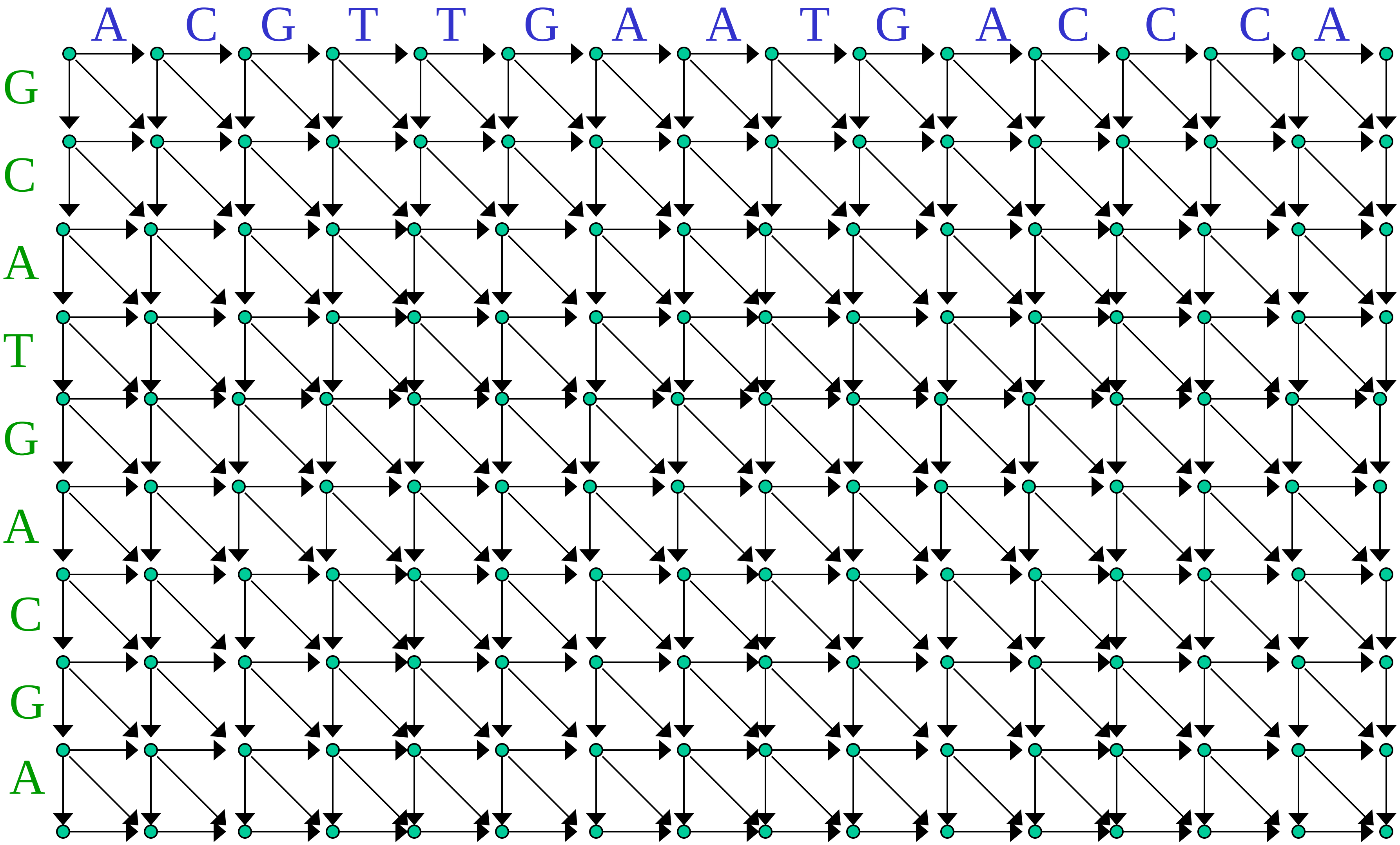
Today’s Lecture

- Edit graph & alignment algorithms
  - Smith-Waterman algorithm
  - Needleman-Wunsch algorithm
- Local vs global
- Computational complexity of pairwise alignment

