Lecture Notes for INF281 Basics of Bioinformatics Sequence Analysis

Takaya Saito

Contents

Part I

1 Introduction

1.1 Introduction to Molecular Biology

Molecular biology is the study of biology focusing on organisms and cells at the molecular level.

Five essential facts about cells

1. Two primary types of cells - eukaryotes and prokaryotes

- Eukaryote: animals & plants
- Prokaryote: bacteria & archaea

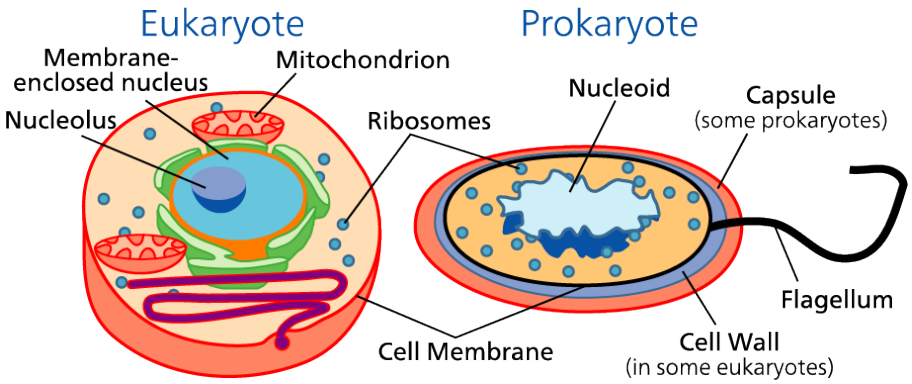


Figure 1.1: Eukaryotic and prokaryotic cells (source: Science Primer, Wikimedia Commons)

2. Cell size - around 1 to 100 micrometers

- Cell Size and Scale: `http://learn.genetics.utah.edu/content/cells/scale`

3. The number of cells

- Prokaryotes: 1 cell
- Human: Estimate of 15 trillion cells

4. An animal cell and cell organelles

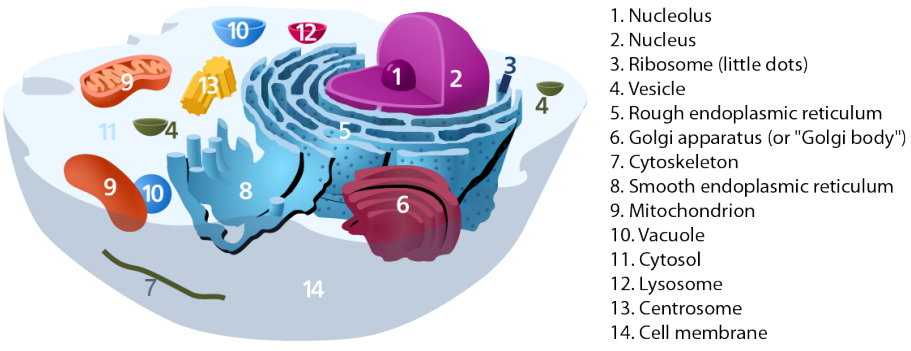


Figure 1.2: An animal cell and organelles (source: Kelvinsong, Wikimedia Commons)

5. Cellular processes

- Cell growth, cell development, cell signaling,
- Example: `http://www.nature.com/nrg/multimedia/rnai`

Central dogma of molecular biology

It describes the information flow within a cell.

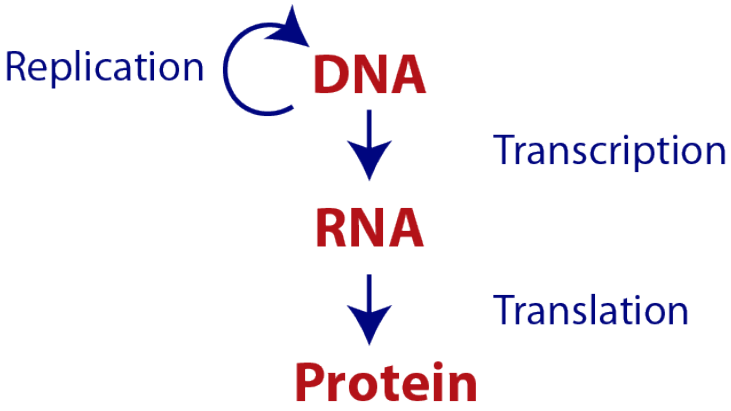


Figure 1.3: Central dogma of molecular biology

DNA (deoxyribonucleic acid)

DNA stores genetic information. It has four different bases: cytosine (C), guanine (G), adenine (A), and thymine (T).

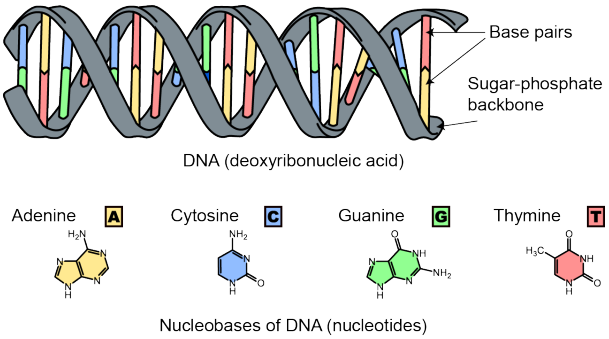


Figure 1.4: DNA double helix and base pairs
(modified from the original version by Sponk, Wikimedia Commons)

Base pair matching (Watson-Crick base pair)

Adenine (A) pairs with thymine (T), whereas cytosine (C) pairs with guanine (G).

```
DNA strand1: ACGT
             ||||
DNA strand2: TGCA
```

RNA (Ribonucleic acid)

RNA has various biological roles and several sub-classes. Messenger RNAs (mRNAs) convey genetic information. It has four different bases: cytosine (C), guanine (G), adenine (A), and uracil (U).

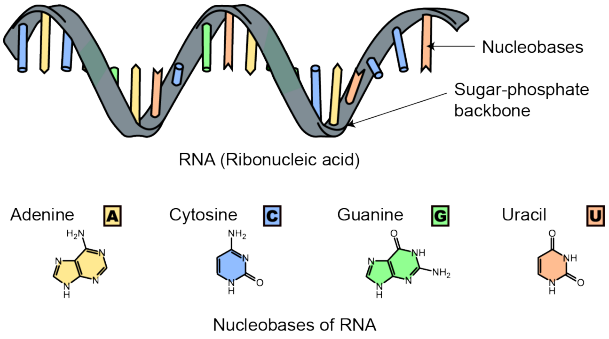


Figure 1.5: Single strand RNA
(modified from the original version by Sponk, Wikimedia Commons)

Transcription: mRNAs are transcribed from DNAs

```
DNA: ACGT -------> RNA: ACGU
        Transcription
```

Protein

Proteins are large molecules consisting of amino acids. There are 20 common amino acids.

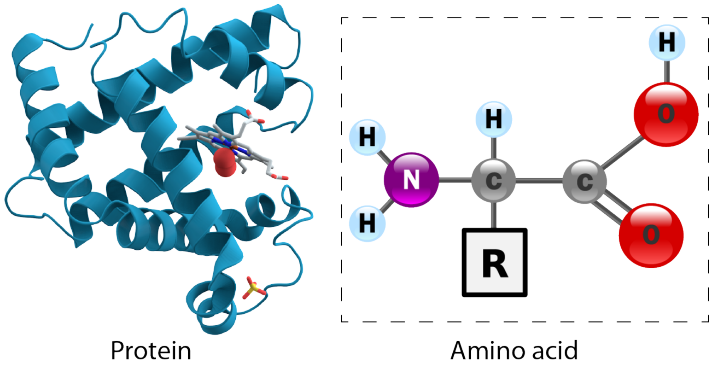


Figure 1.6: Protein 3D structure and amino acids
(sources: AzaToth, Wikimedia Commons, YassineMrabet, Wikimedia Commons)

Translation: Amino-acids are translated from mRNAs

```
mRNA: GUC -------> AA: Valine
        Translation
```

Universal genetic code

A codon consists of three nucleic acids. Single-letter or three-letter names can be used for amino acids.

