String Matching – Applicable Algorithmics

Esko Ukkonen
University of Helsinki
Esko.Ukkonen@Helsinki.Fi

FDPW 2004 June 8 Petrozavodsk

Where is string matching?

- 'Every question in theoretical computer science can be formulated as a problem on strings' *-Zvi Galil*
- 'Biology has challenging problems for computer science for one hundred years' *–Donald Knuth*

Theoretical Computer Science ⟺ **Combinatorial algorithms on strings** ⟺

- Bioinformatics
- Natural language processing
- Information retrieval
- Data compression
- …

Basic question

Text: **KALEVALANKALAVALE**

Pattern: **LANKA**

```
KALEVALANKALAVALE
LANKA
 LANKA
  LANKA
   LANKA
    LANKA
     LANKA
      LANKA
       LANKA
        LANKA
         LANKA
          LANKA
           LANKA
            LANKA
```

AAAAAAAAAAAAAAAA..

AAAAAAB

=> O(mn)

Exact string matching

- pattern $P = p_1...p_m$ and text $T = t_1...t_n$ in Σ^*
- finite alphabet Σ
- $m << n$
- find the occurrences of P in T:
  Knuth-Morris-Pratt O(m+n),
  multipattern matching Aho-Corasick O(m+n),
  Boyer-Moore O(n/m), …
- more general patterns: gaps, don't-care-symbols, regular expressions, …

Text indexing

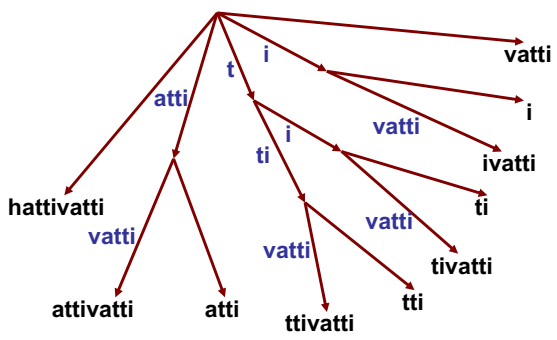
- assume: text T stays the same (human genome,…)
- need to answer queries for several different P
- build an *index* for T that makes the search for different P fast: get rid of O(|T|)
- suffix tree, suffix array, suffix automaton

Suffix tree

- cleverly implemented TRIE that represents all the suffixes of T
- T = hattivatti

hattivatti
attivatti
ttivatti
tivatti
ivatti
vatti
atti
tti
ti
i

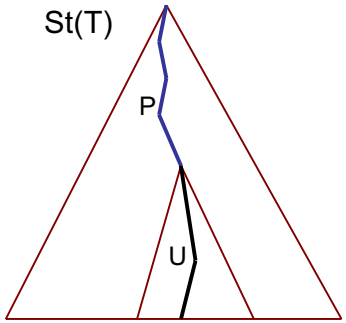
hattivatti
attivatti
ttivatti
tivatti
ivatti
vatti
atti
tti
ti
i



The suffix tree St(T) of T

- tree St(T) represents all suffixes of T
- the size of St(T) is O(|T|)
- can be constructed in linear time O(|T|) (Weiner 1974, …)
- 'myriads' of nice properties and applications:
  search time O(|P| + #occurrences) (optimal!)
- *'The best data structure in the world'*

Search of P in St(T)



Path for P exists ⇔ P is a prefix of some suffix of T
⇔ P occurs in T

Approximate string matching: string distance/similarity functions

STOCKHOLM vs TUKHOLMA

STOCKHOLM_
_TU_ KHOLMA

=> 2 deletions, 1 insertion, 1 change

PETROSKOI vs PETROZAVODSK

PETRO_____SKOI
PETROZAVODSK__

=> 5 insertions, 2 deletions

Approximate string matching

- A: S T O C K H O L M

  B: T U K H O L M A
- minimum number of 'mutation' steps
- a -> b a -> ϵ ϵ -> b …
- $d_{ID}(A,B) = 5$ $d_{Levenshtein}(A,B) = 4$
- unit cost vs general cost (=> probabilistic modeling)
- evaluation by dynamic programming

