Pairwise Sequence Alignment (I)

Jianlin Cheng, PhD

School of Electrical Engineering and Computer Science
University of Central Florida

2006

Sequence Data

- **DNA Sequence (gene)**

  gaagagagcttcaggtttggggaagagaca
  acaactcccgctcagaagcaggggccgata
- **RNA Sequence**
  caaaucacuacacacaggguagaagguggaacgcacaggagcaugu
  caacggggugc
- **Protein Sequence**
  RELQVWGRDNNSLSEAGANRQGDVSFNLPQIT
  LWQRPLVTIKIGGQLKEALLDTGADDTVLEDID
  LPGKWKP

Fundamental Problems

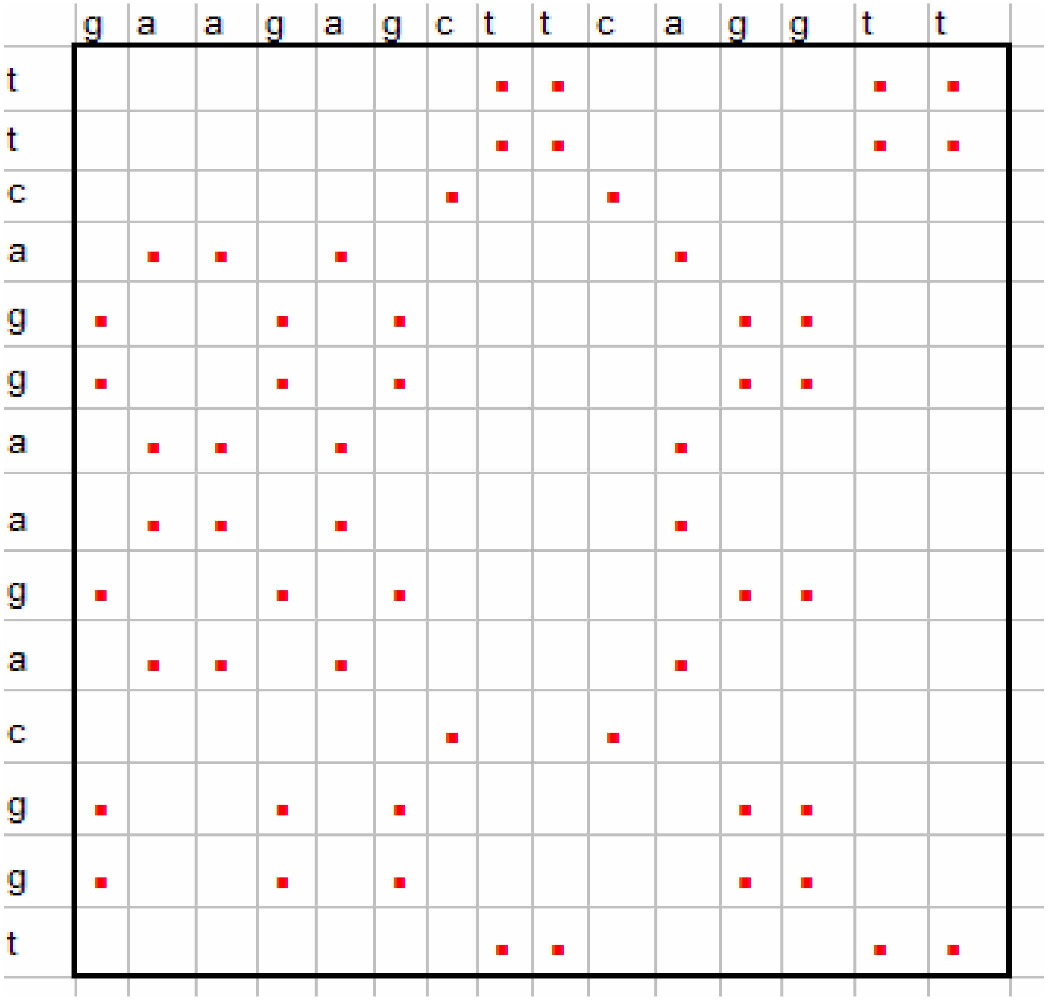
- Why do we compare sequences?
- What’s similarity between two sequences?
- What methods should be used to compare sequences?
- How similar are two sequences are?
- Is similarity significant?

Importance of Similarity Comparison

- Identify evolutionary relationship between genes and proteins
- Similar genes/proteins have similar functions
- Similar proteins have similar structures
- Classification and Clustering

Similarity comparison of biological sequences is the most fundamental tool of Bioinformatics. It is used in almost all Bioinformatics areas and is still an active research area after more than 50-year research.

