String Matching Problem

Computer Science Fundamentals

- Specify an input-output description of the problem.
- Design a conceptual algorithm and analyze it.
- Design data structures to refine the algorithm.
- Write the program in parts and test the parts separately.

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

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.

Evolution of Data Structures

- Complex problems often required complex data structures.
- Simple data types: Lists. Applications of lists include: students roster, voters list, grocery list, list of transactions,
- Array implementation of a list. Advantage – random access.
- Need for list “operations” arose – “Static” vs. “dynamic” lists. “Storing” items in a list vs. “Maintaining” items in a list.
- Lot of research on “Sorting” and “Searching” resulted.
- “Inserting” in a specified location in a list caused the following evolution: Array implementation → Linked list implementation.
- Other linear structures e.g., stacks, queues, etc.

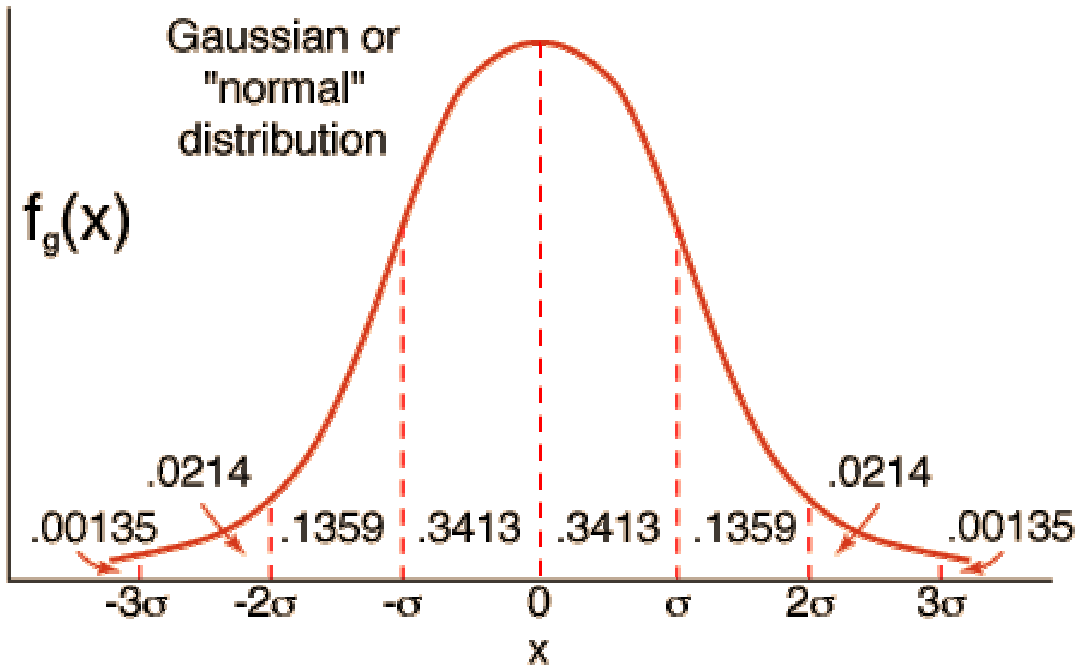
Evolution of Data Structures (Cont'd)

- Trees made hierarchical organization of data easy to handle. Applications of trees: administrative hierarchy in a business set up, storing an arithmetic expression, organization of the functions calls of a recursive program, etc.
- Search trees (e.g., BST) were designed to make search and retrieval efficient in trees. A BST may not allow fast search/retrieval, if it is very unbalanced, since the time complexities of the operations depended on the height of the tree. Hence the study of "balanced" trees and "nearly balanced" trees. Examples: AVL trees, 2-3 trees, 2-3-4 trees, RB trees, Skip lists, etc.
- Graphs generalize trees; model more general networks.
- Abstract data types. Advantages include: Encapsulation of data and operations, hiding of unnecessary details, localization and debugging of errors, ease of use since interface is clearly specified, ease of program development, etc.

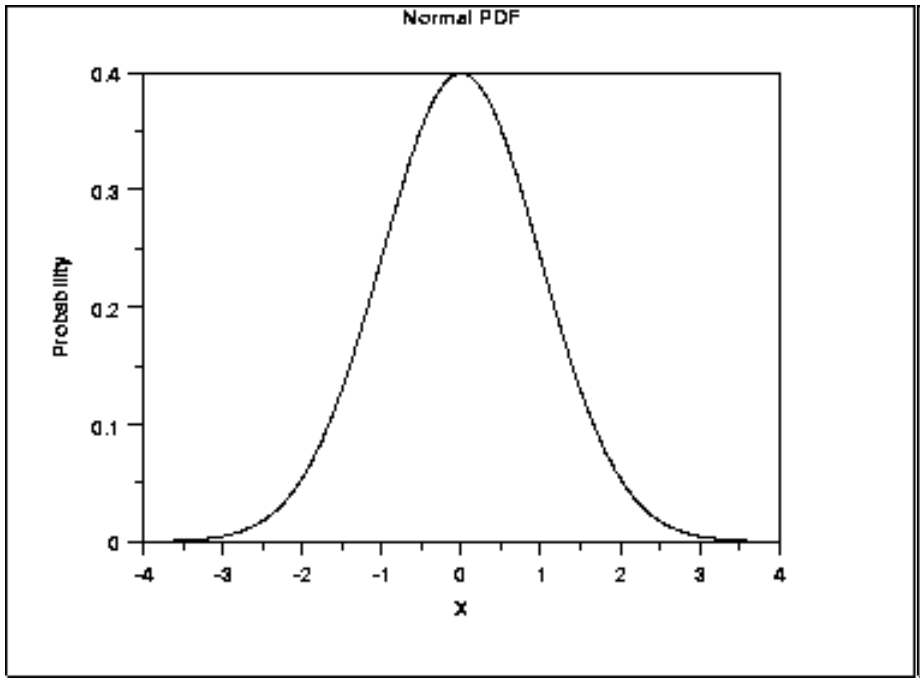
Why is Statistics important in Bioinformatics?

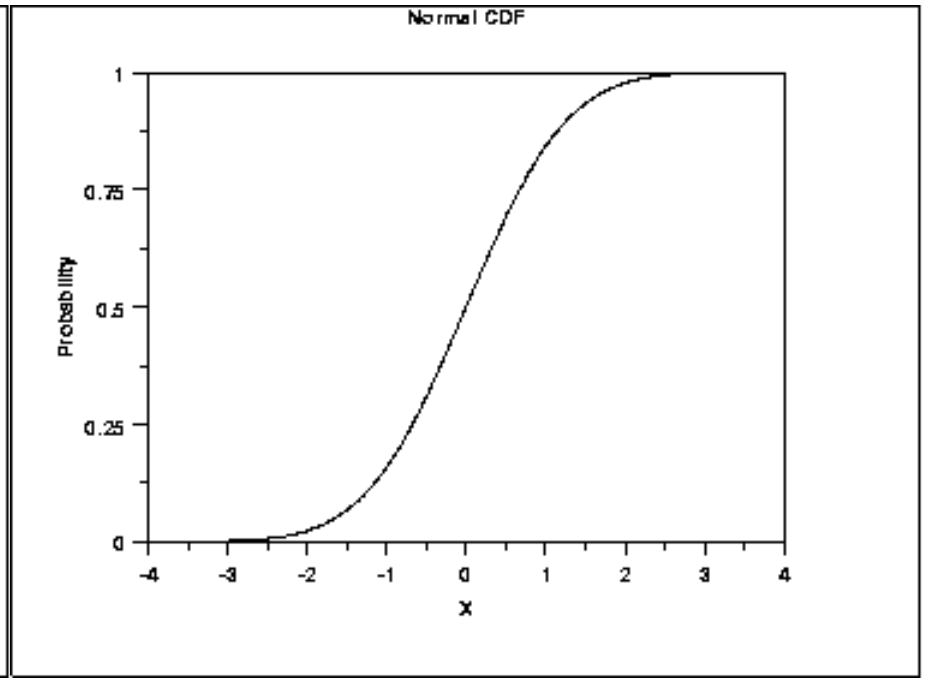
- Random processes are inherent in biological processes (e.g., evolution) & in sampling (data collection).
- Errors are often unavoidable in the data collection process.
- Statistics helps in studying *trends, interpolations, extrapolations, categorizations, classifications, inferences, models, predictions, assigning confidence to predictions, ...*

