# DSCI 575: Advanced Machine Learning

Sequence Mining

Winter 2017

# Sequence Mining

- Finding patterns in data organized according to a sequence:
  - Customer purchases:
    - ‘Star Wars’ followed by ‘Empire Strikes Back’ followed by ‘Return of the Jedi’.
  - Stocks/bonds/markets:
    - Stocks going up followed by bonds going down.
  - Environmental:
    - $CO_2$ going up is followed by temperatures going up.
  - Website/telephone system navigation.
  - Biological sequences.
    - DNA: ATGCTTCGGCAAGACTCAAAAAATA…
    - RNA: ATGCUUCGGCAAGACUCAAAAAAUA…
    - Protein: GIVEQCCTSICSLYQLENYCN

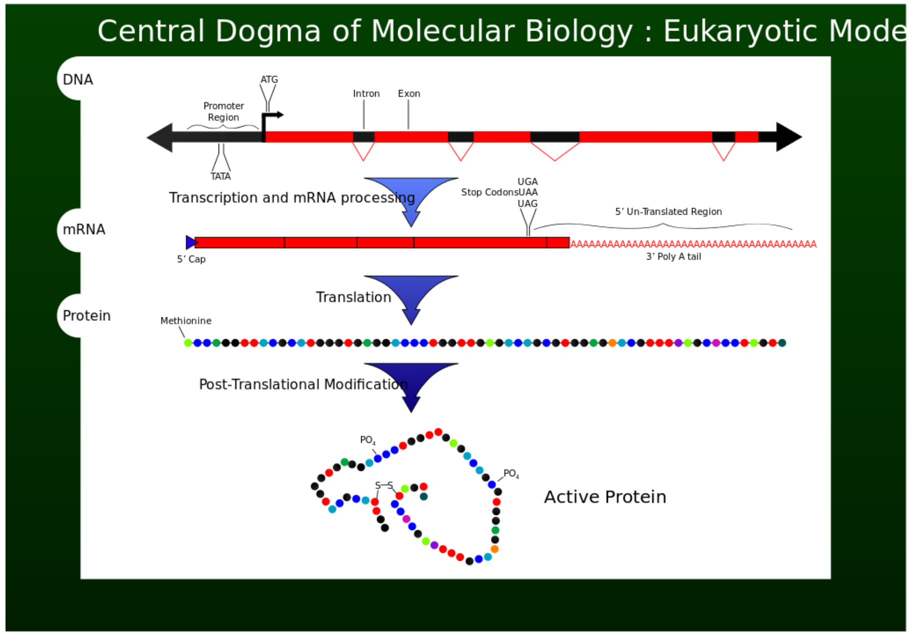
# Sequential Pattern Analysis

- In data mining, called sequential pattern analysis:
  - If you buy product A, are you likely to buy product B at a later time?
- Similar to association rules, but now order matters.
  - Many issues stay the same.
- Exist sequential generalization of association rule methods:
  - Generalized sequential pattern (GSP) algorithm is like a priori algorithm.

- We're going to instead focus on methods from bioinformatics…

## Biological Sequences

- We are generated huge quantities of biological data.
- Much of it is stored as sequences.
  - DNA, RNA, and proteins.



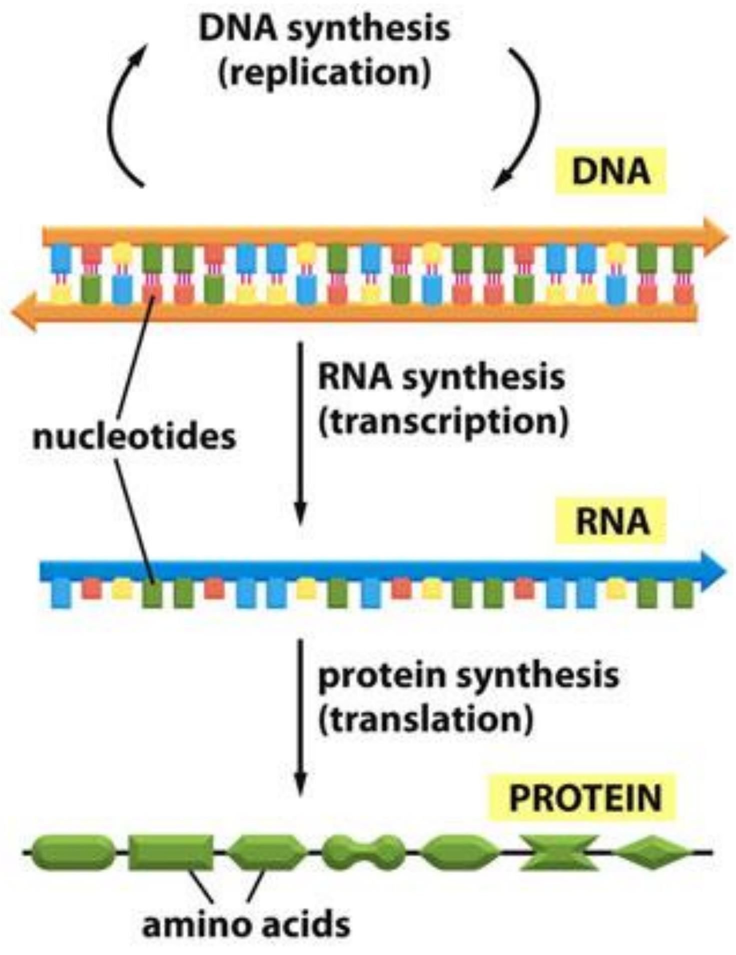
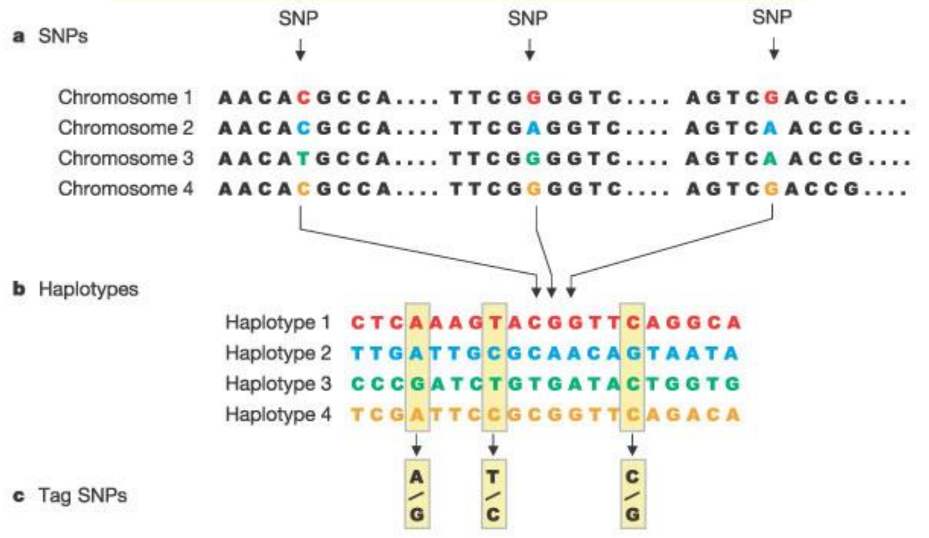