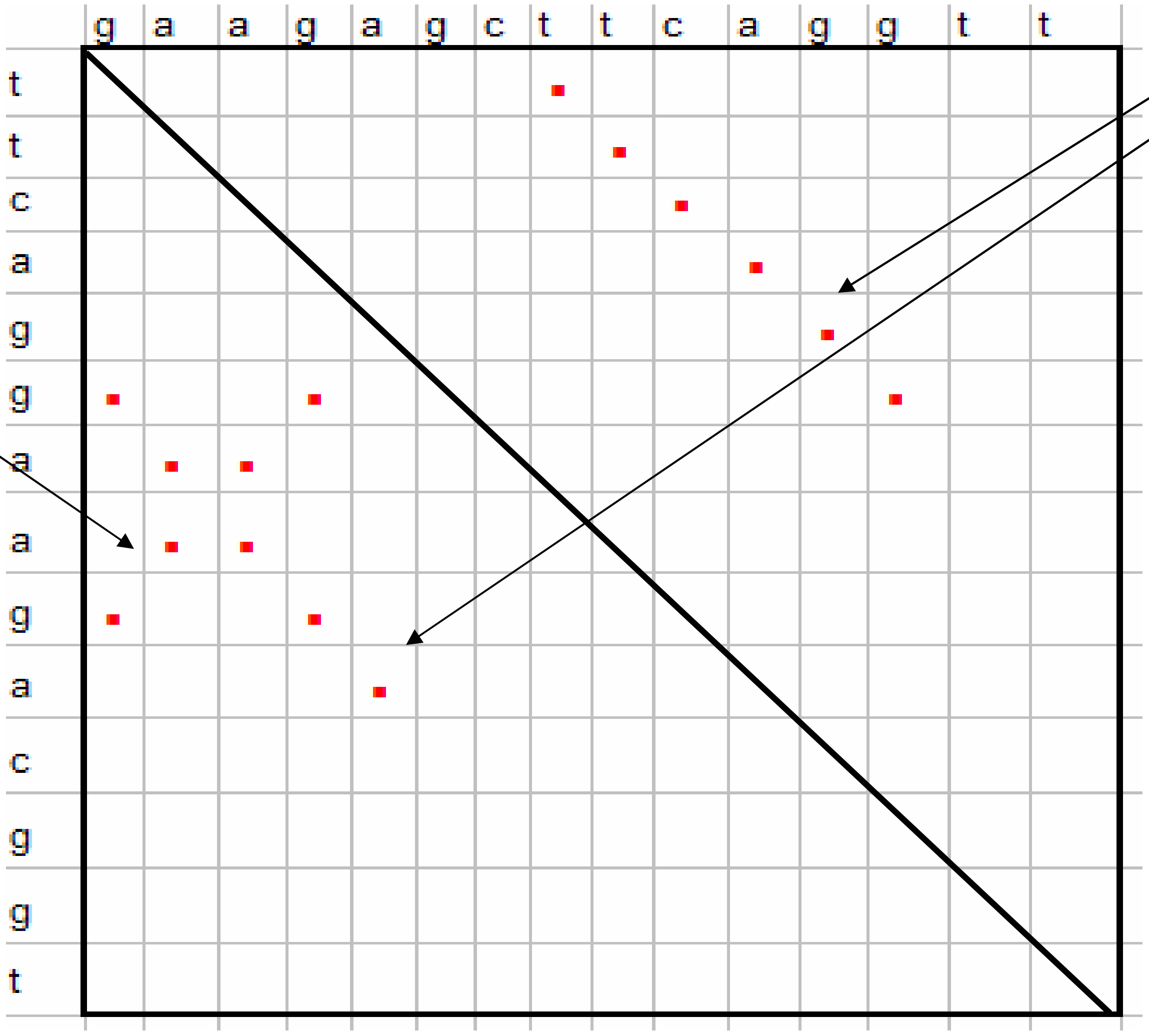
Dot Matrix Approach



	g	a	a	g	a	g	c	t	t	c	a	g	g	t	t
t								●	●					●	●
t								●	●					●	●
c							●			●					
a		●	●		●						●				
g	●			●		●						●	●		
g	●			●		●						●	●		
a		●	●		●						●				
a		●	●		●						●				
g	●			●		●						●	●		
a		●	●		●						●				
c							●			●					
g	●			●		●						●	●		
g	●			●		●						●	●		
t								●	●					●	●

