Multiple Sequence Alignment (MSA)

Outline

- Multiple sequence alignment (MSA)
- Generalize DP to 3 sequence alignment
  - Impractical
- Heuristic approaches to MSA
  - Progressive alignment – ClustalW (using substitution matrix based scoring function)
  - Consistency-based approach – T-Coffee (consistency-based scoring function)
  - MUSCLE (MUSCLE-fast, MUSCLE-prog): reduces time and space complexity

From pairwise to multiple alignment

- Alignment of 2 sequences is represented as a 2-row matrix
- In a similar way, we represent alignment of 3 sequences as a 3-row matrix

```
A T _ G C G _
A _ C G T _ A
A T C A C _ A
```

- Score: more conserved columns, better alignment

What's multiple sequence alignment (MSA)

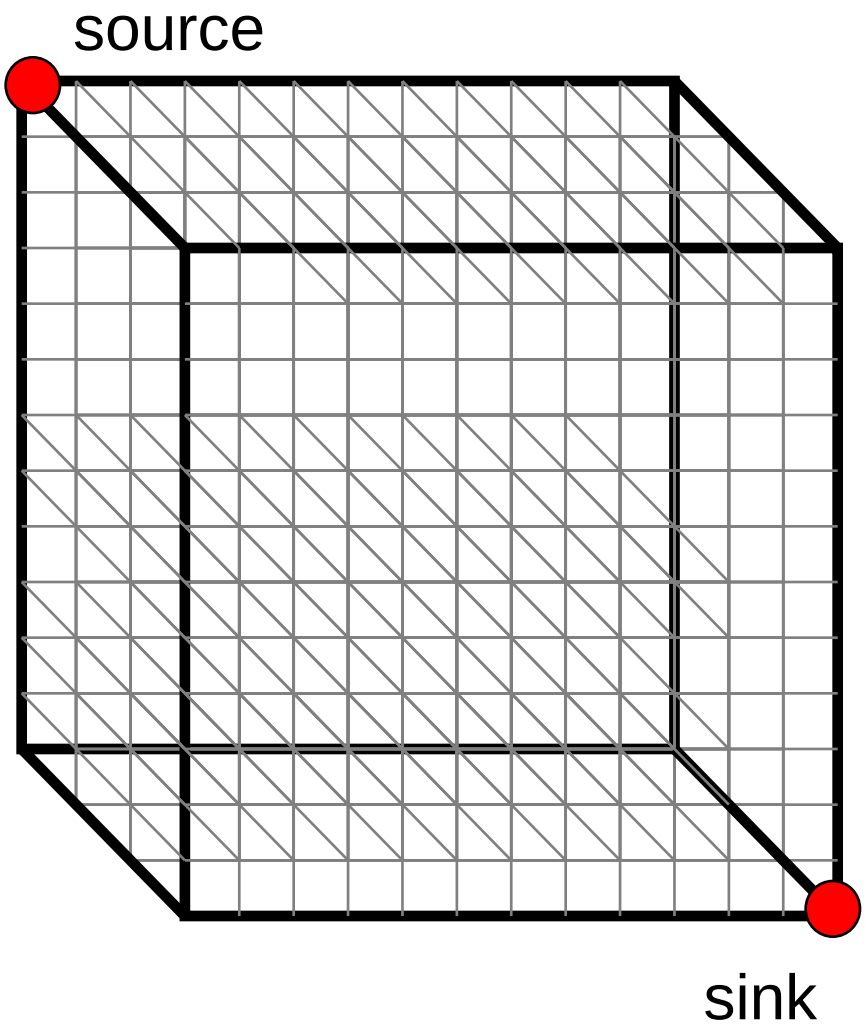
- A model
- Indicates relationship between residues of different sequences
- Reveals similarity/disimilarity

Why we need MSA

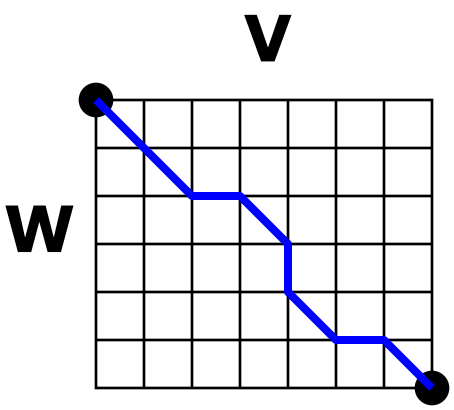
- MSA is central to many bioinformatics applications
- Phylogenetic tree
- Motifs
- Patterns
- Structure prediction (RNA, protein)