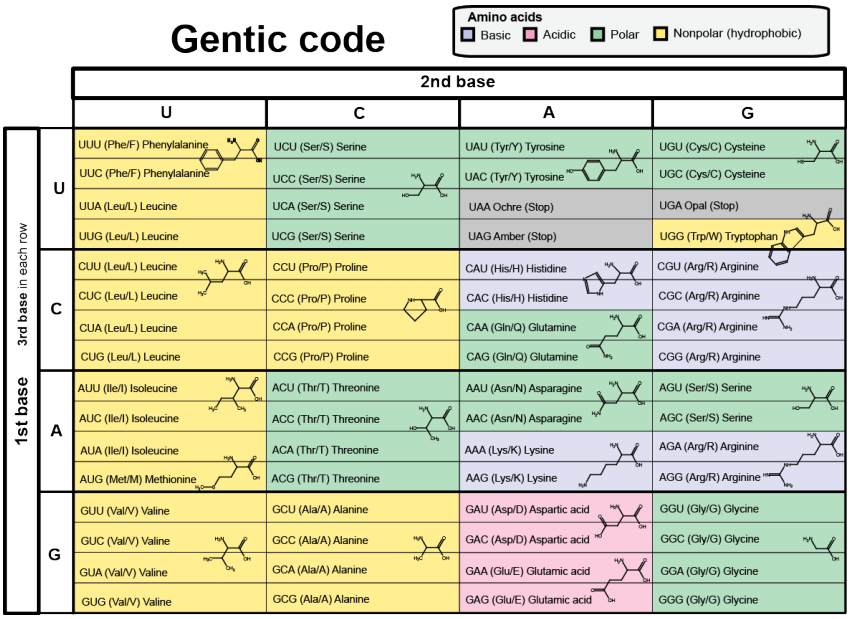


Figure 1.7: Universal genetic code
(modified from the original version by Häggström, Wikimedia Commons)

Cellular functions of proteins

- Enzymes: catalyze chemical reaction
- Cell signaling: hormone (e.g. insulin), antibodies,
- Structural: collagen, cartilage, keratin,

Exercises 1.1

1. Draw a simple diagram of the central dogma of molecular biology and briefly explain the information flow of the molecules.
2. What are the DNA sequences of the opposite strand for the following DNA sequences?

```
Seq1 CCGATT
Seq2 TTACGC
Seq3 ACGCGC
```

3. What are the mRNA sequences transcribed from the following DNA sequences?
4. What are the polypeptide sequences translated from the following mRNA sequences? Answer them with both one-letter and three letter names.

```
Seq1 AUGUUUUAA
Seq2 GCAGCAAAA
```

1.2 Introduction to Biotechnology

Biotechnology is the use of laboratory techniques to study living organism and cells.

Applications of biotechnology

Branches of biotechnology can be explained with different colors.

- Red: medical processes
- Green: agricultural processes
- White: industrial processes
- Blue: marine and aquatic applications

Laboratory tools and equipment

Figure 1.8: Pipette, centrifuge, thermal cycler, and DNA sequencer (sources: Domain, Manske, Rror, RE73 via Wikimedia Commons)

Human genome project

It was a large-scale international research project to determine the whole DNA sequences of human.

- 1990 - 2003
- $2.7 billion

Next generation sequencing

Sequence technologies have been rapidly advanced since the human genome project.
Example: sequence a whole human genome with Illumina HiSeq X Ten.

- One day
- $1000

Protein sequencing

Proteins are generally more studied than DNAs and RNAs, but the whole proteome is generally harder to analyze than the whole genome. MS (mass-spectrometry) based technologies are widely used to sequence proteins.

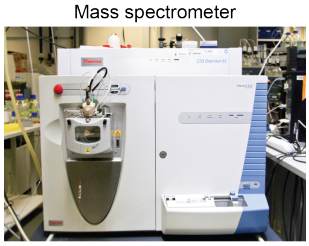


Figure 1.9: Orbitrap mass spectrometer (source: Wiòrkiewicz, Wikimedia Commons)

1.3 Bioinformatics

Bioinformatics uses computational approaches to solve problems in life sciences. It is based on computer science.

Similar or almost equivalent disciplines

- Biostatistics
- Biophysics
- Systems biology
- Computational biology

Not much related with bioinformatics

- Health informatics
- Forensic science

Scope of INF281

We mainly cover the following fields of bioinformatics in this course.

- Pairwise alignment
- Database search
- Statistical evaluation
- Multiple alignment
- Phylogenetic tree
- Scoring scheme
- Sequence patterns

Popular bioinformatics programs

BLAST and ClustalW are popular tools for sequence analysis.

- BLAST: a program for database search

  URL: `http://blast.ncbi.nlm.nih.gov`

- ClustalW: a program for multiple alignments

  URL: `http://www.ch.embnet.org/software/ClustalW.html`

Rank	Title	Times cited
1	Protein measurement with the folin phenol reagent	305148
2	Cleavage of structural proteins during the assembly of the head of bacteriophage T4	213005
3	A rapid and sensitive method for the quantitation of microgram quantities of protein utilizing the principle of protein-dye binding	155530
4	DNA sequencing with chain-terminating inhibitors	65335
5	Single-step method of RNA isolation by acid guanidinium thiocyanate-phenol-chloroform extraction	60397
6	Electrophoretic transfer of proteins from polyacrylamide gels to nitrocellulose sheets: procedure and some applications	53349
7	Development of the Colle-Salvetti correlation-energy formula into a functional of the electron density	46702
8	Density-functional thermochemistry. III. The role of exact exchange	46145
9	A simple method for the isolation and purification of total lipides from animal tissues	45131
10	**Clustal W**: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice	40289
11	Nonparametric estimation from incomplete observations	38600
12	**Basic local alignment search tool**	38380
13	A short history of SHELX	37978
14	**Gapped BLAST and PSI-BLAST**: A new generation of protein database search programs	36410
15	A revised medium for rapid growth and bio assays with tobacco tissue cultures	36132

Table 1.1: The 15 most cited papers of all time
(The top 100 papers, Van Noorden, Maher, and Nuzzo, *Nature*, 2014)

Part II

2 Global pairwise alignment

2.1 Pairwise alignment

A pairwise alignment is a basic sequence structure that consists of two sequences. A global alignment stretches to the whole part of two sequences, whereas a local alignment usually contains only part of the sequences.

Components of pairwise alignment

We name two sequences as database or d and query or q through this course. They may represent sequences from two different species or organisms.

Identical sequences.

```
q: ACGT
d: ACGT
```

One mismatch.

```
q: ACGT
d: ACGA
```

The '-' symbol represents a blank. A single or a set of multiple blanks further represents a gap, which is an indication of insertion or deletion in the course of evolution between two organisms.

```
q: ACGT
d: A-GT
```

N.B. A gap cannot be aligned with another gap.

Example of a simple scoring scheme

- Match: 1
- Mismatch: 0
- Gap penalty: 1 (use -1 for the actual calculation)

We may use the following notation.