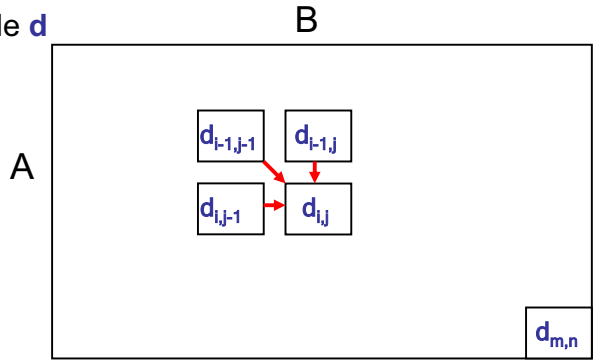
Distance evaluation problem

- strings A and B, |A| = m, |B| = n, m≤n
- evaluate distance d(A,B)
- dynamic programming O(mn); also O(mn/logm) (Crochemore at al SODA2002)
- string alignment

Dynamic programming

$d_{i,j}$ = min(if $a_i=b_j$ then $d_{i-1,j-1}$ else ∞,
$d_{i-1,j} + 1$,
$d_{i,j-1} + 1$)
= distance between i-prefix of A and j-prefix of B

mxn table **d**



$d_{i,j}$ = min(if $a_i=b_j$ then $d_{i-1,j-1}$ else ∞,
$d_{i-1,j} + 1$,
$d_{i,j-1} + 1$)

A\B		s	t	o	c	k	h	o	l	m
	0	1	2	3	4	5	6	7	8	9
t	1	2	1	2	3	4	5	6	7	8
u	2	3	2	3	4	5	6	7	8	9
k	3	4	3	4	5	4	5	6	7	8
h	4	5	4	5	6	5	4	5	6	7
o	5	6	5	4	5	6	5	4	5	6
l	6	7	6	5	6	7	6	5	4	5
m	7	8	7	6	7	8	7	6	5	4
a	8	9	8	7	8	9	8	7	6	5

optimal alignment by trace-back

$d_{ID}(A,B)$

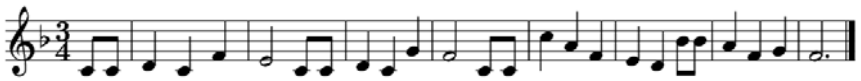
Search problem

- find approximate occurrences of P in T: substrings P' of T such that d(P,P') small
- dyn progr with a small modification: O(mn)
- lots of (practical) improvements:
  distance bound k => O(kn) search;
  utilizing regularities of the dp table;
  filtration approach;
  bit-parallelism (Gonnet & Baeza-Yates)

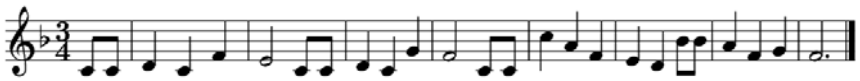
Summary: Good for something?

- deep beauty? +-
- shallow beauty? +
- applications? ++
- intensive algorithmic miniatures
- techniques: finite automata, 'the best data structure', dynamic programming, filtration techniques, bit-parallelism, …
- sources of new problems:
  text processing, DNA, music,…