Aligning three sequences

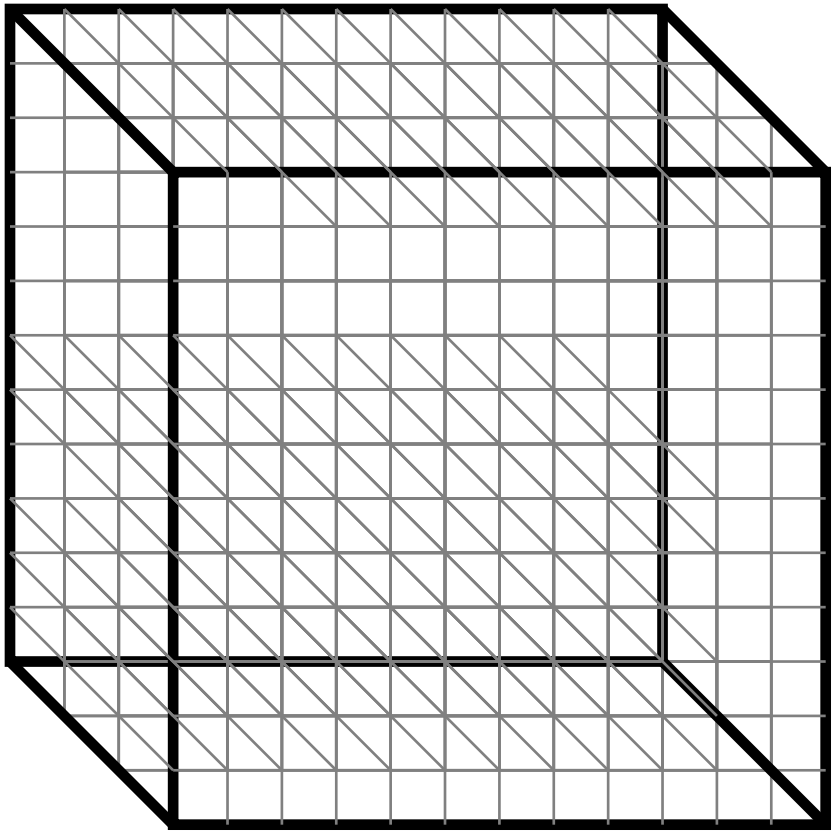
- Same strategy as aligning two sequences
- Use a 3-D “Manhattan Cube”, with each axis representing a sequence to align
- For global alignments, go from source to sink