- $R_{ab} = 1$ for a = b
- $R_{ab} = 0$ for a $\neq$ b
- g = 1

Exercise 2.1

Use the simple scoring scheme above and calculate the scores of the following two alignments.

```
Alignment 1
    q: GCA-GCA
    d: GA-TG-A
```

```
Alignment 2
    q: GCA-GCA
d: G-ATG-A
```

2.2 Alignment by brute–force

A brute–force approach finds the alignment with the highest score by simply considering all possible alignments and calculates the score for each of them.

An example of brute–force approach

We find the optimal alignment for the following sequences by using the scoring scheme below.

Sequences:

```
q: AG, d: ACG
```

Scoring scheme:
$R_{ab} = 1$ for a = b
$R_{ab} = 0$ for a $\neq$ b
g = 1

1. The length of alignment

- Maximum length: length(q) + length(d)
- Minimum length: max(length(q), length(d))

2. All possible alignments when length = 5

```
---AG    A---G    A--G-    AG---    --A-G
ACG--    -ACG-    -AC-G    --ACG    AC-G-

--AG-    -AG--    -A--G    -A-G-    A-G--
AC--G    A--CG    A-CG-    A-C-G    -A-CG
```

3. All possible alignments when length = 4

```
A--G    A-G-    AG--    A--G    -A-G    -AG-
ACG-    AC-G    A-CG    -ACG    ACG-    AC-G

-AG-    A-G-    --AG    --AG    -A-G    AG--
A-CG    -ACG    ACG-    AC-G    A-CG    -ACG
```

4. All possible alignments when length = 3

```
-AG    A-G    AG-
ACG    ACG    ACG
```

5. Alignment with the best score

```
ACG
A-G
```

Score: 1

Search space size of the brute-force approach

The search space size is the number of all possible alignments. It is 25 (10 + 12 + 3) for the example above.

Rapid growth of search space size

```
Example 1
    q: ACGACG, d: AGAG
```

Search space size: 1289

```
Example 2
    q: ACGACGACGACG, d: AGAGAGAG
```

Search space size: 4,673,345

Exercise 2.2

Find the alignment with the best score for the sequences. Use the simple scoring scheme below.

Sequences:

```
q: A, d: AC
```

Scoring scheme:

$R_{ab} = 1$ for a = b
$R_{ab} = 0$ for a $\neq$ b
g = 1

1. What are the maximum and minimum lengths of the alignment?
2. Identify all possible alignments.
3. What is the best score?
4. What is the search space size when the brute-force approach is used?

2.3 Table representation of alignment

Several data structures can be used to represent an alignment. The table representation is frequently used and also makes the process clear when we combine it with dynamic programming (DP) later.

Data structures and algorithms

It is important to consider the following aspects before solving computational problems.

1. Identify and analyze the problem you want to solve
2. Pick up an algorithm that can efficiently solve the problem
3. Decide a data structure that works with the algorithm of your choice

We use a table format (2D array) to solve global alignments by dynamic programming.

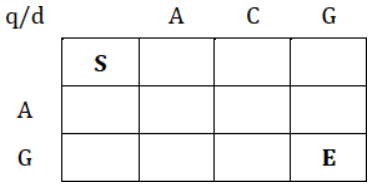
Example of table format

Alignment:

```
q: -AG-
d: A-CG
```

1. Initial setup

1. Make a table with the size of (1 + length(q)) by (1 + length(d))
2. Add the database sequence as column labels
3. And the query sequence as row labels

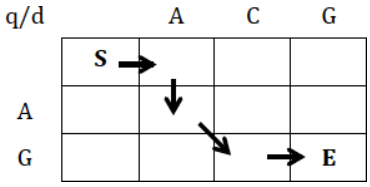


2. Add arrows

We use three types of arrows to form an alignment.

- Move diagonally: add the letters from q and d to the alignment
- Move vertically: add - and the letter from d to the alignment
- Move horizontally: add the letter from q and - to the alignment

It should start from S and stops at E.



Exercise 2.3

Find the corresponding alignments for Table 1, 2 and 3.

Table 1

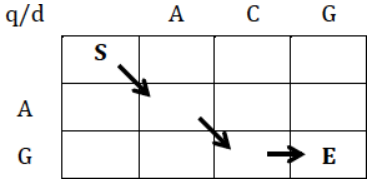


Table 2

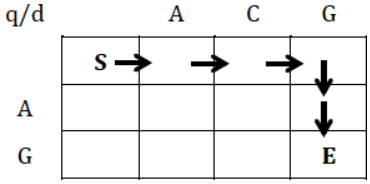
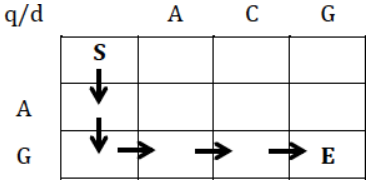


Table 3



2.4 Global alignment with DP

Dynamic programming (DP) provides a solution for a multi-stage decision process, in which larger decisions recursively nest smaller decisions.

Memorize the best score in a table cell

For global alignment, the core procedure of DP is updating a cell with the highest score from the three different scores calculated from its adjacent cells. DP ends when the entire table is updated.

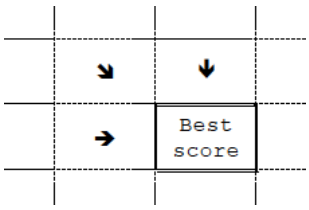
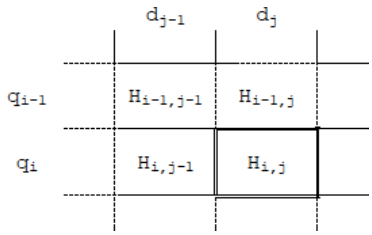


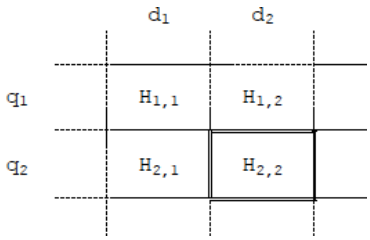
Table notation and indices

$H_{i,j}$ represents the score of the cell for the current update. $H_{i-1,j}$, $H_{i,j-1}$,and $H_{i-1,j-1}$ are the scores of the adjacent cells.

Cell $H_{i,j}$ and its adjacent cells



Example



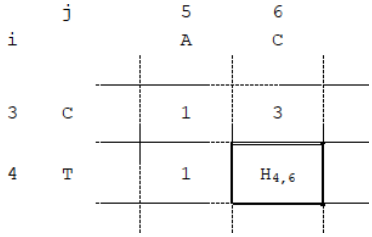
Calculation of three candidate scores

$H_{i,j}^{(0)}$, $H_{i,j}^{(1)}$, and $H_{i,j}^{(2)}$ represent the three candidate scores of $H_{i,j}$. They are respectively calculated as:

$$H_{i,j}^{(0)} = H_{i-1,j} - g \qquad (vertical)$$
$$H_{i,j}^{(1)} = H_{i,j-1} - g \qquad (horizontal)$$
$$H_{i,j}^{(2)} = H_{i-1,j-1} + R_{a,b} \qquad (diagonal)$$

Exercise 2.4

Calculate the scores of $H_{4,6}^{(0)}$, $H_{4,6}^{(1)}$, and $H_{4,6}^{(2)}$ first and then update $H_{4,6}$.