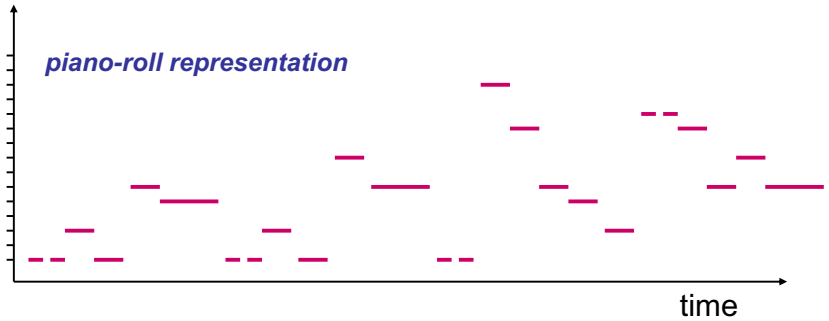
Applications 1: Music retrieval

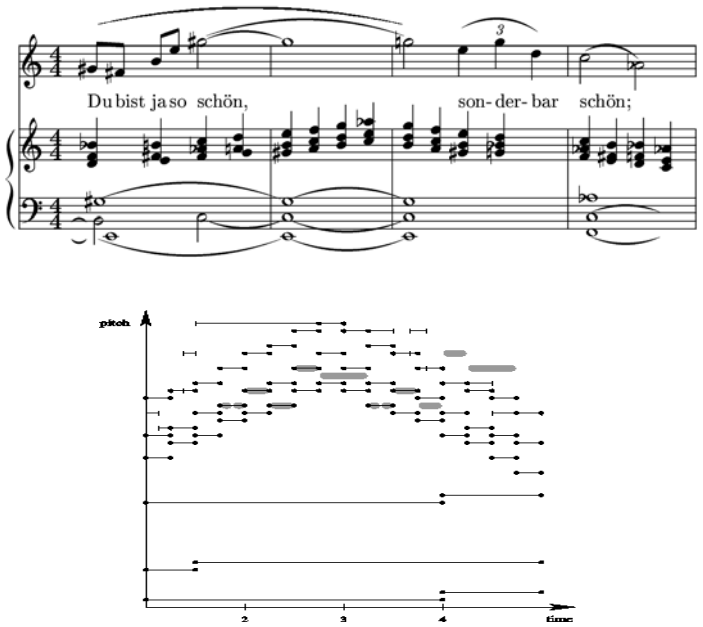


C C D C F E C C D C G F C C C^1 A F E D B B A F G F



C C D C F E C C D C G F C C C^1 A F E D B B A F G F





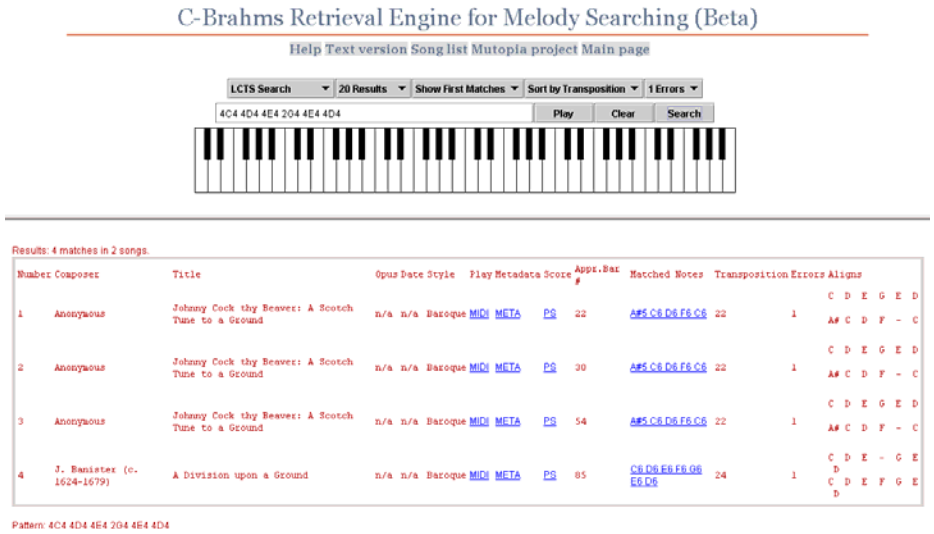
Music retrieval and analysis

- symbolically (using notes) encoded music:
  pitch levels < 300 (typically less)
  durations < 1000 (typically less)
- sequence of discrete symbols, interpretation in 2D
- not necessarily evenly spaced as traditional strings (different durations)
- many symbols can be 'on' simultaneously

