For written notes on this lecture, please read chapter 10 of *The Practical Bioinformatician,* and chapter 2 and 5 of *Algorithms in Bioinformatics*.

CS2220: Introduction to Computational Biology

Sequence Comparison Essence

Wong Limsoon

Outline

Protein evolution

Global sequence comparison

Needleman-Wunsch's algo

Ukkonen's idea

Hirschberg's idea

Local sequence alignment

Smith-Waterman's algo

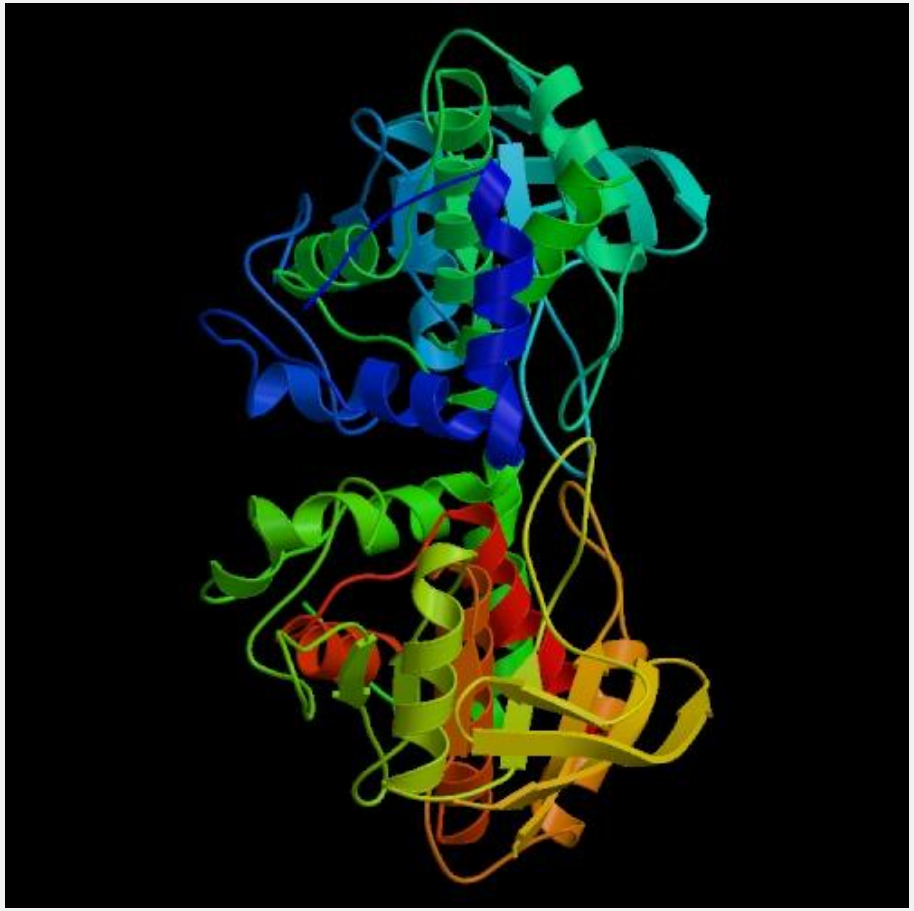
Scoring functions

Protein evolution

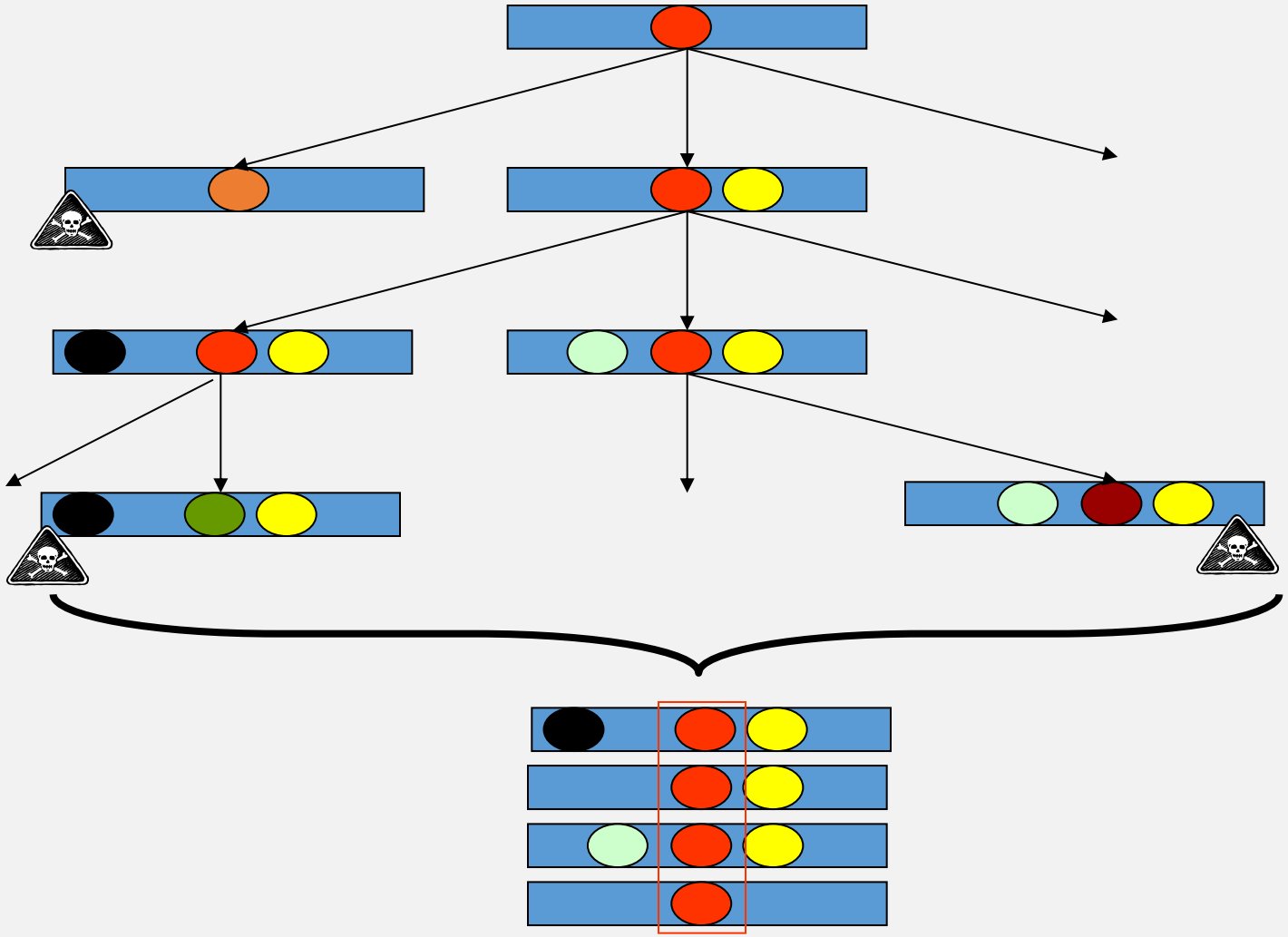
A protein is a ...

A protein is a large complex molecule made up of one or more chains of amino acids

Proteins perform a wide variety of activities in the cell



In the course of evolution…



Remember this exercise?

Let a = AFPHQHRVP

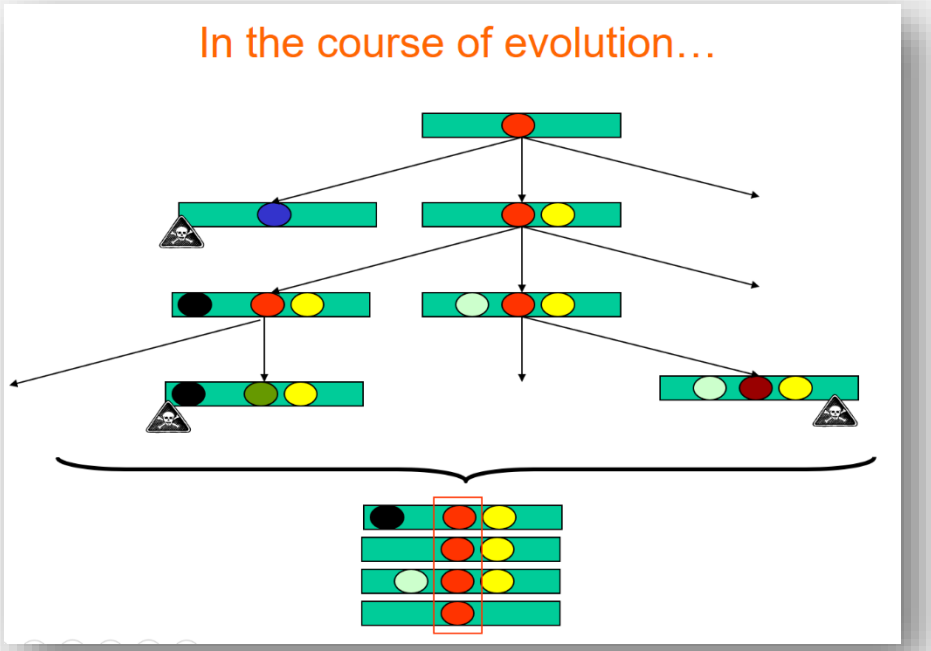
Let b = PQVYNIMKE

Suppose each generation differs from the previous by 1 residue

What is the max difference between the 2nd generation of a?

