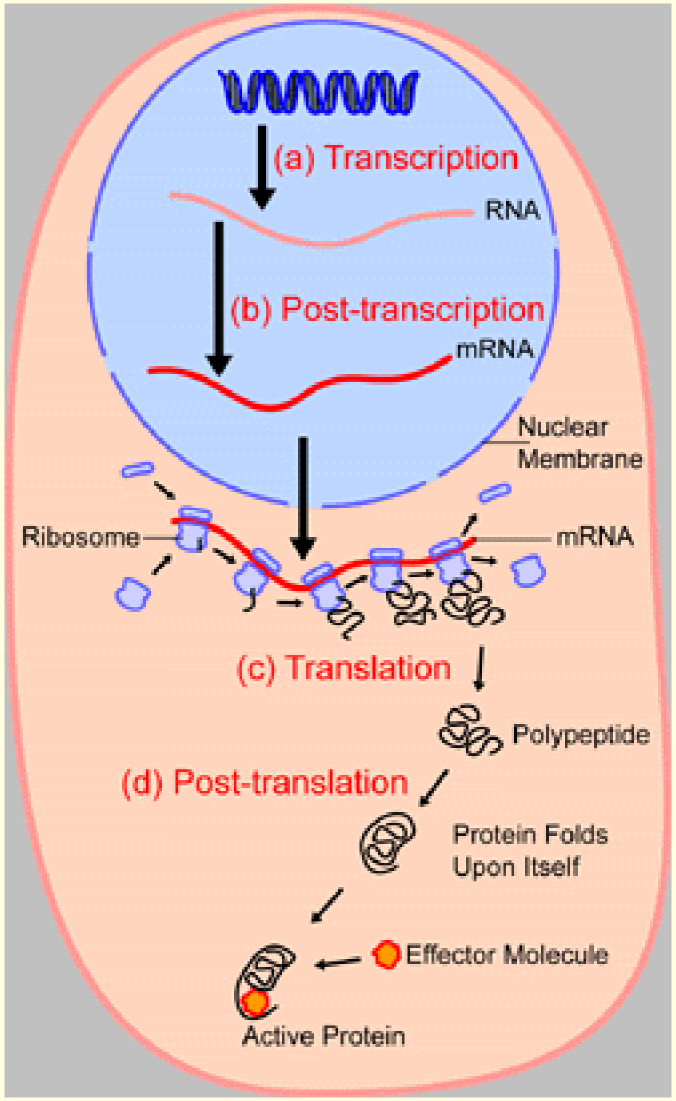
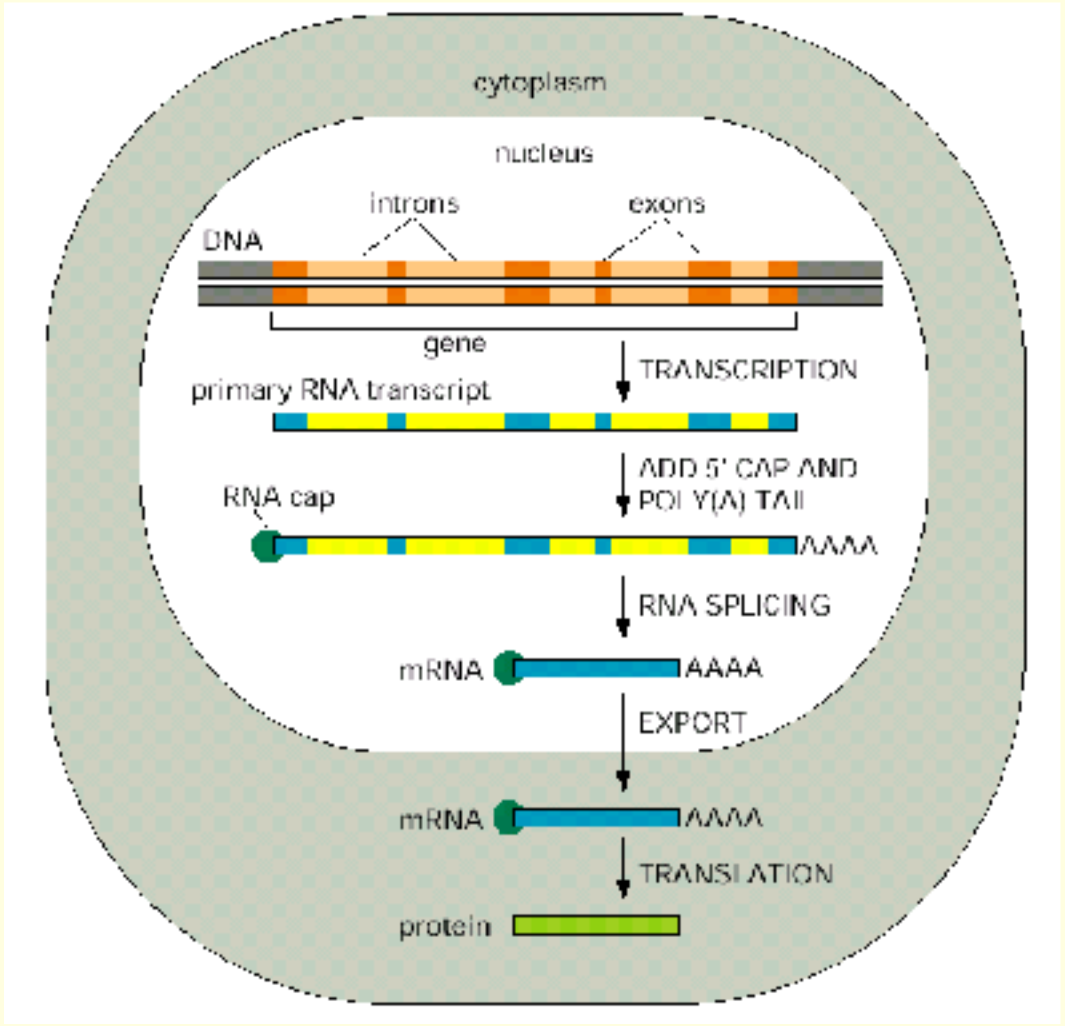
CMPS 6630: Introduction to Computational Biology and Bioinformatics