Normal/Gaussian Distribution

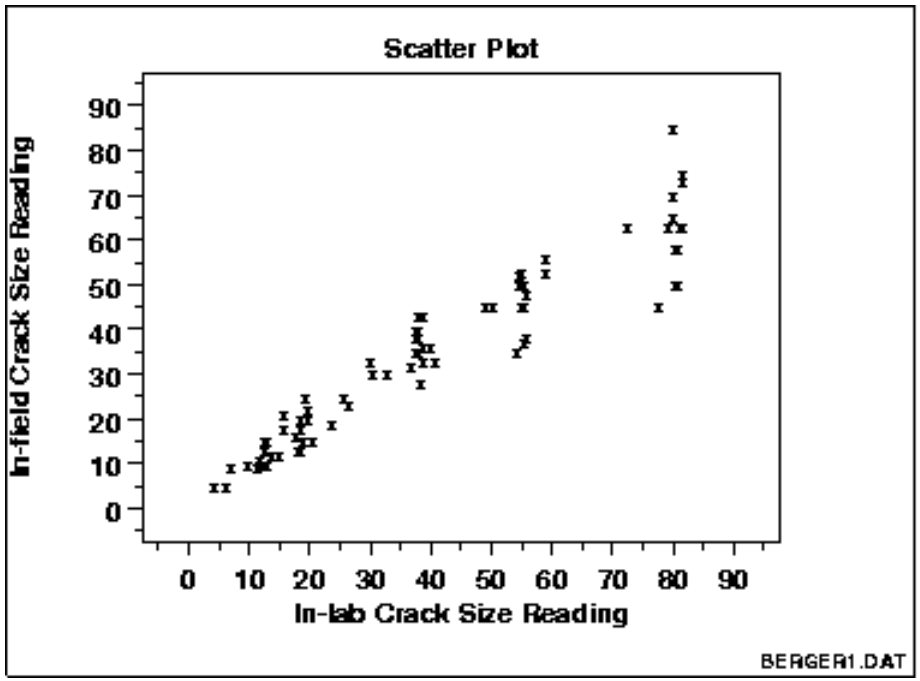


Density & Cumulative Distribution Functions

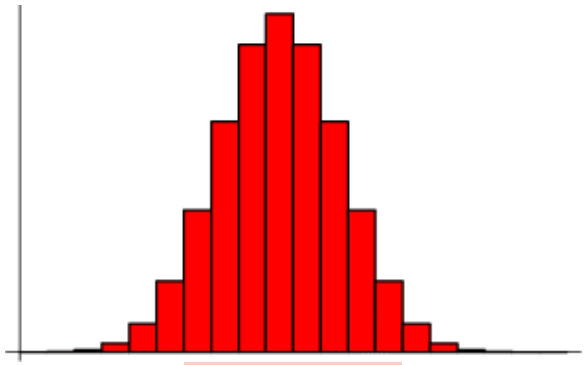




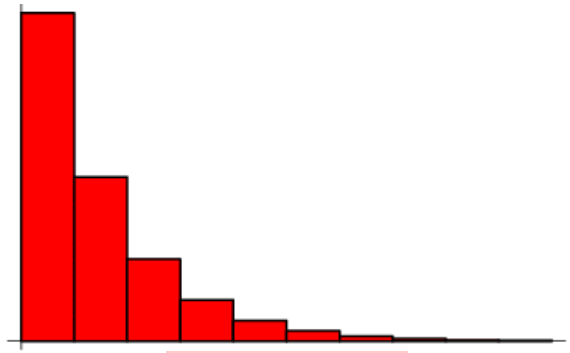
Graphical Techniques: Scatter Plot



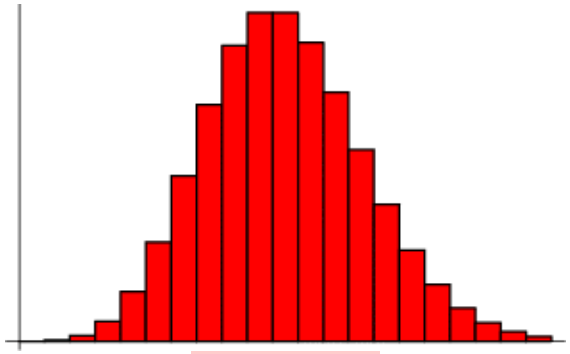
Common Discrete Distributions



Binomial



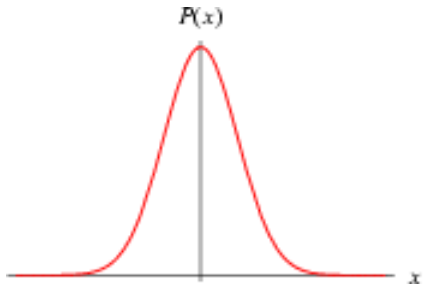
Geometric

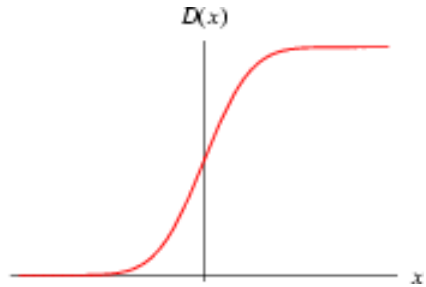


Poisson

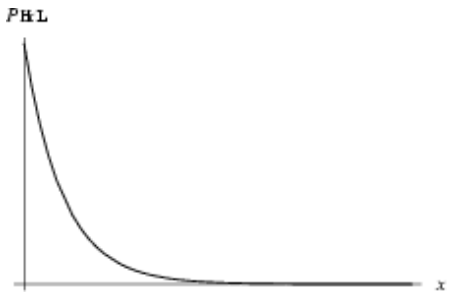
Common Continuous Distributions

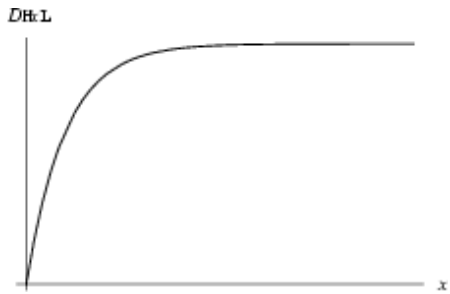
Normal



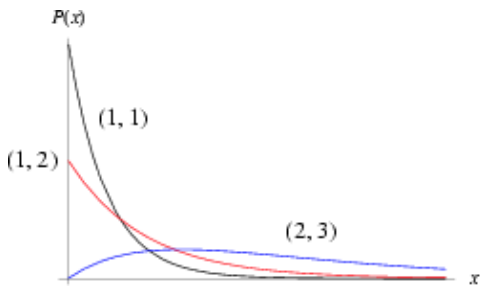


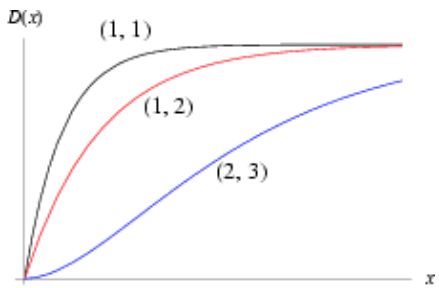
Exponential





Gamma





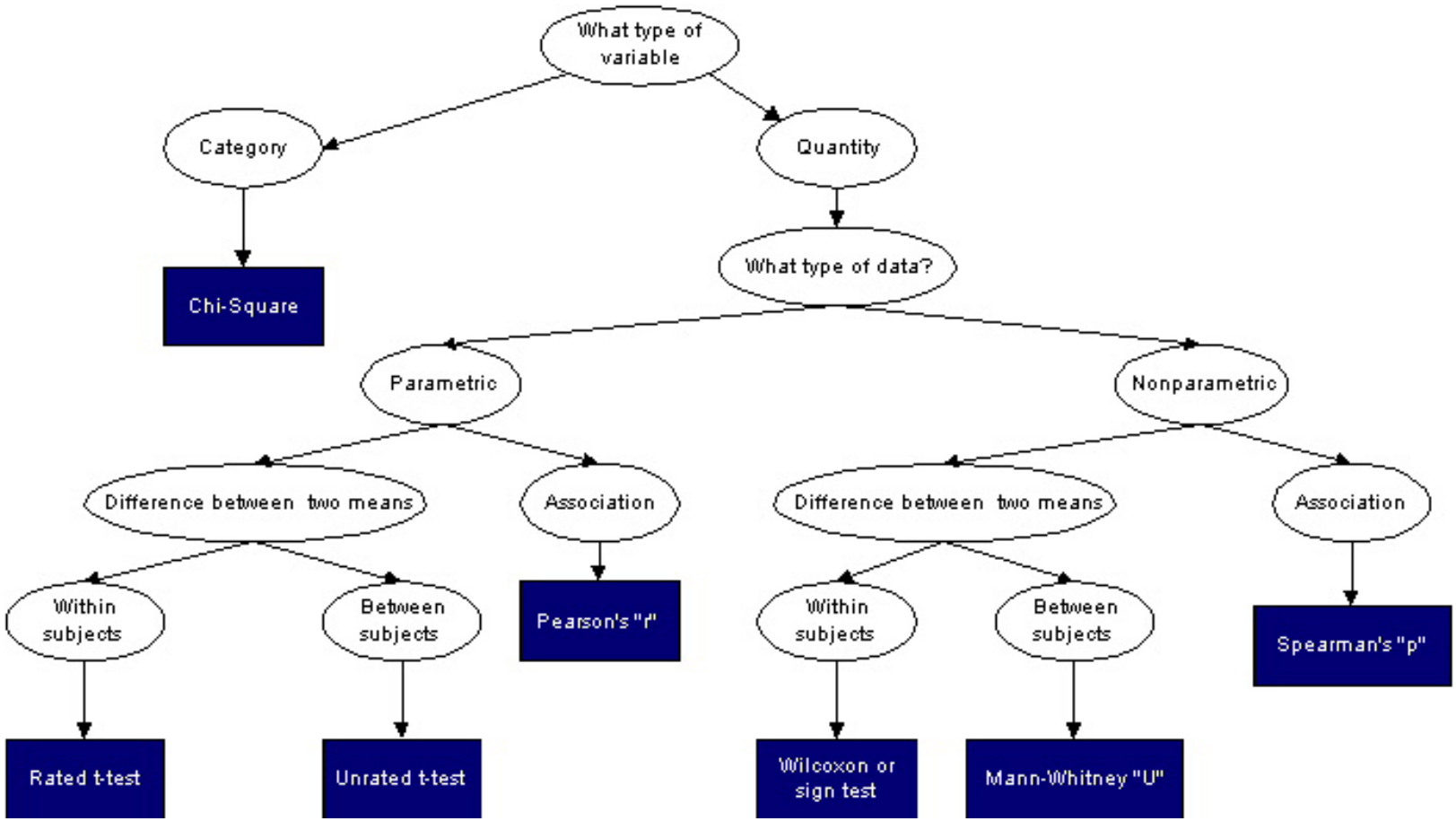
Monte Carlo methods

- Numerical statistical simulation methods that utilize sequences of random numbers to perform the simulation.
- The primary components of a Monte Carlo simulation method include the following:
  - *Probability distribution functions (pdf's)* --- the system described by a set of pdf's.
  - *Random number generator* --- uniformly distributed on the unit interval.