DNA synthesis (replication)
DNA
nucleotides
RNA synthesis (transcription)
RNA
protein synthesis (translation)
PROTEIN
amino acids

Figure 1-2 Essential Cell Biology 3/e (© Garland Science 2010)

http://trenchesofdiscovery.blogspot.ca/2012/09/the-human-machine-coding-and-uncoding.html
https://en.wikipedia.org/wiki/Central_dogma_of_molecular_biology

# Whole Genome Sequencing

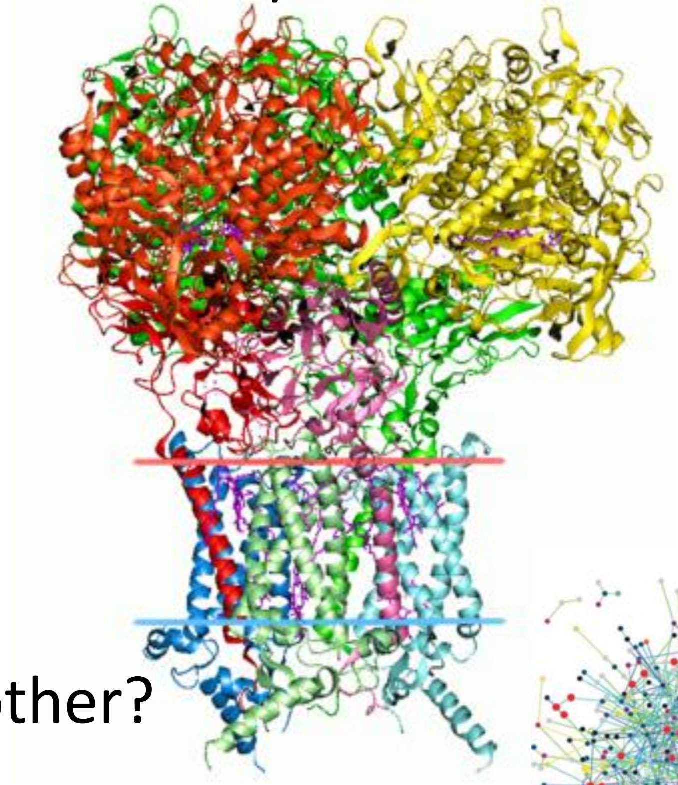
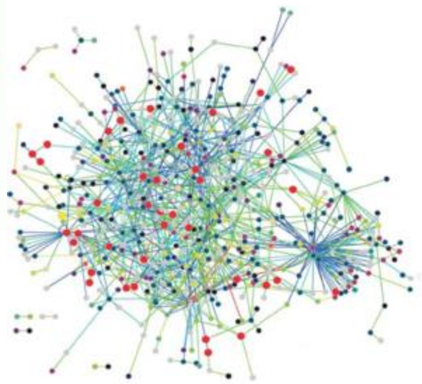
- First single-celled organisms' genomes sequenced in late 90s.
- Many animals/plants in early 2000s.
- Human genome project finished in 2003.
- Late 2000s and 2010s:
  - Characterizing variation and function.
  - HapMap, ENCODE, 1000 genomes, 23andMe.
  - Potential to study infrequent variations.
  - New insights into rare diseases.
  - Promise of personalized medicine.
- Way more data than understanding:
  - One of most important scientific problems.


a SNPs
SNP
SNP
SNP
Chromosome 1 AACACGCCA.... TTCGGGGTC.... AGTCGACCG....
Chromosome 2 AACACGCCA.... TTCGAGGTC.... AGTCA ACCG....
Chromosome 3 AACATGCCA.... TTCGGGGTC.... AGTCA ACCG....
Chromosome 4 AACACGCCA.... TTCGGGGTC.... AGTCGACCG....
b Haplotypes
Haplotype 1 CTCAAAGTACGGTTCAGGCA
Haplotype 2 TTGATTGCGCAACAGTAATA
Haplotype 3 CCCGATCTGTGATACTGGTG
Haplotype 4 TCGATTCGCGGTTCAGACA
c Tag SNPs
A/G
T/C
C/G

http://hapmap.ncbi.nlm.nih.gov/whatishapmap.html

# Bioinformatics

- Bioinformatics: biology and databases and data analysis.
  - It's a huge area, with many interesting variations on DM/ML methods.
- Big focus on analyzing sequences.
  - We'll discuss some of the classic ideas today.
- But sequences aren't everything:
  - How do molecules 'fold' in three-dimensions?
  - Which molecules can 'fit' together?
  - What genes perform similar functions?
  - How do molecule concentrations affect each other?
  - What are signaling 'pathways'?