What is the min difference between the 2nd generation of a and b?

Therefore…



Two proteins inheriting their function from a common ancestor have very similar amino acid sequences

Sequence comparison!

In other words…

The structure of a protein defines its function

In order for a protein to have a specific function, it must satisfy specific structural constraints

The “Law”: Protein evolves → amino acid seq changes → protein structure changes → breaks those structural constraints → protein loses function

By abduction: The more similar two proteins’ amino acid sequences are, the more likely they come from the same ancestor → the more likely they have the same structure and function

Uses of sequence comparison

Protein Structure Prediction

Similar sequence have similar structure & function

Homology-based protein structure prediction

Genome Annotation

Homology-based gene prediction

Function assignment

Evolutionary studies

Searching drug targets

Searching sequence present or absent across genomes

Global sequence alignment

Earliest research in seq comparison

Doolittle et al. (Science, July 1983) searched for platelet-derived growth factor (PDGF) in his own DB. He found that PDGF is similar to v-sis oncogene

```
PDGF-2   1        SLGSLTIAEPAMIAECKTREEVFCICRRL?DR??  34
p28sis  61 LARGKRSLGSLSVAEPAMIAECKTRTEVFEISRRLIDRTN 100
```


Source: Ken Sung

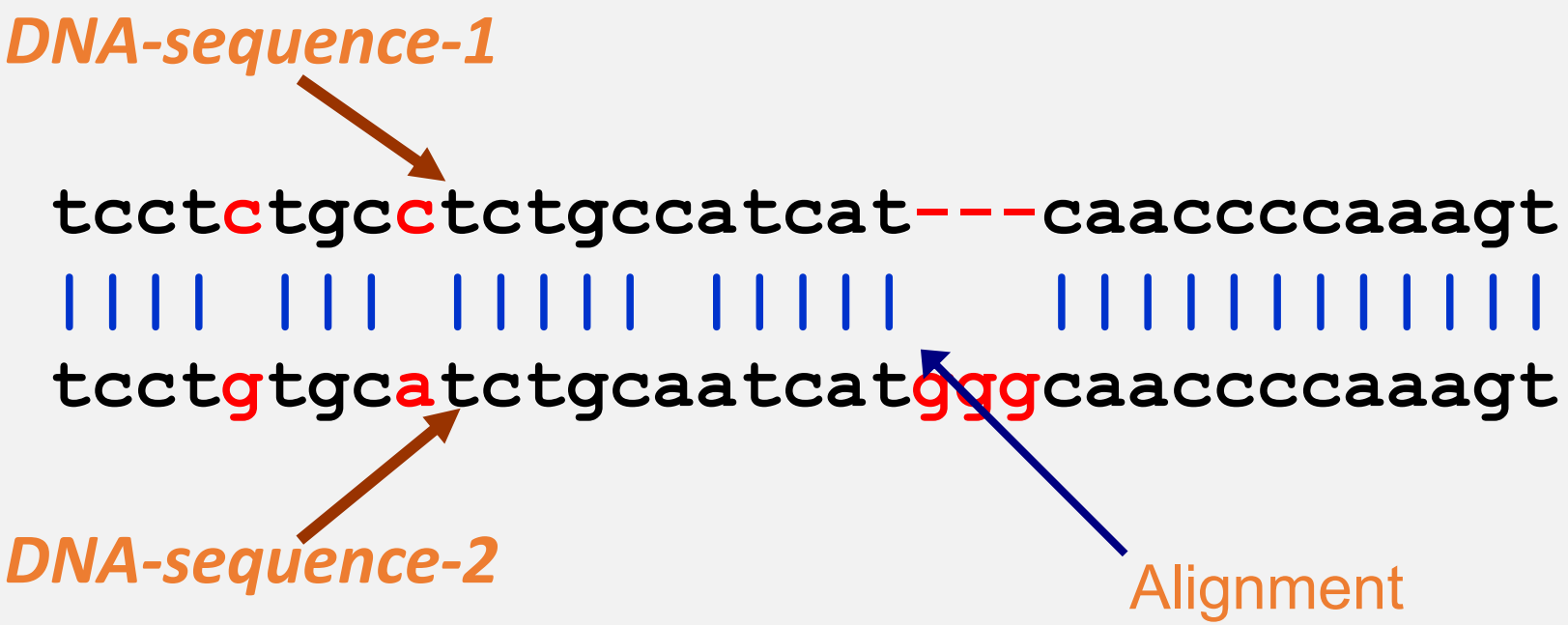
Exercise

```
PDGF-2   1        SLGSLTIAEPAMIAECKTREEVFCICRRL?DR?? 34
p28sis  61 LARGKRSLGSLSVAEPAMIAECKTRTEVFEISRRLIDRTN 100
```

Are these pairs of amino acids compatible (T, S), (I, V), and (C, S)?

I.e., are these conservative substitutions that do not change the structure and function of a protein?

Sequence alignment



Key aspect of sequence comparison is sequence alignment, which maximizes the # of positions that are in agreement between the sequences

Terminologies

Substitution

A nucleotide is replaced by another, e.g. ATA → AGA

Insertion

A new nucleotide is inserted between two existing nucleotides, e.g. AA → AGA

Deletion

An existing nucleotide is deleted. e.g. ACTG → AC-G

Indel

An insertion or a deletion

Poor sequence alignment

Poor sequence alignment shows few matched positions

The two proteins are not likely to be homologous

Alignment by FASTA of the sequences of amicyanin and domain 1 of ascorbate oxidase

```
                              60         70         80         90         100
Amicyanin              MPHNVHFVAGVLGEAALKGPMMKKEQAYSLTFTEAGTYDYHCTPHPFMRGKVVVE
                                                  :..:   . ::. ::
Ascorbate Oxidase ILQRGTPWADGTASISQCAINPGETFFYNFTVDNPGTFFYHGHLGMQRSAGLYGSLI
                        70         80         90         100        110        120
```

No obvious match between
Amicyanin and Ascorbate Oxidase

Good sequence alignment

Good alignment usually has clusters of extensive matched positions

The two proteins are likely to be homologous

```
>gi|13476732|ref|NP_108301.1|    unknown protein [Mesorhizobium loti]
gi|14027493|dbj|BAB53762.1|    unknown protein [Mesorhizobium loti]
          Length = 105

 Score =  105 bits (262), Expect = 1e-22
 Identities = 61/106 (57%), Positives = 73/106 (68%), Gaps = 1/106 (0%)

Query: 1    MKPGRLASIALAIIFLPMAVPAHAATIEITMENLVISPTEVSAKVGDTIRWVNKDVFAHT 60
             MK G L  ++       MA PA AATIE+T++ LV SP  V AKVGDTI WVN DV AHT
Sbjct: 1    MKAGALIRLSWLAALALMAAPAAAATIEVTIDKLVFSPATVEAKVGDTIEWVNNDVVAHT 60
```

good match between

Amicyanin and unknown M. loti protein

Exercise

Poor sequence alignment

Poor sequence alignment shows few matched positions
The two proteins are not likely to be homologous

Alignment by FASTA of the sequences of amicyanin and domain 1 of ascorbate oxidase

```
                    60        70        80        90       100
Amicyanin          MPHNVHFVAGVLGEAALKGPMMKKEQAYSLTFTEAGTYDYHCTPHPFMRGKVVVE
                                          :..:  . ::. ::
Ascorbate Oxidase ILQRGTPWADGTASISQCAINPGETFFYNFTVDNPGTFFYHGHLGMQRSAGLYGSLI
                    70        80        90       100       110       120
```

No obvious match between
Amicyanin and Ascorbate Oxidase

Good sequence alignment

Good alignment usually has clusters of extensive matched positions
The two proteins are likely to be homologous

```
>gi|134476732|ref|NP_108301.1|  unknown protein [Mesorhizobium loti]
gi|14027493|dbj|BAB53762.1|  unknown protein [Mesorhizobium loti]
          Length = 105

 Score =  105 bits (262), Expect = 1e-22
 Identities = 61/106 (57%), Positives = 73/106 (68%), Gaps = 1/106 (0%)

Query: 1   MKPGRLASIALAIIFLPMAVPAHAATIEITMENLVISPTEVSAKVGDTIRWVNKDVFAHT 60
           MK G L  ++       MA PA AATIE+T++ LV SP  V AKVGDTI WVN DV AHT
Sbjct: 1   MKAGALIRLSWLAALALMAAPAAAATIEVTIDKLVFSPATVEAKVGDTIEWVNNDVVAHT 60
```

good match between
Amicyanin and unknown M. loti protein

How to formally define what a good alignment is?