  - *Sampling rule* --- a prescription for sampling from the specified pdf's.
  - *Scoring (or tallying)* --- the outcomes must be accumulated into overall tallies or scores for the quantities of interest.
  - *Error estimation* --- an estimate of the statistical error (variance) as a function of the number of trials and other quantities must be determined.
  - *Variance reduction techniques* --- methods for reducing the variance in the estimated solution to reduce the computational time for Monte Carlo simulation.

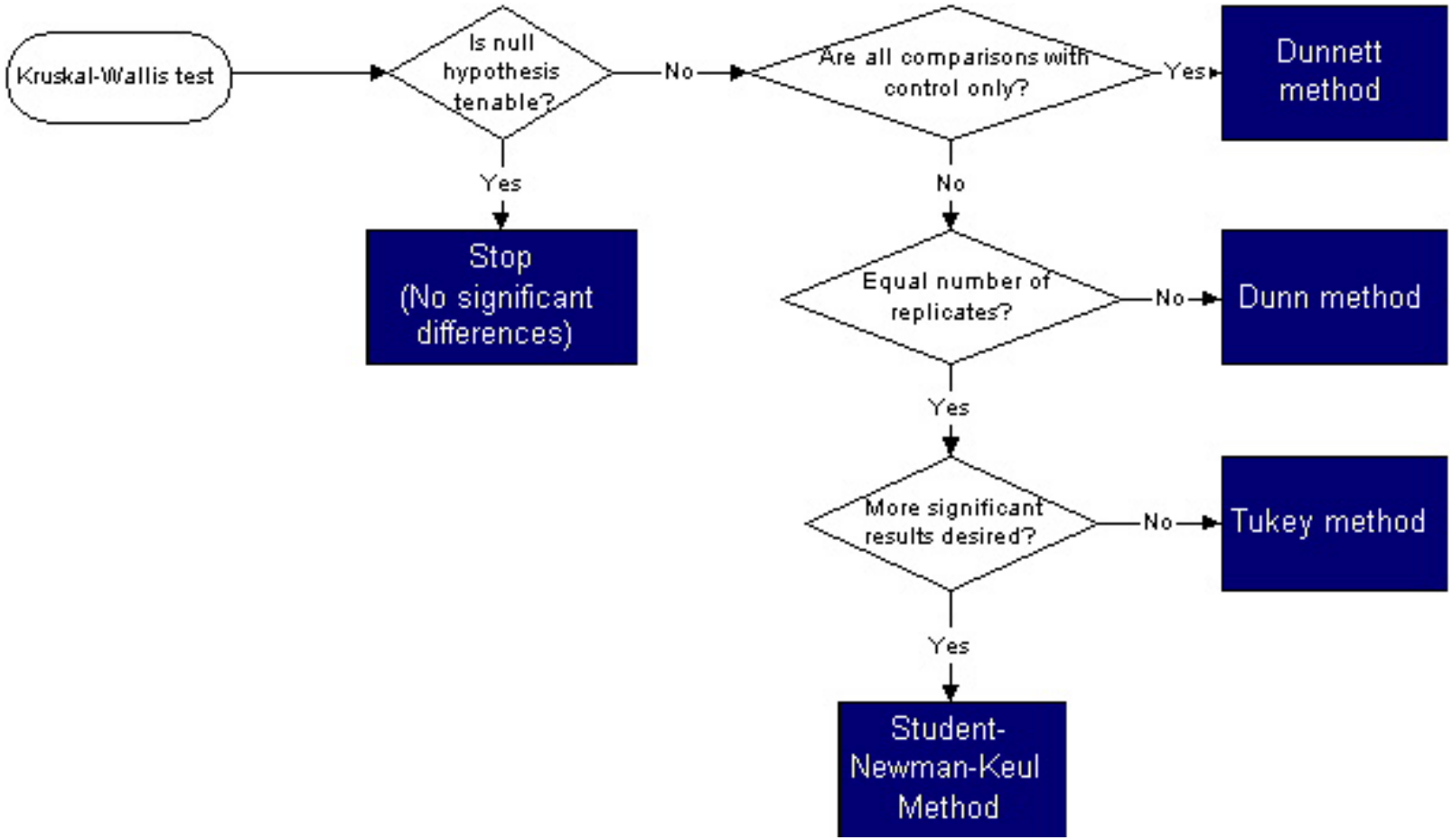
Parametric & Non-parametric equivalents

Type of test	Parametric test	Nonparametric test
2-sample	t-test	Mann-Whitney U-test
Paired sample	Paired t-test	Wilcoxon
Distribution	Chi-square	Kolmogorov-Smirnov
>2 samples	1-way ANOVA	Kruskal-Wallis
Correlation	Pearson's correlation	Spearman's correlation
Crossed comparisons	Factorial ANOVA	Friedman's Quade
Multiple comparisons	Tukey, SNK, Dunnett's, Scheffe's	Nonparametric version of its parametric equivalents