Similarity: same sub-strings / repeats shared by two sequences

Filter out short segments



repeat

Reverse repeat

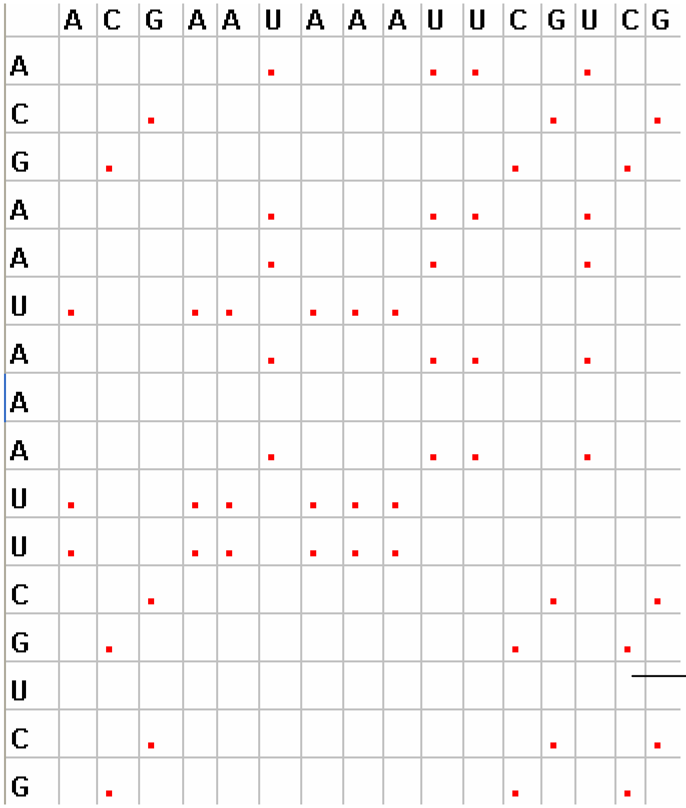
Threshold = 4

Another: window Smoothing approach

Applications

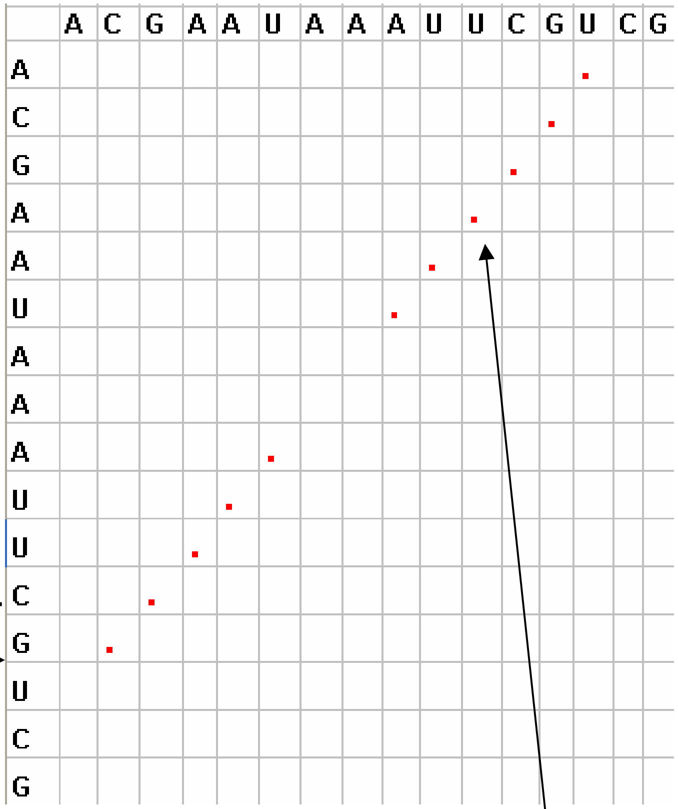
- Find repeats in two sequences
- Find reverse repeats in two sequences
- Find repeats/reverse repeats in the same sequences
- Find complementary repeats / reverse repeats in the same sequences in RNA (base pairs)

Reverse Complementary Repeats

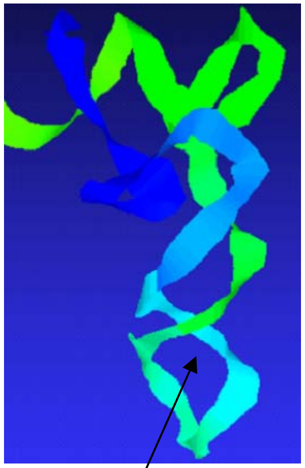


	A	C	G	A	A	U	A	A	A	U	U	C	G	U	C	G
A						●				●	●			●		
C			●										●			●
G		●										●			●	
A						●				●	●			●		
A						●				●				●		
U	●			●	●		●	●	●							
A						●				●	●			●		
A																
A						●				●	●			●		
U	●			●	●		●	●	●							
U	●			●	●		●	●	●							
C			●										●			●
G		●										●			●	
U																
C			●										●			●
G		●										●			●	

Filter



	A	C	G	A	A	U	A	A	A	U	U	C	G	U	C	G
A														●		
C													●			
G												●				
A											●					
A										●						
U									●							
A																
A																
A						●										
U					●											
U				●												
C			●													
G		●														
U																
C																
G																



Complementary Base Pairs

Complexity, Tools, and Limitation

- Time complexity for two sequences with length m and n (fill out matrix: m*n, thresholding by diagonal: at most m*n)
- Dotmatcher: http://bioweb.pasteur.fr/seqanal/interfaces/dotmatcher.html
- Limitation: no gaps allowed, not mismatch, not optimal.