Longest common subsequence (LCS)

Given two sequences $X = x_1\ x_2\ \ldots\ x_m$ and $Y = y_1\ y_2\ \ldots\ y_n$

The LCS is the longest sequence $Z = z_1\ z_2\ \ldots\ z_k$ that appear in X and Y in the same order but not necessarily contiguously

Example:

X = **ACD**B**E**

Y = **A**B**CDE**

LCS is Z = **ACDE**

LCS can be computed using this recurrence:

$$L(i,j) = \begin{cases} L(i-1, j-1) + 1, & \text{if } x_i = y_j \\ \max(L(i-1, j), L(i, j-1)), & \text{if } x_i \neq y_j \end{cases}$$

Exercise

Under what situation is finding the best sequence alignment problem equivalent to finding the LCS of the two sequences being aligned?

Global pairwise alignment

Find an alignment A so that S(A) is max among exponential number of possible alternatives

Given sequences U and V of lengths n and m, then number of possible alignments is given by

$f(n, m) = f(n – 1, m – 1) + f(n – 1, m) + f(n, m – 1)$

$f(n, n) \sim (1 + \sqrt{2})^{2n+1} / \sqrt{n}$

Global pairwise alignment

Define an indel-similarity matrix s(.,.); e.g.,

s(x, x) = 2

s(x, y) = -μ, if x ≠ y

Then Let U and V be two sequences of length n and m. Then their global pairwise alignment can be extracted from the dynamic programming computation of $S_{n,m}$, where

$$S_{i,j} = \max \left\{ \begin{array}{l} S_{i-1,j-1} + s(u'_i, v'_j) \\ S_{i-1,j} - \delta \\ S_{i,j-1} - \delta \end{array} \right\}$$

This is the basic idea of the Needleman-Wunsch algorithm

Exercise

Let U and V be two sequences of length n and m. Then their global pairwise alignment can be extracted from the dynamic programming computation of $S_{n,m}$, where

$$S_{i,j} = \max \left\{ \begin{array}{l} S_{i-1,j-1} + s(u'_i, v'_j) \\ S_{i-1,j} - \delta \\ S_{i,j-1} - \delta \end{array} \right\}$$

What happens when δ is large?

Needleman-Wunsch algorithm for global alignment

Consider two strings

S[1..n] and T[1..m]

Let V(i, j) be score of optimal alignment between S[1..i] and T[1..j]

Basis:

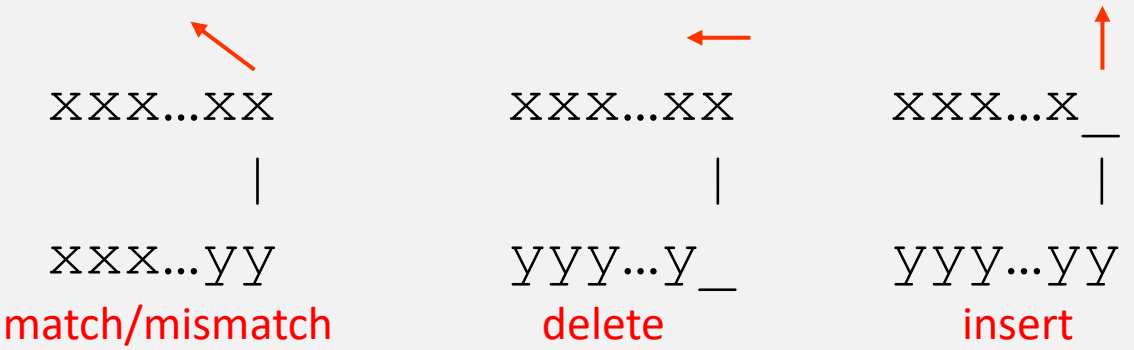
V(0, 0) = 0

V(0, j) = V(0, j –1) – δ, insert j times

V(i, 0) = V(i – 1, 0) – δ, delete i times

For i, j > 0:

$$V(i,j) = \max \begin{cases} V(i-1,j-1) + s(S[i], T[j]) & \text{(mis)match} \\ V(i-1,j) - \delta & \text{delete} \\ V(i,j-1) - \delta & \text{insert} \end{cases}$$



Source: Ken Sung

Example

s(x, x) = 2
s(x, y) = -1, if x ≠ y

	_	A	G	C	A	T	G	C
_	0	–1	– 2	– 3	– 4	– 5	– 6	– 7
A	– 1							
C	– 2							
A	– 3							
A	– 4							
T	– 5							
C	– 6							
C	– 7							

Source: Ken Sung

Example, cont'd

s(x, x) = 2
s(x, y) = -1, if x ≠ y

	_	A	G	C	A	T	G	C
_	0	–1	– 2	– 3	– 4	– 5	– 6	– 7
A	– 1	2						
C	– 2							
A	– 3							
A	– 4							
T	– 5							
C	– 6							
C	– 7							

$$S_{1,1} = \max \begin{cases} S_{0,0} + s(A,A) \\ S_{0,1} - 1 \\ S_{1,0} - 1 \end{cases} = \max \begin{cases} 0 + 2 \\ -1 - 1 = 2 \\ -1 - 1 \end{cases}$$

Source: Ken Sung

Example, further cont'd

s(x, x) = 2
s(x, y) = -1, if x ≠ y

	_	A	G	C	A	T	G	C
_	0	–1	– 2	– 3	– 4	– 5	– 6	– 7
A	– 1	2	1					
C	– 2							
A	– 3							
A	– 4							
T	– 5							
C	– 6							
C	– 7							

$$S_{1,2} = \max\begin{cases} S_{0,1} + s(A,G) \\ S_{0,2} - 1 \\ S_{1,1} - 1 \end{cases} = \max\begin{cases} -1 + -1 \\ -2 - 1 \\ 2 - 1 \end{cases} = 1$$

