Sequence Alignment

Example:

>AVP78042.1 spike protein [Bat SARS-like coronavirus]

```
MLFFLFLQFALVNSQCDLTGRTPLNPNYTNSSQRGVYYPDTIYRSDTLVLSQGYFLPFYSNVSWYYSLTT
NNAATKRTDNPILDFKDGIYFAATEHSNIVRGWIFGTTLDNTSQSLLIVNNATNVIIKVCNFDFCYDPYL
SGYYHNNKTWSIREFAVYSFYANCTFEYVSKSFMLNISGNGGLFNTLREFVFRNVDGHFKIYSKFTPVNL
NRGLPTGLSVLQPLVELPVSINITKFRTLLTIHRGDPMSNNGWTAFSAAYFVGYLKPRTFMLKYNENGTI
TDAVDCALDPLSETKCTLKSLSVQKGIYQTSNFRVQPTQSIVRFPNITNVCPFHKVFNATRFPSVYAWER
TKISDCIADYTVFYNSTSFSTFKCYGVSPSKLIDLCFTSVYADTFLIRFSEVRQVAPGQTGVIADYNYKL
PDDFTGCVIAWNTAKQDTGHYFYRSHRSTKLKPFERDLSSDENGVRTLSTYDFNPNVPLEYQATRVVVLS
FELLNAPATVCGPKLSTQLVKNQCVNFNFNGLKGTGVLTDSSKRFQSFQQFGKDASDFIDSVRDPQTLEI
LDITPCSFGGVSVITPGTNTSSEVAVLYQDVNCTDVPTTIHADQLTPAWRIYAIGTSVFQTQAGCLIGAE
HVNASYECDIPIGAGICASYHTASILRSTGQKAIVAYTMSLGAENSIAYANNSIAIPTNFSISVTTEVMP
VSMAKTSVDCTMYICGDSIECSNLLLQYGSFCTQLNRALSGIAIEQDKNTQEVFAQVKQIYKTPPIKDFG
GFNFSQILPDPSKPSKRSFIEDLLFNKVTLADAGFIKQYGDCLGDISARDLICAQKFNGLTVLPPLLTDE
MIAAYTAALISGTATAGWTFGAGAALQIPFAMQMAYRFNGIGVTQNVLYENQKLIANQFNSAIGKIQESL
TSTASALGKLQDVVNQNAQALNTLVKQLSSNFGAISSVLNDILSRLDKVEAEVQIDRLITGRLQSLQTYV
TQQLIRAAEIRASANLAATKMSECVLGQSKRVDFCGKGYHLMSFPQSAPHGVVFLHVTYIPSQEKNFTTA
PAICHEGKAHFPREGVFVSNGTHWFVTQRNFYEPQIITTDNTFVSGNCDVVIGIINNTVYDPLQPELDSF
KEELDKYFKNHTSPDIDLGDISGINASVVNIQKEIDRLNEVARNLNESLIDLQELGKYEHYIKWPWYVWL
GFIAGLIAIVMVTILLCCMTSCCSCLKGCCSCGFCCKFDEDDSEPVLKGVKLHYT
```

