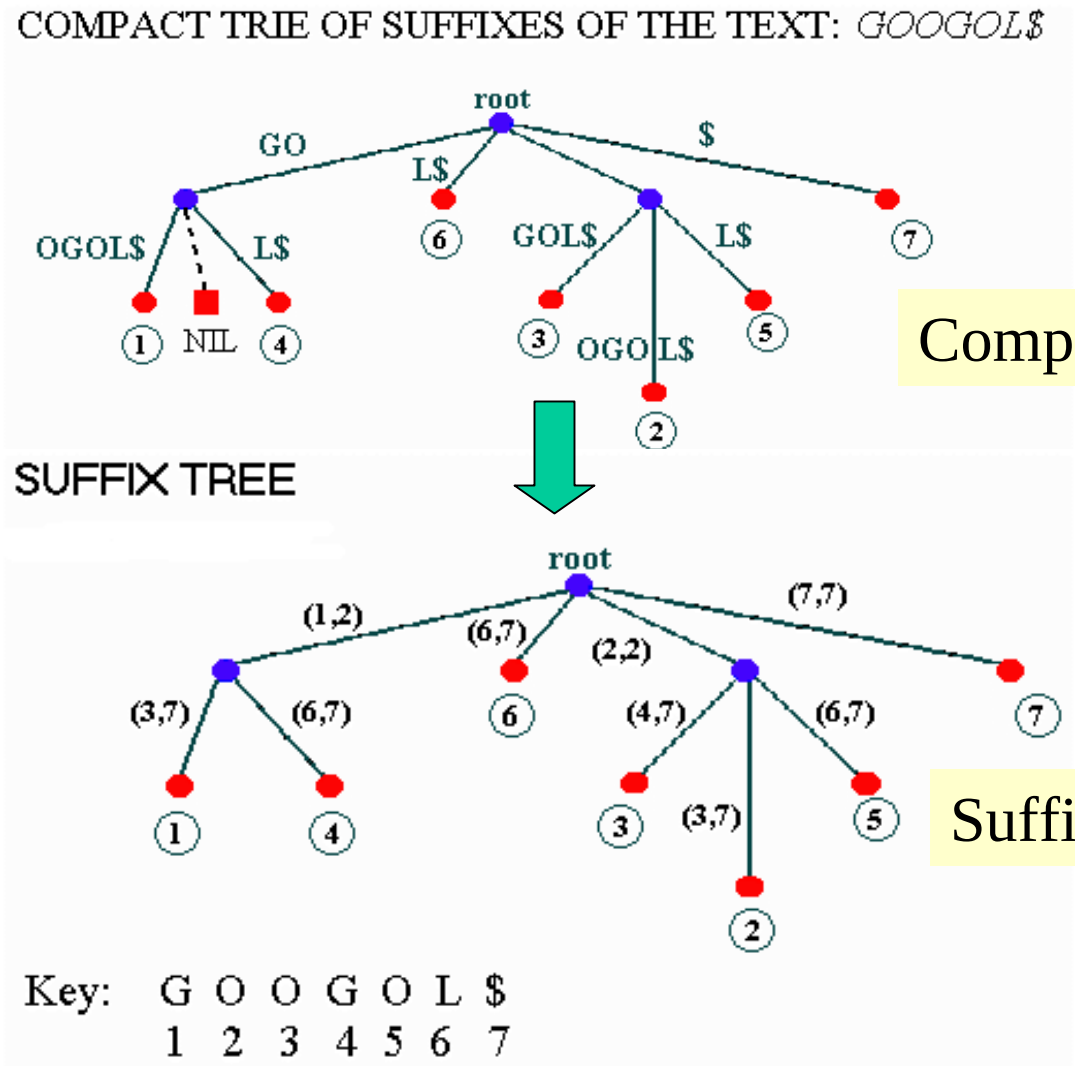
Suffix Trie

Store all suffixes of GOOGOL$



Compact 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