2D vs 3D alignment grid

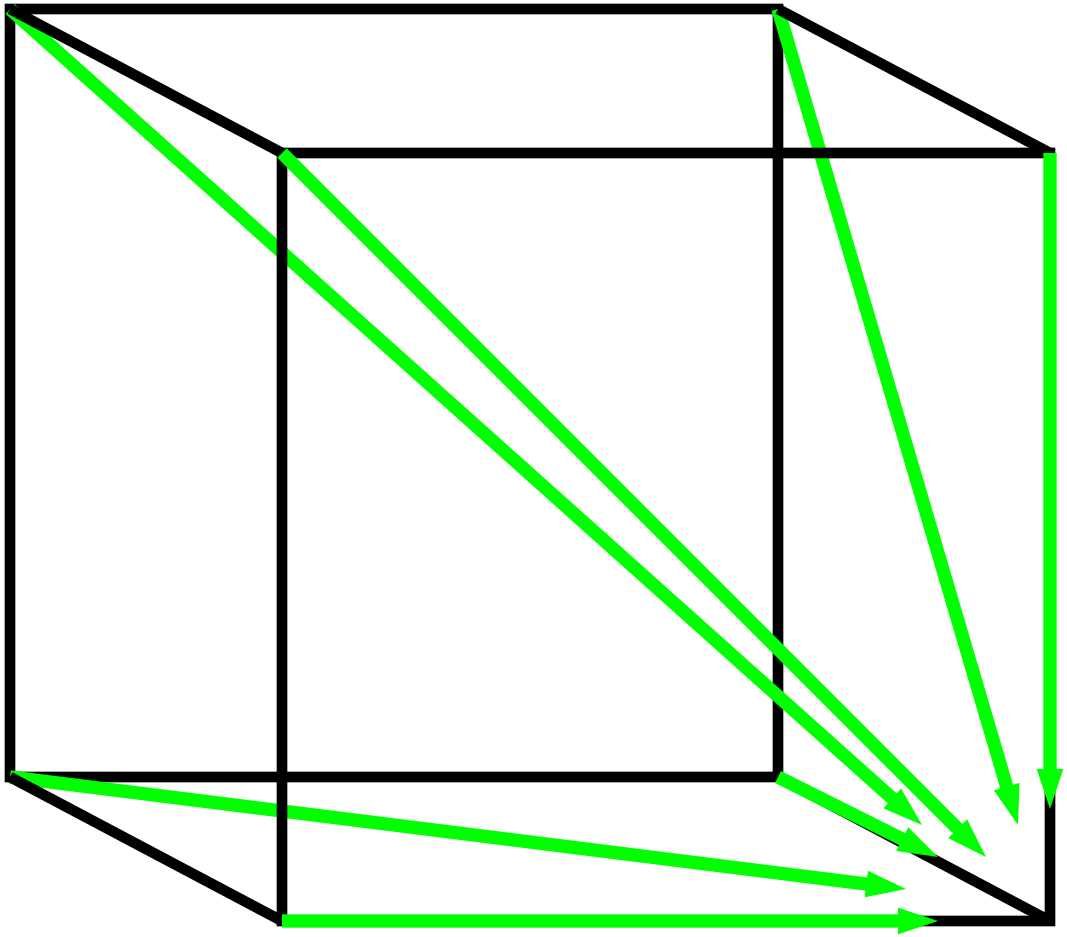


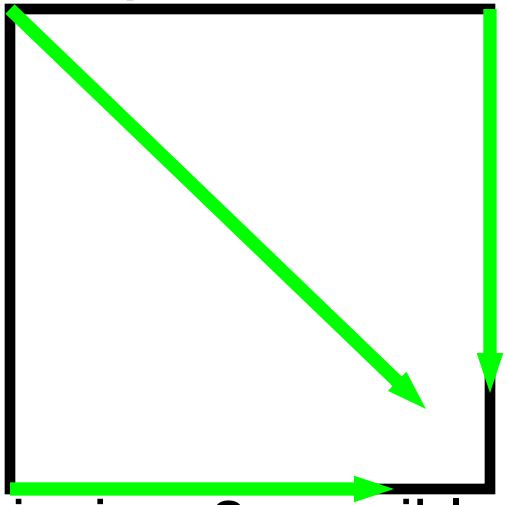
2D table



3D graph

DP recursion (3 edges vs 7)