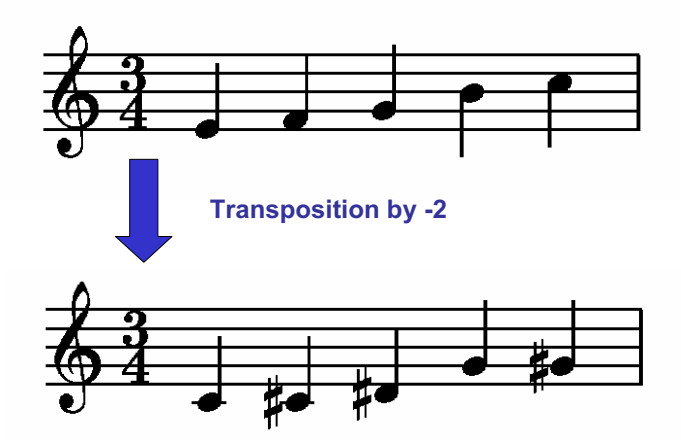
Music…

- seminal application: **query-by-humming**
  inaccurate pitches and durations
  => approximate melody matching
  (Ghias et al 1995; MELDEX 1997)
- Mongeau & Sankoff (1990)
- music information retrieval (MIR)
- monophonic vs polyphonic music
- music analysis: comparative analysis, evolution of musical ideas, characterization of the style of composers, …

http://www.cs.helsinki.fi/group/cbrahms/demoengine/



Transposition invariance



Transposition invariance…

- transposition (systematic shift) of pitch levels keeps the music the same
- the intervals between the notes do not change

Transposition invariant distances

- alphabet Σ: closed with respect to addition and subtraction
- say, Σ = integers
- $A = a_1 \ldots a_m$ translated by t:

  $A+t = a_1+t, \ldots, a_m+t$
- *transposition invariant distance*

  $d^T(A,B) = \min_{t \in \Sigma} d(A+t,B)$
- 'longest common hidden melody'

Exact case is trivial

- interval: $a_{i+1} - a_i$
- intervals(A+t) = intervals(A)
- use interval sequences

  $a_2-a_1, a_3-a_2, \ldots, a_m-a_{m-1}$

  instead of originals in exact matching
- transposition invariance for free

Approximate case

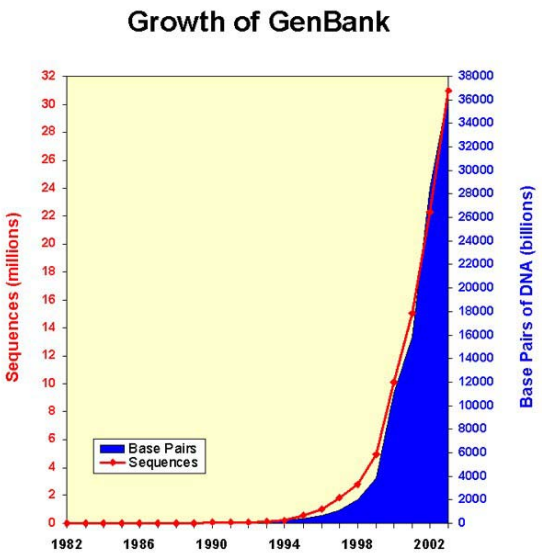
- repeat dp for all O(mn) relevant transpositions: $O(m^2n^2)$
- sparse dynamic programming (Epstein at al 1992) => transposition invariant distance & search O(mn loglog n) (Veli Mäkinen's PhD thesis)