## Finding/Testing Similar Sequences

- A classic bioinformatics problem:
  - You find an interesting part of a biological sequence.
    - E.g., this gene makes your mice live much longer or immune to a disease.
  - Do similar sequences appear elsewhere?
    - Either in the same organism, or in other organisms.
- Want to test relatedness of sequences and find related sequences.
  - Heavy use of dynamic programming.
  - Other tricks to handle huge datasets.
- We'll start from simplest case, and get more complicated.

# String Search

- Simplest variant is string search:
  - We have a sequence of length ‘n’
  - We have a query of length ‘m’.
  - Does query occur in sequence?
- Example:
  - Sequence: “GIVEQCCTSICSLYQLENYCN” (insulin).
  - Query: “TSI”.
- Naïve algorithm:
  - For each of ‘n’ positions, test whether the string starts there.
  - Cost is O(nm).
- Several algorithms reduce this to O(n + m) (e.g., Knurth-Morris-Pratt).

# Longest Common Substring

- What if we have multiple queries for same sequence?
  - Sequence: "GIVEQCCTSICSLYQLENYCN".
  - Queries: "TSI", "CCT", "CST" (diabetes).
  - With 'k' queries of length <= 3, cost is O(n + km) with suffix trees.
- A related problem is longest common substring:
  - Sequence 1: "GIVEQCCTSICSLYQLENYCN" (human).
  - Sequence 2: "GIVEQCCASVCSLYQLENYCN" (cow).
  - What is longest string that occurs in both sequences?
    - In this case it's "CSLYQLENYCN".
- Suffix trees solve this problem in O(n + m).

(pause)

# Longest Common Substring vs. Subsequence

- Consider human/pig/cow insulin:
  - Sequence 1: “GIVEQCCTSICSLYQLENYCN” (human).
  - Sequence 2: “GIVEQCCASVCSLYQLENYCN” (cow).
  - Sequence 3: “GIVEQCCTSICSLYQLENYCN” (pig).
- Longest substring between human/pig is 22 (entire sequence).
- Longest substring between human/cow is 11: “CSLYQLENYCN”.
  - But have we really cut the similarity in half?
- Longest common subsequence:
  - Longest exact match by deleting characters.
  - For human/cow it’s 20: “GIVEQCCSCSLYQLENYCN” (still 22 for human/pig).

# Longest Common Subsequence

- Longest common subsequence (LCS):
  - Sequence 1: “GIVEQCCTSICSLYQLENYCN” (human).
  - Sequence 2: “GIVEQCCASVCSLYQLENYCN” (cow).
  - LCS: “GIVEQCC[ ]S[]CSLYQLENYCN”.
- Basis of most ‘diff’ commands (and version control like git).
- Finding LCS by brute force:
  - $2^n$ possible deletions in sequence 1.
  - $2^m$ possible deletions in sequence 2.
  - $O(\min(n,m)2^{n+m})$.
- Can we do better?

# Longest Common Subsequence

- Suppose we have the LCS for two sequences:
  - Sequence 1: “ACE”.
  - Sequence 2:“ABCD”.
  - LCS: “AC”.
- Key idea: it’s easy to update LCS if we append one character.
  - Updated sequence 2: “ABCDE”.
  - New LCS: “ACE”.
  - Either the new character extends LCS or not: compute this in O(m).
- O(mn)-time Algorithm:
  1. Start with all of sequence 1 and empty sequence 2 (LCS = []).
  2. Sequentially append sequence 2 to sequence 2, tracking LCS.

# Dynamic Programming

- LCS algorithm is special case of dynamic programming.
- Dynamic programming efficiency requires two ingredients:
  1. Optimal substructure:
     - Can efficiently solve the problem given solutions to ‘sub-problems’ (i.e. recursion).
     - For LCS: we can quickly solve problem of length ‘m’ given solution of length (m-1).
  2. Overlapping sub-problems:
     - Limited number of *different* possible sub-problems.
     - For LCS: there are only O(mn) possible lengths for the two strings.

- Key trick: store solutions of sub-problems, instead re-computing.
  – Guarantees each sub-problem is solved at most once.

# LCS with Dynamic Programming

- Let’s define the LCS recursively:

$$LCS\left(X_i, Y_j\right) = \begin{cases} \emptyset & \text{if } \; i = 0 \text{ or } j = 0 \\ LCS\left(X_{i-1}, Y_{j-1}\right) \frown x_i & \text{if } x_i = y_j \\ \text{longest}\left(LCS\left(X_i, Y_{j-1}\right), LCS\left(X_{i-1}, Y_j\right)\right) & \text{if } x_i \neq y_j \end{cases}$$

- Exponential number of recursive calls in naïve method:

$$\begin{tikzcd}
 & & & (n,m) \arrow[dll] \arrow[drr] & & & & \\
 & (n,m-1) \arrow[dl] \arrow[dr] & & & & (n-1,m) \arrow[dl] \arrow[dr] & & \\
(n,m-2) \arrow[d] \arrow[dr] & & (n-1,m-1) \arrow[d] \arrow[dr] & & (n-1,m-1) \arrow[d] \arrow[dr] & & (n-2,m) \arrow[d] \arrow[dr] & \\
(n,m-3) & (n-1,m-2) & (n-1,m-2) & (n-2,m-1) & (n-1,m-2) & (n-2,m-1) & (n-2,m-1) & (n-3,m)
\end{tikzcd}$$