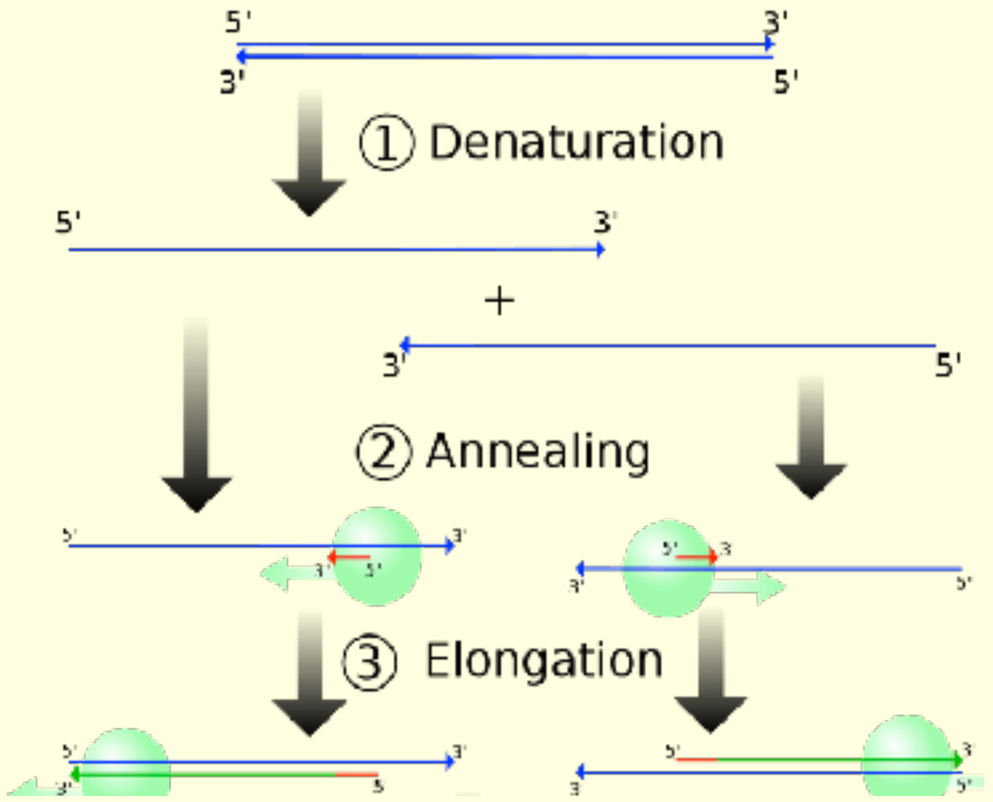
Sequence Comparison Methods





DNA sequence “codes for” biological function. How do we get a sequence of DNA or mRNA?

Polymerase Chain Reaction



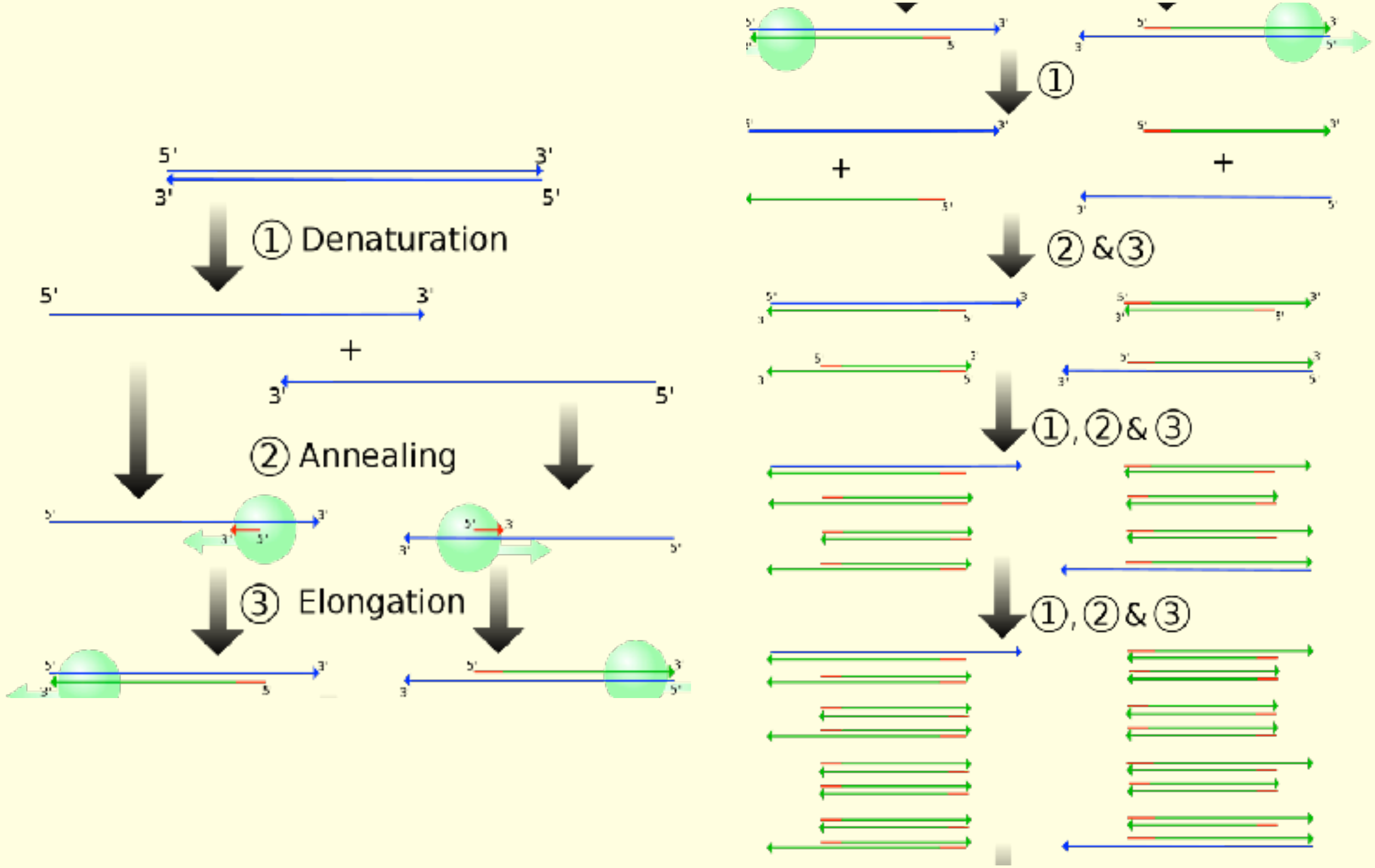
DNA is denatured at 94-96C; primers bind to single strands.

Taq-polymerase is used to extend primers.