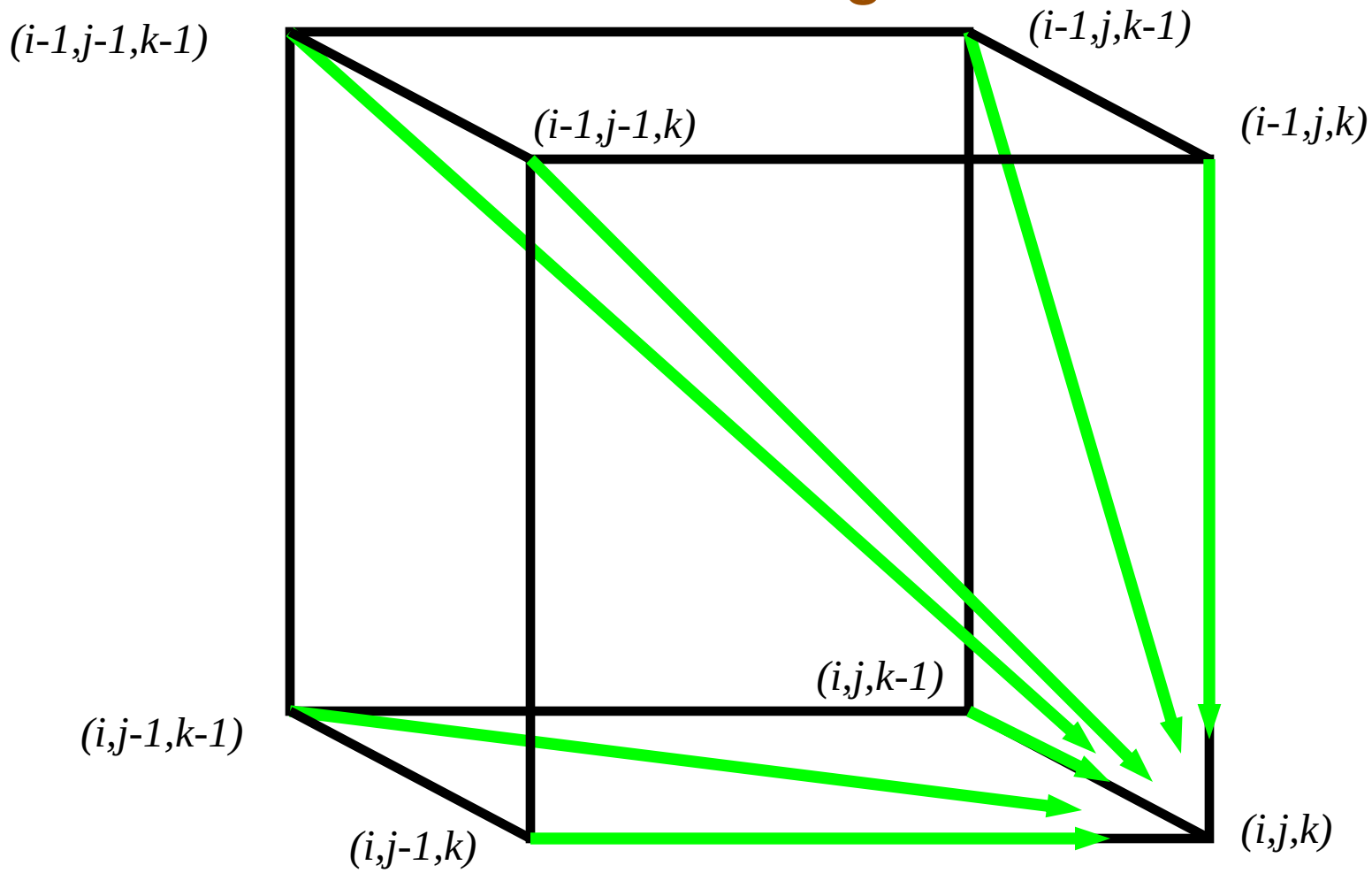




Pairwise: 3 possible paths (match/mismatch, insertion, and deletion)

In **3-D**, 7 edges in each unit cube

Architecture of 3D alignment cell



Multiple alignment: dynamic programming

- $s_{i,j,k} = \max \begin{cases} s_{i-1,j-1,k-1} + \delta(v_i, w_j, u_k) \\ s_{i-1,j-1,k} + \delta(v_i, w_j, \_) \\ s_{i-1,j,k-1} + \delta(v_i, \_, u_k) \\ s_{i,j-1,k-1} + \delta(\_, w_j, u_k) \\ s_{i-1,j,k} + \delta(v_i, \_, \_) \\ s_{i,j-1,k} + \delta(\_, w_j, \_) \\ s_{i,j,k-1} + \delta(\_, \_, u_k) \end{cases}$

  cube diagonal: no indels

  face diagonal: one indel

  edge diagonal: two indels

- $\delta(x, y, z)$ is an entry in the 3D scoring matrix

MSA: running time

- For 3 sequences of length n, the run time is $7n^3$; $O(n^3)$
- For k sequences, build a k-dimensional Manhattan, with run time $(2^k-1)(n^k)$; $O(2^k n^k)$
- Conclusion: dynamic programming approach for alignment between two sequences is easily extended to k sequences (simultaneous approach) but it is impractical due to exponential running time.
- Computing exact MSA is computationally almost impossible, and in practice heuristics are used (progressive alignment)

Profile representation of multiple alignment

```
  -  A  G  G  C  T  A  T  C  A  C  C  T  G
  T  A  G  -  C  T  A  C  C  A  -  -  -  G
  C  A  G  -  C  T  A  C  C  A  -  -  -  G
  C  A  G  -  C  T  A  T  C  A  C  -  G  G
  C  A  G  -  C  T  A  T  C  G  C  -  G  G

A     1              1        .8
C  .6        1           .4  1     .6 .2
G        1 .2                    .2        .4  1
T  .2                 1     .6              .2
-  .2           .8                    .4 .8 .4
```