https://en.wikipedia.org/wiki/Longest_common_subsequence_problem

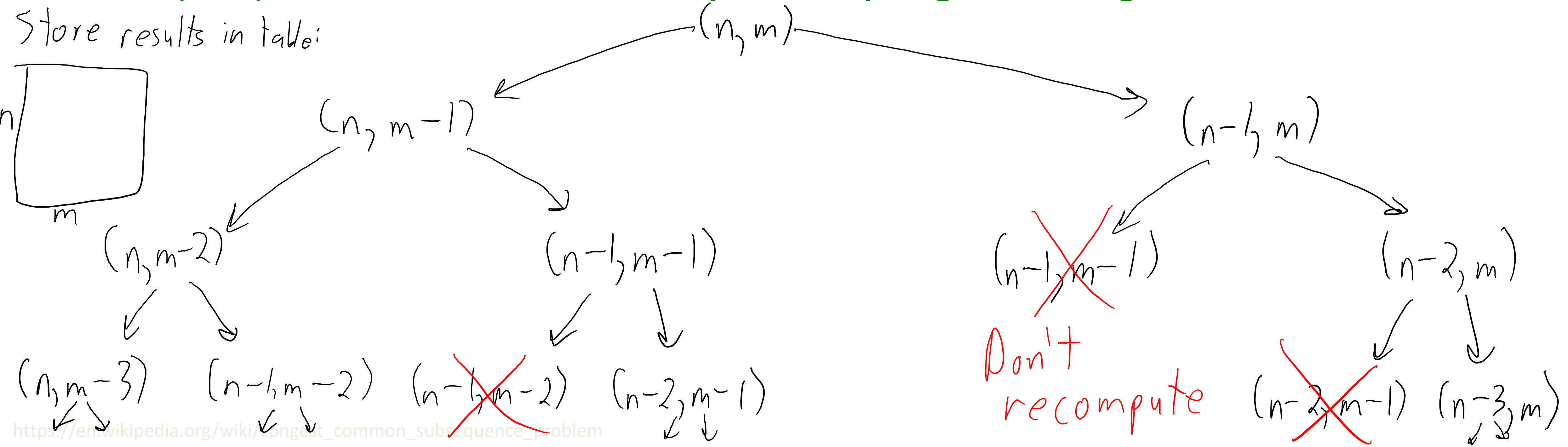
# LCS with Dynamic Programming

- Let's define the LCS recursively:

$$LCS(X_i, Y_j) = \begin{cases} \emptyset & \text{if } \ i = 0 \text{ or } j = 0 \\ LCS(X_{i-1}, Y_{j-1}) \frown x_i & \text{if } x_i = y_j \\ \text{longest}(LCS(X_i, Y_{j-1}), LCS(X_{i-1}, Y_j)) & \text{if } x_i \neq y_j \end{cases}$$

- O(mn) recursive calls with dynamic programming method:

Store results in table:



https://en.wikipedia.org/wiki/Longest_common_subsequence_problem

# Edit Distance

```
GAATTCAG       GAATTCAG
| | || |       | || | |
GGA-TC-G       GCAT-C-G

GAATTC-A       GAATTC-A
| | || |       | || | |
GGA-TCGA       GCAT-CGA
```

- LCS considers deletions of elements.
- We might also consider replacements:
  - Sequence 1: “GIVEQCCTSICSLYQLENYCN”.
  - Sequence 2: “GIVEQCCASVCSLYQLENYCN”.
  - Where different replacements have different ‘costs’.
    - Some proteins can be substituted and molecule will be similar, some are disastrous.
- Edit distance:
  - Min ‘cost’ of turning string 1 into 2 via additions/deletions/replacements.
  - Can also be computed by dynamic programming:
    - Minimize over the 3 operations.

# Edit Distance

- Edit distance between strings ‘X’ and ‘Y’ is $ED(X_m,Y_n)$ where is min of:

$$ED(X_i, Y_j) = \begin{cases} ED(X_{i-1}, Y_{i-1}) & \text{if } X_i = Y \\ \min\begin{cases} ED(X_{i-1}, Y_j) + \text{cost}(\text{'delete } X_i\text{'}) \\ ED(X_i, Y_{j-1}) + \text{cost}(\text{'insert } Y_j\text{'}) \\ ED(X_{i-1}, Y_{j-1}) + \text{cost}(\text{'replace } X_i \text{ with } Y_j\text{'}) \end{cases} & \text{if } \begin{matrix} 1 < i \leq n \\ 1 < j \leq m \end{matrix} \text{ and } X_i \neq Y_i \\ \sum_{k=1}^{i} \text{cost}(\text{'delete } X_k\text{'}) & \text{if } j = 0 \\ \sum_{k=1}^{j} \text{cost}(\text{'insert } Y_k\text{'}) & \text{if } i = 0 \end{cases}$$

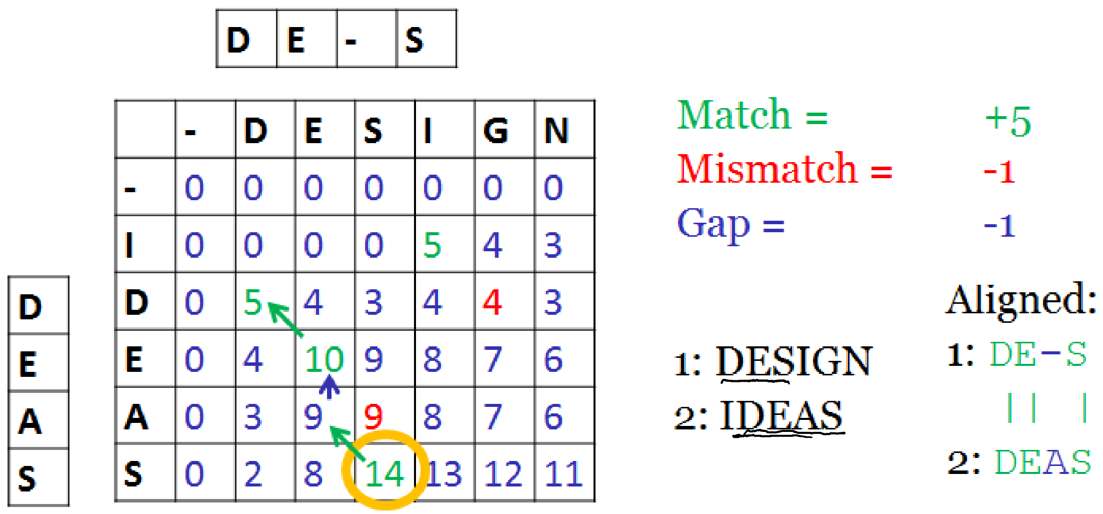
- Cost is still $O(mn)$, and if costs are non-negative this is a distance.