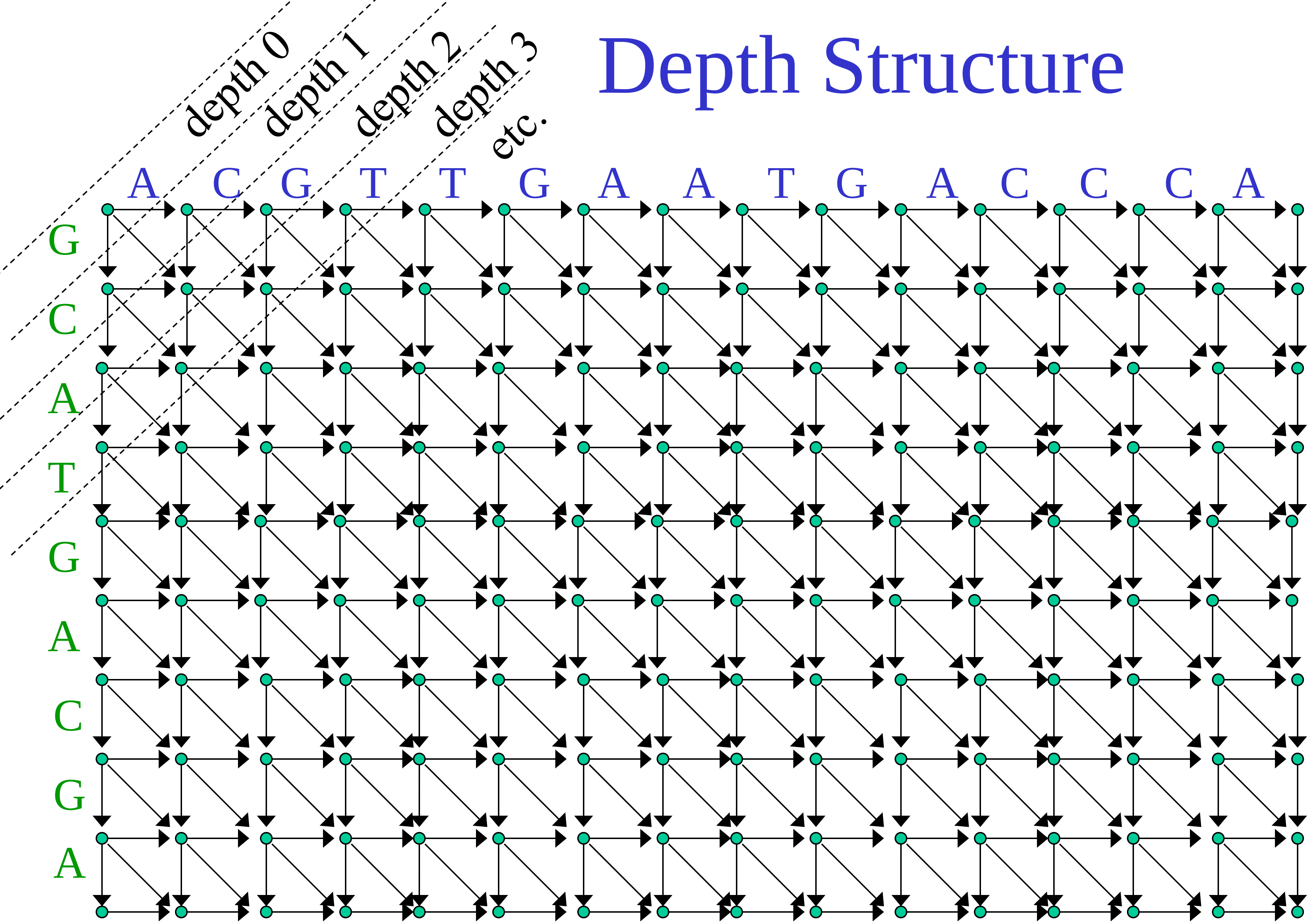
Sequence alignments correspond to *paths* in a *DAG*!