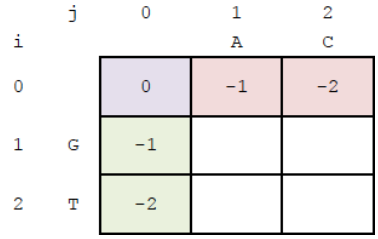
Scoring scheme:
$R_{ab} = 1$ for a = b
$R_{ab} = 0$ for a $\neq$ b
g = 1

Initialization

The first row and the first column can be calculated independently from the adjacent cells.

$$H_{0,j} : j * -1 * g$$
$$H_{i,0} : i * -1 * g$$

Example



Exercise 2.5

Update all cells of Table 1 and 2. Use the scoring scheme in Exercise 2.4.

Table 1

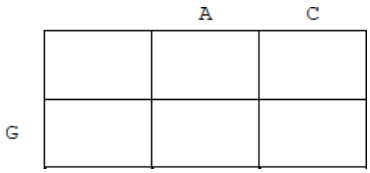
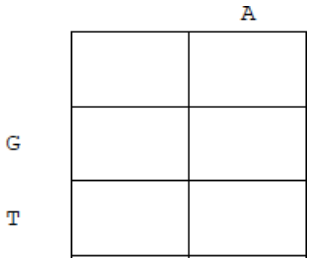


Table 2



Sub-solutions

In DP, larger decisions recursively nest smaller decisions. For instance, Table S is included in Table L.

Table S

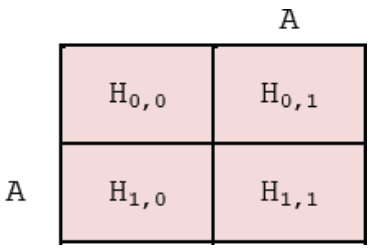
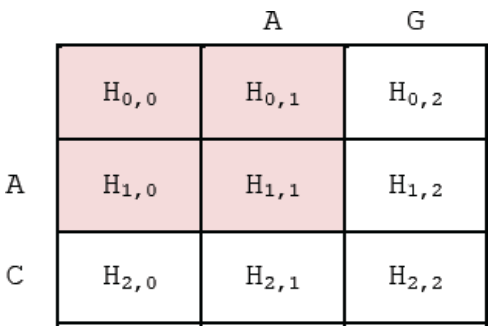


Table L



Pseudo-code of updating DP table for global alignment

Algorithm 2.1: Update dynamic programming table for global alignment

$H_{i,j}$: Dynamic programming table
$R_{a,b}$: Match/mismatch scores
g : Gap penalty

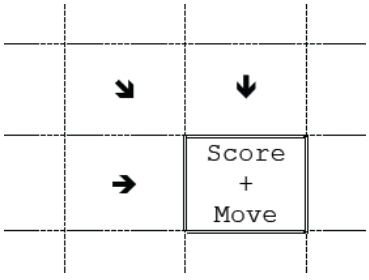
```
// Initialization
for i ← 0 to m do
    H_{i,0} ← i * −1 * g;
end
for j ← 1 to n do
    H_{0,j} ← j * −1 * g;
end

// Main loop for table update
for i ← 1 to m do
    for j ← 1 to n do
        H_{i,j} ← max(H_{i−1,j} − g, H_{i,j−1} − g, H_{i−1,j−1} + R_{a,b});
    end
end
```

2.5 Backtracking

Backtracking is a post-processing procedure to find the alignments that have yielded the best score.

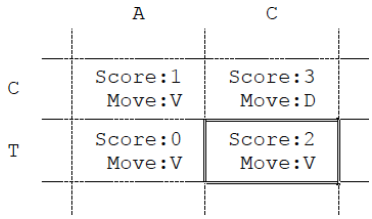
Store movement in cells

A table cell can be used for storing the movement.

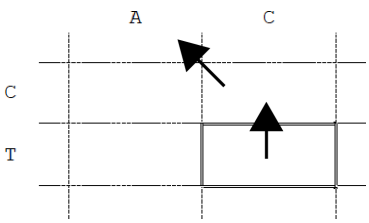


Example

Cells with scores and directions

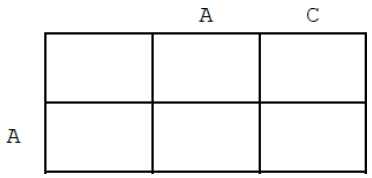


Use arrows to indicate backtracking



Exercise 2.6

Complete the DP table with scores and directions. What is the alignment with the best score?



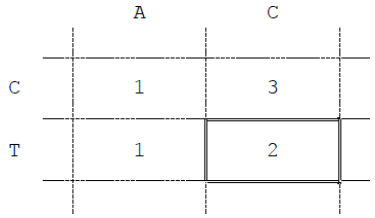
Scoring scheme:
$R_{ab} = 1$ for a = b
$R_{ab} = 0$ for a $\neq$ b
g = 1

Re-calculate candidate scores

Re-calculating the three candidate scores also reveals the movement.

$$H_{i,j}^{(0)} = H_{i-1,j} - g \qquad (vertical)$$
$$H_{i,j}^{(1)} = H_{i,j-1} - g \qquad (horizontal)$$
$$H_{i,j}^{(2)} = H_{i-1,j-1} + R_{a,b} \qquad (diagonal)$$

Example



$$H_{i,j}^{(0)} = 3 - 1 = 2 = H_{i,j} \qquad \checkmark (vertical)$$
$$H_{i,j}^{(1)} = 1 - 1 = 0 \neq H_{i,j} \qquad (horizontal)$$
$$H_{i,j}^{(2)} = 1 + 0 = 1 \neq H_{i,j} \qquad (diagonal)$$

Common mistake with backtracking

For the re-calculation approach, it is not to find $max(H_{i-1,j}, H_{i,j-1}, H_{i-1,j-1})$. You must re-calculate the candidates and then $max(H_{i,j}^{(0)}, H_{i,j}^{(1)}, H_{i,j}^{(2)})$ to find the actual direction.

Implementation with recursive call

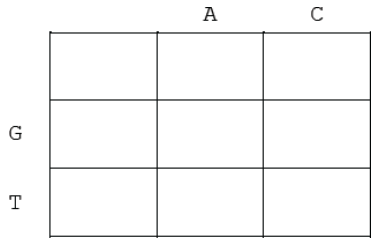
Recursive calls are usually used to implement DP backtracking.

Algorithm 2.2: DP backtracking

```
S_q  : Sequence q
S_d  : Sequence d
H_i,j : Dynamic programming table
R_a,b: Match/mismatch scores
g    : Gap penalty

proc backTrack(i, j, A_q, A_d, k)
    i   : Index of sequence q
    j   : Index of sequence d
    A_q : q part of alignment (stored in reverse order)
    A_d : d part of alignment (stored in reverse order)
    k   : Index for A_q and A_d
    //
    // Need to implement recursion termination here
    // ...
    //
    if H_i,j = H_i-1,j - g then                                  // vertical
        A_q,k ← S_q,i;
        A_d,k ← '-';
        backTrack(i − 1, j, A_q, A_d, k + 1);
    end
    if H_i,j = H_i,j-1 - g then                                  // horizontal
        A_q,k ← '-';
        A_d,k ← S_d,i;
        backTrack(i, j − 1, A_q, A_d, k + 1);
    end
    if H_i,j = H_i-1,j-1 + R_Sq,i,Sd,i then                      // diagonal
        A_q,k ← S_q,i;
        A_d,k ← S_d,i;
        backTrack(i − 1, j − 1, A_q, A_d, k + 1);
    end
end
```

Exercise 2.7

Find the alignment with the best score.



Scoring scheme:
$R_{ab} = 1$ for a = b
$R_{ab} = 0$ for a ≠ b
g = 1

2.6 Needleman-Wunsch algorithm

The method of using DP to solve global pairwise alignment is called the Needleman-Wunsch algorithm in the field of bioinformatics.