(pause)

# Local Edit Distance / Local Alignment

- Local alignment (Smith-Waterman):
  - Positive 'score' for matches, negative 'score' for add/delete/replace.
  - Set negative '$d_{ij}$' values to zero, and maximize $d_{ij}$ over 'i' and 'j'.
    - Note that in bioinformatics you maximize 'score' rather than minimize 'distance'.
  - Finds substrings with small edit distance:

**Smith-Waterman Scoring**



| D | E | - | S |
|---|---|---|---|

|   | - | D | E | S | I | G | N |
|---|---|---|---|---|---|---|---|
| - | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| I | 0 | 0 | 0 | 0 | 5 | 4 | 3 |
| D | 0 | 5 | 4 | 3 | 4 | 4 | 3 |
| E | 0 | 4 | 10 | 9 | 8 | 7 | 6 |
| A | 0 | 3 | 9 | 9 | 8 | 7 | 6 |
| S | 0 | 2 | 8 | 14 | 13 | 12 | 11 |

| D | E | A | S |
|---|---|---|---|

| | |
|---|---|
| Match = | +5 |
| Mismatch = | -1 |
| Gap = | -1 |

1: DESIGN
2: IDEAS

Aligned:
1: DE−S
   | |  |
2: DEAS

http://2012.igem.org/Team:Johns_Hopkins-Software/Cloud

## BLAST

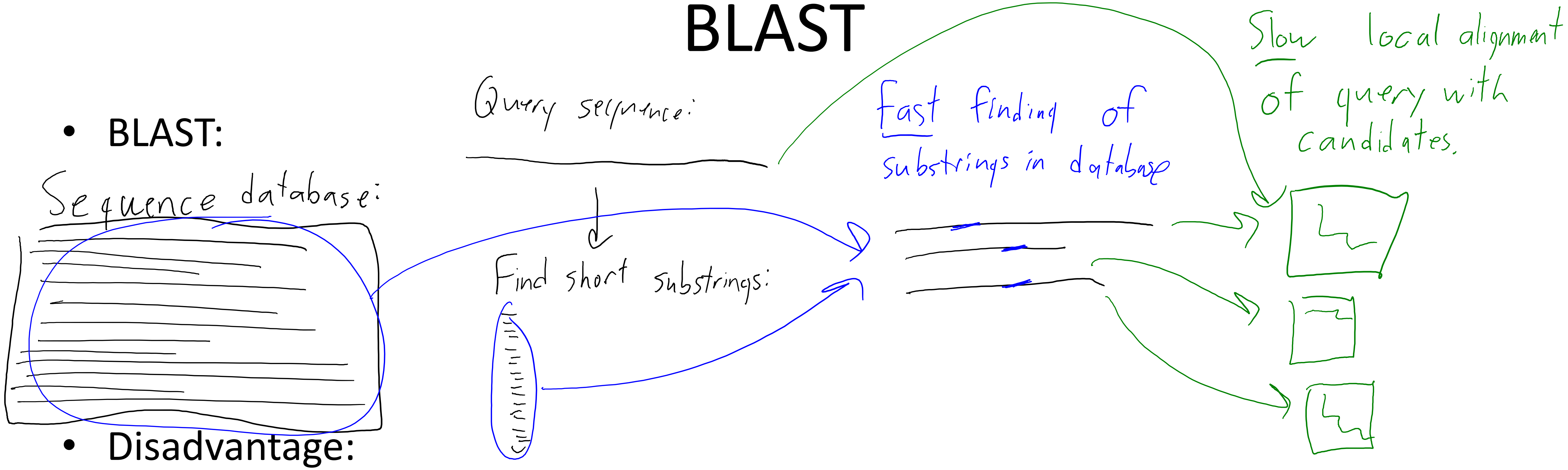
- Basic Local Alignment Search Tool (BLAST):
  - A method for searching biological sequences.
  - Most cited paper in 1990s of all of science.
- Setup:
  - We have a huge database of sequences.
  - Individual sequences may be very long (human genome: ~3.2 billion).
  - Quickly find similar sequences to a query sequence.
- Key ideas:
  - Find interesting and short substrings in query.
  - Fast phase: Find 'candidates' that contain any substring.
  - Slow phase: apply dynamic programming on the 'candidates'.
  - Some other tricks to make it faster.

If there all 'ℓ' strings in database, finding indices of 'k' substrings of length 'm' costs O(kmℓ).

No dependence on length of database sequences.

# BLAST

- BLAST:

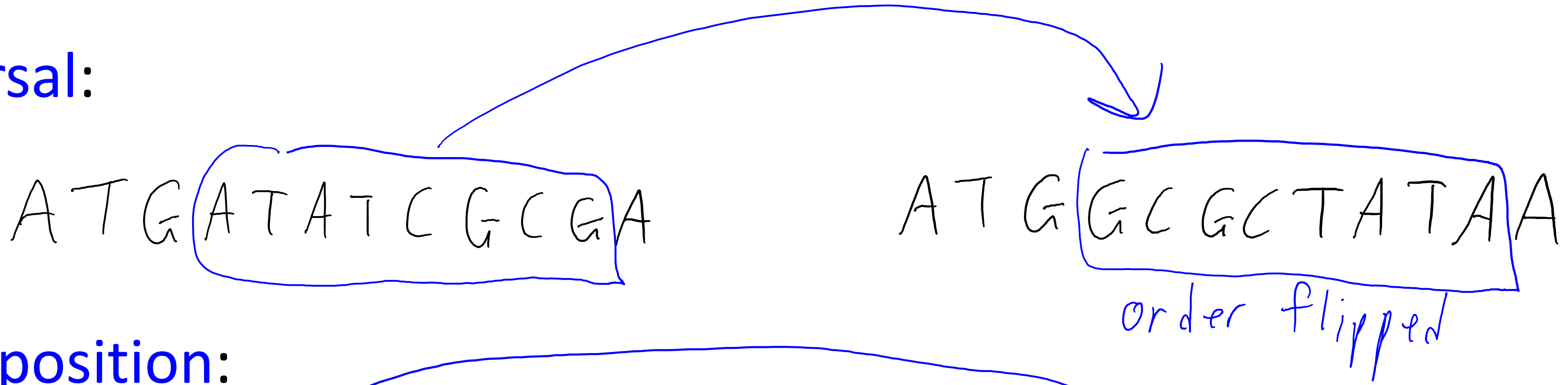


Sequence database:
Query sequence:
Find short substrings:
Fast finding of substrings in database
Slow local alignment of query with candidates.