The *Edit Graph* for a Pair of Sequences

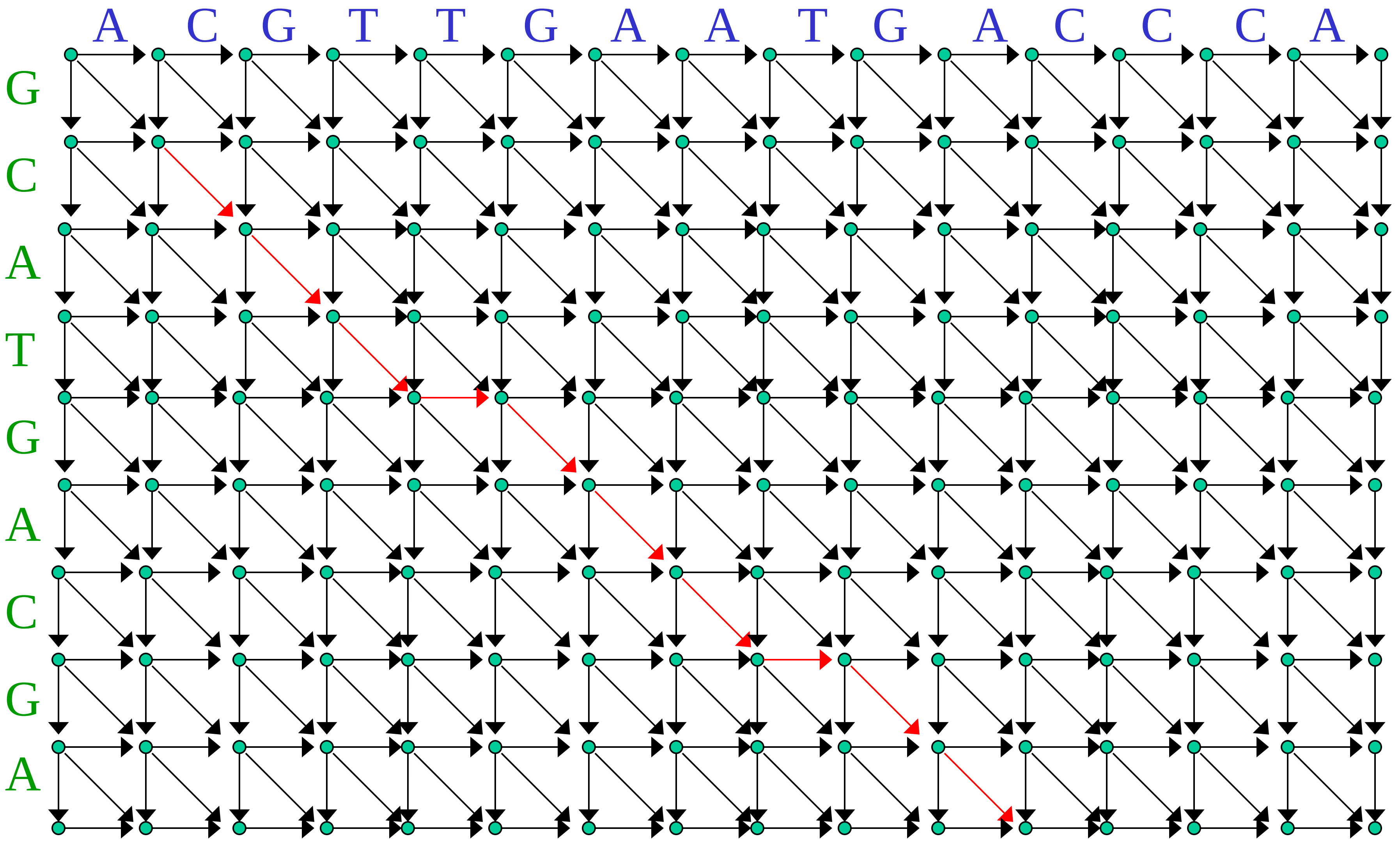


- The edit graph is a DAG.
  - Except on the boundaries, the nodes have in-degree and out-degree both 3.
- The depth structure is as shown on the next slide. Child of node of depth *n* always has
  - depth $n + 1$ (for a horizontal or vertical edge), or
  - depth $n + 2$ (for a diagonal edge).

Depth Structure



- *Paths* in edit graph correspond to *alignments* of subsequences
  - each edge on path corresponds to alignment column.
  - diagonal edges correspond to column of two aligned residues;
  - horizontal edges correspond to column with
    - residue in 1st (top, horizontal) sequence
    - gap in the 2d (vertical) sequence
  - vertical edges correspond to column with
    - residue in 2d sequence
    - gap in 1st sequence



Above path corresponds to following alignment (w/ lower case letters considered unaligned):

```
aCGTTGAATGAccca
gCAT-GAC-GA
```

Weights on Edit Graphs

- Edge weights correspond to scores on alignment columns.
- Highest weight path corresponds to highest-scoring alignment for that scoring system.
- Weights may be assigned using
  - a *substitution score matrix*,
    - assigns a score to each possible pair of residues occurring as alignment column