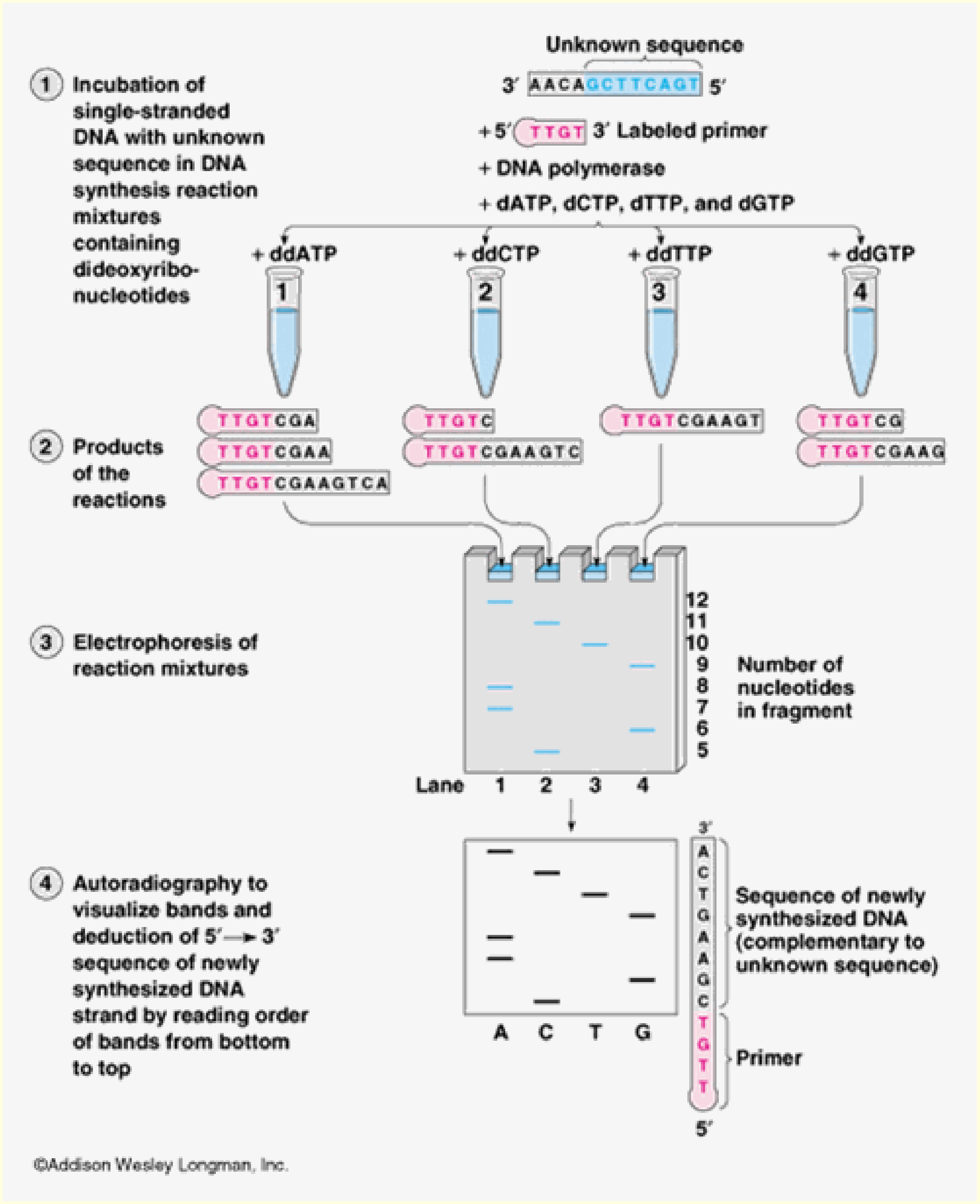
Primers bind at 50-60C, Taq works at 72C.

Invented by Kary Mullis (Nobel, 1993).

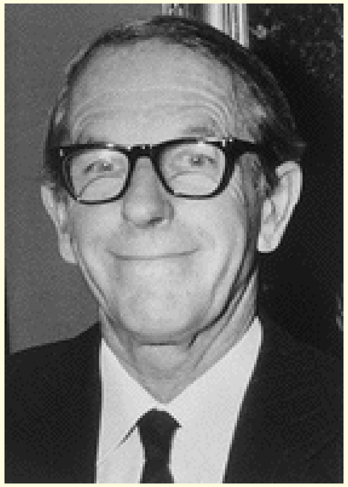
Polymerase Chain Reaction



Polymerase Chain Reaction



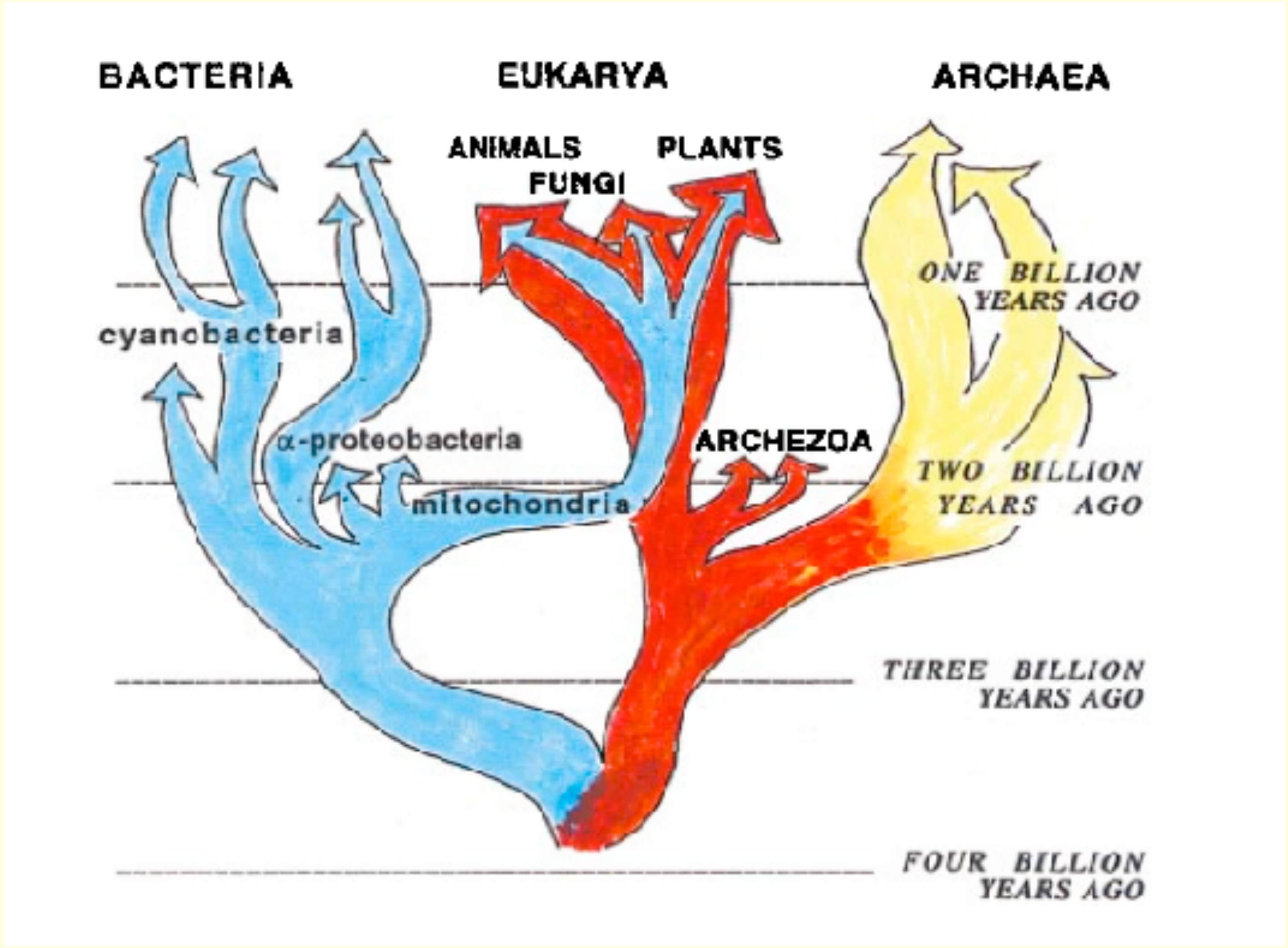
Unknown sequence
3′ AACAGCTTCAGT 5′
① Incubation of single-stranded DNA with unknown sequence in DNA synthesis reaction mixtures containing dideoxyribonucleotides
+5′ TTGT 3′ Labeled primer
+ DNA polymerase
+ dATP, dCTP, dTTP, and dGTP
+ ddATP
+ ddCTP
+ ddTTP
+ ddGTP
1
2
3
4
② Products of the reactions
TTGTCGA
TTGTCGAA
TTGTCGAAGTCA
TTGTC
TTGTCGAAGTC
TTGTCGAAGT
TTGTCG
TTGTCGAAG
③ Electrophoresis of reaction mixtures
12
11
10
9
8
7
6
5
Number of nucleotides in fragment
Lane 1 2 3 4
④ Autoradiography to visualize bands and deduction of 5′→3′ sequence of newly synthesized DNA strand by reading order of bands from bottom to top
A C T G
3′
A C T G A A G C T G T T
5′
Sequence of newly synthesized DNA (complementary to unknown sequence)
Primer



Sanger (1982) introduced chain-termination sequencing.