>YP_009724390.1 surface glycoprotein [Severe acute respiratory syndrome coronavirus 2]

```
MFVFLVLLPLVSSQCVNLTTRTQLPPAYTNSFTRGVYYPDKVFRSSVLHSTQDLFLPFFSNVTWFHAIHV
SGTNGTKRFDNPVLPFNDGVYFASTEKSNIIRGWIFGTTLDSKTQSLLIVNNATNVVIKVCEFQFCNDPF
LGVYYHKNNKSWMESEFRVYSSANNCTFEYVSQPFLMDLEGKQGNFKNLREFVFKNIDGYFKIYSKHTPI
NLVRDLPQGFSALEPLVDLPIGINITRFQTLLALHRSYLTPGDSSSGWTAGAAAYYVGYLQPRTFLLKYN
ENGTITDAVDCALDPLSETKCTLKSFTVEKGIYQTSNFRVQPTESIVRFPNITNLCPFGEVFNATRFASV
YAWNRKRISNCVADYSVLYNSASFSTFKCYGVSPTKLNDLCFTNVYADSFVIRGDEVRQIAPGQTGKIAD
YNYKLPDDFTGCVIAWNSNNLDSKVGGNYNYLYRLFRKSNLKPFERDISTEIYQAGSTPCNGVEGFNCYF
PLQSYGFQPTNGVGYQPYRVVVLSFELLHAPATVCGPKKSTNLVKNKCVNFNFNGLTGTGVLTESNKKFL
PFQQFGRDIADTTDAVRDPQTLEILDITPCSFGGVSVITPGTNTSNQVAVLYQDVNCTEVPVAIHADQLT
PTWRVYSTGSNVFQTRAGCLIGAEHVNNSYECDIPIGAGICASYQTQTNSPRRARSVASQSIIAYTMSLG
AENSVAYSNNSIAIPTNFTISVTTEILPVSMTKTSVDCTMYICGDSTECSNLLLQYGSFCTQLNRALTGI
AVEQDKNTQEVFAQVKQIYKTPPIKDFGGFNFSQILPDPSKPSKRSFIEDLLFNKVTLADAGFIKQYGDC
LGDIAARDLICAQKFNGLTVLPPLLTDEMIAQYTSALLAGTITSGWTFGAGAALQIPFAMQMAYRFNGIG
VTQNVLYENQKLIANQFNSAIGKIQDSLSSTASALGKLQDVVNQNAQALNTLVKQLSSNFGAISSVLNDI
LSRLDKVEAEVQIDRLITGRLQSLQTYVTQQLIRAAEIRASANLAATKMSECVLGQSKRVDFCGKGYHLM
SFPQSAPHGVVFLHVTYVPAQEKNFTTAPAICHDGKAHFPREGVFVSNGTHWFVTQRNFYEPQIITTDNT
FVSGNCDVVIGIVNNTVYDPLQPELDSFKEELDKYFKNHTSPDVDLGDISGINASVVNIQKEIDRLNEVA
KNLNESLIDLQELGKYEQYIKWPWYIWLGFIAGLIAIVMVTIMLCCMTSCCSCLKGCCSCGSCCKFDEDD
SEPVLKGVKLHYT
```

- How do we know these two proteins are similar?
- Many existing tools: such as Clustal Omega.