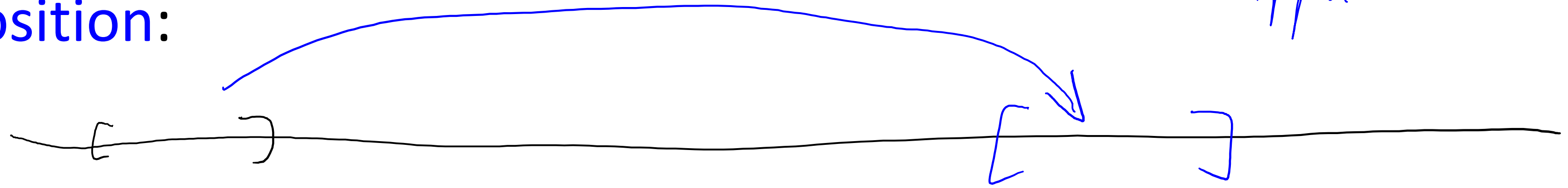
- Disadvantage:
  - You could have false negatives in the first phase (you miss distantly-related sequences).
- PSI-BLAST:
  - Re-run with related sequences to find more distantly-related sequences.
- Related to hashing tricks for finding elements of a set:
  - Bloom filter: guaranteed to have no false negatives.
  - Count-min sketch: more recent probabilistic/online method.

# Generalizations of Edit Distance

- We can have score based on insertion/deletion length ('gap score')
- Other common mutations:
  - Reversal:



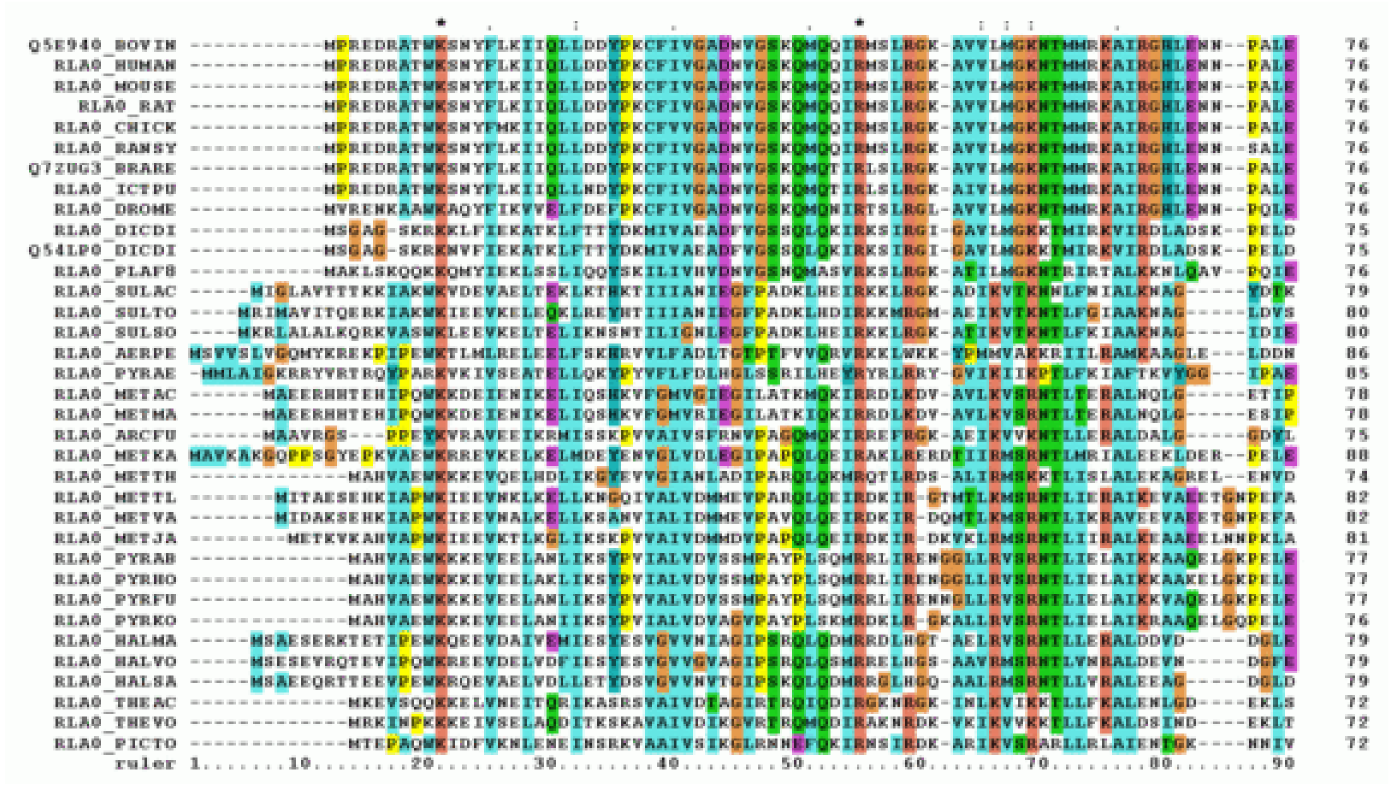
  - Transposition:



  - In general, we can't handle these efficiently: sub-problems don't overlap.
- But some special cases exist:
  - If reversals are 'contained' in each other, solve as 'context-free grammar'.