Selection of Statistical Tests



Selection of Multiple Comparison Tests



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

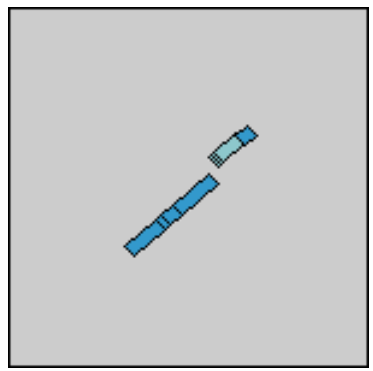
Sequences producing significant alignments:	Score (bits)	E Value
gi|332623|gb|J02396.1|SEG_SSVPCS2 Simian sarcoma virus v-si...	4591	0.0
gi|61774|emb|V01201.1|RESSV1 Simian sarcoma virus proviral ...	4504	0.0
gi|332622|gb|J02395.1|SEG_SSVPCS1 Simian sarcoma virus LTR ...	1283	0.0
gi|885929|gb|U20589.1|GLU20589 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

Homework #1

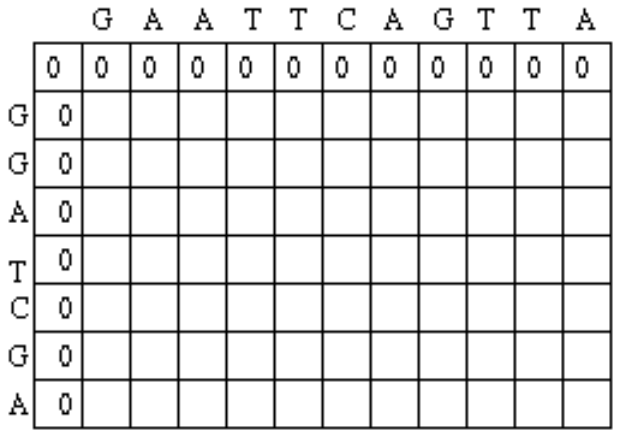
- Check out homework #1 on course homepage.

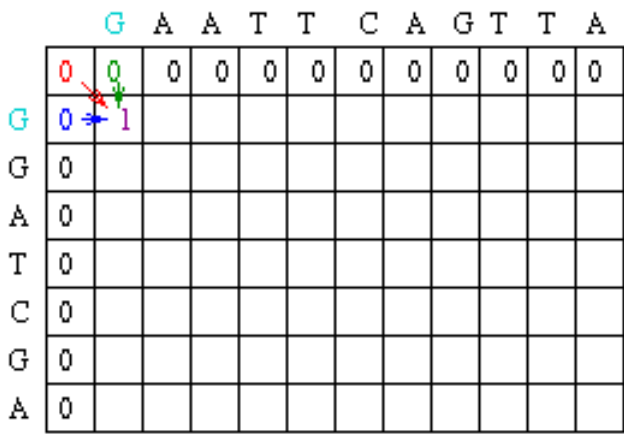
Global Sequence Alignment

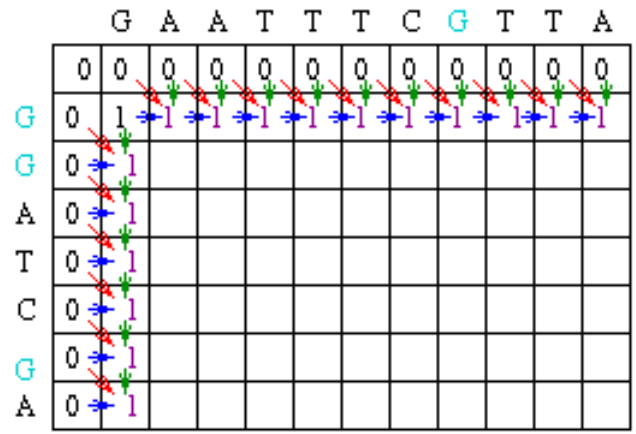
- Needleman-Wunsch-Sellers algorithm.
- Dynamic Programming (DP) based.
  - Overlapping Subproblems
  - Recurrence Relation
  - Table to store solutions to subproblems
  - Ordering of subproblems to fill table
  - Traceback to find solution

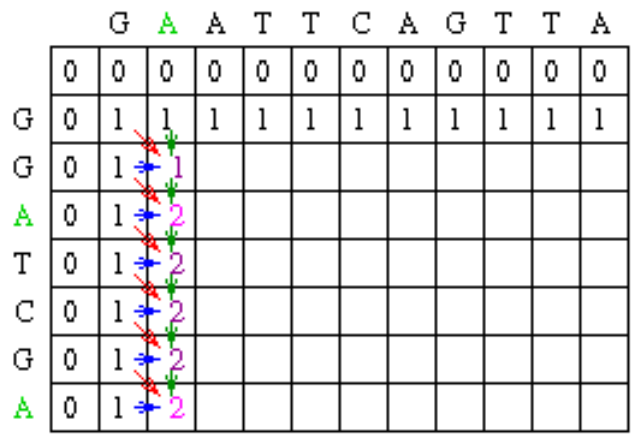
Global Alignment: An example

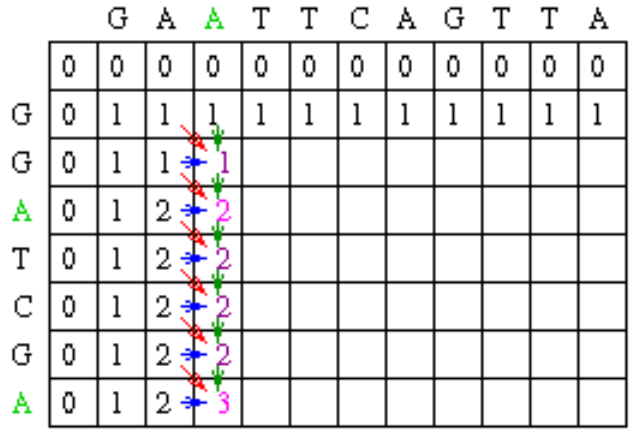
```
V: G A A T T C A G T T A
W: G G A T C G A
```