Example:

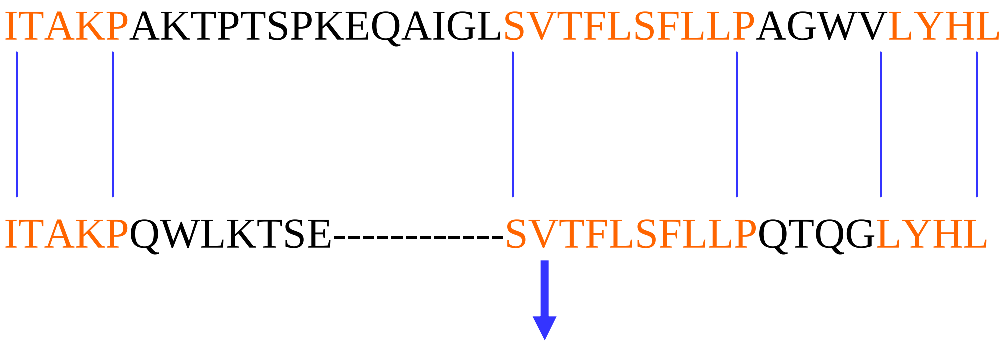
aaggtccttagga
aaggccttagga

→

aaggtccttagga
aagg ccttagga

Global Pairwise Sequence Alignment

(Needleman and Wunsch, 1970)


ITAKPAKTPTSPKEQAIGLSVTFLSFLLPAGWVLYHL

ITAKPQWLKTSE------------SVTFLSFLLPQTQGLYHL

Alignment (similarity) score

Optimization Problem:
Align two sequences into the same length by adding gaps if necessary and maximizing alignment scores.

Three Main Issues

1. Definition of alignment score
2. Algorithms of finding the optimal alignment
3. Evaluation of significance of alignment score

A simple scoring scheme

- Score of character pair: S(match)=1, S(not_match) = -1, S(gap-char) = -1
- Score of an alignment = $\sum_{1}^{n} S_i$
- Discuss more advanced scoring scheme later.

```
ITAKPAKTPTSPKEQAIGLSVTFLSFLLPAGWVLYHL

ITAKPQWLKTSE-------SVTFLSFLLPQTQGLYHL
```

5 + 10 + 4 – 7 – 7 -4 = 1

Optimization

- How can we find the best alignment to maximize alignment score?
- Time complexity of brute force enumeration. How many possible alignments exist for two sequences with length m and n?

Shortest Alignments

Sequence 1: AGATCAGAAATGG

Sequence 2: ATAGAATCC

Shortest Alignment

Sequence 1: AGATCAGAAATGG
Sequence 2: ATAGAATCC

```
AGATCAGAAATGG
ATAGAATCC----
```

Length = max (m,n)

Longest Alignment

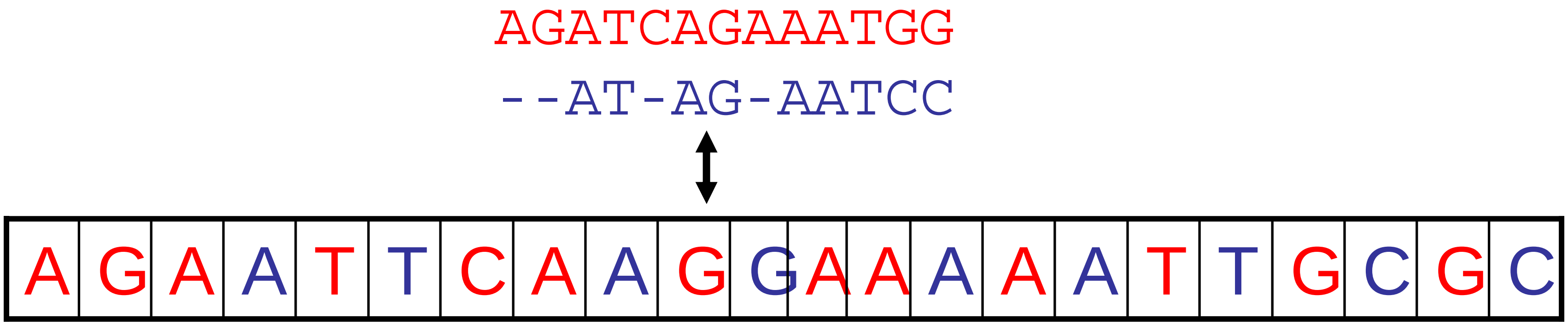
```
AGATCAGAAATGG---------
-------------ATAGAATCC
```

Length = m + n

Total Number of Possible Alignments

Hard combinatorial problem

Use another way to represent all possible alignments



AGATCAGAAATGG

--AT-AG-AATCC

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

Convert one alignment of two sequences into one linear string (1-1 correspondence)

Consider there are m+n positions. Each character in each sequence takes one position (the characters in the same sequence must be in the sequence order.

Total number of alignments

Select m positions out of *m*+*n* possible positions:

$$\binom{m+n}{m} = \frac{(m+n)!}{m!n!}$$

Exponential!

If m = 300, n = 300, total = 10^37

Divide and Conquer

Goal: align prefix P[1..i] and prefix Q[1..j]

```
                i
Seq P: AGATCAGAAATGG
Seq Q: ATAGAATCC
          j
```

Three possibilities assuming we know the optimal alignment of smaller prefixes:

Case 1

Use alignment of P[1..i-1] and Q[1..j-1], pair P[i] and Q[j]

```
      i
AGATCAG
--AT-AG
      j
```

Case 2

Use alignment of P[1..i] And Q[1..j-1], pair Q[j] with gap

```
      i
AGATCAG-
--AT-A-G
       j
```

Case 3

Use alignment of P[1..i-1] and Q[1..j], pair P[i] with gap.

```
       i
AGATCA-G
--AT-AG-
      j
```