<u>Main idea</u>: Obtain fragments of all possible lengths, ending in A, C, T, G.

Using gel electrophoresis, we can separate fragments of differing lengths, and then assemble them.



Genes evolve and define the evolutionary relationship between organisms.

Sequence Comparison

- Comparing sequences is critical to understanding functional similarities and differences.

- DNA (and thus proteins) can be modified by:
  - insertions/deletions/substitutions
  - repeats/rearrangements

- 1.5% of mammalian DNA codes for proteins, 5-7% is functional.

$$\begin{tikzcd}
& \mathtt{ACTTTGGATT} \arrow[dl] \arrow[dr] & \\
\mathtt{A\cancel{C}TTTTGGG\cancel{A}TT} \arrow[d] & & \mathtt{ATC\cancel{T}GTTG\cancel{G}A\cancel{T}T} \arrow[d] \\
\mathtt{ATTTTGGGTT} & & \mathtt{ATCGTTGAT}
\end{tikzcd}$$

“Indels” occur naturally, how do we assess the similarity between two sequences?

g : ACTTTGGATT

?

h : ATCGTTGAT

<u>Candidate Alignments</u>

```
ACTTTGGATT
A-TCGTTGAT
1 -1 1 0 0 0 0 0 0 1
```

cost = 2

```
A-C-TTTGGATT
ATCGTT--GAT-
1 -1 1 -1 1 1 -1 -1 1 1 1 -1
```

cost = 2

When we compare two sequences, we want to “score” an alignment so that it can be used to explain how one sequence “evolved” in to another.

Problem Definition

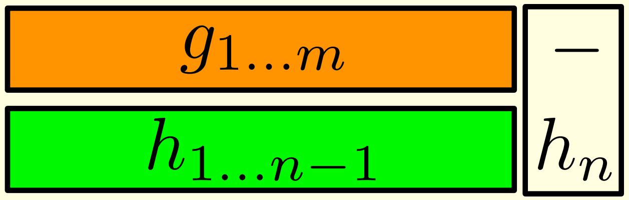
Let g and h be two given sequences of lengths m and n, and let $cost(g,h)$ denote the minimum number of “indels” required to change g into h, with fixed penalties.

Can we calculate $cost(g,h)$ exactly? What about the alignment?

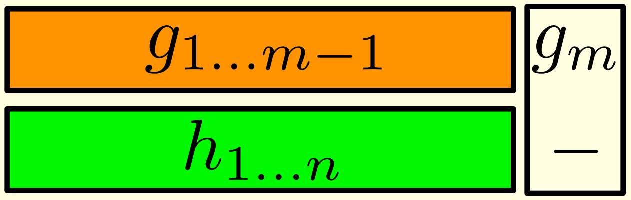
How long will it take (in terms of m and n) ?

Local Optimality

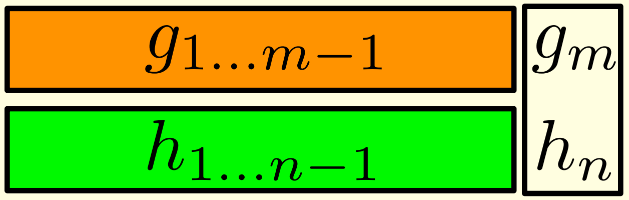
The best alignment between g and h must be one of the following:



$$cost(g, h_{1\ldots n-1}) - 1$$



$$cost(g_{1\ldots m-1}, h) - 1$$



$$cost(g_{1\ldots m-1}, h_{1\ldots n-1}) + cost(g_m, h_n)$$

Local Optimality

We have that:

$$cost(g,h) = \max \left\{ \begin{array}{c} cost(g, h_{1\ldots n-1}) - 1, \\ \\ cost(g_{1\ldots m-1}, h) - 1, \\ \\ cost(g_{1\ldots m-1}, h_{1\ldots n-1}) + cost(g_m, h_n) \end{array} \right\}$$

How do we calculate the recursive terms?

Calculating Alignment Score

Suppose we save $cost(g_{1 \ldots i}, h_{1 \ldots j})$ instead of computing from scratch:

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

Calculating Alignment Score

Suppose we save $cost(g_{1 \ldots i}, h_{1 \ldots j})$ instead of computing from scratch:

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

Calculating Alignment Score

Suppose we save $cost(g_{1 \ldots i}, h_{1 \ldots j})$ instead of computing from scratch:

$$
\begin{array}{cc|c|c|c|c|c|}
 & & 0 & 1 & 2 & 3 & 4 \\
 & & & \mathtt{C} & \mathtt{C} & \mathtt{G} & \mathtt{T} \\
\hline
0 & & 0 & -1 & -2 & -3 & -4 \\
\hline
1 & \mathtt{G} & -1 & 0 & -1 & & \\
\hline
2 & \mathtt{C} & -2 & & & & \\
\hline
3 & \mathtt{C} & -3 & & & & \\
\hline
4 & \mathtt{G} & -4 & & & & \\
\hline
\end{array}
$$

Calculating Alignment Score

Suppose we save $cost(g_{1\ldots i}, h_{1\ldots j})$ instead of computing from scratch:

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

Calculating Alignment Score

Suppose we save $cost(g_{1\ldots i}, h_{1\ldots j})$ instead of computing from scratch:

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

Calculating Alignment Score

Suppose we save $cost(g_{1...i}, h_{1...j})$ instead of computing from scratch:

$$
\begin{array}{cc|ccccc}
 & & 0 & 1 & 2 & 3 & 4 \\
 & & & \texttt{C} & \texttt{C} & \texttt{G} & \texttt{T} \\
\hline
0 & & 0 & -1 & -2 & -3 & -4 \\
1 & \texttt{G} & -1 & 0 & -1 & -1 & -2 \\
2 & \texttt{C} & -2 & 0 & & & \\
3 & \texttt{C} & -3 & & & & \\
4 & \texttt{G} & -4 & & & & \\
\end{array}
$$

Calculating Alignment Score

Suppose we save $cost(g_{1...i}, h_{1...j})$ instead of computing from scratch:

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

Calculating Alignment Score

Suppose we save $cost(g_{1\ldots i}, h_{1\ldots j})$ instead of computing from scratch:

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

Calculating Alignment Score

Suppose we save $cost(g_{1 \ldots i}, h_{1 \ldots j})$ instead of computing from scratch:

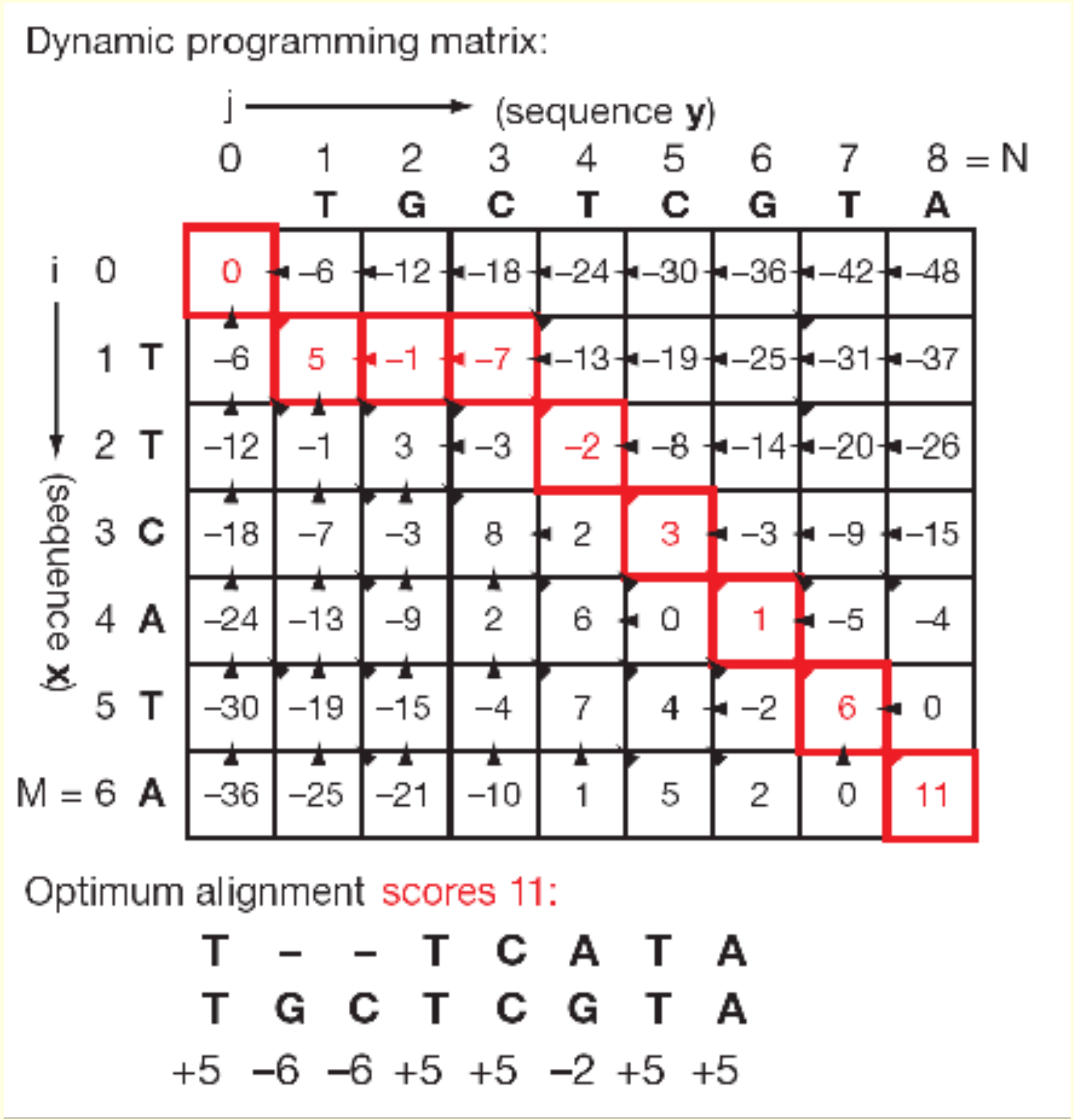
		0	1	2	3	4
			C	C	G	T
0		0	-1	-2	-3	-4
1	G	-1	0	-1	-1	-2
2	C	-2	0	1	0	-1
3	C	-3	-1	1	1	0
4	G	-4	-2	0	2	1

Running Time

- This method compute the optimal alignment by definition, the table only makes the approach more efficient.

- To perform an alignment of two sequences of lengths m and n takes $O(m \cdot n)$ time and space.

- What about aligning k sequences?



Dynamic programming matrix:

j → (sequence **y**)

i ↓ (sequence **x**)		0	1 T	2 G	3 C	4 T	5 C	6 G	7 T	8 = N A
0		0	–6	–12	–18	–24	–30	–36	–42	–48
1	T	–6	5	–1	–7	–13	–19	–25	–31	–37
2	T	–12	–1	3	–3	–2	–8	–14	–20	–26
3	C	–18	–7	–3	8	2	3	–3	–9	–15
4	A	–24	–13	–9	2	6	0	1	–5	–4
5	T	–30	–19	–15	–4	7	4	–2	6	0
M = 6	A	–36	–25	–21	–10	1	5	2	0	11

Optimum alignment scores 11:

```
 T   –   –   T   C   A   T   A
 T   G   C   T   C   G   T   A
+5  –6  –6  +5  +5  –2  +5  +5
```

Once we compute the table, the actual alignment can be obtained by backtracking.

Note that our algorithm can handle any fixed choice of gap penalties and matching costs.

Dynamic Programming

- Problems that can be solved by dynamic programming have a *local optimality* property.
- The term “dynamic programming” was invented by Richard Bellman (1954) for marketing purposes.
- Needleman and Wunsch (1970) were the first to apply it to biosequences.

Alignment Scoring Matrices

- We can generalize our approach, by having a scoring matrix as input.
- Why did sequence change in the first place? Evolution!
- PAM (Point Accepted Mutations) and BLOSUM (Block Substitution Matrix) are methods to statistically estimate the mutation rates in protein sequences.
- Mutation rates for DNA are roughly 3×10^{-6} (mitochondrial), and 2.5×10^{-8} (nuclear).

Generalized Gap Penalties

What if gap penalties are not a simple linear function of their length?

$$cost(g,h) = \max \left\{ \begin{array}{c} cost(g, h_{1 \ldots n-1}) - 1, \\ cost(g_{1 \ldots m-1}, h) - 1, \\ cost(g_{1 \ldots m-1}, h_{1 \ldots n-1}) + cost(g_m, h_n) \end{array} \right\}$$

$$\Big\downarrow$$

?

Generalized Gap Penalties

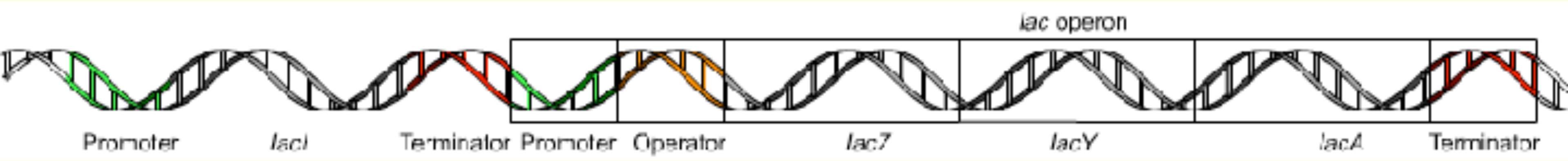
$$cost(g,h) = \max \left\{ \begin{array}{c} cost(g, h_{1\ldots n-1}) - 1, \\ cost(g_{1\ldots m-1}, h) - 1, \\ cost(g_{1\ldots m-1}, h_{1\ldots n-1}) + cost(g_m, h_n) \end{array} \right\}$$