Sequence Alignment

```
AVP78042.1          VNNATNVIIKVCNFDFCYDPYLSGYYHN-NKTWSIREFAVYSFYANCTFEYVSKSFMLNI        177
YP_009724390.1      VNNATNVVIKVCEFQFCNDPFLGVYYHKNNKSWMESEFRVYSSANNCTFEYVSQPFLMDL        179
                    *******:****:*:** **:*. ***: **:*   ** ***   ********: *::::

AVP78042.1          SGNGGLFNTLREFVFRNVDGHFKIYSKFTPVNLNRGLPTGLSVLQPLVELPVSINITKFR        237
YP_009724390.1      EGKQGNFKNLREFVFKNIDGYFKIYSKHTPINLVRDLPQGFSALEPLVDLPIGINITRFQ        239
                    .*: * *:.******:*:**:******.**:** *.** *:*.*:***:**:.****:*:

AVP78042.1          TLLTIHRGDP---MSNNGWTAFSAAYFVGYLKPRTFMLKYNENGTITDAVDCALDPLSET        294
YP_009724390.1      TLLALHRSYLTPGDSSSGWTAGAAAYYVGYLQPRTFLLKYNENGTITDAVDCALDPLSET        299
                    ***::**.      *..**** :***:****:****:************************

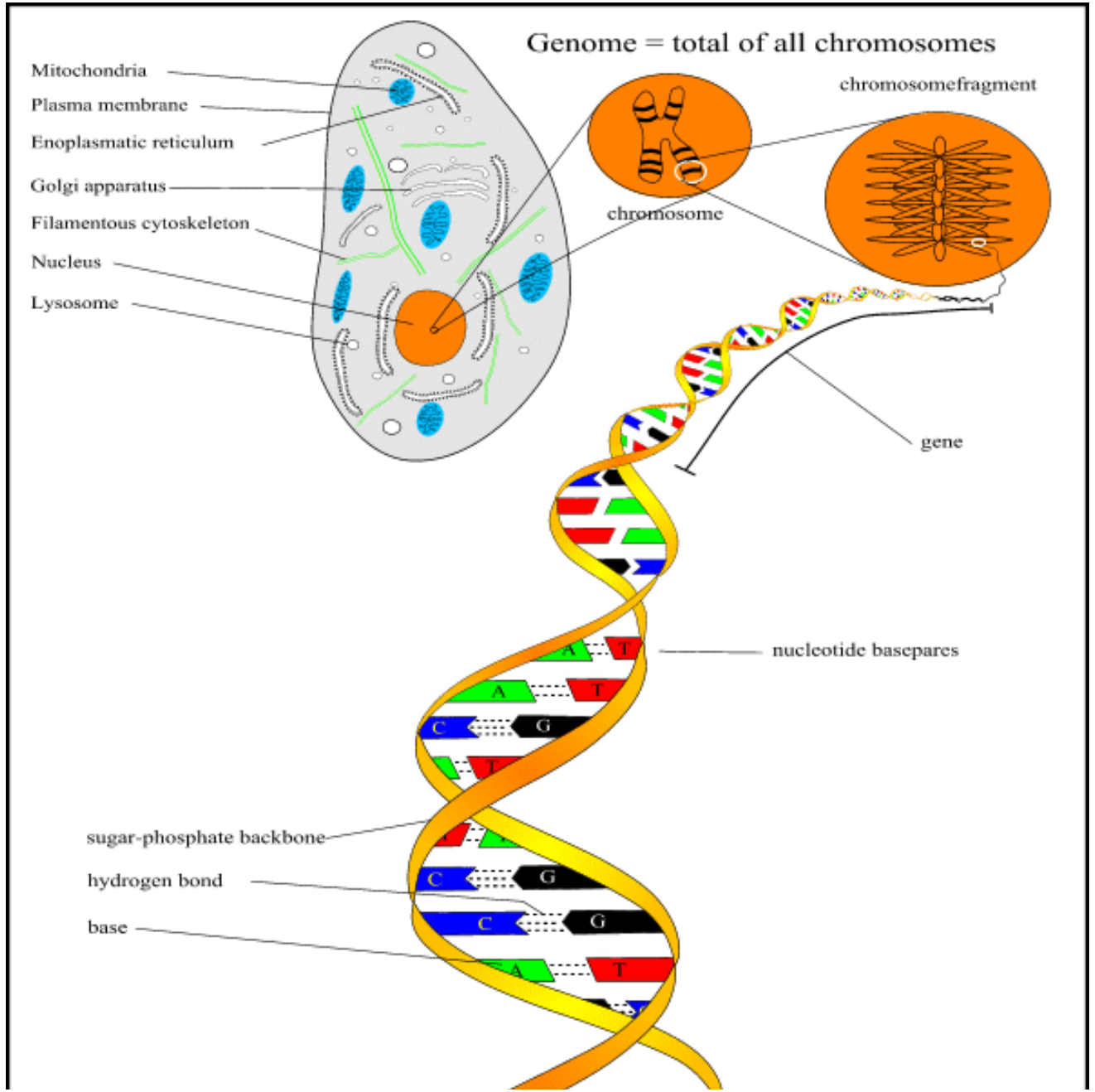