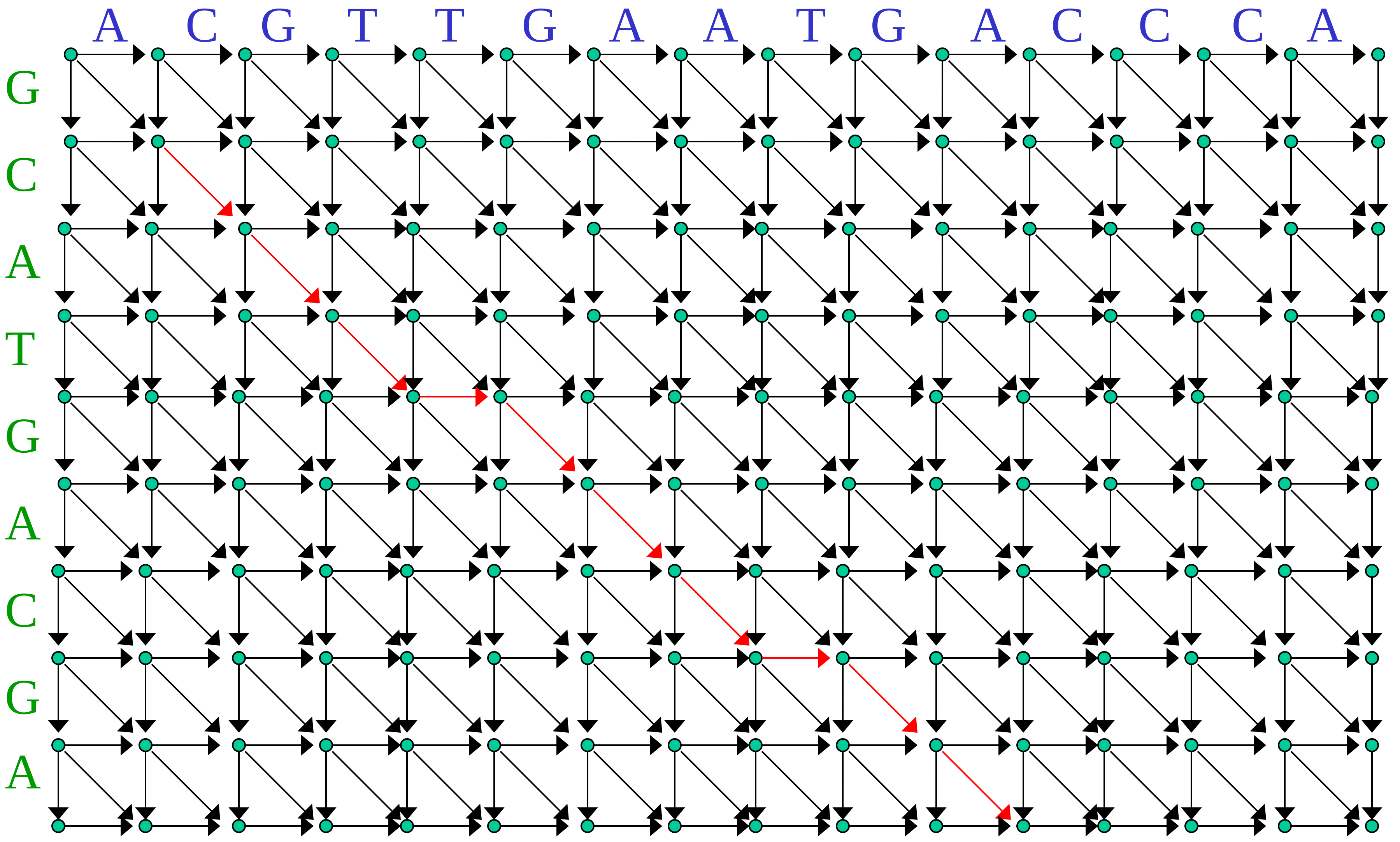
  and

  - a *gap penalty*
    - assigns a score to column consisting of residue opposite a gap.
  - Example for protein sequences: BLOSUM62

BLOSUM62 Score Matrix

GAP -12 -2

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	B	Z	X	*
A	4	-1	-2	-2	0	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-3	-2	0	-2	-1	0	-4
R	-1	5	0	-2	-3	1	0	-2	0	-3	-2	2	-1	-3	-2	-1	-1	-3	-2	-3	-1	0	-1	-4
N	-2	0	6	1	-3	0	0	0	1	-3	-3	0	-2	-3	-2	1	0	-4	-2	-3	3	0	-1	-4
D	-2	-2	1	6	-3	0	2	-1	-1	-3	-4	-1	-3	-3	-1	0	-1	-4	-3	-3	4	1	-1	-4
C	0	-3	-3	-3	9	-3	-4	-3	-3	-1	-1	-3	-1	-2	-3	-1	-1	-2	-2	-1	-3	-3	-2	-4
Q	-1	1	0	0	-3	5	2	-2	0	-3	-2	1	0	-3	-1	0	-1	-2	-1	-2	0	3	-1	-4
E	-1	0	0	2	-4	2	5	-2	0	-3	-3	1	-2	-3	-1	0	-1	-3	-2	-2	1	4	-1	-4
G	0	-2	0	-1	-3	-2	-2	6	-2	-4	-4	-2	-3	-3	-2	0	-2	-2	-3	-3	-1	-2	-1	-4
H	-2	0	1	-1	-3	0	0	-2	8	-3	-3	-1	-2	-1	-2	-1	-2	-2	2	-3	0	0	-1	-4
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4	2	-3	1	0	-3	-2	-1	-3	-1	3	-3	-3	-1	-4
L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4	-2	2	0	-3	-2	-1	-2	-1	1	-4	-3	-1	-4
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5	-1	-3	-1	0	-1	-3	-2	-2	0	1	-1	-4
M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5	0	-2	-1	-1	-1	-1	1	-3	-1	-1	-4
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6	-4	-2	-2	1	3	-1	-3	-3	-1	-4
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7	-1	-1	-4	-3	-2	-2	-1	-2	-4
S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4	1	-3	-2	-2	0	0	0	-4
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5	-2	-2	0	-1	-1	0	-4
W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11	2	-3	-4	-3	-2	-4
Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	-1	-3	-2	-1	-4
V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4	-3	-2	-1	-4
B	-2	-1	3	4	-3	0	1	-1	0	-3	-4	0	-3	-3	-2	0	-1	-4	-3	-3	4	1	-1	-4
Z	-1	0	0	1	-3	3	4	-2	0	-3	-3	1	-1	-3	-1	0	-1	-3	-2	-2	1	4	-1	-4
X	0	-1	-1	-1	-2	-1	-1	-1	-1	-1	-1	-1	-1	-1	-2	0	0	-2	-1	-1	-1	-1	-1	-4
*	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	1