# Multiple Sequence Alignment

- Multiple Sequence Alignment:
  - We have several sequences and want to jointly align them:

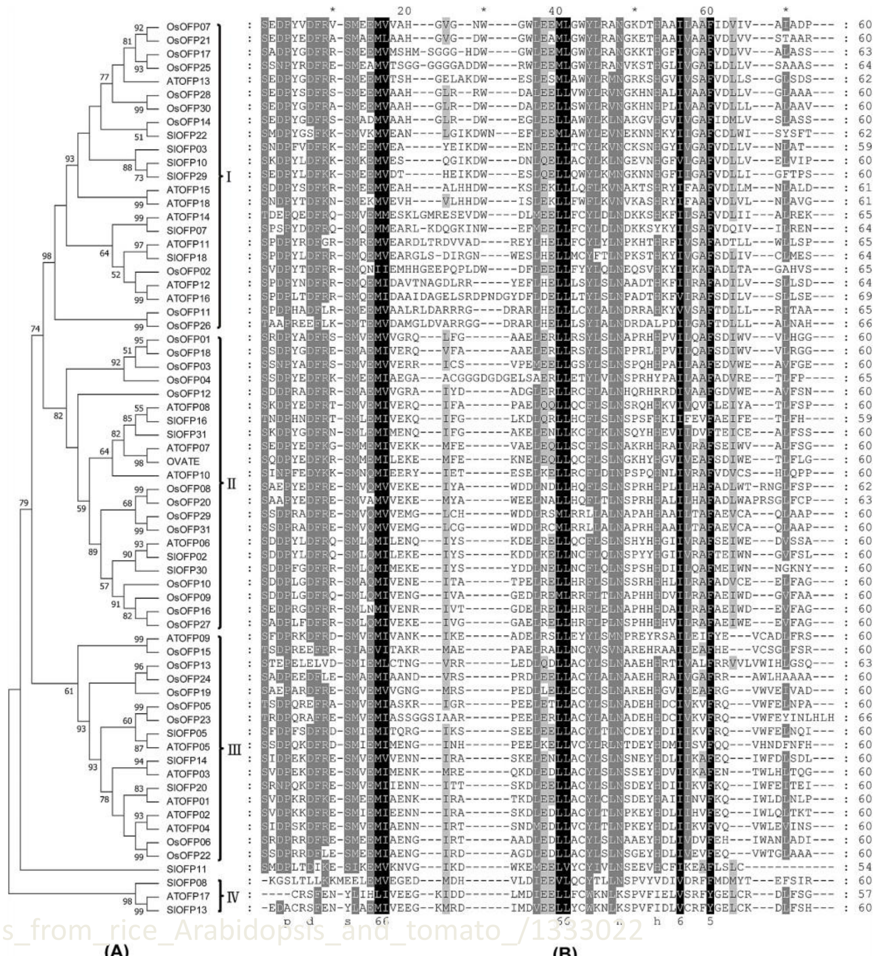


```
                                 *   .      :       .          .   *         : : :      .
Q5E940_BOVIN -----------MPREDRATWKSNYFLKIIQLLDDYPKCFIVGADNVGSKQMQQIRMSLRGK-AVVLMGKNTMMRKAIRGHLENN--PALE    76
  RLA0_HUMAN -----------MPREDRATWKSNYFLKIIQLLDDYPKCFIVGADNVGSKQMQQIRMSLRGK-AVVLMGKNTMMRKAIRGHLENN--PALE    76
  RLA0_MOUSE -----------MPREDRATWKSNYFLKIIQLLDDYPKCFIVGADNVGSKQMQQIRMSLRGK-AVVLMGKNTMMRKAIRGHLENN--PALE    76
    RLA0_RAT -----------MPREDRATWKSNYFLKIIQLLDDYPKCFIVGADNVGSKQMQQIRMSLRGK-AVVLMGKNTMMRKAIRGHLENN--PALE    76
  RLA0_CHICK -----------MPREDRATWKSNYFMKIIQLLDDYPKCFVVGADNVGSKQMQQIRMSLRGK-AVVLMGKNTMMRKAIRGHLENN--PALE    76
  RLA0_RANSY -----------MPREDRATWKSNYFLKIIQLLDDYPKCFIVGADNVGSKQMQQIRMSLRGK-AVVLMGKNTMMRKAIRGHLENN--SALE    76
Q7ZUG3_BRARE -----------MPREDRATWKSNYFLKIIQLLDDYPKCFIVGADNVGSKQMQTIRLSLRGK-AVVLMGKNTMMRKAIRGHLENN--PALE    76
  RLA0_ICTPU -----------MPREDRATWKSNYFLKIIQLLNDYPKCFIVGADNVGSKQMQTIRLSLRGK-AIVLMGKNTMMRKAIRGHLENN--PALE    76
  RLA0_DROME -----------MVRENKAAWKAQYFIKVVELFDEFPKCFIVGADNVGSKQMQNIRTSLRGL-AVVLMGKNTMMRKAIRGHLENN--PQLE    76
  RLA0_DICDI -----------MSGAG-SKRKKLFIEKATKLFTTYDKMIVAEADFVGSSQLQKIRKSIRGI-GAVLMGKKTMIRKVIRDLADSK--PELD    75
Q54LP0_DICDI -----------MSGAG-SKRKNVFIEKATKLFTTYDKMIVAEADFVGSSQLQKIRKSIRGI-GAVLMGKKTMIRKVIRDLADSK--PELD    75
  RLA0_PLAF8 -----------MAKLSKQQKKQMYIEKLSSLIQQYSKILIVHVDNVGSNQMASVRKSLRGK-ATILMGKNTRIRTALKKNLQAV--PQIE    76
  RLA0_SULAC -----MIGLAVTTTKKIAKWKVDEVAELTEKLKTHKTIIIANIEGFPADKLHEIRKKLRGK-ADIKVTKNNLFNIALKNAG-----YDTK    79
  RLA0_SULTO ----MRIMAVITQERKIAKWKIEEVKELEQKLREYHTIIIANIEGFPADKLHDIRKKMRGM-AEIKVTKNTLFGIAAKNAG-----LDVS    80
  RLA0_SULSO ----MKRLALALKQRKVASWKLEEVKELTELIKNSNTILIGNLEGFPADKLHEIRKKLRGK-ATIKVTKNTLFKIAAKNAG-----IDIE    80
  RLA0_AERPE MSVVSLVGQMYKREKPIPEWKTLMLRELEELFSKHRVVLFADLTGTPTFVVQRVRKKLWKK-YPMMVAKKRIILRAMKAAGLE---LDDN    86
  RLA0_PYRAE -MMLAIGKRRYVRTRQYPARKVKIVSEATELLQKYPYVFLFDLHGLSSRILHEYRYRLRRY-GVIKIIKPTLFKIAFTKVYGG---IPAE    85
  RLA0_METAC ------MAEERHHTEHIPQWKKDEIENIKELIQSHKVFGMVGIEGILATKMQKIRRDLKDV-AVLKVSRNTLTERALNQLG-----ETIP    78
  RLA0_METMA ------MAEERHHTEHIPQWKKDEIENIKELIQSHKVFGMVRIEGILATKIQKIRRDLKDV-AVLKVSRNTLTERALNQLG-----ESIP    78
  RLA0_ARCFU ------MAAVRGS---PPEYKVRAVEEIKRMISSKPVVAIVSFRNVPAGQMQKIRREFRGK-AEIKVVKNTLLERALDALG-----GDYL    75
  RLA0_METKA MAVKAKGQPPSGYEPKVAEWKRREVKELKELMDEYENVGLVDLEGIPAPQLQEIRAKLRERDTIIRMSRNTLMRIALEEKLDER--PELE    88