AVP78042.1          KCTLKSLSVQKGIYQTSNFRVQPTQSIVRFPNITNVCPFHKVFNATRFPSVYAWERTKIS        354
YP_009724390.1      KCTLKSFTVEKGIYQTSNFRVQPTESIVRFPNITNLCPFGEVFNATRFASVYAWNRKRIS        359
                    ******::*:***************:**********:*** :******* *****:*.:**

AVP78042.1          DCIADYTVFYNSTSFSTFKCYGVSPSKLIDLCFTSVYADTFLIRFSEVRQVAPGQTGVIA        414
YP_009724390.1      NCVADYSVLYNSASFSTFKCYGVSPTKLNDLCFTNVYADSFVIRGDEVRQIAPGQTGKIA        419
                    :*:***:*:***:************:** *****.****:*:** .****:****** **

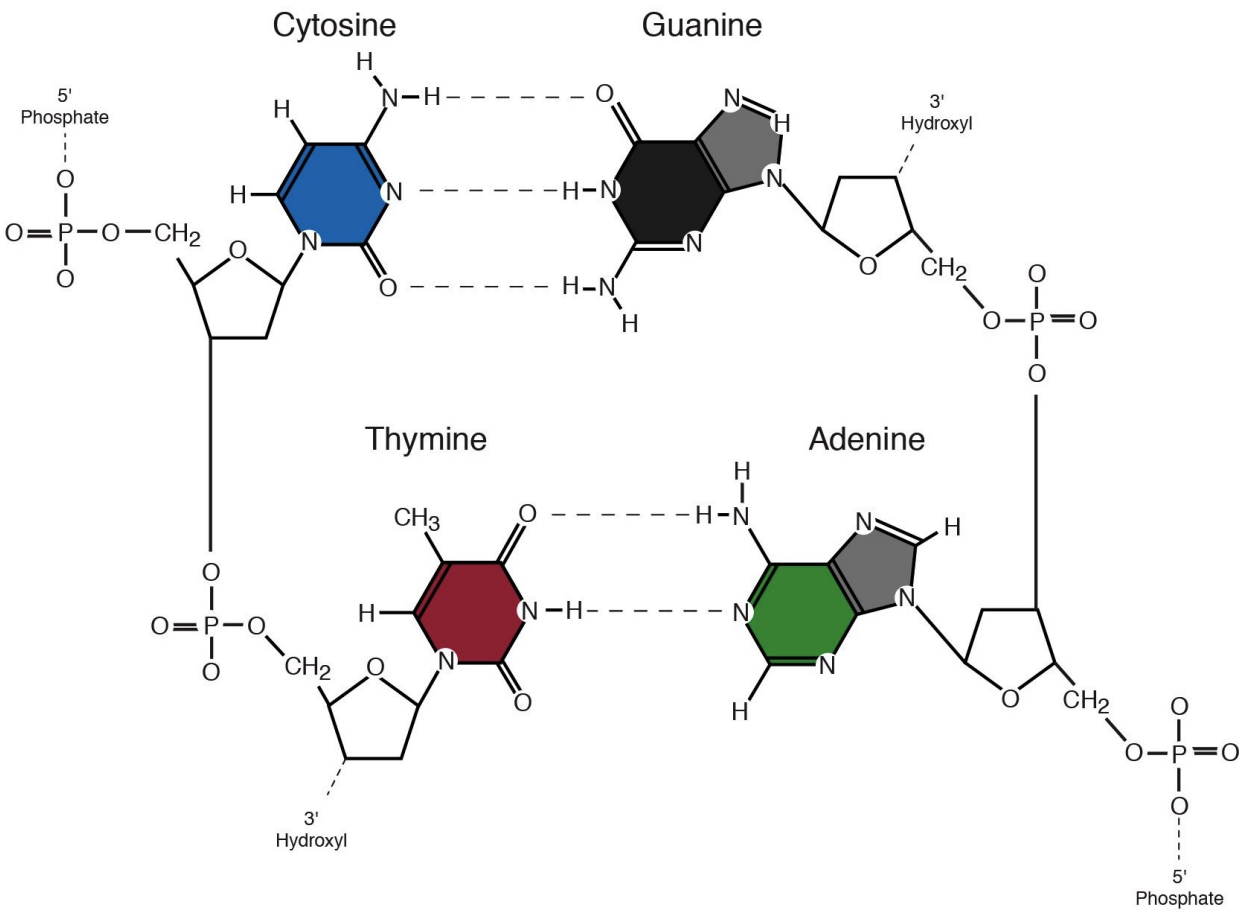
AVP78042.1          DYNYKLPDDFTGCVIAWNTAKQDT-----GHYFYRSHRSTKLKPFERDLSSDEN------        463
YP_009724390.1      DYNYKLPDDFTGCVIAWNSNNLDSKVGGNYNYLYRLFRKSNLKPFERDISTEIYQAGSTP        479
                    ******************: : *:      :*:** .*.::********:*::
```