Source: Ken Sung

Exercise: fill up the rest of the table

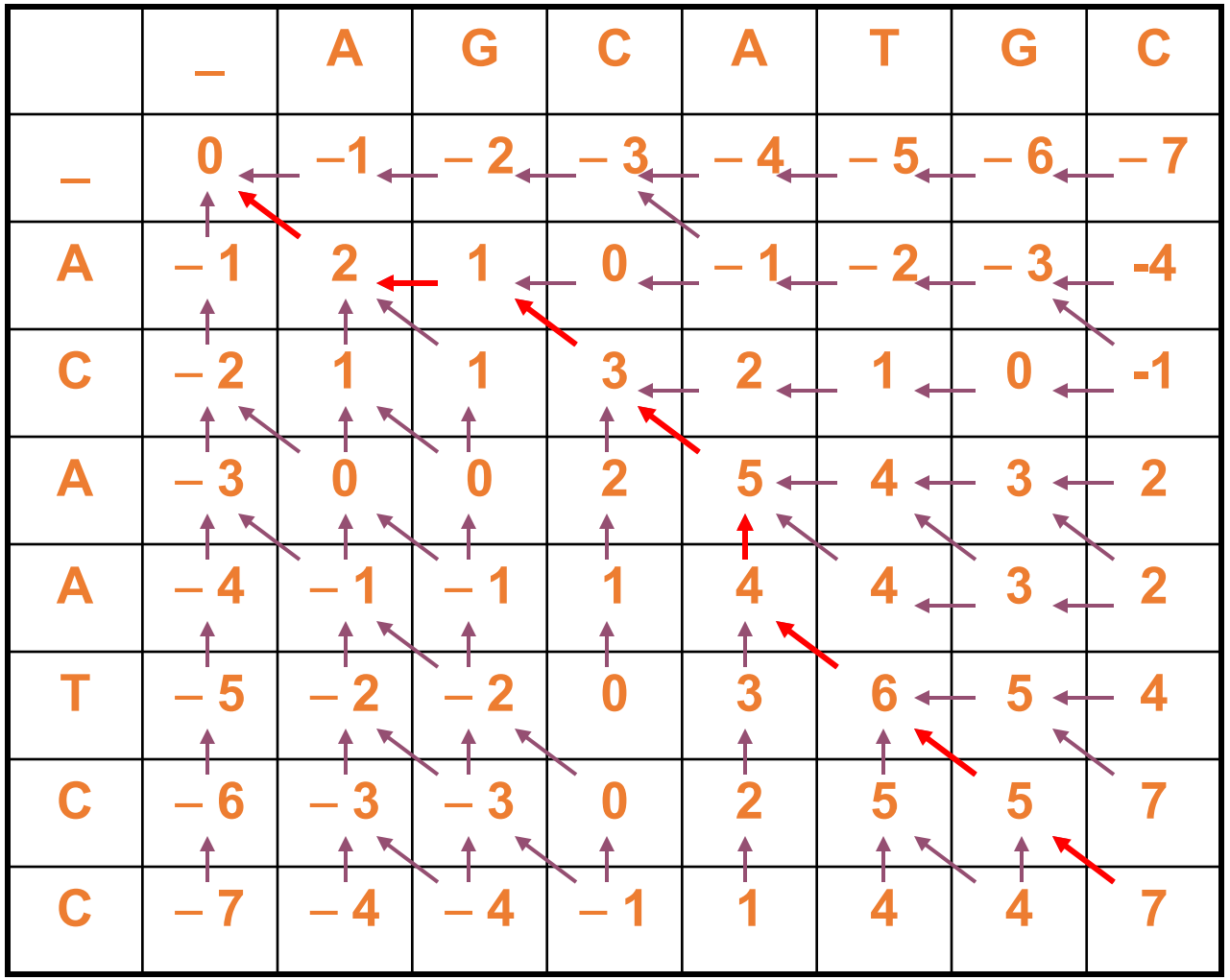
s(x, x) = 2
s(x, y) = -1, if x ≠ y

	_	A	G	C	A	T	G	C
_	0	– 1	– 2	– 3	– 4	– 5	– 6	– 7
A	– 1	2	1	0	– 1	– 2	– 3	– 4
C	– 2	1	1	3	2			
A	– 3							
A	– 4							
T	– 5							
C	– 6							
C	– 7							

Exercise

What is the alignment that corresponds to this table?

Source: Ken Sung



	_	A	G	C	A	T	G	C
_	0	–1	– 2	– 3	– 4	– 5	– 6	– 7
A	– 1	2	1	0	– 1	– 2	– 3	-4
C	– 2	1	1	3	2	1	0	-1
A	– 3	0	0	2	5	4	3	2
A	– 4	– 1	– 1	1	4	4	3	2
T	– 5	– 2	– 2	0	3	6	5	4
C	– 6	– 3	– 3	0	2	5	5	7
C	– 7	– 4	– 4	– 1	1	4	4	7

Source: Ken Sung

Pseudo codes

```
Create table V[0..n, 0..m] and P[1..n, 1..m];
V[0,0] = 0;
For j= 1 to m, set V[0, j] := v[0, j – 1] – δ ;
For i= 1 to n, set V[i, 0] := V[i – 1, 0] – δ ;
For j= 1 to m {
     For i = 1 to n {
          set V[i, j] := V[i, j – 1] – δ ;
          set P[i, j] := (0, – 1);
          if V[i, j] < V[i – 1, j] – δ then
               set V[i, j] := V[i – 1, j] – δ ;
               set P[i, j] := (– 1, 0);
          if (V[i, j] < V[i – 1, j – 1] + s(S[i], T[j])) then
               set V[i, j] := V[i – 1, j – 1] + s(S[i], T[j]);
               set P[i, j] := (– 1, – 1);
     }
}
Backtracking P[n,m] to P[0,0] to find optimal alignment;
```


Source: Ken Sung

Analysis

We need to fill in all entries in the $n \times m$ matrix

Each entry can be computed in O(1) time

Time complexity = O(nm)

Space complexity = O(nm)

Source: Ken Sung

Problem on speed

Aho, Hirschberg, Ullman 1976

If can only compare whether two symbols are equal, the string alignment problem can be solved in $\Omega(nm)$ time

Hirschberg 1978

If symbols are ordered and can be compared, the string alignment problem can be solved in $\Omega(n \log n)$ time

Masek and Paterson 1980

Based on Four-Russian's paradigm, the string alignment problem can be solved in O(nm/log2 n) time

Let d be the total number of inserts and deletes. Thus $0 \leq d \leq n + m$. If d is smaller than n + m, can we get a better algorithm? Yes!

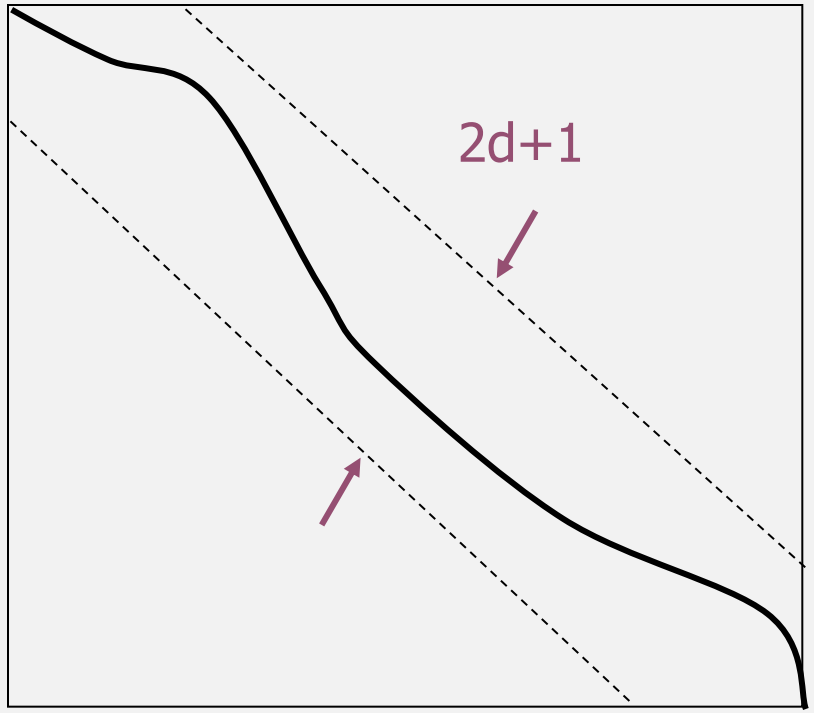
Source: Ken Sung

Ukkonen's idea for an O(dn)-time algorithm

The alignment should be inside the 2d+1 band

No need to fill-in the lower and upper triangle

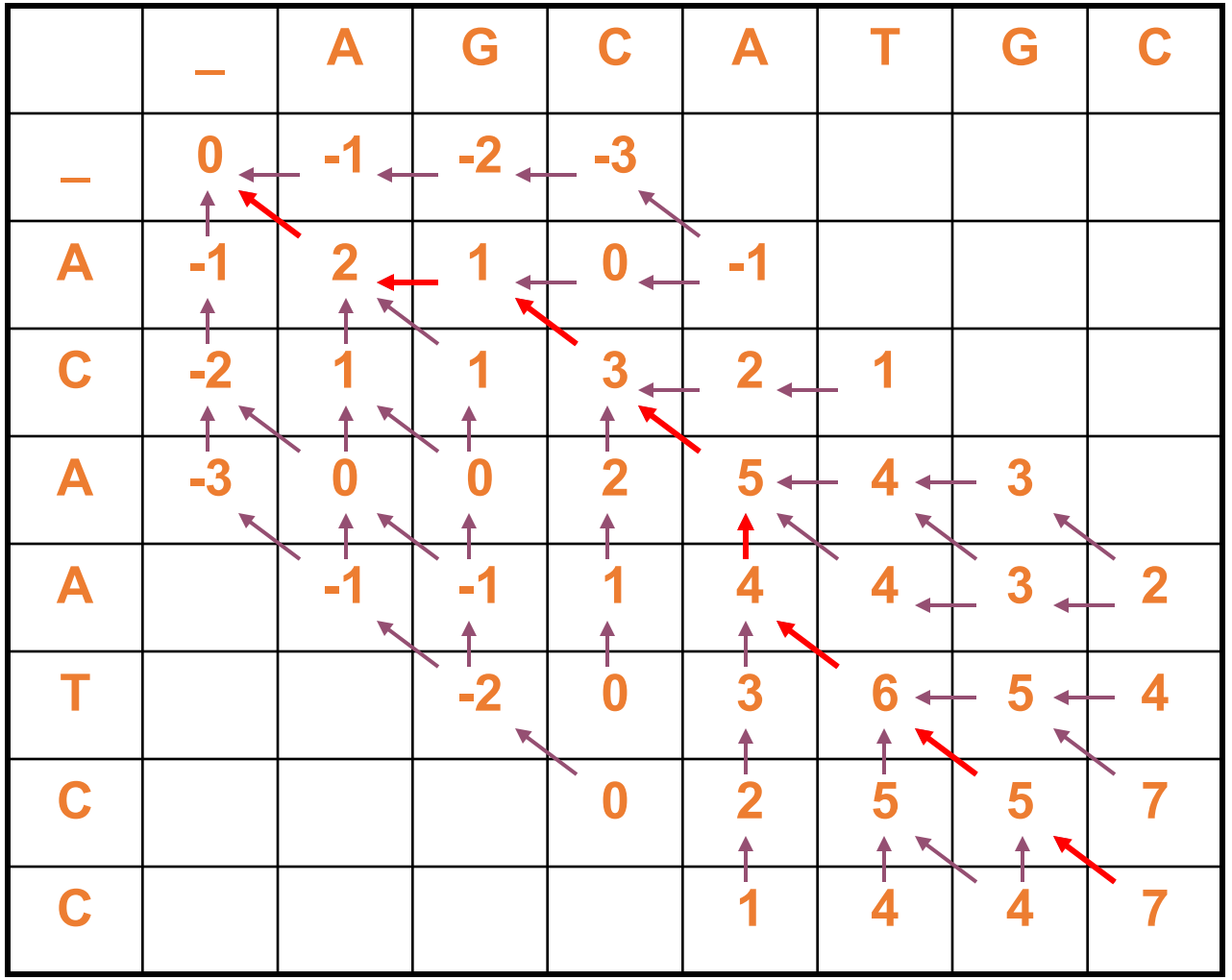
Time complexity: O(dn)