Applications 2: Bioinformatics/computational biology

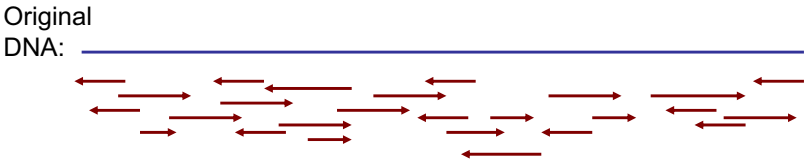
- DNA sequencing
- multiple alignment and motif discovery
- lots of more …

Genome projects

```
cgccgagtgacagagacgctaatcaggc
tgtgttctcaggatgcgtaccgagtggg
agacagcagcacgaccagcggtggcaga
gacccttgcagacatcaagctctttggg
aacaagtggagcaccgatgatgtacagc
cgatcaatgacatttccctaatgcagga
ttacattgcagtgcccaaggagaagtat
gccaagtaatacctccctcacagtg...
```



Shotgun sequencing



'Gel readings': random location and orientation

The fragment assembly problem

```
cctcgagttaagtactgcccgcggcttcaacggatctgtcgggagtcg
```

reconstruction?

```
cctcgagtttaa
tacttaactcgag
cgggcagtacttaa
aagtactgcccgcg
gcccgcggcttcaacggat
cccgcggcttcaacggatctgtg
cccgacacagat
tgtgtcgggagtcg
```

```
cctcgagtttaa
tacttaactcgag
cgggcagtacttaa
aagtactgcccgcg
gcccgcggcttcaacggat
cccgcggcttcaacggatctgtg
cccgacacagat
tgtgtcgggagtcg
```

```
cctcgagtttaa
 ctcgagt-taagta
       t-taagtactgcccg
          aagtactgcccgcg
                 gcccgcggcttcaacggat
                  cccgcggcttcaacggatctgtg
                                  atctgtgtcggg
                                      tgtgtcgggagtcg
```

```
cctcgagt-taagtactgcccgcggcttcaacggatctgtgtcgggagtcg
```

Fragment assembly (cont.)

- T = DNA sequence (say, the genome of a human), a string over alphabet A, C, G, T
- fragments F1, …, Fm: pieces of T of length 150 – 800, taken from randon locations and in random orientation, some errors can occur
- construct T from its fragments F1, … , Fm.

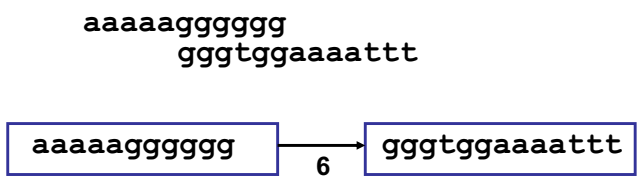
Some numbers (Human Genome)

- length about 3 000 000 000 bp
- Celera's fragment data (Feb 2001): 27.27 million fragments, each 150-800 bp
- at least 7-fold coverage by fragments needed => total data length 7 x 3 billion bp