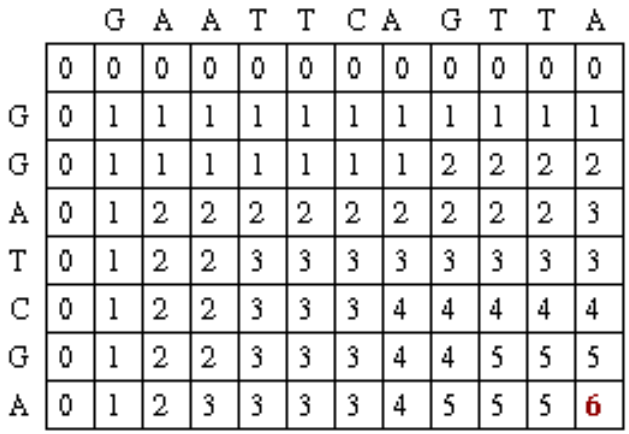












Recurrence Relation for Needleman-Wunsch-Sellers

- $S[I, J] = \text{MAXIMUM} \{$

  $S[I-1, J-1] + \delta(V[I], W[J]),$

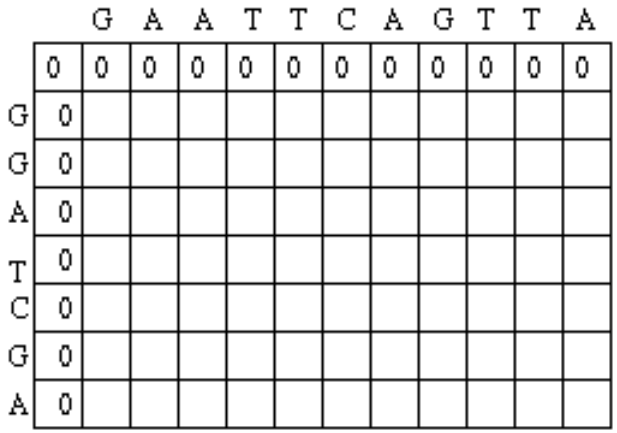
  $S[I-1, J] + \delta(V[I], —),$

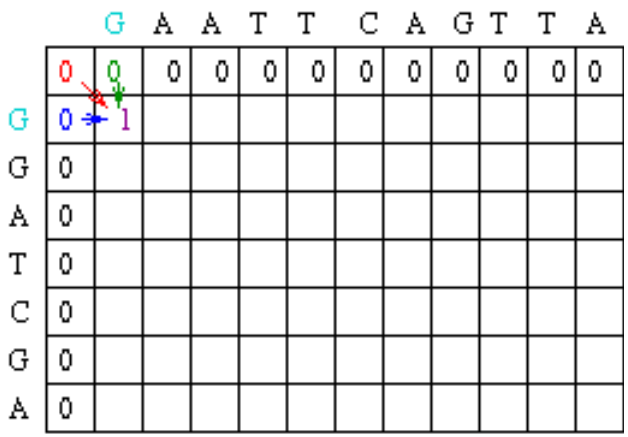
  $S[I, J-1] + \delta(— , W[J])$

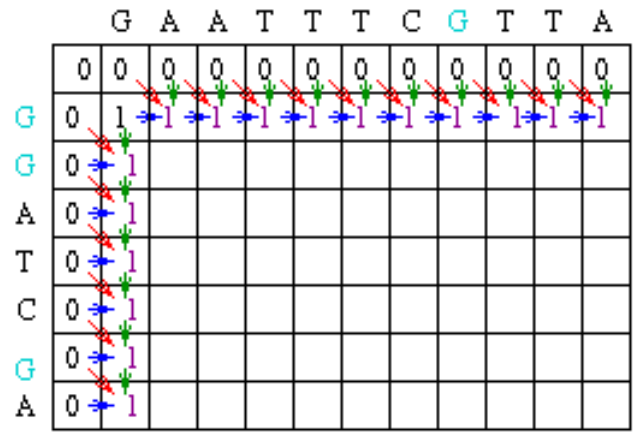
  $\}$

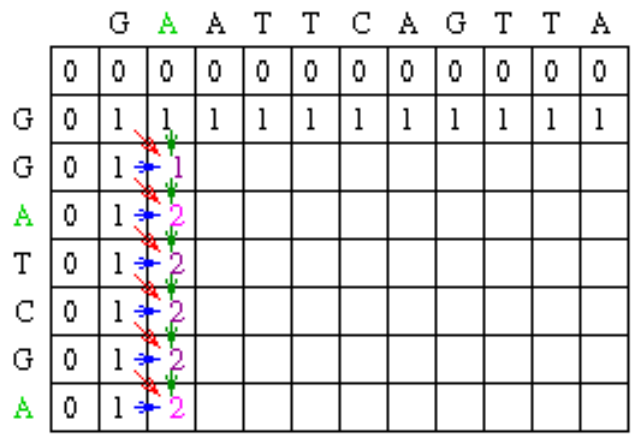
Global Alignment: An example

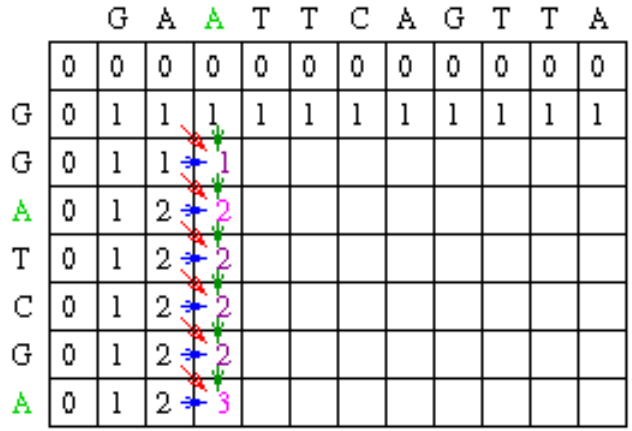
```
V: G A A T T C A G T T A
W: G G A T C G A
```