Above path corresponds to following alignment (w/ lower case letters considered unaligned):

```
aCGTTGAATGAccca
gCAT-GAC-GA
```

Alignment algorithms

- *Smith-Waterman* algorithm to find highest scoring alignment

  = dynamic programming algorithm to find highest-weight path
  - Is a *local* alignment algorithm:
    - finds alignment of subsequences rather than the full sequences.
- Can process nodes in any order in which parents precede children. Commonly used alternatives are
  - depth order
  - row order
  - column order

- If constrain path to
  - start at upper-left corner node and
  - extend to lower-right corner node,

  get a *global* alignment instead
- This sometimes called *Needleman-Wunsch algorithm*
  - (altho original N-W alg treated gaps differently)
- ∃ variants which constrain path to
  - start on the left or top boundary,
  - extend to the right or bottom boundary.

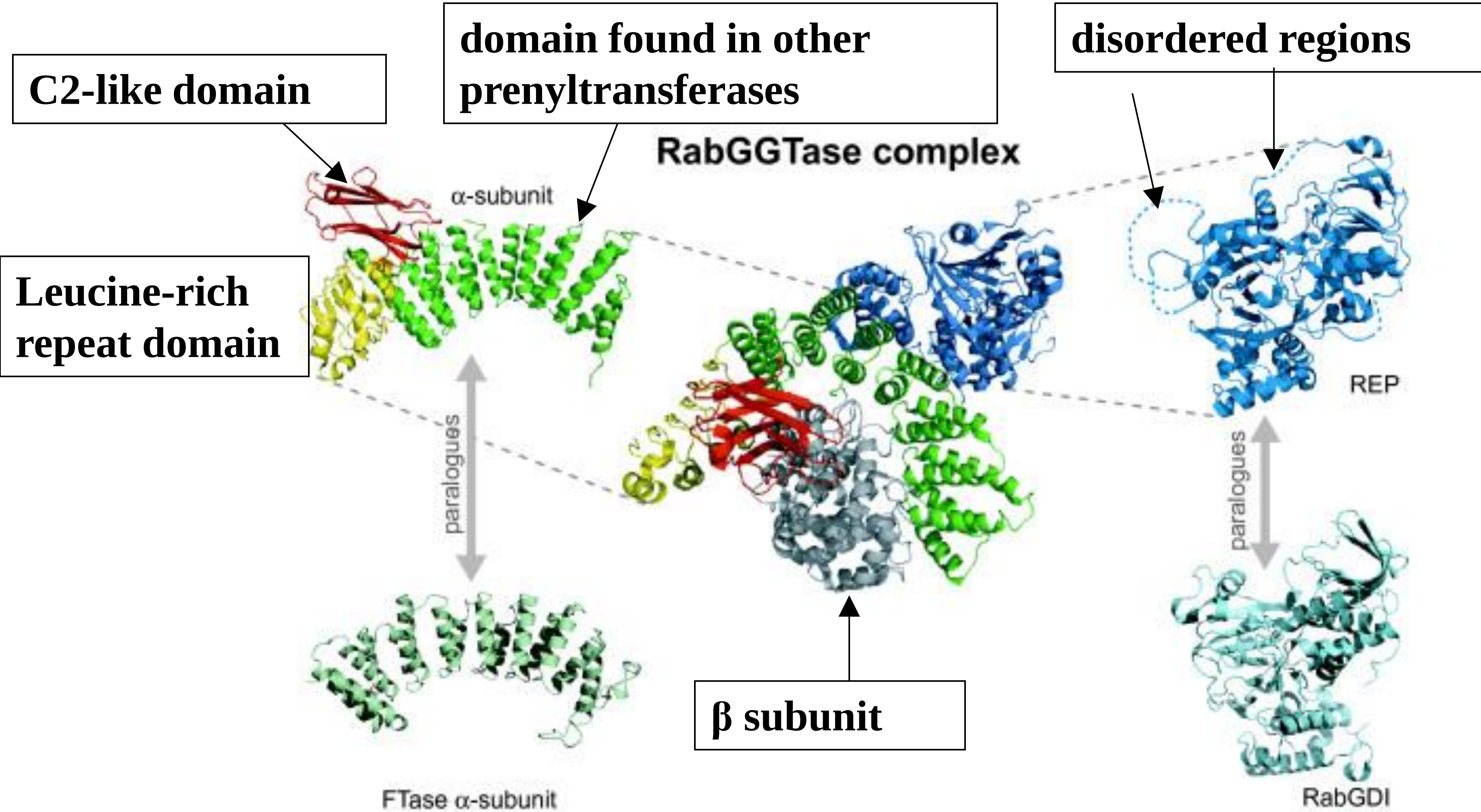
Local vs. Global Alignments: Biological Considerations