Complexity

- Time: O(nm)
- Space: O(nm)

Comparisons with other algorithms

The Needleman-Wunsch algorithm is similar to several algorithms.

Divide and conquer algorithms

Sub-solutions must be independent with divide and conquer.

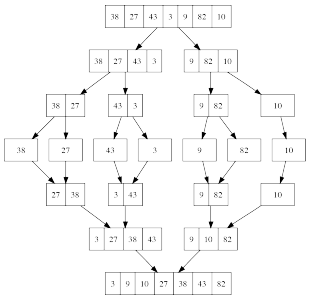


Figure 2.1: Merge sort (source: VineetKumar, Wikimedia Commons)

Dijkstra's algorithm

Worst-case performance of Dijkstra: $O(|E| + |V| \log |V|)$

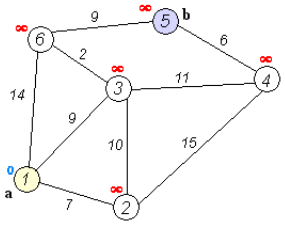


Figure 2.2: Dijkstra's algorithm (source: Ibmua, Wikimedia Commons)

3 Extension of global alignment

3.1 Homology at the sequence level

Constructing alignments can be useful to understand homology among different species. Finding homologies is important to reveal a common evolutionary ancestor.

Evolution and homology

All species are derived from a common ancestor at some point during the course of evolution.

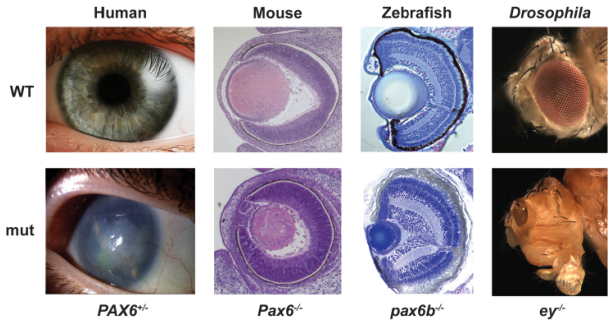


Figure 3.1: PAX6 alterations result in similar changes to eye morphology
(source: Washington et al, doi: 10.1371/journal.pbio.1000247 via Wikimedia Commons)

Homologous and analogous

It is useful to check similarity at the molecular level because there are cases that analogous structures may not indicate homologous.

Figure 3.2: Homologous and analogous structures
(source: John Romanes, 1892, Darwin and after Darwin via Wikimedia Commons)

Sequence homology

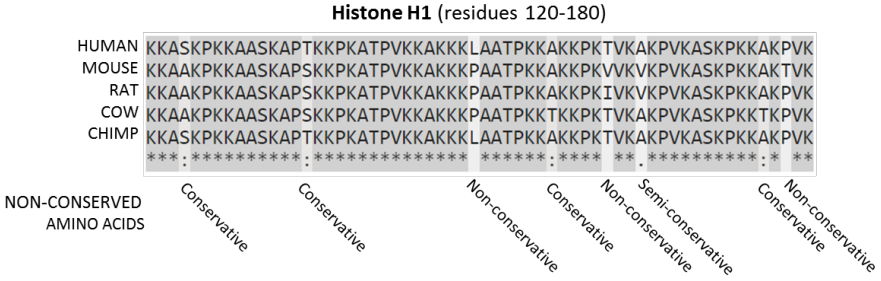


Figure 3.3: Multiple sequence alignment of histone sequences
(source: Shafee, Wikimedia Commons)

Evolution at the sequence level

Sequence differences in DNA

- Substitution (a mismatch in alignment)
- Insertion (a gap in alignment)
- Deletion (a gap in alignment)
- Inversion

Sources of variations

- Mutation
- Recombination
- Insertional mutagenesis
- ...

A mutation of the third nucleotide in a codon often does not affect which amino acid is synthesized.

- GCU → Ala (Alanine)
- GCC → Ala (Alanine)
- GCA → Ala (Alanine)
- GCG → Ala (Alanine)

An amino acid can be replaced by a different amino acid that has similar properties in some cases.

- AUU, AUC, AUA → Ile (Isoleucine)
- CUU, CUC, CUA → Leu (Leucine)

Extension of global alignment with DP

- Score matrix
  DNA, RNA, and protein
- Gap penalty
  Linear, affine, and constant

3.2 Introduction of score matrix

We will expand our simple scoring scheme to score matrices. This expansion allows us to solve general alignment problems with DNA, RNA, and protein sequences.

Extension of a scoring scheme to a score matrix

The matrix below is equivalent with match: 1 and mismatch: 0.

	a	b
a	1	0
b	0	1

Example of a DNA score matrix

The matrix below is equivalent with match: 5 and mismatch: -4.

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

Applications of score matrix

Score matrices are more flexible than the simple scoring scheme. For instance, they can be used for the following cases.

- DNA pairs
- RNA pairs
- Similarity of protein sequences by amino acid properties

DNA pairs (Watson-Crick pairs)

A thymine pairs with an adenine, and a cytosine pairs with a guanine.

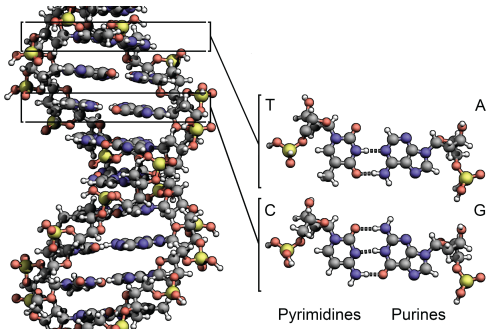


Figure 3.4: Watson-Crick pairs (source: Zephyris, Wikimedia Commons)

Example of score matrix for DNA pairs

The matrix reflects the differences of hydrogen bonds.

	A	T	G	C
A	-3	4	-3	-3
T		-3	-3	-3
G			-3	5
C				-3

Example of DP for DNA pairs

You can use DP to find a DNA alignment with Watson-Crick pairs. For instance, the DP table below is used to solve the optimal alignment for two DNA sequences: q = AC and d = GT with gap penalty g = 4.

DP table:

q/d		G	T
	0	-4	-8
A	-4	-3	0
C	-8	1	-3

Alignment:

```
q: AC-
d: -GT
```

RNA pairs

A single stand of RNA can form a 3D structure that has a biological function. The secondary structure of RNA is a two-dimensional representation of the structure.

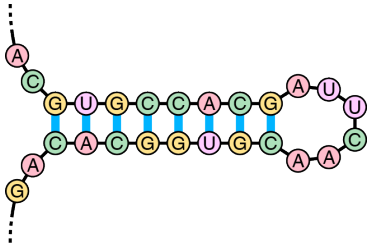


Figure 3.5: RNA stem-loop (source: Sakurambo, Wikimedia Commons)

Wobble pairs

Wobble pairs are not canonical Watson-Crick pairs, but they can still form hydrogen bonds.

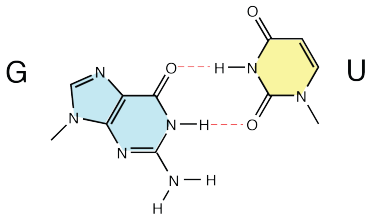


Figure 3.6: GU wobble pairs
(modified from the original version by Fdardel, Wikimedia Commons)

Example of score matrix for RNA pairs

The matrix takes GU wobbles into consideration.