Needleman and Wunsch Algorithm

- Given sequences P and Q, we use a matrix M to record the optimal alignment scores of all prefixes of P and Q. M[i,j] is the best alignment score for the prefixes P[1..i] and Q[1..j].
- **M[i,j] =**

```
**max [**
        M[i-1,j-1] + S(P[i],Q[j]),
        M[i,j-1] + S(-, Q[j])
        M[i-1,j] + S(P[i], -)
      ]
```

Comments:

We know M[0,0], M[i,0], and M[0,j], where i<=m, j <=n,
m and n are sequence lengths.
Start from lower index and run to the end of sequences
Global optimum = combination of local optimum

Dynamic Programming

Dynamic Programming Algorithm

Three-Step Algorithm:

- Initialization
- Matrix fill (scoring)
- Trace back (alignment)

1. Initialization of Matrix M

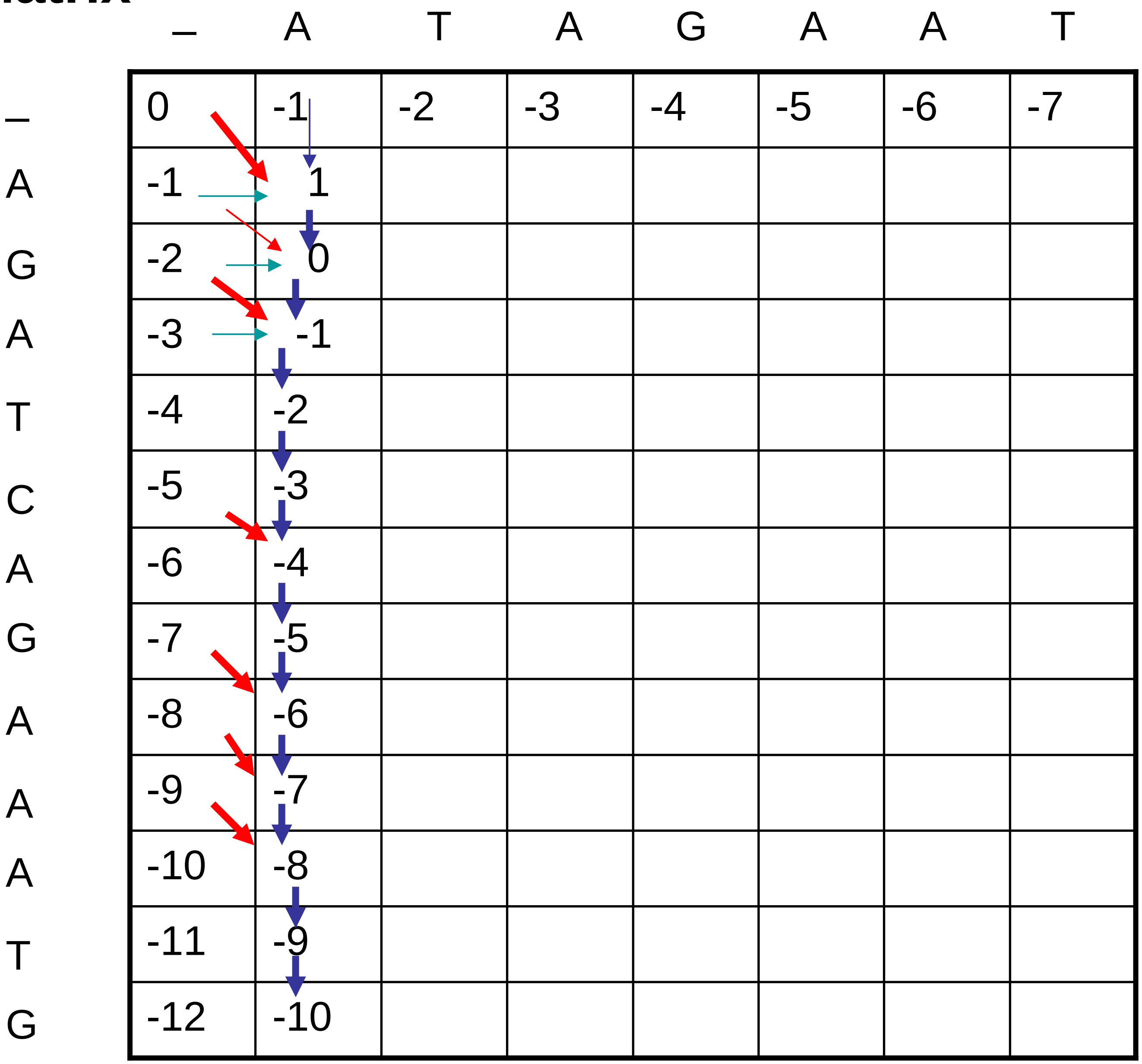
j

i

	_	A	T	A	G	A	A	T
_	0	-1	-2	-3	-4	-5	-6	-7
A	-1							
G	-2							
A	-3							
T	-4							
C	-5							
A	-6							
G	-7							
A	-8							
A	-9							
A	-10							
T	-11							
G	-12							