Source: Ken Sung

Example

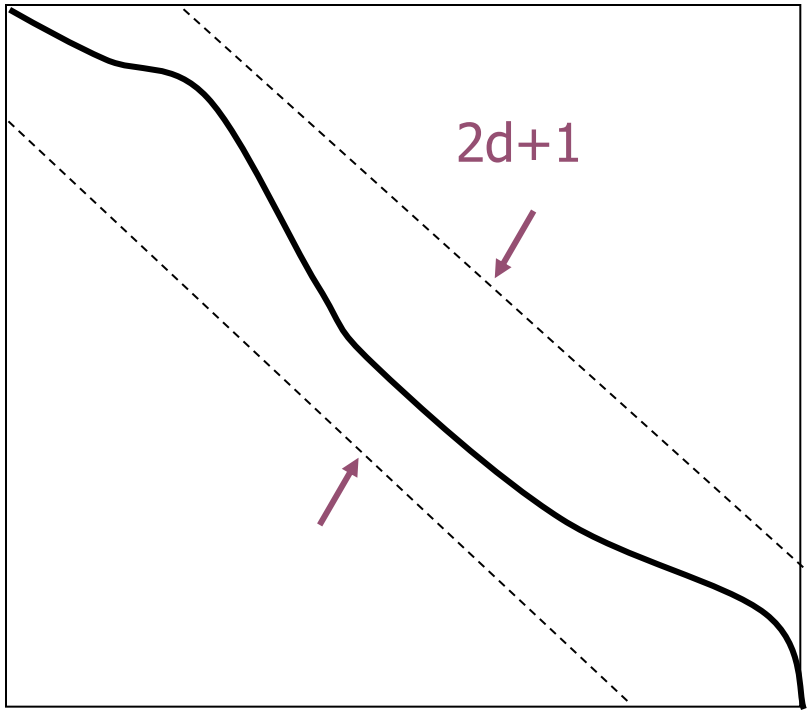
d = 3
A_CAATCC
AGCA_TGC



	_	A	G	C	A	T	G	C
_	0	-1	-2	-3				
A	-1	2	1	0	-1			
C	-2	1	1	3	2	1		
A	-3	0	0	2	5	4	3	
A		-1	-1	1	4	4	3	2
T			-2	0	3	6	5	4
C				0	2	5	5	7
C					1	4	4	7

Exercise

Write down Ukkonen's algorithm



Problem on space

Dynamic programming requires O(mn) space

When we compare two very long sequences, space may be a limiting factor

Can we solve the string alignment problem in linear space?

Easy, if no need to recover alignment

When filling row 3, it depends only on row 2
No need to keep rows 0 and 1

I.e., we only need to keep two rows

“Cost only” algo

	_	A	G	C	A	T	G	C
_	0	-1	-2	-3	-4	-5	-6	-7
A	-1	2	1	0	-1	-2	-3	-4
C	-2	1	1	3	2	1	0	-1
A	-3	0	0	2	5	4	3	2
A	-4	-1	-1	1	4	4	3	2
T	-5	-2	-2	0	3	6	5	4
C	-6	-3	-3	0	2	5	5	7
C	-7	-4	-4	-1	1	4	4	7

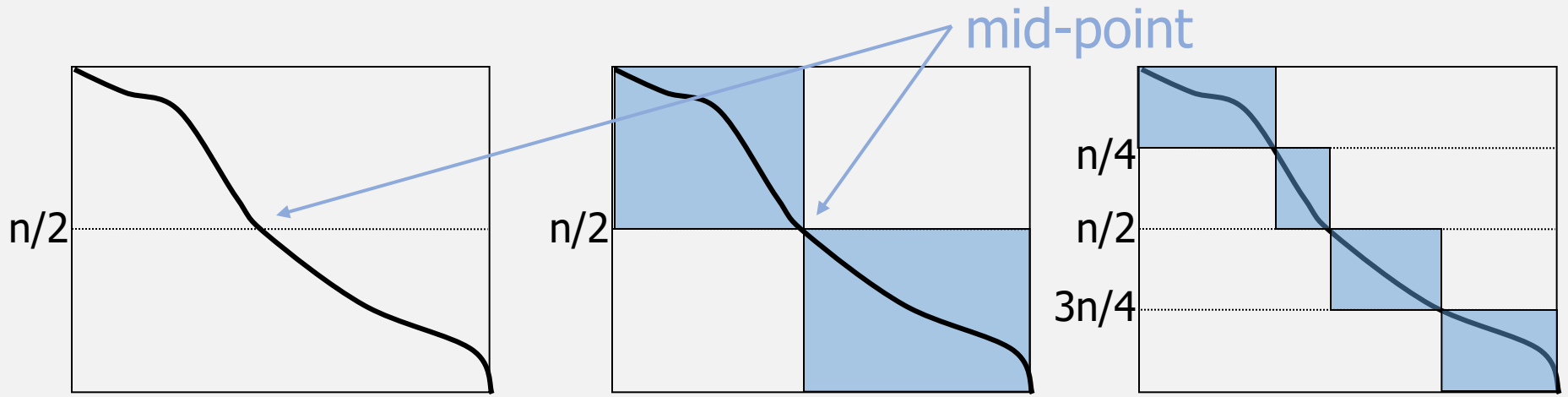
Source: Ken Sung

Recovering alignment in O(n + m) space

Use cost-only algo to find mid-point of alignment

Divide the problem into two halves

Recursively deduce alignments for the two halves



Source: Ken Sung

How to find mid-point: Hirschberg's idea

$$V(S[1..n], T[1..m]) =$$
$$\max_{0 \le j \le m} \left\{ V(S[1..\tfrac{n}{2}], T[1..j]) + V(S[\tfrac{n}{2}+1..n], T[j+1..m]) \right\}$$

Do cost-only dynamic programming for 1st half
I.e., find V(S[1..n/2], T[1..j]) for all j

Do cost-only dynamic programming for 2nd half
i.e., find V(S[n/2+1..n], T[j+1..m]) for all j

Determine j which maximizes the sum above

Source: Ken Sung

Example

Step 1

	_	A	G	C	A	T	G	C	_
_	0	-1	-2	-3	-4	-5	-6	-7	
A	-1	2	1	0	-1	-2	-3	-4	
C	-2	1	1	3	2	1	0	-1	
A	-3	0	0	2	5	4	3	2	
A	-4	-1	-1	1	4	4	3	2	
T									
C									
C									
_									

Step 2

	_	A	G	C	A	T	G	C	_
_									
A									
C									
A									
A	-4	-1	-1	1	4	4	3	2	
T		-1	0	1	2	3	0	0	-3
C		-2	-1	1	-1	0	1	1	-2
C		-4	-3	-2	-1	0	1	2	-1
_		-7	-6	-5	-4	-3	-2	-1	0