	A	U	G	C
A	-3	5	-3	-3
U		-3	2	-3
G			-3	5
C				-3

Example of DP for RNA pairs

You can form the following DP table for two RNA sequences: q = AU and d = UGA with gap penalty g = 9.

DP table:

q/d		U	G	A
	0	-9	-18	-27
A	-9	5	-4	-11
U	-18	-4	7	1

Alignment:

```
q: A-U
d: UGA
```

Similarity of protein sequences

Amino acids can be categorized into several groups by their properties. Proteins alignments often need to take these properties into consideration.

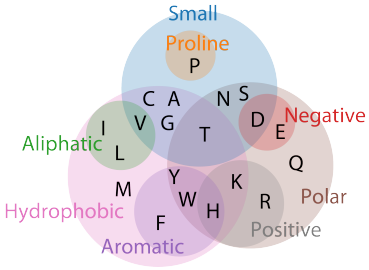


Figure 3.7: Venn diagram of amino acid properties

Example of a protein score matrix

It can be used to compare the similarity between two protein sequences.

	A	**R**	**N**	**D**	**C**	**Q**	**E**	**G**	**H**	**I**	**L**	**K**	**M**	**F**	**P**	**S**	**T**	**W**	**Y**	**V**
A	13	6	9	9	5	8	9	12	6	8	6	7	7	4	11	11	11	2	4	9
R	3	17	4	3	2	5	3	2	6	3	2	9	4	1	4	4	3	7	2	2
N	4	4	6	7	2	5	6	4	6	3	2	5	3	2	4	5	4	2	3	3
D	5	4	8	11	1	7	10	5	6	3	2	5	3	1	4	5	5	1	2	3
C	2	1	1	1	52	1	1	2	2	2	1	1	1	1	2	3	2	1	4	2
Q	3	5	5	6	1	10	7	3	7	2	3	5	3	1	4	3	3	1	2	3
E	5	4	7	11	1	9	12	5	6	3	2	5	3	1	4	5	5	1	2	3
G	12	5	10	10	4	7	9	27	5	5	4	6	5	3	8	11	9	2	3	7
H	2	5	5	4	2	7	4	2	15	2	2	3	2	2	3	3	2	2	3	2
I	3	2	2	2	2	2	2	2	2	10	6	2	6	5	2	3	4	1	3	9
L	6	4	4	3	2	6	4	3	5	15	34	4	20	13	5	4	6	6	7	13
K	6	18	10	8	2	10	8	5	8	5	4	24	9	2	6	8	8	4	3	5
M	1	1	1	1	0	1	1	1	1	2	3	2	6	2	1	1	1	1	1	2
F	2	1	2	1	1	1	1	1	3	5	6	1	4	32	1	2	2	4	20	3
P	7	5	5	4	3	5	4	5	5	3	3	4	3	2	20	6	5	1	2	4
S	9	6	8	7	7	6	7	9	6	5	4	7	5	3	9	10	9	4	4	6
T	8	5	6	6	4	5	5	6	4	6	4	6	5	3	6	8	11	2	3	6
W	0	2	0	0	0	0	0	0	1	0	1	0	0	1	0	1	0	55	1	0
Y	1	1	2	1	3	1	1	1	3	2	2	1	2	15	1	2	2	3	31	2
V	7	4	4	4	4	4	4	4	5	4	15	10	4	10	5	5	5	72	4	17

Table 3.1: Mutation probability matrix for the evolutionary distance of 250 PAMs (in percentage) (Chapter 22: A model of evolutionary change in proteins, Dayhoff and Schwartz, Atlas of Protein Sequence and Structure, 1978)

Exercise 3.1

1. Use the DNA score matrix below with g = 10 and find the optimal alignment for q = TG and d = TCG.

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

2. The 250 PAM mutation matrix above can not directly be used for global alignments. Explain what kind of matrix you need for calculating alignment scores.

3.3 Extension of gap penalties

Types of gap penalties

Three types of gap penalties can be considered when creating an alignment. They treat a gap penalty differently depending on the gap length.

- Linear
- Affine
- Constant

Gap penalty notation

- g: single gap penalty
- l: length of a gap
- g_l: gap penalty of length l
- g_{open}: initial gap penalty
- g_{extend}: extended gap penalty

Linear gap penalty

It is the same as our simple scoring scheme. It treats a gap with multiple blanks as a result of several mutations. A gap of length l can be calculated as: $g_l = g * l$.

Example of a gap of length 2

```
q: ACCCGT
d: AC--GT
```

The score of the gap (only the gap part) is 10 when $g = 5$.

Affine gap penalty

It treats a gap with multiple blanks as a result of a single mutation. A gap with length l can be calculated as: $g_l = g_{open} + (l1) * g_{extend}$.

Example of a gap of length 2

```
q: ACCCGT
d: AC--GT
```

The score of the gap (only the gap part) is 5.5 when g_{open} and g_{extend} are 5 and 0.5 respectively.

Constant gap penalty

It is similar to the affine gap penalty, but the score is independent form the gap length. A gap with length l can be calculated as: $g_l = g$

Example of a gap of length 2

```
q: ACCCGT
d: AC--GT
```

The score of the gap (only the gap part) for the alignment above is 5 when $g = 5$.

Exercise 3.2

Calculate all three types of gap penalties for the gap in alignment 1 & 2.

- g: 5
- g_{open}: 5
- g_{extend}: 0.5

```
Alignment 1
    q: CCCGG
    d: CC-CG
```

```
Alignment 2
    q: CCCGG
    d: C---G
```

3.4 Affine gap penalties with a single DP table

DP for general gap penalty

We need to modify DP so that extra cells are checked to find the optimal score of a cell.

Cell update rule of general gap penalty

$$H_{i,j} = \max \left[H_{i-1,j-1} + R_{q_i d_j}, \quad \max_{1 \le l \le j} (H_{i,j-l} - g_l), \quad \max_{1 \le l \le i} (H_{i-l,j} - g_l) \right]$$

Example of cell update

Sequences:

```
q: AG, d: ACG
```

Scoring scheme:

$g_{open} = 1$
$g_{extend} = 0.1$
$R_{ab} = 1$ for a = b
$R_{ab} = 0$ for a $\neq$ b

Update $H_{2,1}$

		A	C	T	T
	0	-1	-1.1	-1.2	-1.3
A	-1	1			
T	-1.1	0			

- vertical: $max(1-1, \quad -1-1-0.1) = 0$
- horizontal: $-1.1-1 = -2.1$
- diagonal: $-1-0 = -1$

Update $H_{1,2}$

		A	C	T	T
	0	-1	-1.1	-1.2	-1.3
A	-1	1	0		
T	-1.1	0			

- vertical: $-1.1-1 = -2.1$
- horizontal: $max(1-1, \quad -1-1-0.1) = 0$
- diagonal: $-1-0 = -1$

Update $H_{1,3}$

		A	C	T	T
	0	-1	-1.1	-1.2	-1.3
A	-1	1	0	**-0.1**	
T	-1.1	0	1		

- vertical: $-1.2-1 = -2.2$
- horizontal: $max(0-1, \quad 1-1-0.1, \quad -1-1-0.1-0.1) = -0.1$
- diagonal: $-1.1-0 = -1.1$

Exercise 3.3

Complete the DP table below.

Sequences:

`q: AT, d: ACTT`

Scoring scheme:

$g_{open} = 1$
$g_{extend} = 0.1$
$R_{ab} = 1$ for a = b
$R_{ab} = 0$ for a $\neq$ b

		A	C	T	T
	0	-1	-1.1	-1.2	-1.3
A	-1	1	0	-0.1	
T	-1.1	0	1		

3.5 Affine gap penalties with three DP tables

DP can effectively solve affine gap penalties with three tables.