M is a (m+1) * (n+1) matrix. M[i,j] is to record the best alignment score of P[1..i] and Q[1..j]

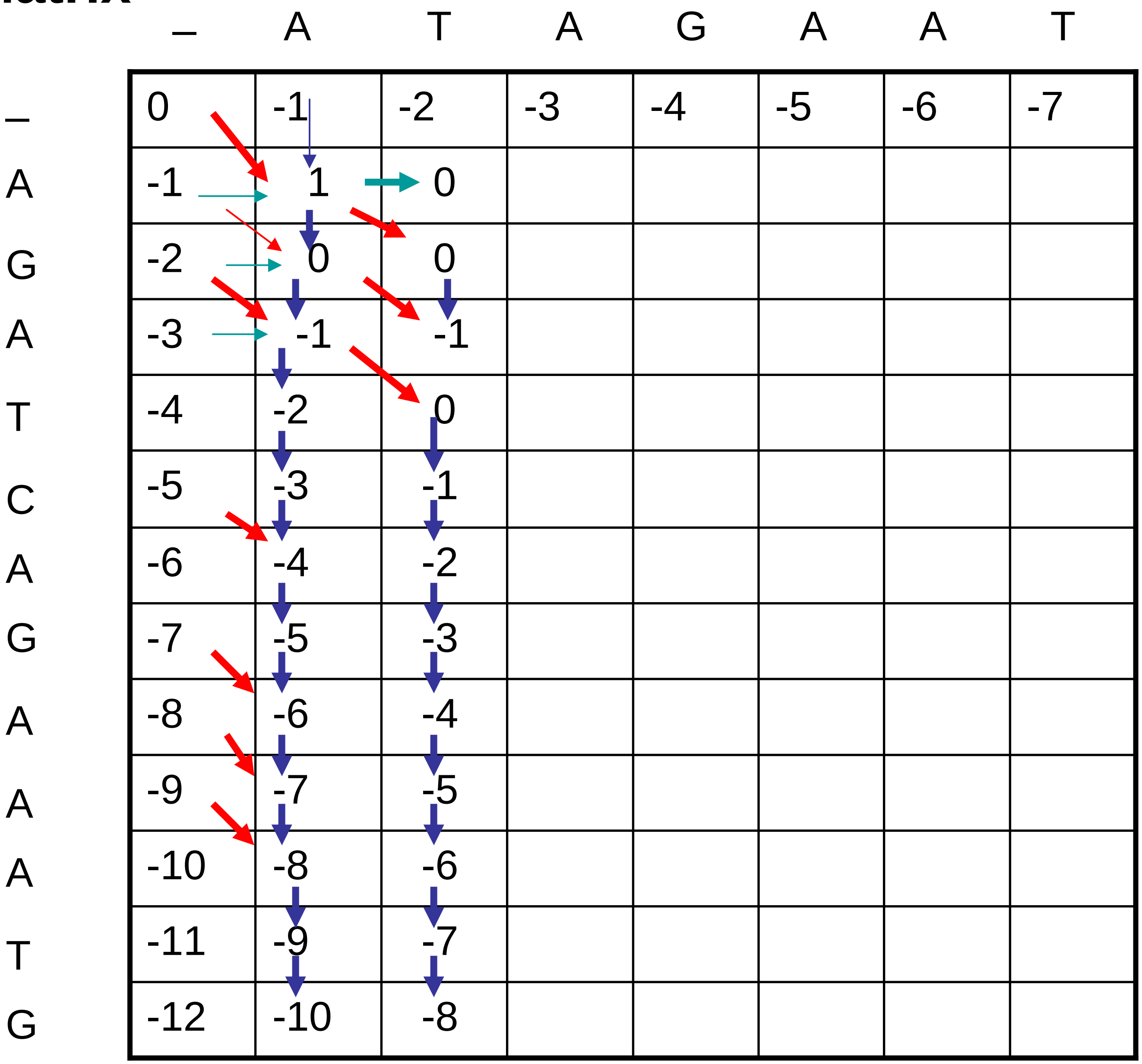
2. Fill Matrix



	_	A	T	A	G	A	A	T
_	0	-1	-2	-3	-4	-5	-6	-7
A	-1	1						
G	-2	0						
A	-3	-1						
T	-4	-2						
C	-5	-3						
A	-6	-4						
G	-7	-5						
A	-8	-6						
A	-9	-7						
A	-10	-8						
T	-11	-9						
G	-12	-10						

M[i,j] =
max [
M[i-1,j-1] + S(P[i],Q[j]),
M[i,j-1] + S(-, Q[j])
M[i-1,j] + S(P[i], -)
]