Aligning alignments/profiles

- Given two alignments, can we align them?

```
x GGGCACTGCAT
y GGTTACGTC--     Alignment 1
z GGGAACTGCAG

w GGACGTACC--     Alignment 2
v GGACCT-----
```

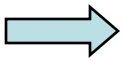
Aligning alignments/profiles

- Given two alignments, can we align them?
- Hint: use alignment of corresponding profiles

```
x GGGCACTGCAT
y GGTTACGTC--     Combined Alignment
z GGGAACTGCAG
w GGACGTACC--
v GGACCT-----
```

Aligning a sequence to a profile

K	L	M	–	K
K	L	K	L	K
K	M	M	L	–
M	L	–	L	M



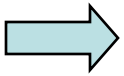
	1	2	3	4	5
K	.75		.25		.75
L		.75		.75	
M	.25	.25	.50		.25
–			.25	.25	.25

New sequence:

K K L L M

Align with profile:

K	K	L	–	L	M
1	–	2	3	4	5



K	K	L	–	L	M
K	–	L	M	–	K
K	–	L	K	L	K
K	–	M	M	L	–
M	–	L	–	L	M

Scoring a sequence-to-profile alignment

- Score each column separately according to PSSM
- Each character contributes to score, weighed by its frequency

	1	2	3	4	5
K	.75		.25		.75
L		.75		.75	
M	.25	.25	.50		.25
–			.25	.25	.25

K	K	L	–	L	M
1	–	2	3	4	5

Column 1 score:
0.75 s(K,K) + 0.25 s(K,M)