Step 3

	_	A	G	C	A	T	G	C	_
_									
A									
C									
A									
A	-4	-1	-1	1	4	4	3	2	
T		-1	0	1	2	3	0	0	-3
C									
C									
_									

Step 4: Recursive on subproblems

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

Source: Ken Sung

Exercise

Write down Hirschberg's algorithm for computing sequence alignment in linear space

Complexity analysis

Space

O(m) working memory for finding mid-point

Once mid-point is found, can free working memory → In each recursive call, we only need to store the alignment path

Alignment subpaths are disjoint → total space required is O(n+m)

Time? This one is for you to think about ☺

More Realistic Handling of Indels

In Nature, indels of several adjacent letters are not the sum of single indels, but the result of one event

So reformulate as follows:

Let $g(k)$ be the indel weight for an indel of k letters. Typically, $g(k) \leq k \cdot g(1)$. Let U and V be two sequences of length n and m. Then their global pairwise alignment can be extracted from the dynamic programming computation of $S_{n,m}$, where

$$S_{0,0} = 0, \quad S_{0,j} = -g(j), \quad S_{i,0} = -g(i)$$

$$S_{i,j} = \max \left\{ \begin{array}{l} S_{i-1,j-1} + s(u'_i, v'_j) \\ \max_{1 \leq k \leq j} \{S_{i,j-k} - g(k)\} \\ \max_{1 \leq k \leq i} \{S_{i-k,j} - g(k)\} \end{array} \right\}$$

Affine gap penalty

g(q):$\mathbb{N}\rightarrow\Re$ is the penalty of a gap of length q

Note g() is subadditive, i.e, $g(p+q) \leq g(p) + g(q)$

If $g(k) = \alpha + \beta k$, the gap penalty is called affine

A penalty (α) for initiating the gap

A penalty (β) for the length of the gap

Needleman-Wunsch w/ general gap penalty

Global alignment of

S[1..n] and T[1..m]:

Denote V(i, j) be the score for global alignment between S[1..i] and T[1..j]

Base cases:

V(0, 0) = 0
V(0, j) = g(j)
V(i, 0) = g(i)

For i, j > 0:

$$V(i,j) = \max \begin{cases} V(i-1, j-1) + \delta(S[i], T[j]) & \text{(mis)match} \\ \max_{0 \le k \le j-1} \{V(i,k) + g(j-k)\} & \text{insert T[k+1..j]} \\ \max_{0 \le k \le i-1} \{V(k,j) + g(i-k)\} & \text{delete S[k+1..i]} \end{cases}$$

Source: Ken Sung

Analysis

We need to fill in all entries in the $n \times m$ table

Each entry can be computed in $O(\max\{n, m\})$ time

$\Rightarrow$Time complexity = $O(nm \max\{n, m\})$

$\Rightarrow$Space complexity = $O(nm)$

Source: Ken Sung

For you to learn yourself: Gotoh's idea

$$V(i,j) = \max \begin{cases} V(i-1, j-1) + \delta(S[i], T[j]) & \text{(mis)match} \\ \max\limits_{0 \le k \le j-1} \{V(i,k) + g(j-k)\} & \text{insert T[k+1..j]} \\ \max\limits_{0 \le k \le i-1} \{V(k,j) + g(i-k)\} & \text{delete S[k+1..i]} \end{cases}$$

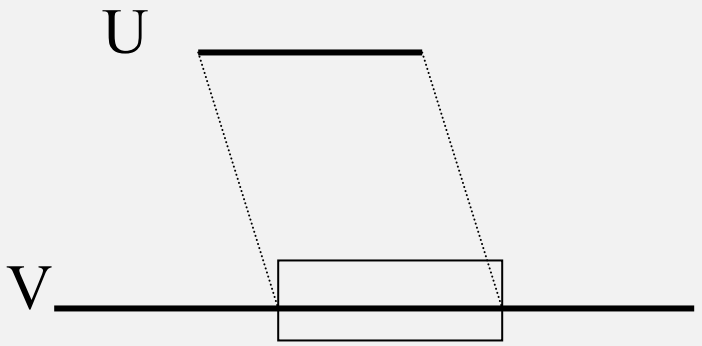
Direct dynamic programming implementation of this has cubic time complexity

Osamu Gotoh described a simple and elegant solution in 1982

Local sequence alignment

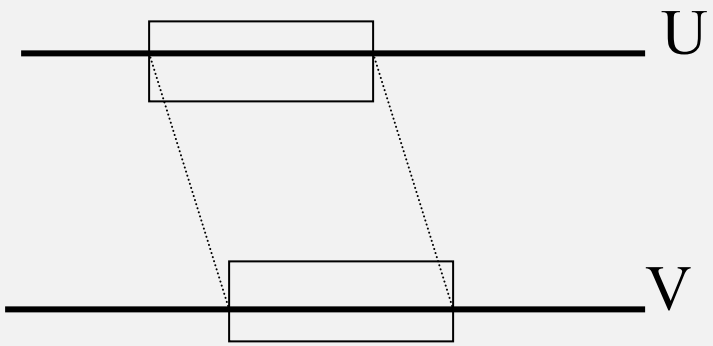
Variations of pairwise alignment

Fitting a “short” seq to a “long” seq



Indels at beginning and end are not penalized

Find “local” alignment



Find *i, j, k, l,* so that
S(A) is maximized,
A is alignment of $u_i \ldots u_j$ and $v_k \ldots v_l$

Local vs global alignment

Source: Somayyeh Koohi

Global alignment may not find local similarities

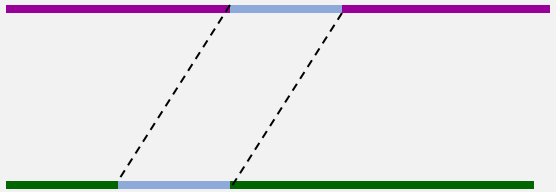
Global alignment

```
--T--CC-C-AGT--TATGT-CAGGGGACACG-A-GCATGCAGA-GAC
  |   || |  ||   | | | |||     || | | |   | ||||   |
AATTGCCGCC-GTCGT-T-TTCAG----CA-GTTATG-T-CAGAT--C
```

Local alignment – found the conserved segment

```
                    tccCAGTTATGTCAGgggacacgagcatgcagagac
                       ||||||||||||
aattgccgccgtcgttttcagCAGTTATGTCAGatc
```

Local alignment problem



Given two long DNAs, both of them contain the same gene or closely related gene. Can we identify the gene?

Local alignment problem: Given two strings S[1..n] and T[1..m], among all substrings of S and T, find substrings A of S and B of T whose global alignment has the highest score

Source: Ken Sung

Brute-force solution

Algorithm

For every substring A of S, for every substring B of T, compute the global alignment of A and B

Return the pair (A, B) with the highest score

Time

There are n^2 choices of A and m^2 choices of B

Global alignment computable in O(nm) time

In total, time complexity = $O(n^3m^3)$

Can we do better?

Suffix and prefix

X is a suffix of S[1..n] if X = S[k..n] for some $k \geq 1$

X is a prefix of S[1..n] if X = S[1..k] for some $k \leq n$

E.g.

Consider S[1..7] = ACCGATT

ACC is a prefix of S, GATT is a suffix of S

Empty string is both prefix and suffix of S

Which string is both a prefix and a suffix of S?

Smith-Waterman's algorithm for local alignment

Define V(i, j) as max score of global alignment of (A,B) over all suffixes A of S[1..i] and all suffixes B of T[1..j]

Then, local alignment score is $\max_{i,j}$ V(i ,j)

i.e., ignore end segment

Basis:

V(i, 0) = V(0, j) = 0

For i, j>0:

$$V(i,j) = \max \begin{cases} 0 & \text{Ignore initial segment} \\ V(i-1,j-1) + s(S[i], T[j]) & \text{(mis)match} \\ V(i-1,j) - \delta & \text{delete} \\ V(i,j-1) - \delta & \text{insert} \end{cases}$$

Source: Ken Sung

Example

Score for match = 2

Score for insert, delete, mismatch = –1

	_	C	T	C	A	T	G	C
_	0	0	0	0	0	0	0	0
A	0							
C	0							
A	0							
A	0							
T	0							
C	0							
G	0							

Source: Ken Sung

Example, cont'd

Score for match = 2

Score for insert, delete, mismatch = –1

	_	C	T	C	A	T	G	C
_	0	0	0	0	0	0	0	0
A	0	0	0	0	2	1	0	0
C	0	2	1	2	1	1	0	2
A	0	1	1	1	4	3	2	1
A	0	0	0	0	3	3	2	1
T	0	0	?					
C								
G								

Source: Ken Sung

Exercise

Score for match = 2

Score for insert, delete, mismatch = –1

Fill in the rest of this table

	_	C	T	C	A	T	G	C
_	0	0	0	0	0	0	0	0
A	0	0	0	0	2	1	0	0
C	0	2	1	2	1	1	0	2
A	0	1	1	1	4	3	2	1
A	0	0	0	0	3	3	2	1
T	0	0	2	1	2			
C								
G								

Source: Ken Sung

Exercise

Score for match = 2

Score for insert, delete, mismatch = –1

What is the alignment corresponding to this table?