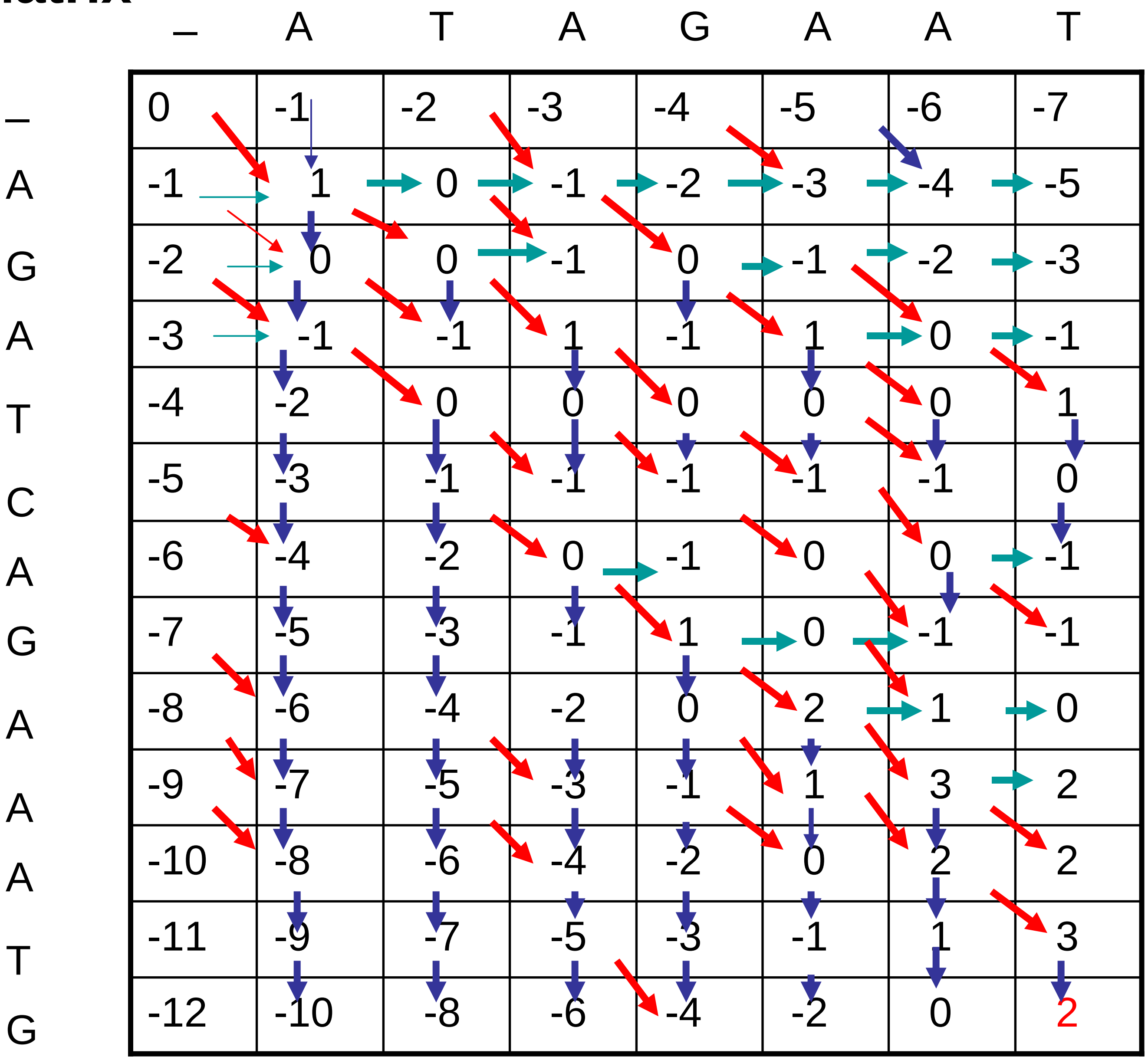
2. Fill Matrix



	_	A	T	A	G	A	A	T
_	0	-1	-2	-3	-4	-5	-6	-7
A	-1	1	0					
G	-2	0	0					
A	-3	-1	-1					
T	-4	-2	0					
C	-5	-3	-1					
A	-6	-4	-2					
G	-7	-5	-3					
A	-8	-6	-4					
A	-9	-7	-5					
A	-10	-8	-6					
T	-11	-9	-7					
G	-12	-10	-8					

M[i,j] =
max [
M[i-1,j-1] + S(P[i],Q[j]),
M[i,j-1] + S(-, Q[j])
M[i-1,j] + S(P[i], -)
]