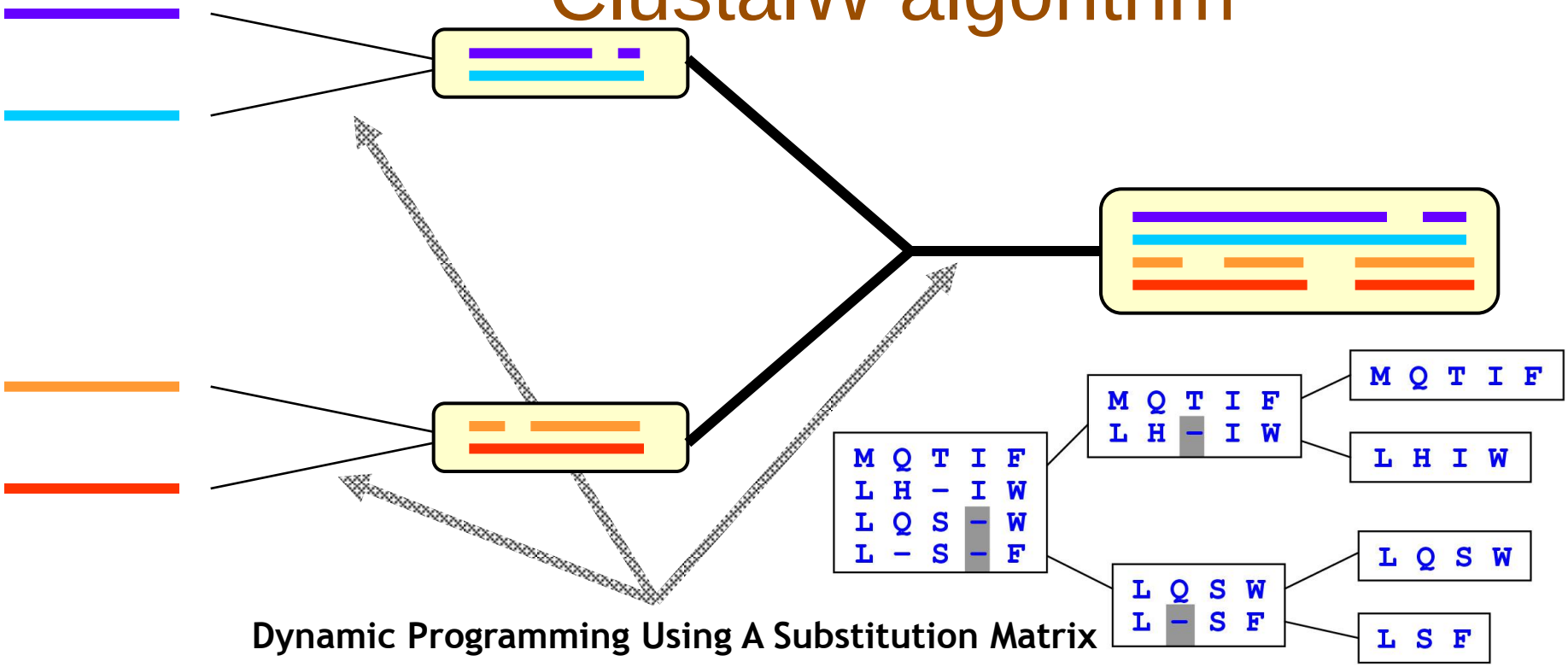
Progressive alignment

- *Progressive alignment* uses guide tree
- Sequence weighting & scoring scheme and gap penalties
- Progressive alignment works well for close sequences, but deteriorates for distant sequences
  - Gaps in consensus string are permanent
  - Use profiles to compare sequences

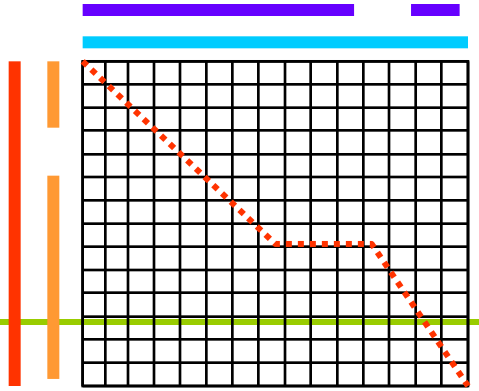
ClustalW

- Popular multiple alignment tool today
- ‘W’ stands for ‘weighted’ (sequences are weighted differently).
- Three-step process
  1.) Construct pairwise alignments
  2.) Build guide tree
  3.) Progressive alignment guided by the tree

ClustalW algorithm



Dynamic Programming Using A Substitution Matrix



Step 1: Pairwise alignment

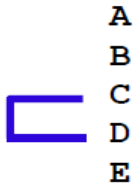
- Aligns each sequence again each other giving a similarity matrix
- Similarity = exact matches / sequence length (percent identity)

	v_1	v_2	v_3	v_4
v_1	–			
v_2	.17	–		
v_3	.87	.28	–	
v_4	.59	.33	.62	–

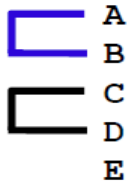
(.17 means 17 % identical)

Step 2: Guide tree

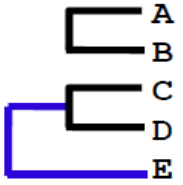
	A	B	C	D	E
A		17	59	59	77
B			37	61	53
C				13	41
D					21

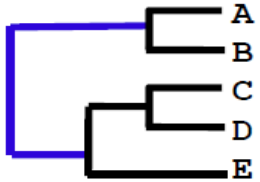


	A	B	E	CD
A		17	77	59
B			53	49
E				31



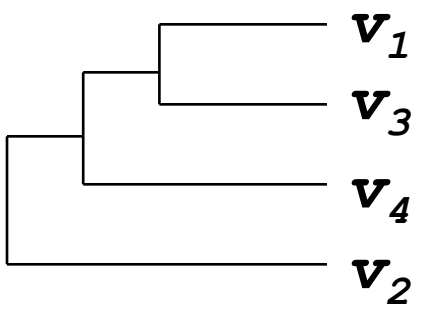
	E	CD	AB
E		31	65
CD			54





Step 2: Guide tree

	v_1	v_2	v_3	v_4
v_1	–			
v_2	.17	–		
v_3	.87	.28	–	
v_4	.59	.33	.62	–



Calculate:

$v_{1,3}$ = alignment (v_1, v_3)

$v_{1,3,4}$ = alignment$((v_{1,3}), v_4)$

$v_{1,2,3,4}$ = alignment$((v_{1,3,4}), v_2)$

ClustalW uses NJ to build guide tree;

Guide tree *roughly* reflects evolutionary relations

Step 3: Tree based recursion

```
Align ( Node N)
        {

        if ( N->left_child is a Node)
                A1=Align ( N->left_child)

        else if ( N->left_child is a Sequence)
                A1=N->left_child

        if (N->right_child is a node)
                A2=Align (N->right_child)

        else if ( N->right_child is a Sequence)
                A2=N->right_child

        Return dp_alignment (A1, A2)
        }
```