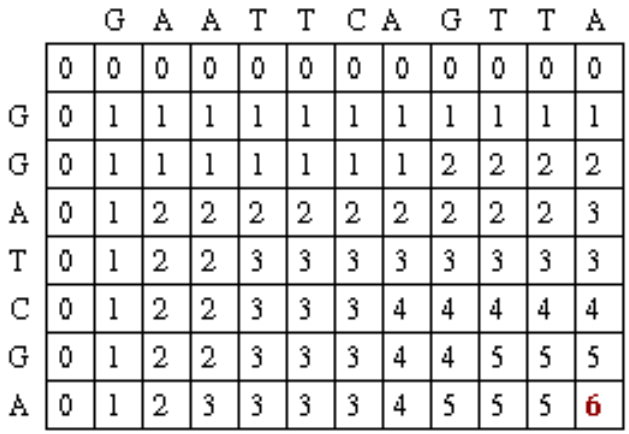




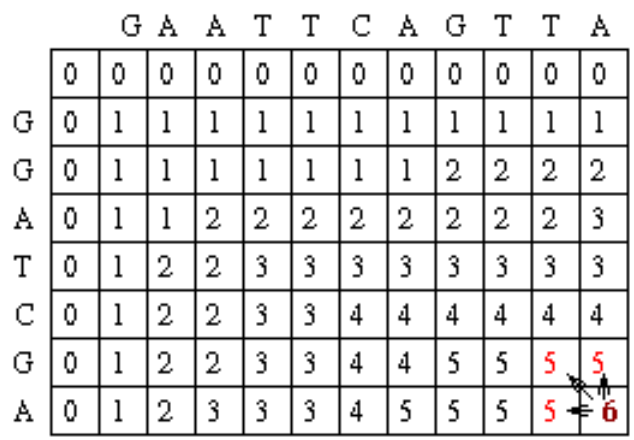


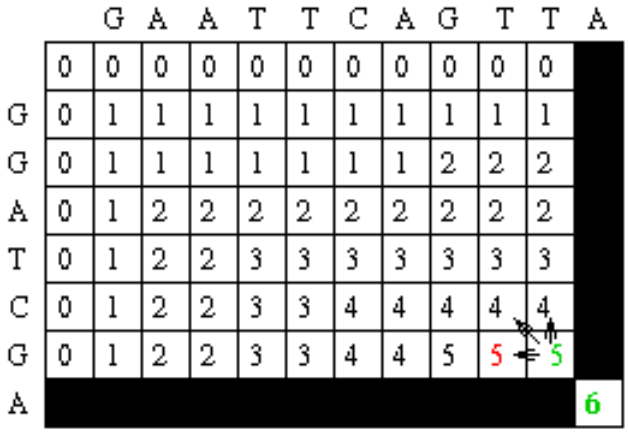


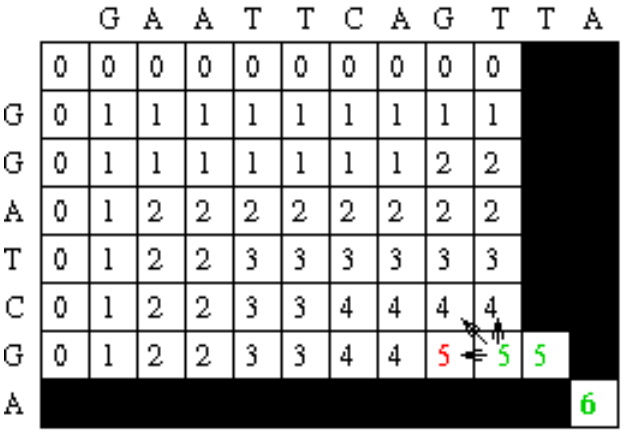


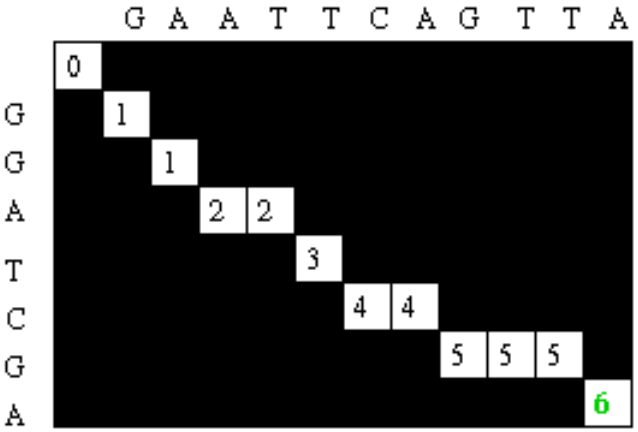


Traceback



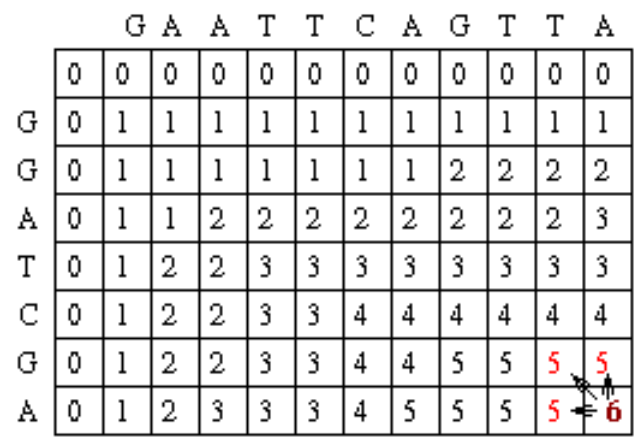


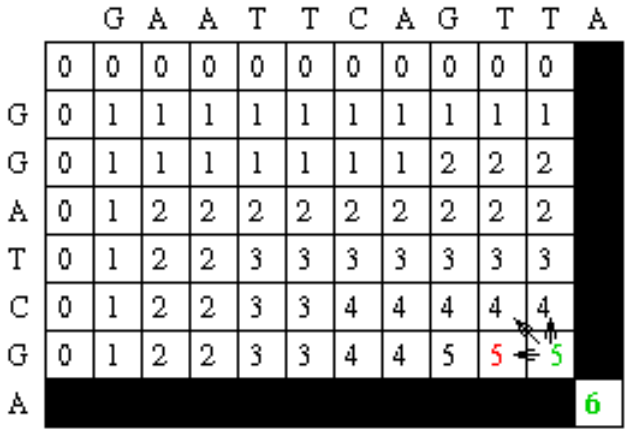


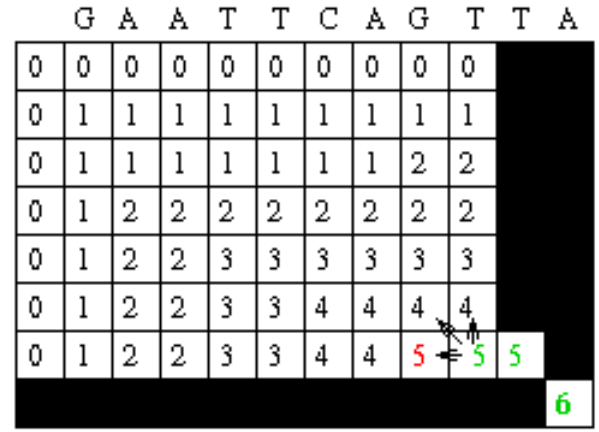


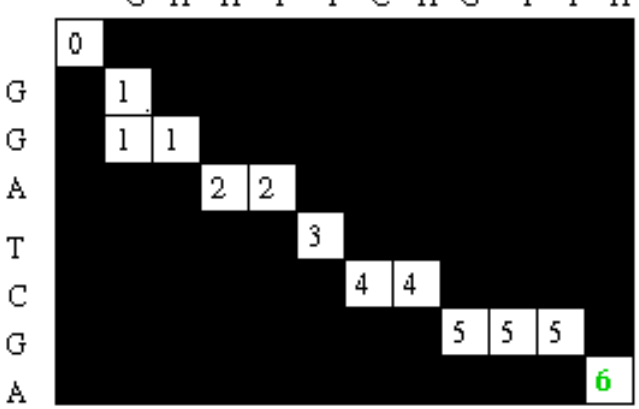

```
V: G A A T T C A G T T A
   |   |   | |   |     |
W: G G A - T C - G - - A
```

Alternative Traceback










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

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]

Time Complexity

- 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)

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

Global Alignment

- $S[I, J] = \text{MAXIMUM} \{ 0,$
  $S[I-1, J-1] + \delta(V[I], W[J]),$
  $S[I-1, J] + \delta(V[I], —),$
  $S[I, J-1] + \delta(— , W[J]) \}$

Local Alignment

Local Alignment: Example

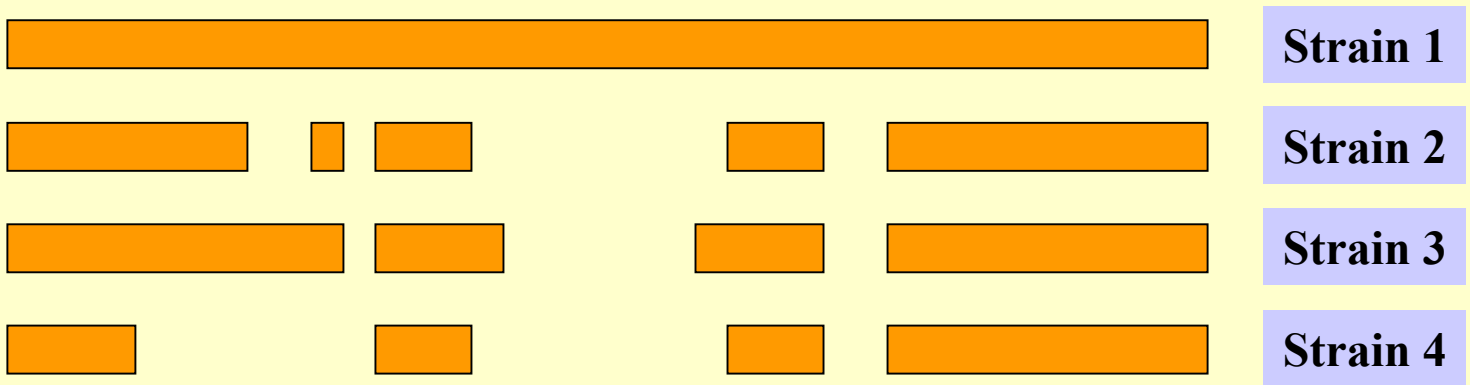
		G	A	A	T	T	C	A	G	T	T	A
	0	0	0	0	0	0	0	0	0	0	0	0
G	0	× 1	0	0	0	0	0	0	0	0	0	0
G	0	× 1	← 0	0	0	0	0	0	× 1	0	0	0
A	0	0	× 2	× 1	0	0	0	× 1	0	0	0	× 1
T	0	0	↑ 0	× 1	× 2	← 1	0	0	0	× 1	× 1	0
C	0	0	0	0	↑ 0	× 0	× 2	0	0	0	0	0
G	0	0	0	0	0	0	0	0	× 1	0	0	0
A	0	0	× 1	× 1	0	0	0	× 1	0	0	0	× 1