  RLA0_METTH -------------MAHVAEWKKKEVQELHDLIKGYEVVGIANLADIPARQLQKMRQTLRDS-ALIRMSKKTLISLALEKAGREL--ENVD    74
  RLA0_METTL -------MITAESEHKIAPWKIEEVNKLKELLKNGQIVALVDMMEVPARQLQEIRDKIR-GTMTLKMSRNTLIERAIKEVAEETGNPEFA    82
  RLA0_METVA -------MIDAKSEHKIAPWKIEEVNALKELLKSANVIALIDMMEVPAVQLQEIRDKIR-DQMTLKMSRNTLIKRAVEEVAEETGNPEFA    82
  RLA0_METJA --------METKVKAHVAPWKIEEVKTLKGLIKSKPVVAIVDMMDVPAPQLQEIRDKIR-DKVKLRMSRNTLIIRALKEAAEELNNPKLA    81
  RLA0_PYRAB -------------MAHVAEWKKKEVEELANLIKSYPVIALVDVSSMPAYPLSQMRRLIRENGGLLRVSRNTLIELAIKKAAQELGKPELE    77
  RLA0_PYRHO -------------MAHVAEWKKKEVEELAKLIKSYPVIALVDVSSMPAYPLSQMRRLIRENGGLLRVSRNTLIELAIKKAAKELGKPELE    77
  RLA0_PYRFU -------------MAHVAEWKKKEVEELANLIKSYPVVALVDVSSMPAYPLSQMRRLIRENNGLLRVSRNTLIELAIKKVAQELGKPELE    77
  RLA0_PYRKO -------------MAHVAEWKKKEVEELANIIKSYPVIALVDVAGVPAYPLSKMRDKLR-GKALLRVSRNTLIELAIKRAAQELGQPELE    76
  RLA0_HALMA -----MSAESERKTETIPEWKQEEVDAIVEMIESYESVGVVNIAGIPSRQLQDMRRDLHGT-AELRVSRNTLLERALDDVD-----DGLE    79
  RLA0_HALVO -----MSESEVRQTEVIPQWKREEVDELVDFIESYESVGVVGVAGIPSRQLQSMRRELHGS-AAVRMSRNTLVNRALDEVN-----DGFE    79
  RLA0_HALSA -----MSAEEQRTTEEVPEWKRQEVAELVDLLETYDSVGVVNVTGIPSKQLQDMRRGLHGQ-AALRMSRNTLLVRALEEAG-----DGLD    79
  RLA0_THEAC -------------MKEVSQQKKELVNEITQRIKASRSVAIVDTAGIRTRQIQDIRGKNRGK-INLKVIKKTLLFKALENLGD----EKLS    72
  RLA0_THEVO -------------MRKINPKKKEIVSELAQDITKSKAVAIVDIKGVRTRQMQDIRAKNRDK-VKIKVVKKTLLFKALDSIND----EKLT    72
  RLA0_PICTO -------------MTEPAQWKIDFVKNLENEINSRKVAAIVSIKGLRNNEFQKIRNSIRDK-ARIKVSRARLLRLAIENTGK----NNIV    72
       ruler 1.......10........20........30........40........50........60........70........80........90
```

  - Dynamic programming is exponential in number of sequences.

# Multiple Sequence Alignment and Clustering

- Heuristic to avoid exponential cost of multiple sequence alignment:
  - First perform hierarchical clustering.
    - Clustering coud be interesting on its own.
  - Align sequences as we go up the tree.
- Popular method is Clustal:
  - 3 of top 15 all-time most-cited science papers:
    - BLAST, PSI-BLAST, Clustal.



http://figshare.com/articles/_Multiple_sequence_alignment_and_phylogenetic_tree_of_the_OVATE_domains_of_OFP_proteins_from_rice_Arabidopsis_and_tomato_/1333022

# Summary

- Sequence data arises in applications involving time/strings.
  - Common substrings can be found in linear time.
  - Edit distance can be found efficiently using dynamic programming.
  - BLAST combines the above two with other tricks.
- Multiple sequence alignment considers multiple sequences.