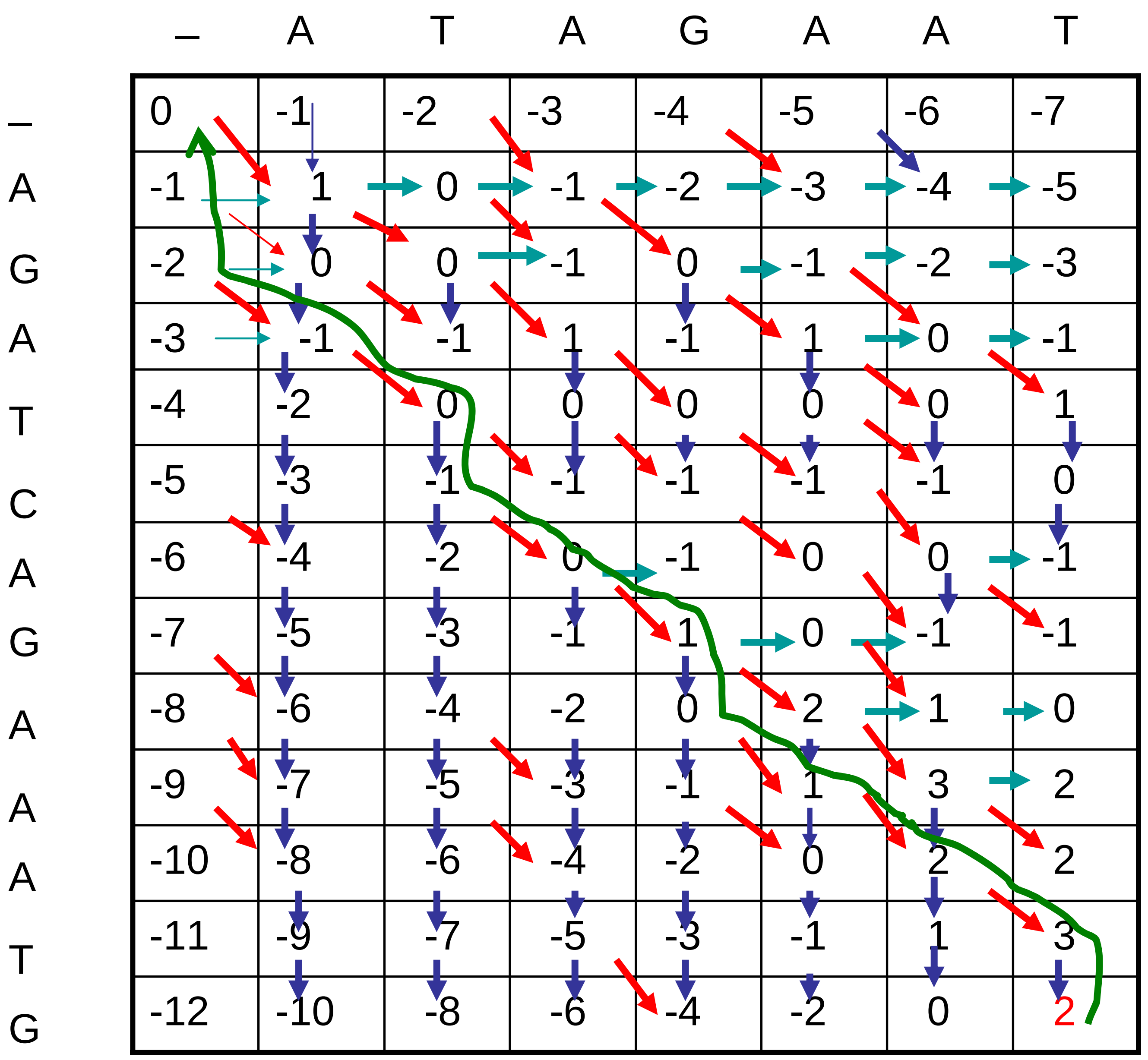
2. Fill Matrix



	_	A	T	A	G	A	A	T
_	0	-1	-2	-3	-4	-5	-6	-7
A	-1	1	0	-1	-2	-3	-4	-5
G	-2	0	0	-1	0	-1	-2	-3
A	-3	-1	-1	1	-1	1	0	-1
T	-4	-2	0	0	0	0	0	1
C	-5	-3	-1	-1	-1	-1	-1	0
A	-6	-4	-2	0	-1	0	0	-1
G	-7	-5	-3	-1	1	0	-1	-1
A	-8	-6	-4	-2	0	2	1	0
A	-9	-7	-5	-3	-1	1	3	2
A	-10	-8	-6	-4	-2	0	2	2
T	-11	-9	-7	-5	-3	-1	1	3
G	-12	-10	-8	-6	-4	-2	0	2

M[i,j] =
max [
M[i-1,j-1] + S(P[i],Q[j]),
M[i,j-1] + S(-, Q[j])
M[i-1,j] + S(P[i], -)
]

3. Trace Back



	_	A	T	A	G	A	A	T
_	0	-1	-2	-3	-4	-5	-6	-7
A	-1	1	0	-1	-2	-3	-4	-5
G	-2	0	0	-1	0	-1	-2	-3
A	-3	-1	-1	1	-1	1	0	-1
T	-4	-2	0	0	0	0	0	1
C	-5	-3	-1	-1	-1	-1	-1	0
A	-6	-4	-2	0	-1	0	0	-1
G	-7	-5	-3	-1	1	0	-1	-1
A	-8	-6	-4	-2	0	2	1	0
A	-9	-7	-5	-3	-1	1	3	2
A	-10	-8	-6	-4	-2	0	2	2
T	-11	-9	-7	-5	-3	-1	1	3
G	-12	-10	-8	-6	-4	-2	0	2

```
AGATCAGAAATG

--AT-AG-AAT-
```

Insights

- Each path from top left to right bottom corresponds to an alignment. (how many paths? Which path is the shortest alignment? Longest alignment?
- More than one optimal alignment
- Global alignment
- Biological meaning of a gap: deletion or insertion (another view of DP: minimize editing distance)

More Insights

- Time complexity and space complexity is O(m*n). Why?
- Can we run DP from right to left? What optimal alignment score do we get?