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
```

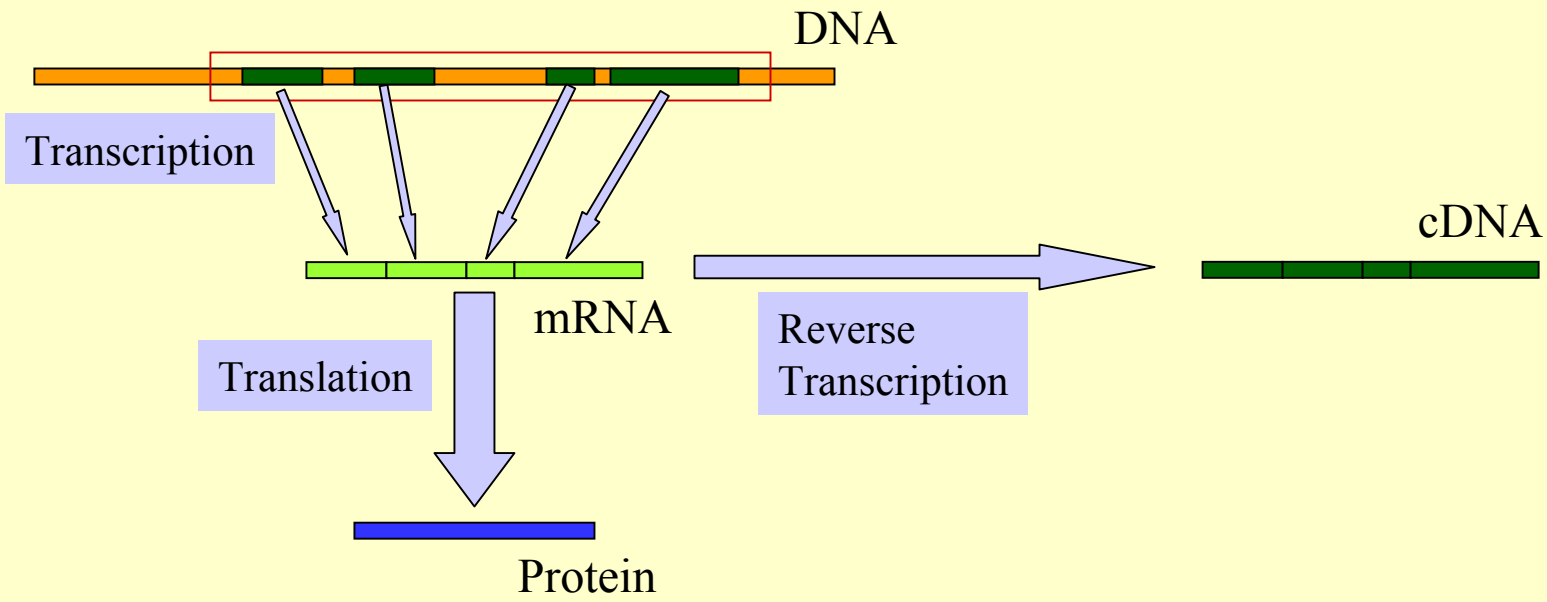
Why Gaps?

- Example: Finding the gene site for a given (eukaryotic) cDNA requires “gaps”.
- Example: HIV-virus strains



What is cDNA?

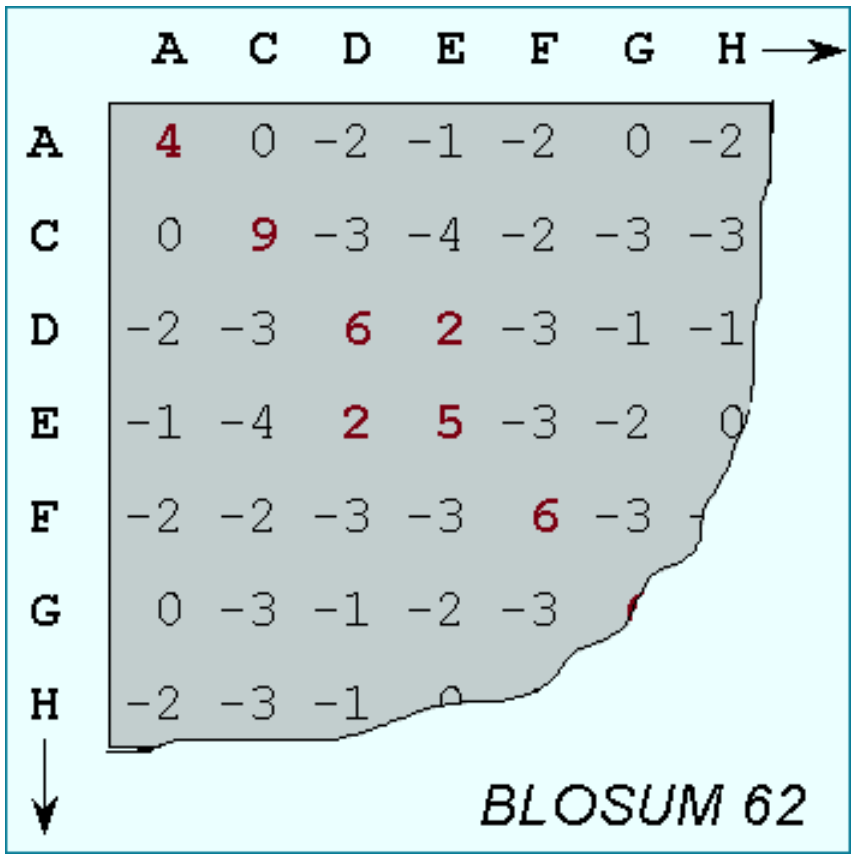
- cDNA = Copy DNA



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**.

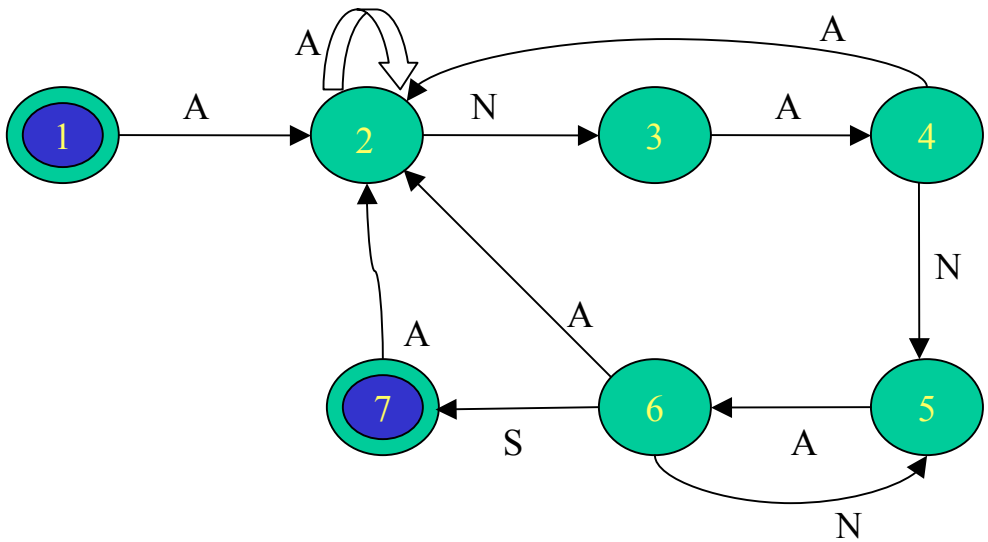
Methods:

- 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

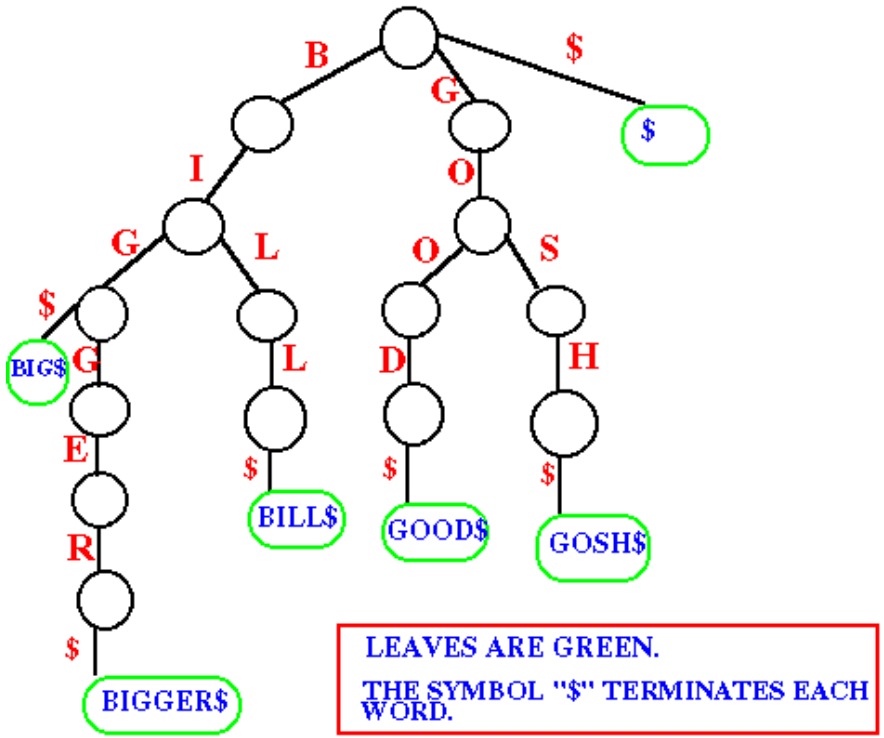