- Many proteins consist of multiple ‘domains’ (modules), some of which may be present
  - with similar, but not identical sequence

  in many other proteins
  - e.g. ATP binding domains, DNA binding domains, protein-protein interaction domains ...

  Need *local alignment* to detect presence of similar regions in otherwise dissimilar proteins.
- Other proteins consist of single domain evolving as a unit
  - e.g. many enzymes, globins.

  Global alignment sometimes best in such cases
  - ... but even here, some regions are more highly conserved (more slowly evolving) than others, and most sensitive similarity detection may be local alignment.

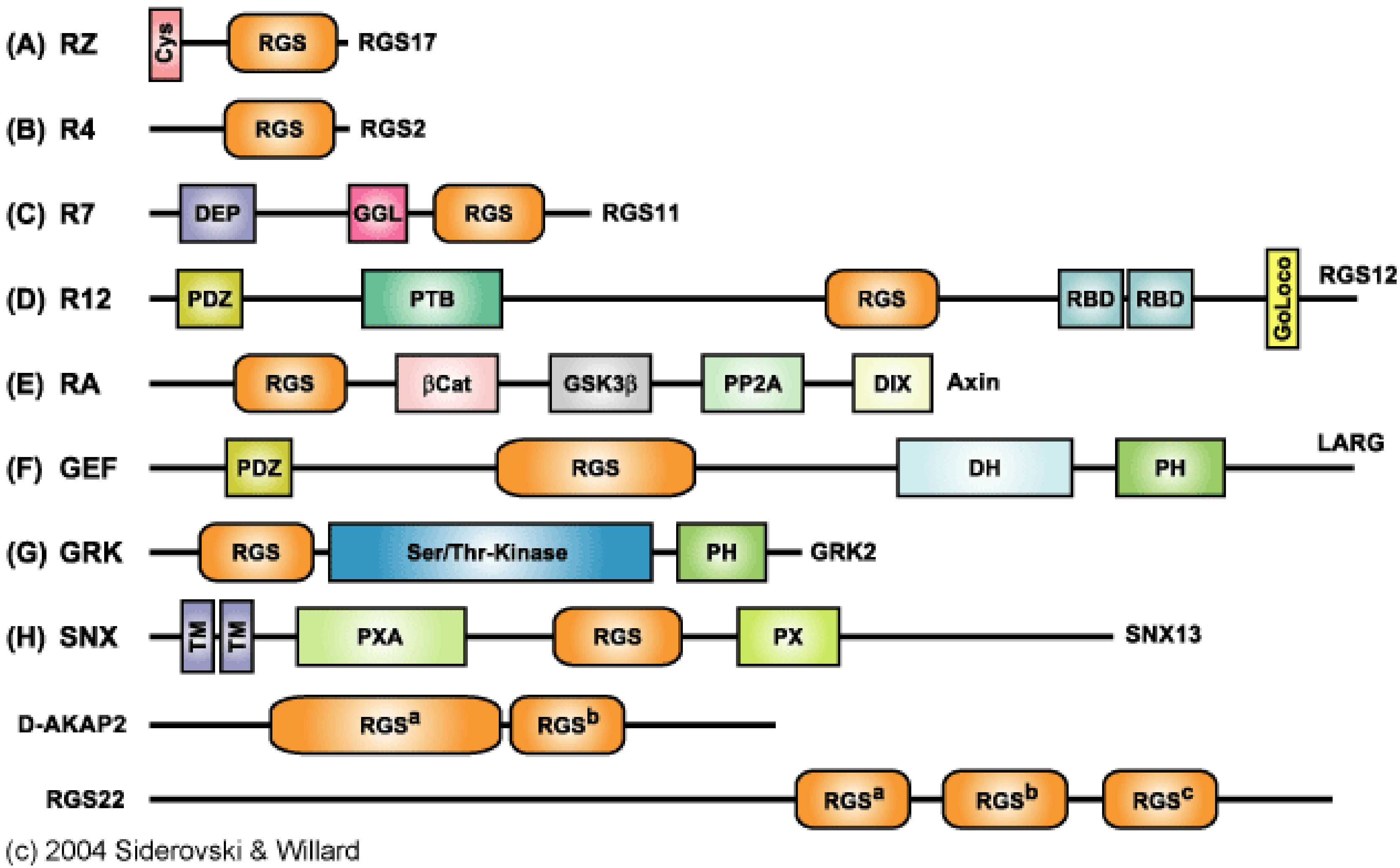


3-D structures of rat Rab Geranylgeranyl Transferase complexed with REP-1, + paralogs.

***adapted from* Rasteiro and Pereira-Leal *BMC Evolutionary Biology* 2007 7:140**

Multidomain architecture of representative members from all subfamilies of the mammalian RGS protein superfamily.

from www.unc.edu/~dsiderov/page2.htm



Similar considerations apply to aligning DNA sequences:

- (semi-)global alignment may be preferred for aligning
  - cDNA to genome
  - recently diverged genomic sequences (e.g. human / chimp)

  but local alignment often gives same result!
- between more highly diverged sequences, have
  - rearrangements (or large indels) in one sequence vs the other,
  - variable distribution of sequence conservation,

  & these usually make local alignments preferable.

Complexity

- For two sequences of lengths M and N, edit graph has
  - $(M+1)(N+1)$ nodes,
  - $3MN+M+N$ edges,
- time complexity: $O(MN)$
- space complexity to find

    highest score and beginning & end of alignment

  is $O(\min(M,N))$

    (since only need store node's values until children processed)
- space complexity to reconstruct highest-scoring alignment: $O(MN)$

- For genomic comparisons may have
  - M, $N \approx 10^6$ (if comparing two large genomic segments), or
  - $M \approx 10^3$, $N \approx 10^9$ (if searching gene sequence against entire genome);

  in either case $MN \approx 10^{12}$.
- Time complexity 10^{12} is (marginally) acceptable.
- ∃ speedups which reduce constant by
  - reducing calculations per matrix cell, using fact that score often 0
    - (our program *swat*).
    - still guaranteed to find highest-scoring alignment.
  - reducing # cells considered, using nucleating word matches
    - (*BLAST*, or *cross_match*).
    - Lose guarantee to find highest-scoring alignment.