- Too many identical positions to be random.
- Insertion/deletion (indel) needed for a proper comparison.

DNA


Genome = total of all chromosomes
Mitochondria
Plasma membrane
Enoplasmatic reticulum
Golgi apparatus
Filamentous cytoskeleton
Nucleus
Lysosome
chromosomefragment
chromosome
gene
nucleotide basepares
sugar-phosphate backbone
hydrogen bond
base
A
T
C
G

Nucleotide and Base Pairs



- Two classes of necleutide bases:
  - Purine: A and G
  - Pyrimidine: T and C
- Base pairs are due to hydrogen bonds.
- G-C bind stronger because of 3 H-bonds.
- DNA moleculre is oriented (5’ -> 3’).

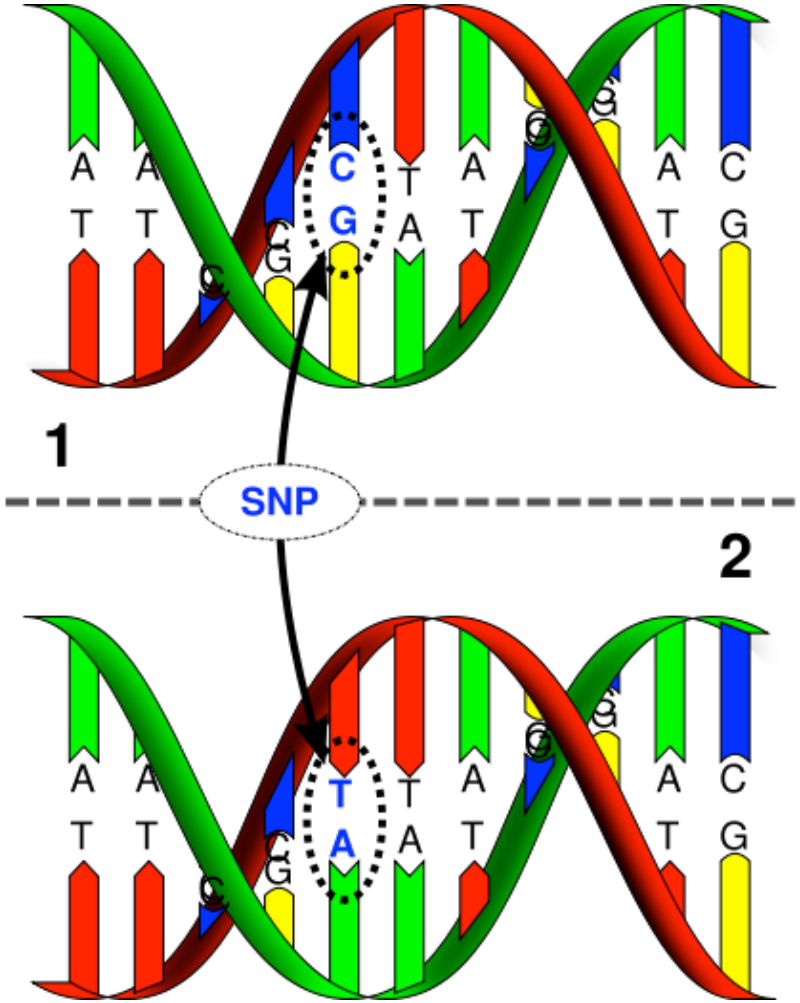
Reverse Complement a DNA Sequence

- DNA is double-helical, with two complementary strands.
- Complementary bases:
  - Adenine (A) - Thymine (T)
  - Guanine (G) - Cytosine (C)
- Example: What is the reverse complement of AAGGTAGC?

DNA Mutation

- DNA mutates with a small probability when inherited by the offspring.
  - For example, one base can be substituted by another.
  - This creates different alleles of the same gene.
  - An **allele** is a variant form of a gene at the same location of the genome among different individuals.
- Also, one only inherits half of each parent's genome.
- These together cause the differences between individuals of the same species.

Single Nucleotide Polymorphisms



- Single base variation between members of a species.
- For Human, 90% of all human genetic variation is caused by SNPs. SNPs occur every 100 to 300 bases along the 3-billion-base human genome.
- Major risk for genetic disease.

Compare DNA sequences

- The most often used distance on strings in computer science is Hamming distance.
  - **AGTTTAATCA**
  - | | | | | | | |
  - **AGTATAACGA**
- This makes some sense on comparing DNA sequences in some cases. But there are other mutations
  - Substitution ACAGT → ACGGT
  - Insertion/deletion (indel) ACAGT → ACGT
- Other DNA rearrangements can also happen. But substitutions and indel are the two mutations we concern the most for this course.

Edit Distance

- Let's focus on substitution and indel only. How "far" away are two sequences from each other?
- E.g. CGATA and GGATTA

- **Edit distance**: the minimum number of edit operations needed to convert one to another. Here edit operations include substitutions and indels.

```
d(ATGCATTTA,ATGTACTTTC)
    ATGCATTTA
    ATGTACTTTC
```

Edit distance is a distance metric

- Identity: d(x,y)=0 iff x=y
- Symmetry: d(x,y) = d(y,x)
- Triangular Inequality: d(x,z) <= d(x,y) + d(y,z)

Preparation for the Algorithm

- For convenience of the proof, we treat each occurrence of the same letter different.
- E.g. ATAA -> ATA can be done by either deleting the 2^{nd} or 3^{rd} letter A from the first string. These are different editing paths.
- This does not affect our definition of edit distance, but makes our later proof more precise.