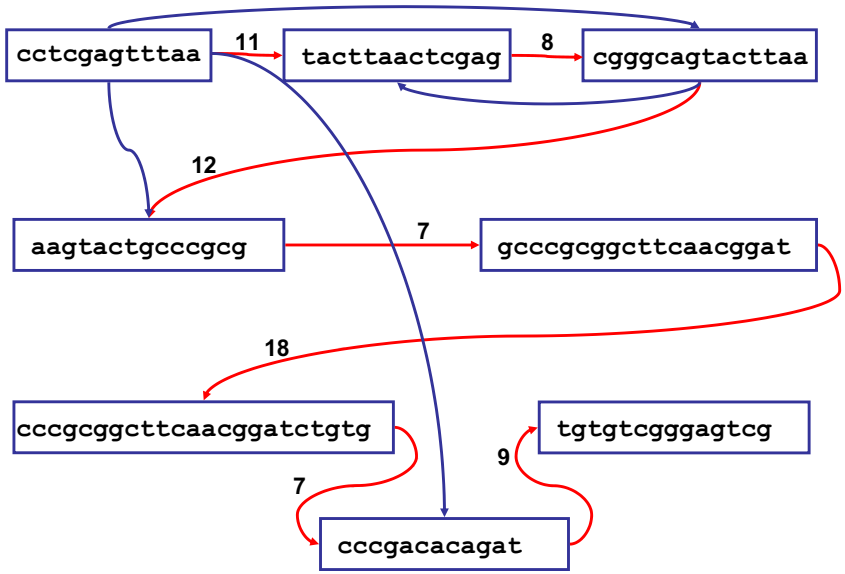
Overlap graph - Layout - Consensus

- 1. For each pair (Fi,Fj) of fragments, find all acceptable overlaps between Fi and Fj and their reverse complements. This gives an overlap graph.
- 2. Using the overlap graph, build a lay-out that gives global alignment of the fragments.
- 3. Construct a consensus string from the lay-out. The consensus is the resulting approximation of the original DNA sequence T

Overlap graph



Finding the overlaps: dynamic programming



Hamiltonian paths ⇔ Lay-outs

The red path gives our example lay-out:

```
cctcgagtttaa
 ctcgagt-taagta
       t-taagtactgcccg
          aagtactgcccgcg
                 gcccgcggcttcaacggat
                  cccgcggcttcaacggatctgtg
                                 atctgtgtcggg
                                    tgtgtcgggagtcg
```

Consensus from lay-out

```
cctcgagtttaa
 ctcgagt-taagta
       t-taagtactgcccg
          aagtactgcccgcg
                 gcccgcggcttcaacggat
                  cccgcggcttcaacggatctgtg
                                 atctgtgtcggg
                                    tgtgtcgggagtcg

cctcgagt-taagtactgcccgcggcttcaacggatctgtgtcgggagtcg
```

Other problems

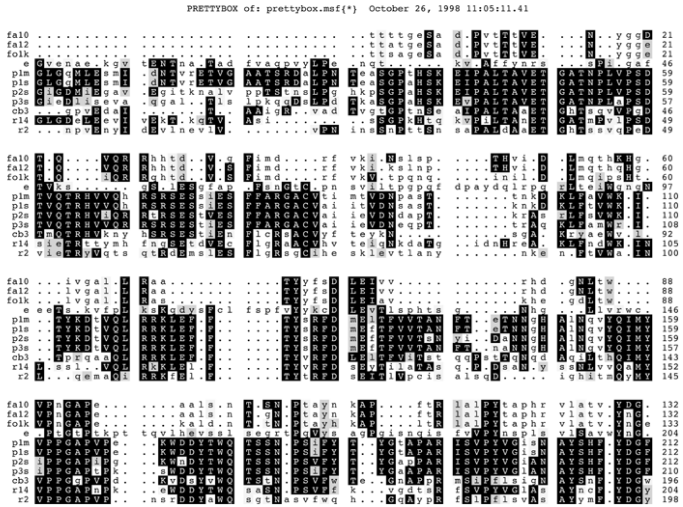
- finding homologous sequences: new sequence vs all old ones – the most popular computational task in present-day molecular biology = approximate string matching (filtration, also dynamic programming a la Smith-Waterman). BLAST
- multiple alignment of sequence families to find interesting motifs: NP-hard => heuristics, Hidden Markov models, MCMC