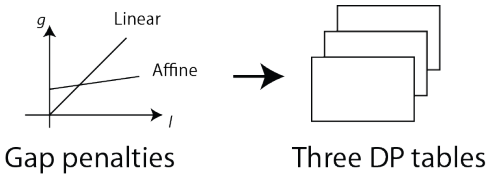


Figure 3.8: Affine gap penalties and three tables

Three DP tables

We need to modify DP so that extra cells are checked to find the optimal score of a cell.

- $E_{i,j}$: alignment ending with a gap extend (vertical)
- $F_{i,j}$: alignment ending with a gap extend (horizontal)
- $G_{i,j}$: alignment ending with a match/mismatch (diagonal)

Cell update rule of the three tables

$$E_{i,j} = max(E_{i-1,j} - g_{extend}, \quad F_{i-1,j} - g_{open}, \quad G_{i-1,j} - g_{open})$$
$$F_{i,j} = max(E_{i,j-1} - g_{open}, \quad F_{i,j-1} - g_{extend}, \quad G_{i,j-1} - g_{open})$$
$$G_{i,j} = max(E_{i-1,j-1} + R_{q_i d_j}, \quad F_{i-1,j-1} + R_{q_i d_j}, \quad G_{i-1,j-1} + R_{q_i d_j})$$

You can calculate H only in the last cell.

$$H_{m,n} = max(E_{m,n}, F_{m,n}, G_{m,n})$$

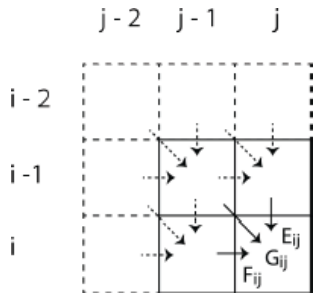


Figure 3.9: Update a cell with E, F, and G

Recurrence rules when i = 0 and j = 0

	$i > 1, j > 1$	$i = 1$	$j = 1$
$E_{i,j}$	$max\begin{cases} E_{i-1,j} - g_{extend} \\ F_{i-1,j} - g_{open} \\ G_{i-1,j} - g_{open} \end{cases}$	$max\begin{cases} E_{i-1,j} - g_{open} \\ F_{i-1,j} - g_{open} \\ G_{i-1,j} - g_{open} \end{cases}$	$max\begin{cases} E_{i-1,j} - g_{extend} \\ F_{i-1,j} - g_{open} \\ G_{i-1,j} - g_{open} \end{cases}$
$F_{i,j}$	$max\begin{cases} E_{i,j-1} - g_{open} \\ F_{i,j-1} - g_{extend} \\ G_{i,j-1} - g_{open} \end{cases}$	$max\begin{cases} E_{i,j-1} - g_{open} \\ F_{i,j-1} - g_{extend} \\ G_{i,j-1} - g_{open} \end{cases}$	$max\begin{cases} E_{i,j-1} - g_{open} \\ F_{i,j-1} - g_{open} \\ G_{i,j-1} - g_{open} \end{cases}$
$G_{i,j}$	$max\begin{cases} E_{i-1,j-1} + R_{q_i d_j} \\ F_{i-1,j-1} + R_{q_i d_j} \\ G_{i-1,j-1} + R_{q_i d_j} \end{cases}$	$max\begin{cases} E_{i-1,j-1} + R_{q_i d_j} \\ F_{i-1,j-1} + R_{q_i d_j} \\ G_{i-1,j-1} + R_{q_i d_j} \end{cases}$	$max\begin{cases} E_{i-1,j-1} + R_{q_i d_j} \\ F_{i-1,j-1} + R_{q_i d_j} \\ G_{i-1,j-1} + R_{q_i d_j} \end{cases}$

Example of updating DP tables with affine gaps

Sequences:

`q: AT, d: ACTT`

Scoring scheme:

$g_{open} = 1$
$g_{extend} = 0.1$
$R_{ab} = 1$ for a = b
$R_{ab} = 0$ for a ≠ b

Initialization

E

		A	C	T	T
	0	-1	-1.1	-1.2	-1.3
A	-1				
T	-1.1				

F

		A	C	T	T
	0	-1	-1.1	-1.2	-1.3
A	-1				
T	-1.1				

G

		A	C	T	T
	0	-1	-1.1	-1.2	-1.3
A	-1				
T	-1.1				

Update the first row

E

		A	C	T	T
	0	-1	-1.1	-1.2	-1.3
A	-1	-2	-2.1	-2.2	-2.3
T	-1.1				

F

		A	C	T	T
	0	-1	-1.1	-1.2	-1.3
A	-1	-2	0	-0.1	-0.2
T	-1.1				

G

		A	C	T	T
	0	-1	-1.1	-1.2	-1.3
A	-1	1	-1	-1.1	-1.2
T	-1.1				

Update the second row

E

		A	C	T	T
	0	-1	-1.1	-1.2	-1.3
A	-1	-2	-2.1	-2.2	-2.3
T	-1.1	0	-1	-1.1	-1.2

F

		A	C	T	T
	0	-1	-1.1	-1.2	-1.3
A	-1	-2	0	-0.1	-0.2
T	-1.1	-2.1	-1	0	0

G

		A	C	T	T
	0	-1	-1.1	-1.2	-1.3
A	-1	1	-1	-1.1	-1.2
T	-1.1	-1	1	1	0.9

H

		A	C	T	T
A					
T					0.9

Update H

E

		A	C	T	T
	0	-1	-1.1	-1.2	-1.3
A	-1	-2	-2.1	-2.2	-2.3
T	-1.1	0	-1	-1.1	-1.2

F

		A	C	T	T
	0	-1	-1.1	-1.2	-1.3
A	-1	-2	0	-0.1	-0.2
T	-1.1	-2.1	-1	0	0

G

		A	C	T	T
	0	-1	-1.1	-1.2	-1.3
A	-1	1	-1	-1.1	-1.2
T	-1.1	-1	1	1	0.9

H

		A	C	T	T
A					
T					0.9

Backtrack

		A	C	T	T
	0	-1	-1.1	-1.2	-1.3
A	-1	-2	-2.1	-2.2	-2.3
T	-1.1	0	-1	-1.1	-1.2

		A	C	T	T
	0	-1	-1.1	-1.2	-1.3
A	-1	-2	0	-0.1	-0.2
T	-1.1	-2.1	-1	0	0

G

		A	C	T	T
	0	-1	-1.1	-1.2	-1.3
A	-1	1	-1	-1.1	-1.2
T	-1.1	-1	1	1	0.9

H

		A	C	T	T
A					
T					0.9

Optimal alignment

```
q: A--T    Score: 0.9
d: ACTT
```

Constant gap penalty

DP with constant gap penalty can be solved in the same way as the affine gap penalty.

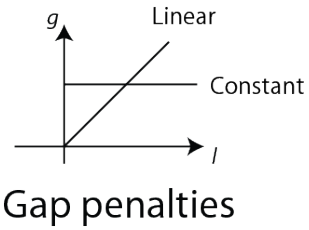


Figure 3.10: Constant gap penalty

3.6 Sequence distance

Distances can be also used to indicate the similarity of an alignment.

Edit distance

The Levenshtein distance is one of the most commonly used edit distances in computer science.

$$\begin{aligned} \text{Insertion}: \quad & AC \to AGC && (\varepsilon \to G) \\ \text{Deletion}: \quad & ATC \to AC && (T \to \varepsilon) \\ \text{Substitution}: \quad & AAA \to ATA && (A \to T) \end{aligned}$$

Scoring scheme for Levenshtein distance

- $R_{ab} = 0$ for a = b
- R_{ab} = -1 for a $\neq$ b
- g = 1

Distance from DP score

Given the best score T from DP, the edit distance d is $-T$.