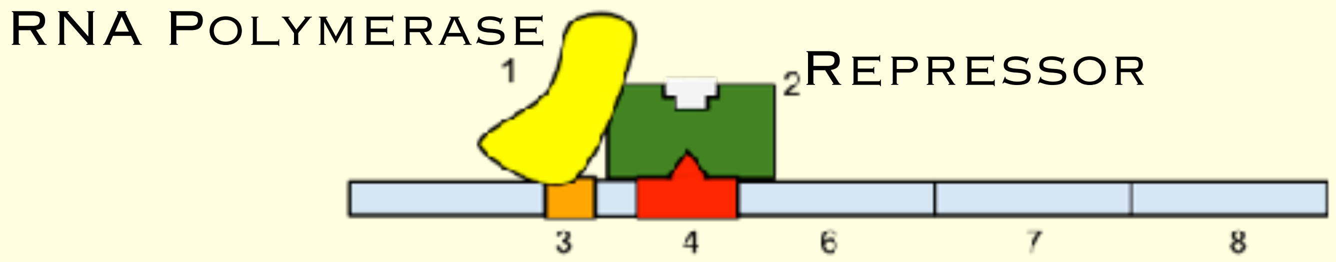
$$\Big\downarrow$$

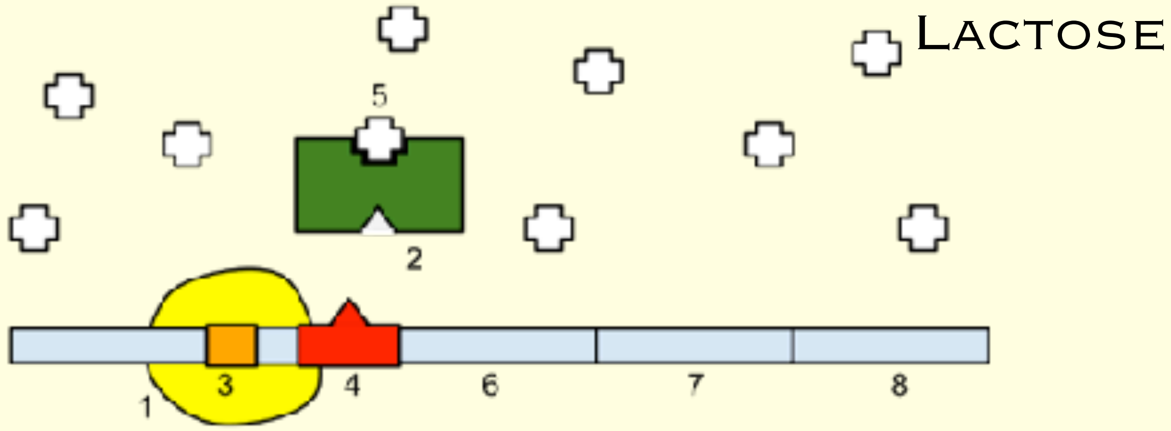
$$cost(g,h) = \max \left\{ \begin{array}{c} \max_\ell \left\{ cost(g, h_{1\ldots n-\ell}) - gap(\ell) \right\}, \\ \max_\ell \left\{ cost(g_{1\ldots m-\ell}, h) - gap(\ell) \right\}, \\ cost(g_{1\ldots m-1}, h_{1\ldots n-1}) + cost(g_m, h_n) \end{array} \right\}$$

Sequence Conservation

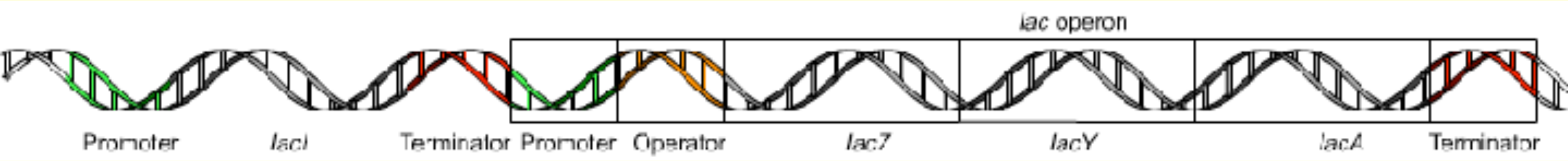
- Suppose we have just sequenced a gene, and want to know its relationship to an existing database of genes.
- Global alignment can identify evolutionarily “close” genes.
- What if we want to study conservation of (sub)sequences?

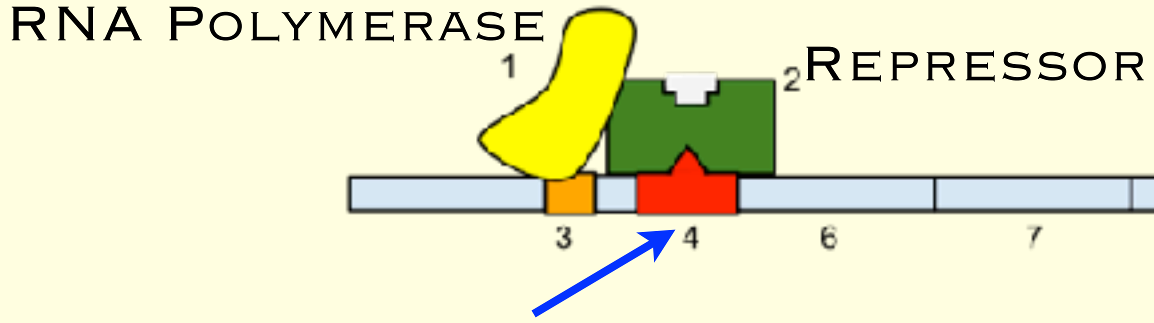

lac operon
Promoter
lacI
Terminator
Promoter
Operator
lacZ
lacY
lacA
Terminator

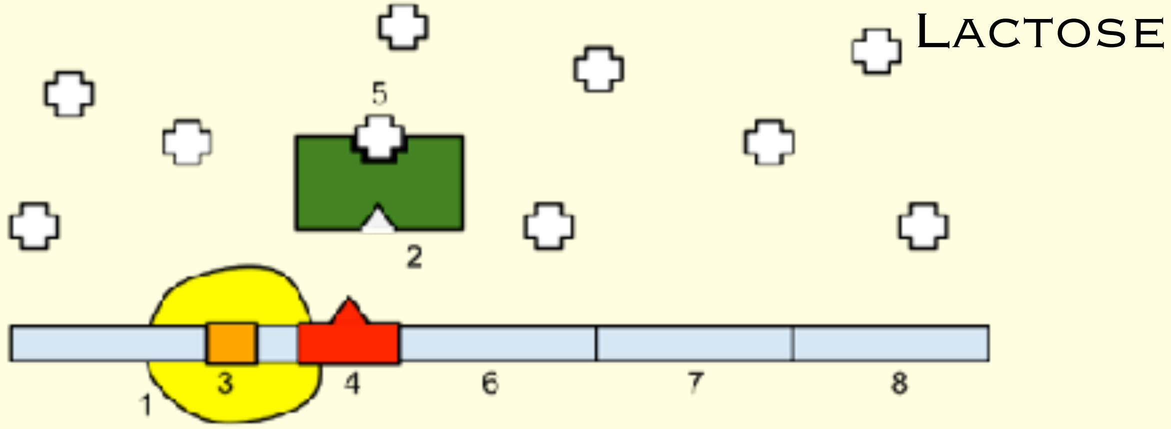

RNA Polymerase
1
2
Repressor
3
4
6
7
8


Lactose
5
2
1
3
4
6
7
8

Wikipedia


lac operon
Promoter
lacI
Terminator
Promoter
Operator
lacZ
lacY
lacA
Terminator


RNA Polymerase
1
2
Repressor
3
4
6
7
8


Lactose
5
2
1
3
4
6
7
8

Wikipedia

Local Alignment

Suppose we have the following protein sequences:

$$\begin{tikzcd} & \begin{array}{l} g : \texttt{ACEDECADE} \\ h : \texttt{REDCEDKL} \end{array} \arrow[dl] \arrow[dr] & \\ \begin{array}{l} g : \texttt{--ACEDECADE} \\ h : \texttt{REDCEDKL---} \end{array} & & \begin{array}{l} g : \texttt{ACEDECADE-} \\ h : \texttt{R-ED-CEDKL} \end{array} \end{tikzcd}$$

cost: -2

cost: 1

Local Alignment

Suppose we have the following protein sequences:

$$\begin{array}{rl} g: & \texttt{ACEDECADE} \\ h: & \texttt{REDCEDKL} \end{array}$$

$$\begin{array}{rl|l|l} \cline{3-3} g: & \texttt{--A} & \texttt{CED} & \texttt{ECADE} \\ h: & \texttt{RED} & \texttt{CED} & \texttt{KL---} \\ \cline{3-3} \end{array}$$

cost: -2

$$\begin{array}{rl} g: & \texttt{ACEDECADE-} \\ h: & \texttt{R-ED-CEDKL} \end{array}$$

cost: 1

Local Alignment

Observation: To find the best local alignment, we must the “best” pair of substrings to align.

$$cost(g,h) = \max_{i,j}\left\{0, \max_{k,\ell}\left\{cost(g_{k\ldots i}, h_{\ell\ldots j})\right\}\right\}$$

Can we use dynamic programming?

		R	E	D	C	E	D	K	L
	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	0	0	0
C	0	0	0	0	0.5	0	0	0	0
E	0	0	0.5	0	0	1	0.5	0	0
D	0	0	0	1	0.5	0.5	1.5	1	0.5
E	0	0	0.5	0.5	0.7	1	1	1.2	0.7
C	0	0	0	0.2	1	0.5	0.7	0.7	0.9
A	0	0	0	0	0.5	0.7	0.2	0.4	0.4
D	0	0	0	0.5	0	0.2	1.2	0.7	0.2
E	0	0	0.5	0	0.2	0.5	0.7	0.9	0.4

match: +0.5

mismatch: -0.3

gap penalty: -0.5

Why should we have a mismatch penalty?

Running time is $O(m^2 n^2)$; how do we get the actual local alignments?

		R	E	D	C	E	D	K	L
	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	0	0	0
C	0	0	0	0	0.5	0	0	0	0
E	0	0	0.5	0	0	1	0.5	0	0
D	0	0	0	1	0.5	0.5	1.5	1	0.5
E	0	0	0.5	0.5	0.7	1	1	1.2	0.7
C	0	0	0	0.2	1	0.5	0.7	0.7	0.9
A	0	0	0	0	0.5	0.7	0.2	0.4	0.4
D	0	0	0	0.5	0	0.2	1.2	0.7	0.2
E	0	0	0.5	0	0.2	0.5	0.7	0.9	0.4

match: +0.5

mismatch: -0.3

gap penalty: -0.5

Why should we have a mismatch penalty?

Running time is $\cancel{O(m^2n^2)}$; how do we get the actual local alignments?

Smith-Waterman Local Alignment

Observation: To find the best local alignment, we must the “best” pair of substrings to align.

$$cost(g,h) = \max \left\{ \begin{array}{c} 0, \\ cost(g, h_{1\ldots n-1}) - 1, \\ cost(g_{1\ldots m-1}, h) - 1, \\ cost(g_{1\ldots m-1}, h_{1\ldots n-1}) + cost(g_m, h_n) \end{array} \right\}$$

Running time is still $O(mn)$!

		R	E	D	C	E	D	K	L
	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	0	0	0
C	0	0	0	0	0.5	0	0	0	0
E	0	0	0.5	0	0	1	0.5	0	0
D	0	0	0	1	0.5	0.5	1.5	1	0.5
E	0	0	0.5	0.5	0.7	1	1	1.2	0.7
C	0	0	0	0.2	1	0.5	0.7	0.7	0.9
A	0	0	0	0	0.5	0.7	0.2	0.4	0.4
D	0	0	0	0.5	0	0.2	1.2	0.7	0.2
E	0	0	0.5	0	0.2	0.5	0.7	0.9	0.4

match: +0.5

mismatch: -0.3

gap penalty: -0.5

Why should we have a mismatch penalty?

Running time is $\overset{O(mn)}{\cancel{O(m^2n^2)}}$; how do we get the actual local alignments?

		R	E	D	C	E	D	K	L
	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	0	0	0
C	0	0	0	0	0.5	0	0	0	0
E	0	0	0.5	0	0	1	0.5	0	0
D	0	0	0	1	0.5	0.5	1.5	1	0.5
E	0	0	0.5	0.5	0.7	1	1	1.2	0.7
C	0	0	0	0.2	1	0.5	0.7	0.7	0.9
A	0	0	0	0	0.5	0.7	0.2	0.4	0.4
D	0	0	0	0.5	0	0.2	1.2	0.7	0.2
E	0	0	0.5	0	0.2	0.5	0.7	0.9	0.4

match: +0.5

mismatch: -0.3

gap penalty: -0.5

Why should we have a mismatch penalty?

Best local alignment:

$$g : \texttt{--A}\boxed{\texttt{CED}}\texttt{ECADE}$$

$$h : \texttt{RED}\boxed{\texttt{CED}}\texttt{KL---}$$

		R	E	D	C	E	D	K	L
	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	0	0	0
C	0	0	0	0	0.5	0	0	0	0
E	0	0	0.5	0	0	1	0.5	0	0
D	0	0	0	1	0.5	0.5	1.5	1	0.5
E	0	0	0.5	0.5	0.7	1	1	1.2	0.7
C	0	0	0	0.2	1	0.5	0.7	0.7	0.9
A	0	0	0	0	0.5	0.7	0.2	0.4	0.4
D	0	0	0	0.5	0	0.2	1.2	0.7	0.2
E	0	0	0.5	0	0.2	0.5	0.7	0.9	0.4

match: +0.5

mismatch: -0.3

gap penalty: -0.5

Best local alignment:

$g:$ --ACEDECADE

$h:$ REDCEDKL---

2nd best local alignment:

$g:$ ACEDECADE

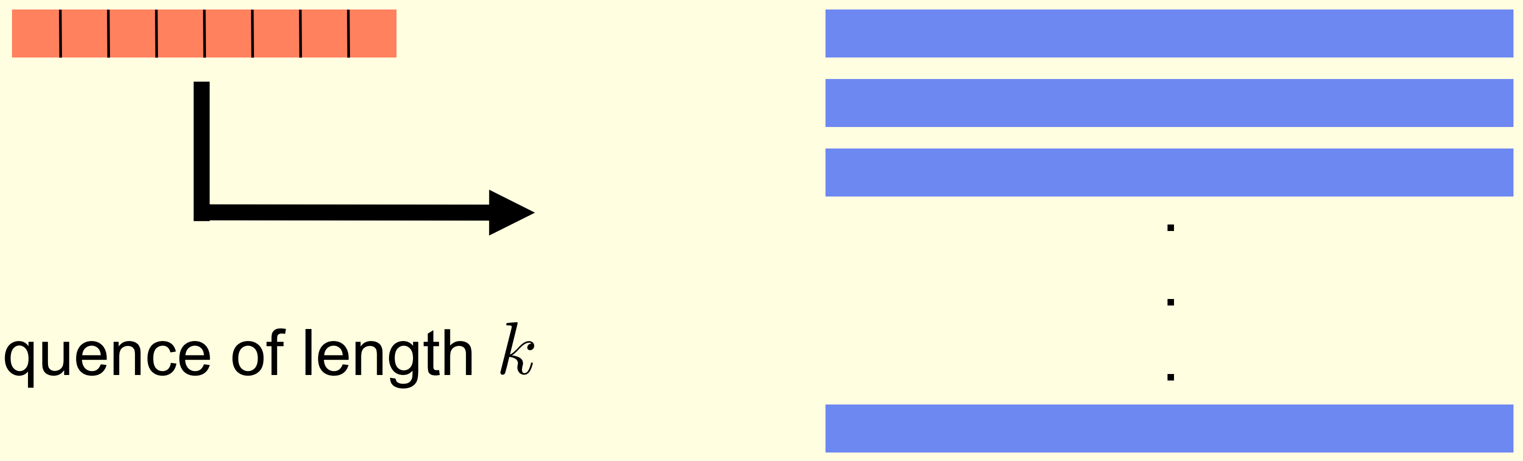
$h:$ -RED-CEDKL

Finding Good Alignments

Suppose we want to compare a new gene against a database of sequenced genes.

Query Sequence

Sequence Library



sequence of length k

n sequences of length $\geq k$

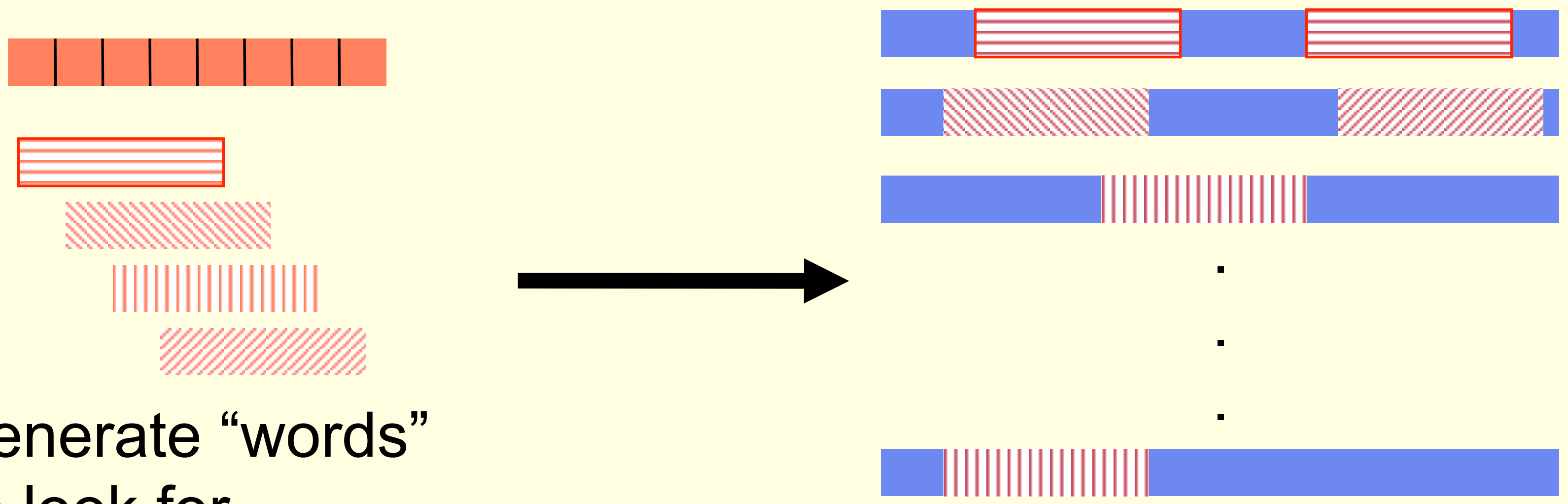
Performing local alignment against the database would take $O(n \cdot k^2)$ time.

BLAST

To perform these alignments quickly, we use a heuristic to find alignment “neighborhoods”:

Query Sequence

Sequence Library



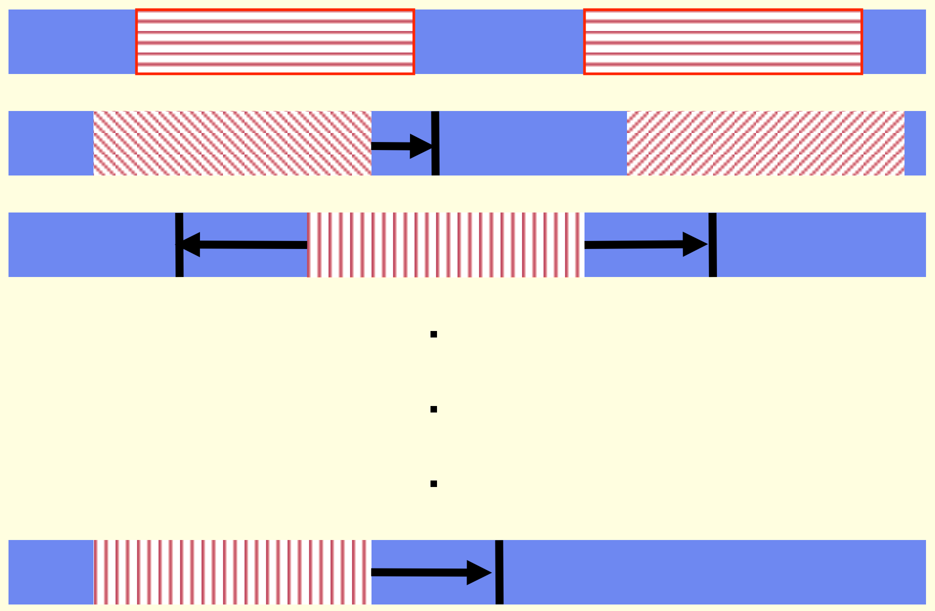
generate “words” to look for

find “high-scoring segment pairs” (HSPs) in database

BLAST

Once words are found, we heuristically extend the local alignments defined by the hits (including gaps if necessary):

Sequence Library



Intuition: The resulting alignment should be good, otherwise it wouldn’t have found many HSPs.

Analysis: ??

BLAST Statistics

BLAST <u>cannot</u> guarantee an optimal solution, but what is the likelihood of a particular score, and how unique is the score?

<u>E-Score</u>: The expected number of HSPs with score at least S :

$$E = Kmn \cdot e^{-\lambda S}$$

<u>Bit-Score</u>: Length-normalized version of HSP score:

$$S' = \frac{\lambda S - \ln K}{\ln 2} \qquad E = mn2^{-S'}$$