    Scan: if (window = P) then report it;

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

endwhile;

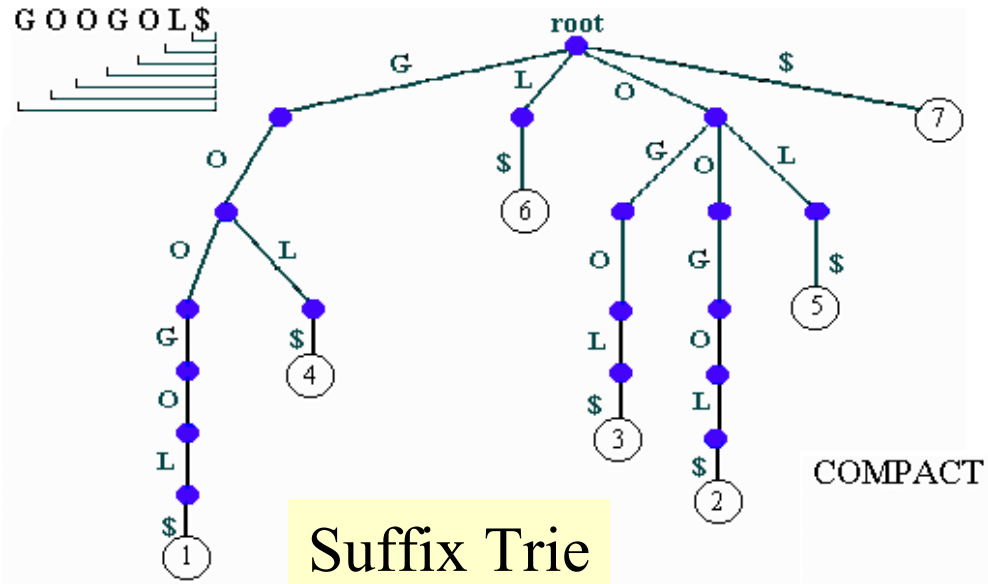
Tries

Storing:
BIG
BIGGER
BILL
GOOD
GOSH



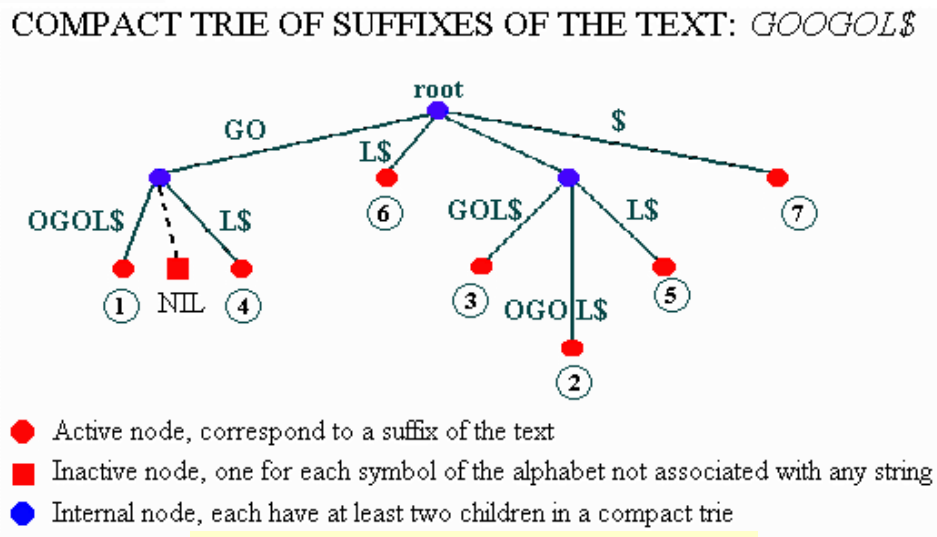
In this figure, the strings either start with B or G. Therefore, the root of the trie is connected to 3 edges called B, G and $.

Suffix Tries & Compact Suffix Tries



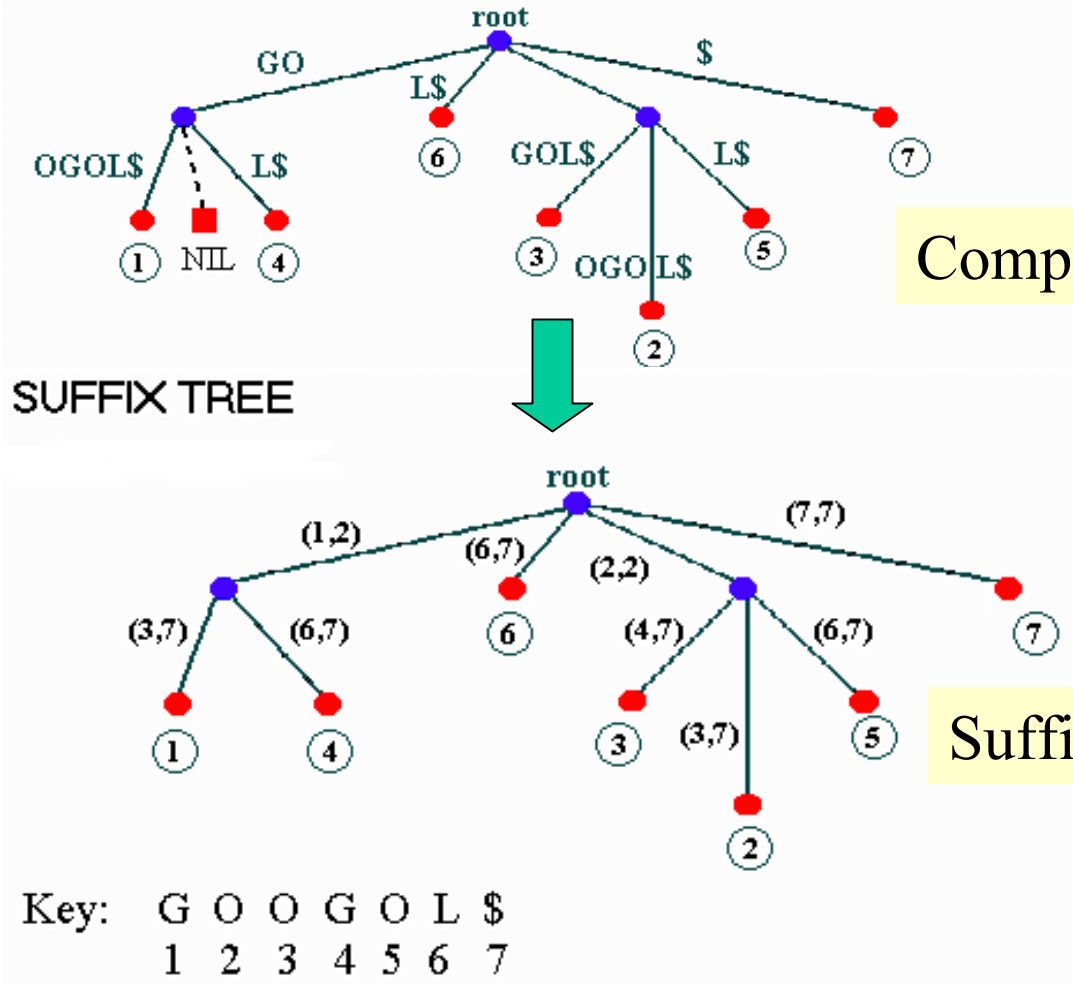
Suffix Trie

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