	_	C	T	C	A	T	G	C
_	0	0	0	0	0	0	0	0
A	0	0	0	0	2	1	0	0
C	0	2	1	2	1	1	0	2
A	0	1	1	1	4	3	2	1
A	0	0	0	0	3	3	2	1
T	0	0	2	1	2	5	4	3
C	0	2	1	4	3	4	4	6
G	0	1	1	3	3	3	6	5

Source: Ken Sung

Analysis

Need to fill in all entries in the $n \times m$ matrix

Each entries can be computed in O(1) time

Finally, finding the entry with the max value

Time complexity = ??

Space complexity = O(nm)

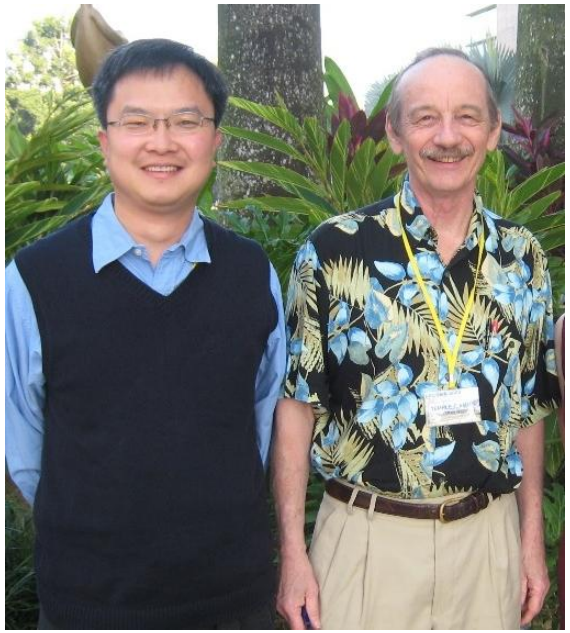
Local alignment constrained to no more than d indels

```
for i = 0..n:
    for j = max(0, i-d)..min(m, i+d):
        H[i][j] = 0   // Smith-Waterman local alignment base case

max_score = 0
max_pos = (0,0)

for i = 1..n:
    for j = max(1, i-d)..min(m, i+d):
        H[i][j] = max(
            0,                          // local alignment: restart
            H[i-1][j-1] + s(A[i],B[j]), // match/mismatch
            H[i-1][j] + g,              // deletion in B
            H[i][j-1] + g               // insertion in B
        )

        if H[i][j] > max_score:
            max_score = H[i][j]
            max_pos = (i,j)
```

```
// Traceback
i,j = max_pos
alignment_A, alignment_B = "", ""
while i > 0 and j > 0 and H[i][j] > 0:
    if H[i][j] == H[i-1][j-1] + s(A[i],B[j]):
        alignment_A = A[i] + alignment_A
        alignment_B = B[j] + alignment_B
        i -= 1; j -= 1
    else if H[i][j] == H[i-1][j] + g:
        alignment_A = A[i] + alignment_A
        alignment_B = "-" + alignment_B
        i -= 1
    else: // H[i][j] == H[i][j-1] + g
        alignment_A = "-" + alignment_A
        alignment_B = B[j] + alignment_B
        j -= 1

return alignment_A, alignment_B, max_score
```

Photos

Limsoon & Temple Smith



Ken & Michael Waterman

Scoring functions

Scoring function for DNA

For DNA, since we only have 4 nucleotides, the score function is simple

BLAST matrix

Transition-transversion matrix: Give mild penalty for replacing purine (A, G) by purine. Similar for replacing pyrimidine (C, T) by pyrimidine

	A	C	G	T
A	5	-4	-4	-4
C	-4	5	-4	-4
G	-4	-4	5	-4
T	-4	-4	-4	5

BLAST Matrix

	A	C	G	T
A	1	-5	-1	-5
C	-5	1	-5	-1
G	-1	-5	1	-5
T	-5	-1	-5	1

Transition-Transversion Matrix

Scoring function for protein

Commonly, it is devised based on two criteria:

Chemical/physical similarity

Observed substitution frequencies

Scoring function for protein using physical/chemical properties

An amino acid is more likely to be substituted by another if they have similar property [Karlin & Ghandour, PNAS, 82:8597, 1985]

The score matrices can be derived based on hydrophobicity, charge, etc.

E.g., give higher score for substituting nonpolar amino acid to another nonpolar amino acid

Scoring function for protein based on statistical model

Two popular matrices:

Point Accepted Mutation (PAM) matrix

BLOSUM

Both methods define the score as the log-odds ratio between the observed substitution rate and the expected substitution rate

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

Point Accepted Mutation (PAM)

PAM was developed by Dayhoff (1978)

A point mutation means substituting one residue by another

It is called an accepted point mutation if the mutation does not change the protein's function or is not fatal

Two sequence S_1 and S_2 are said to be 1 PAM diverged if a series of accepted point mutations can convert S_1 to S_2 with an average of 1 accepted point mutation per 100 residues

PAM-1 matrix, scaled by 10

$$M(a,b) = 10 \times \log_{10}\left(\frac{O(a,b)}{E(a,b)}\right)$$

Symbol	Meaning
$F(a,b)$	Count of substitutions between residues a and b, counting both directions (i.e. $F(a,b) = F(b,a)$).
$S = \sum_{x<y} F(x,y)$	Total number of distinct substitution pairs.
f_a	Background frequency of residue a (fraction of all residues that are a).
$O(a,b)$	Observed joint frequency of aligned residues a, b after 1 PAM of evolution.
$E(a,b)$	Expected joint frequency under random pairing (independent draws).

$$O(a,b) = \begin{cases} \frac{F(a,b)}{200 \sum_{x<y} F(x,y)}, & a \neq b, \\ f_a - \sum_{b \neq a} O(a,b), & a = b, \end{cases}$$

$$E(a,b) = f_a f_b,$$

$$M(a,b) = 10 \log_{10}\left(\frac{O(a,b)}{E(a,b)}\right).$$

$$\sum_a O(a,a) = 0.99, \quad \sum_{a \neq b} O(a,b) = 0.01.$$

Example

Ungapped alignment is constructed for high similarity amino acid sequences (usually >85%)

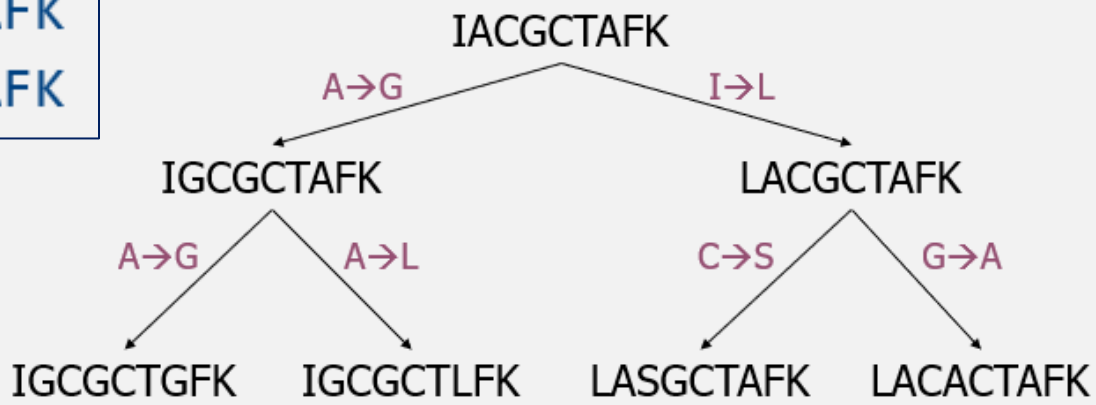
Below is a simplified gap-free global multiple alignment of some highly similar amino acid seqs

```
IACGCTAFK
IGCGCTAFK
LACGCTAFK
IGCGCTGFK
IGCGCTLFK
LASGCTAFK
LACACTAFK
```

Example, cont'd

IACGCTAFK
IGCGCTAFK
LACGCTAFK
IGCGCTGFK
IGCGCTLFK
LASGCTAFK
LACACTAFK

Build the phylogenetic tree for the sequences



$F_{A,G} = 3$, $F_{A,L}=1$

$f_A = f_G = 10/63$

$O_{A,G} = 3/(200 * 6) = 0.0025$

$E_{A,G} = (10/63)(10/63) = 0.0252$

$\delta(A,G) = \log (0.0025 / 0.0252)$
$= \log (0.09925) = -1.0034$

$M(A,G) = 10 * \delta(A,G) = -10$

Exercise

What does a PAM-1 score of -10 mean?