Example of edit distance with DP

		A	T	C
	0	-1	-2	-3
A	-1	0	-1	-2
C	-2	-1	-1	-1

$T = -1$

$d = 1$

Metric space

The edit distance constitutes a metric space.

- $d_{xy} = 0$ for $x = y$
- $d_{xy} > 0$ for $x \neq y$
- $dxy = dyx$
- $dxy \leq dxz + dzy$ for any z (the triangle inequality)

Mutation and distance

Mutations may occur several times on the same position.

Example of single mutations

$ACGT \rightarrow AGT \rightarrow ACT \rightarrow AGT \rightarrow AGCT$

Four mutations have occurred, but the edit distance is 2.

Distance per column

It indicates the number of mutations per column (nucleotide/amino acid).

$D = d/$(length of the longest sequence)

Correction of distance

The distance can be adjusted. Below is a simple correction approach for protein sequences.

$$K = -ln(1 - D - 1/5D^2)$$

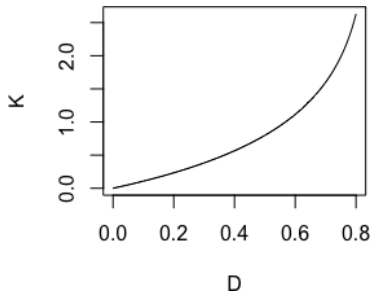


Figure 3.11: Correction of distance D

Example of distance correction

$D = 0.5$
$K = -ln(1 - 0.5 - 1/5 \times 0.25) = -ln(0.45) \approx 0.8$

4 Local alignment

4.1 Local alignments

Local pairwise alignments are aligned pairs of sub–sequences that have centain level of similarities.

Deiffrence between global and local alignments

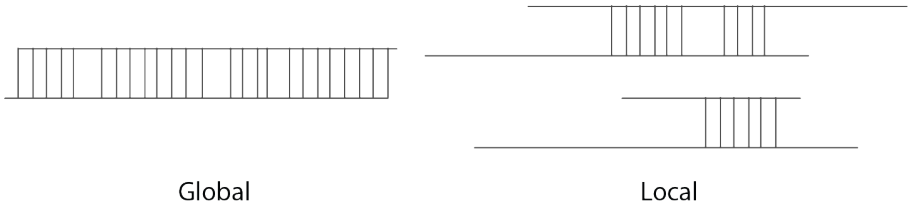


Figure 4.1: Global and local alignments

Elements of local alignment

- Segment: a substring of a sequence
- Segment pair: a pair of segments
- Local alignment: an alignment of a segment pair

Elements of local alignment

- Dynamic programming (Smith–Waterman)
- Dot matrix

Applications

- Sequence motifs
- Conserved regions
- Inverted repeats

4.2 Local alignment with DP

Dynamic programming can be used to find local alignments.

Requirements

- Find all local alignments between two sequences
- Assign scores to all local alignments

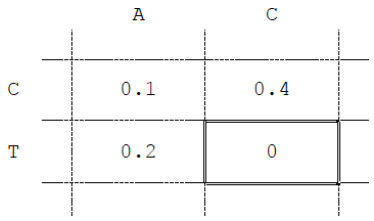
Modification of DP for local alignments

- The minimum alignment score must be 0
- Some entries of the score matrix should be negative
- Backtracking also needs to be modified

Update rule of DP cells

$$
\begin{aligned}
H_{i,j}^{(0)} &= H_{i-1,j} - g && (vertical) \\
H_{i,j}^{(1)} &= H_{i,j-1} - g && (horizontal) \\
H_{i,j}^{(2)} &= H_{i-1,j-1} + R_{a,b} && (diagonal) \\
H_{i,j}^{(2)} &= 0 && (minimumscore)
\end{aligned}
$$

Example of cell udpate



Scoring scheme:
Match: 0.5
Mismatch: -0.3
Gap penalty: 0.5

$$
\begin{aligned}
H_{i,j}^{(0)} &= -0.1 && (vertical) \\
H_{i,j}^{(1)} &= -0.3 && (horizontal) \\
H_{i,j}^{(2)} &= -0.2 && (diagonal) \\
H_{i,j}^{(2)} &= 0 && \checkmark (minimumscore)
\end{aligned}
$$

Backtracking for local alignments

It starts from the cells with the maximum score instead of the right bottom cell.

- Start cells: cells with the maximum score
- End cells: cells with 0

N.B. the end cell with score 0 should not be included in the alignment.

Example of backtracking

		A	C	G	C
	0	0	0	0	0
C	0	0	0.5	0	0.5
G	0	0	0	1	0.5
A	0	0.5	0	0.5	0.7

Local alignment
q: 2 CG 3
d: 2 CG 3

Pseudo-code of updating DP table for local alignment

The cells in the first row and the first column are initialized with 0.

Algorithm 4.1: Update dynamic programming table for global alignment

$H_{i,j}$: Dynamic programming table
$R_{a,b}$: Match/mismatch scores
g : Gap penalty

```
// Initialization
for i ← 0 to m do
    H_{i,0} ← 0;
end
for j ← 1 to n do
    H_{0,j} ← 0;
end

// Main loop for table update
for i ← 1 to m do
    for j ← 1 to n do
        H_{i,j} ← max(0, H_{i−1,j} − g, H_{i,j−1} − g, H_{i−1,j−1} + R_{a,b});
    end
end
```

Exercise 4.1

Use DP to find a local alignment.

q/d		A	G	C	C
A					
G					
C					

Scoring scheme:
Match: 0.2
Mismatch: -0.2
Gap penalty: 0.2

4.3 Dot matrix

Using a dot matrix is an effective and easy way to find local similarities.

Basic concept

It uses an $m \times n$ binary matrix from two sequences.

- A dot: match
- Empty: mismatch

Example of dot matrix

`q: ACATTAG, d: CATTTAGG`

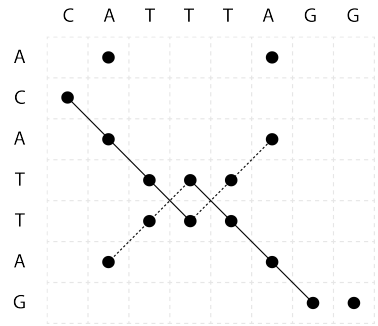


Figure 4.2: Dot matrix of 7×8

It is easy to find segment pairs with a dot matrix. Contiguous dots along diagonals indicate local alignments. It is also easy to find other similarities. For instance, contiguous dots along anti-diagonals indicate reversed substrings.

Filtering of dot matrix

Dot matrices usually get noisy with too many dots. Overlapping windows are usually applied to reduce the noise.

Example of filtering

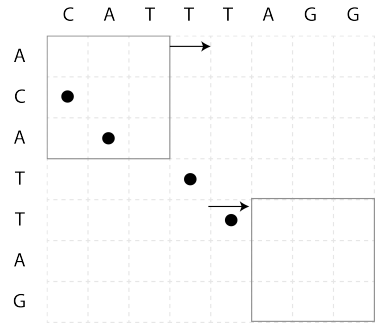


Figure 4.3: Filtered dot matrix with window size 3 and threshold 3.

Exercise 4.2

Find local similarities between two DNA sequences, `q: GATTACA` and `d: GGATTTAC`.

1. Create a dot matrix for the two sequences.
2. Filter dots with overlapping windows size 3 and threshold 3.