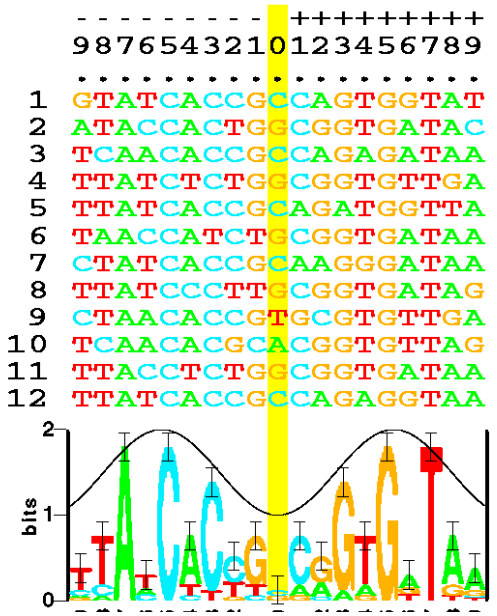
A multiple alignment

```
                         10        20        30        40        50        60
                          .         .         .         .         .         .
Hbb_Human.pep   --------VHLTPEEKSAVTALWGKVN--VDEVGGEALGRLLVVYPWTQRFFESFGDLST
Hbb_Horse.pep   --------VQLSGEEKAAVLALWDKVN--EEEVGGEALGRLLVVYPWTQRFFDSFGDLSN
Hba_Human.pep   ---------VLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHFDLS--
Hba_Horse.pep   ---------VLSAADKTNVKAAWSKVGGHAGEYGAEALERMFLGFPTTKTYFPHFDLS--
Myg_Phyca.pep   ---------VLSEGEWQLVLHVWAKVEADVAGHGQDILIRLFKSHPETLEKFDRFKHLKT
Glb5_Petma.pep  PIVDTGSVAPLSAAEKTKIRSAWAPVYSTYETSGVDILVKFFTSTPAAQEFFPKFKGLTT
Lgb2_Luplu.pep  --------GALTESQAALVKSSWEEFNANIPKHTHRFFILVLEIAPAAKDLFSFLKGTSE
                          *.      .   *                 * .   *

Hbb_Human.pep   PDAVMGNPKVKAHGKKVLGAFSDGLAHLD-----NLKGTFATLSELHCDKLHVDPENFRL
Hbb_Horse.pep   PGAVMGNPKVKAHGKKVLHSFGEGVHHLD-----NLKGTFAALSELHCDKLHVDPENFRL
Hba_Human.pep   ----HGSAQVKGHGKKVADALTNAVAHVD-----DMPNALSALSDLHAHKLRVDPVNFKL
Hba_Horse.pep   ----HGSAQVKAHGKKVGDALTLAVGHLD-----DLPGALSNLSDLHAHKLRVDPVNFKL
Myg_Phyca.pep   EAEMKASEDLKKHGVTVLTALGAILKKKG-----HHEAELKPLAQSHATKHKIPIKYLEF
Glb5_Petma.pep  ADQLKKSADVRWHAERIINAVNDAVASMDDT--EKMSMKLRDLSGKHAKSFQVDPQYFKV
Lgb2_Luplu.pep  VP--QNNPELQAHAGKVFKLVYEAAIQLQVTGVVVTDATLKNLGSVHVSKG-VADAHFPV
                          .. *   .                   *   *     .

Hbb_Human.pep   LGNVLVCVLAHHFGKEFTPPVQAAYQKVVAGVANALAHKYH------
Hbb_Horse.pep   LGNVLVVVLARHFGKDFTPELQASYQKVVAGVANALAHKYH------
Hba_Human.pep   LSHCLLVTLAAHLPAEFTPAVHASLDKFLASVSTVLTSKYR------
Hba_Horse.pep   LSHCLLSTLAVHLPNDFTPAVHASLDKFLSSVSTVLTSKYR------
Myg_Phyca.pep   ISEAIIHVLHSRHPGDFGADAQGAMNKALELFRKDIAAKYKELGYQG
Glb5_Petma.pep  LAAVIADTVAAG---------DAGFEKLMSMICILLRSAY-------
Lgb2_Luplu.pep  VKEAILKTIKEVVGAKWSEELNSAWTIAYDELAIVIKKEMNDAA---
                .   .   .                         .
```





Conclusion

- strings are everywhere: DNA is a string of discrete symbols, music …
- systems biology: cell as a system regulated by the DNA
- pattern discovery rather than pattern matching: statistical techniques, data mining, machine learning