What PAM-2 means

PAM-1 models 1% accepted mutations per site

PAM-2 means evolution that is twice as long, i.e. 2% accepted mutations per site on average

It is *not* simply doubling the PAM-1 matrix entries — because mutation probabilities compound

PAM-1 to PAM-n

$$
\begin{aligned}
P_1(a,b) &= \frac{O(a,b)}{f_a}, \\
P_n &= P_1^n, \\
O_n(a,b) &= f_a P_n(a,b), \\
E(a,b) &= f_a f_b, \\
M_n(a,b) &= 10 \log_{10}\left(\frac{O_n(a,b)}{E(a,b)}\right).
\end{aligned}
$$

E.g., PAM-2: $P_2 = P^2$

Each entry $P_2(a,b)$ gives the probability that starting as "a", you end up as "b" after two independent PAM-1 intervals of mutation

Let P be the **Markov substitution matrix** derived from PAM-1:

$$P(a,b) = \frac{O(a,b)}{f_a}.$$

Here:

- each row a sums to 1 (probabilities of ending as residue b given starting at a);
- $P(a,b)$ is the probability of $a \to b$ after 1 PAM unit of evolution.

Then the matrix for **n PAM units** is:

$$P^n = \underbrace{P \times P \times \cdots \times P}_{n \text{ times}}.$$

After computing $P_n = P^n$, the **observed joint frequencies** are:

$$O_n(a,b) = f_a \, P_n(a,b),$$

and the **expected frequencies** remain $E(a,b) = f_a f_b$.

Then the **PAM-n scoring matrix** is:

$$M_n(a,b) = 10 \log_{10}\left(\frac{O_n(a,b)}{E(a,b)}\right).$$

BLOSUM (BLOck SUbstition Matrix)

PAM did not work well for aligning evolutionarily divergent sequences since the matrix is generated by extrapolation

Henikoff and Henikoff (1992) proposed BLOSUM

Unlike PAM, BLOSUM matrix is constructed directly from the observed alignment (instead of extrapolation)

Generate conserved nonredundant blocks

In BLOSUM, the input is a set of multiple alignments for nonredundant groups of protein families

Based on PROTOMAT, blocks of nongapped local alignments are derived

Each block represents a conserved region of a protein family

1. Start with conserved blocks

- Take a database of **protein sequences**.
- Identify **conserved regions (blocks)**: short ungapped alignments of related proteins. These are regions with strong evolutionary conservation.
- Example: A block could be a segment of 10–20 residues that aligns well across many proteins of a family.

---

2. Cluster sequences to reduce bias

- Some proteins are highly similar; including all of them would **bias the statistics**.
- Sequences with similarity above a threshold $x\%$ are clustered together. Each cluster is treated as **one sequence**.
- For example, in **BLOSUM62**, sequences more than 62% identical are clustered.

Compute probabilities

3. Count amino acid pairs

- For each column in a block, count how often each amino acid pair occurs, including identical and substitution pairs.
- Let f_{ab} be the frequency of observing amino acids a and b together.

---

4. Compute probabilities

- Compute the **observed probability** of a pair:

$$p_{ab} = \frac{f_{ab}}{\text{total pairs counted}}$$

- Compute the **expected probability** if amino acids were independent:

$$q_a = \sum_b p_{ab}, \quad e_{ab} = q_a q_b \quad (\text{or } 2q_a q_b \text{ if } a \neq b)$$

Produce BLOSUM matrix

BLOSUM numbers indicate clustering threshold

BLOSUM62 – sequences > 62% identical are clustered

BLOSUM80 – stricter, for closely related proteins

BLOSUM45 – looser, for more distance sequences

5. Convert to log-odds scores

- The BLOSUM score is a **log-odds ratio**, telling you whether a substitution occurs more or less often than expected by chance:

$$S(a,b) = \log_2 \frac{p_{ab}}{e_{ab}}$$

- If $S(a,b) > 0$: the substitution is **more likely than random**.
- If $S(a,b) < 0$: the substitution is **less likely than random**.

---

6. Scale to integer scores

- Multiply the log-odds by a factor (often 2 or 3) and round to integer to make a practical scoring matrix.
- The result is the **BLOSUM matrix** you see in tools like BLAST.

Example

```
ACGCTAFKI
GCGCTAFKI
ACGCTAFKL
GCGCTGFKI
GCGCTLFKI
ASGCTAFKL
ACACTAFKL
```

1. Single Residue Frequencies (f_a): The Background

The single residue frequency (f_a) is the observed count of residue a divided by the total number of residues (N_{res}).

Residue	Count (C_a)	Frequency (f_a)
Alanine (A)	11	$f_A = \frac{11}{63}$
Glycine (G)	9	$f_G = \frac{9}{63}$

2. Observed Pair Frequency (p_{ab}): The Observed Alignment

The observed pair frequency (p_{AG}) is the count of the (A, G) pair (C_{AG}) divided by the total number of aligned pairs (N_{pairs}). This is the **observed data** from your alignment.

- **Count of (A, G) Pairs (C_{AG}):** 27

$$(4 \times 3) + (2 \times 5) + (5 \times 1) = 12 + 10 + 5 = 27$$

- **Observed Pair Frequency (p_{AG}):**

$$p_{AG} = \frac{C_{AG}}{N_{pairs}} = \frac{27}{189} = \mathbf{\frac{1}{7}}$$

Example, cont’d

ACGCTAFKI
GCGCTAFKI
ACGCTAFKL
GCGCTGFKI
GCGCTLFKI
ASGCTAFKL
ACACTAFKL

A. Expected Frequency (e_{AG})

$$e_{AG} = 2 \times f_A \times f_G = 2 \times \frac{11}{63} \times \frac{9}{63} = \mathbf{\frac{22}{441}}$$

B. Odds Ratio ($\frac{p_{AG}}{e_{AG}}$)

$$\text{Odds Ratio} = \frac{p_{AG}}{e_{AG}} = \frac{1/7}{22/441} = \frac{1}{7} \times \frac{441}{22} = \frac{63}{22} \approx 2.8636$$

C. Final Log-Odds Ratio (Unscaled Score)

$$\text{Log-Odds} = \log_2(\text{Odds Ratio}) = \log_2\left(\frac{63}{22}\right) \approx \mathbf{1.5178}$$

Exercise

Given that a BLOSUM p matrix is created by merging seqs with ≥p% similarity, how are p_{ab}, e_{ab}, etc. calculated?

Work it out using this example:

AVAAA, AVAAA, AVAAA, AVLAA, VVAAL

First 4 seqs have ≥80% similarity

Similarity of last seq with the other 4 sequences is <62%

BLOSUM vs PAM

BLOSUM 80 ≈ PAM 1

BLOSUM 62 ≈ PAM 120

BLOSUM 45 ≈ PAM 250

BLOSUM 62 is the default matrix for BLAST 2.0

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

Acknowledgements

Some slides on Needleman-Wunsch, Smith-Waterman, and scoring functions are based on those given to me by Ken Sung and Somayyeh Koohi

Good to read

S. B. Needleman, C. D. Wunsch. “A general method applicable to the search for similarities in the amino acid sequence of two proteins”, *JMB*, 48:444-453, 1970

T. F. Smith, M. S. Waterman. “Identification of common molecular subsequences”, *JMB*, 147:195-197, 1981

E. Ukkonen. “Algorithms for approximate string matching*”, Information & Control*, 61(1-3):100-118, 1985

D. S. Hirschberg. “A linear space algorithm for computing maximal common subsequences”, *CACM*, 18(6):341-343, 1975

E. W. Myers. “Optimal alignments in linear space”, *Bioinformatics*, 4(1):11-17, 1988

O. Gotoh. “An improved algorithm for matching biological sequences”, JMB, 162(3):705-708, 1982

Good to read

M. O. Dayhoff, R. M. Schwartz, B. C. Orcutt. “A model of evolutionary change in proteins”. In M. O. Dayhoff (ed), *Atlas of Protein Sequence and Structure*, 5(suppl 3):345-352, 1978

S. Henikoff, J.Henikoff, “Amino acid substitution matrices from protein blocks”. *PNAS*, 89(biochemistry):10915-10919, 1992