Dynamic Programming Algorithm for Edit distance

- Let D[i,j] = edit distance between S[1..i] to T[1..j].

- Consider the edit operations associated with S[i] and T[j] in the optimal edit operations. One of the following cases will happen (why?):
    1. S[i] is deleted
    2. T[j] is inserted
    3. S[i] becomes T[j]

Recurrence Relation

- $D[i,j] = \min \begin{cases} D[i-1,j]+1 \\ D[i,j-1]+1 \\ D[i-1,j-1]+\delta(S[i],T[j]) \end{cases}$
- Here $\delta(S[i],T[j]) = 0$ if S[i]=T[j] and 1 if not.

Dynamic Programming Algorithm for Edit distance

- D[0,0] = 0.
- D[0, i] = i for i=1..|S|
- D[i, 0] = i for i=1..|T|
- for i from 1..|S|
- for j from 1..|T|
- D[i,j] = min {D[i-1,j]+1, D[i,j-1]+1, D[i-1, j-1]+d(S[i], T[j])}.
- Return D[|S|, |T|]

A Note about Dynamic Programming

- Define “subproblems”
- Develop recurrence relation to compute subproblems
- Initialization (base cases)
- Determine the computation order for solving the subproblems.
  - Usually bottom-up (smaller to larger)
- Find the solution of the original input

Longest Common Subsequence

- The second way to evaluate the similarity of two sequences is through LCS.
- A subsequence is obtained by deleting some of the letters from the supersequence and concatenating the remaining letters together.
- What is the LCS of the following two sequences?
  - `ATGCATTTA`
  - `ATGTACTTTC`
- LCS can be computed with dynamic programming as well. (Exercise)

Alignment

- The third way to compare to sequences is through sequence alignment.
- Align the two sequences by inserting spaces, so that they are the most **similar** column-wisely.
  - ```
    ATGCA-TTTA
    ||| | || |
    ATGTACTT-A
    ```
- What does “similar” mean? Usually we need a “scoring function” or a “score function”.
- Let’s define the alignment score to be **the total of column scores**. And each column is assigned by a constant score depending on matching conditions.
- E.g. Match = 1, mismatch =-1, indel =-1. This is sometimes called the “score scheme”.
    ```

Two Example Alignments

- AATGCGA-TTTT
  || | |||
  G-TG--ACTTTC

- AATG-CGATTTT
  || | ||
  G-TGAC-TTTC-

- Which of the two alignments better?

- +1 = match
- -1 = mismatch
- -1 = indel

Alignment with Dynamic Programming

- Now we develop the dynamic programming algorithm for alignment.
- Scoring scheme is f(a,b) for a column with a and b. Here one of a and b can be the dash character -.
- f(-,x) and f(x,-) represent scores of indels.

Last column of an alignment

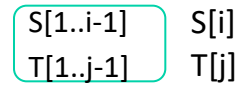
- Suppose we are to align S[1..i] and T[1..j]. Consider the last column of the optimal alignment. Three cases can happen:

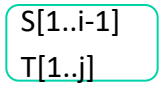
(1)		(2)		(3)	
S[1..i-1]	S[i]	S[1..i-1]	S[i]	S[1..i]	-
T[1..j-1]	T[j]	T[1..j]	-	T[1..j-1]	T[j]

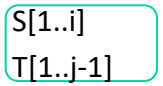
- In each case, the sub-alignment without the last column is an optimal one (why?)

Recurrence Relation

- Denote the optimal alignment score of S[1..i], T[1..j] by D[i,j]. Then D[m,n] is the optimal alignment score.
- Let f(a,b) be the score between two letters a and b.
- Consider last column of the alignment.
- Case 1: S[i] v.s. T[j]
  - D[i,j] = D[i-1, j-1] + f(S[i], T[j]);
- Case 2: S[i] v.s. -
  - D[i,j] = D[i-1, j] + f(S[i], -);
- Case 3: - v.s. T[j]
  - D[i,j] = D[i, j-1] + f(-, T[j]);
- Therefore…







$$D[i,j] = \max \begin{cases} D[i-1, j-1] + f(S[i], T[j]); \\ D[i-1, j] + f(S[i], -); \\ D[i, j-1] + f(-, T[j]); \end{cases}$$

Algorithm

D[0,0] = 0;

for i from 1 to m

D[i,0] = i* indel;

for j from 1 to n

D[0,j] = j* indel;

for i from 1 to m

for j from 1 to n

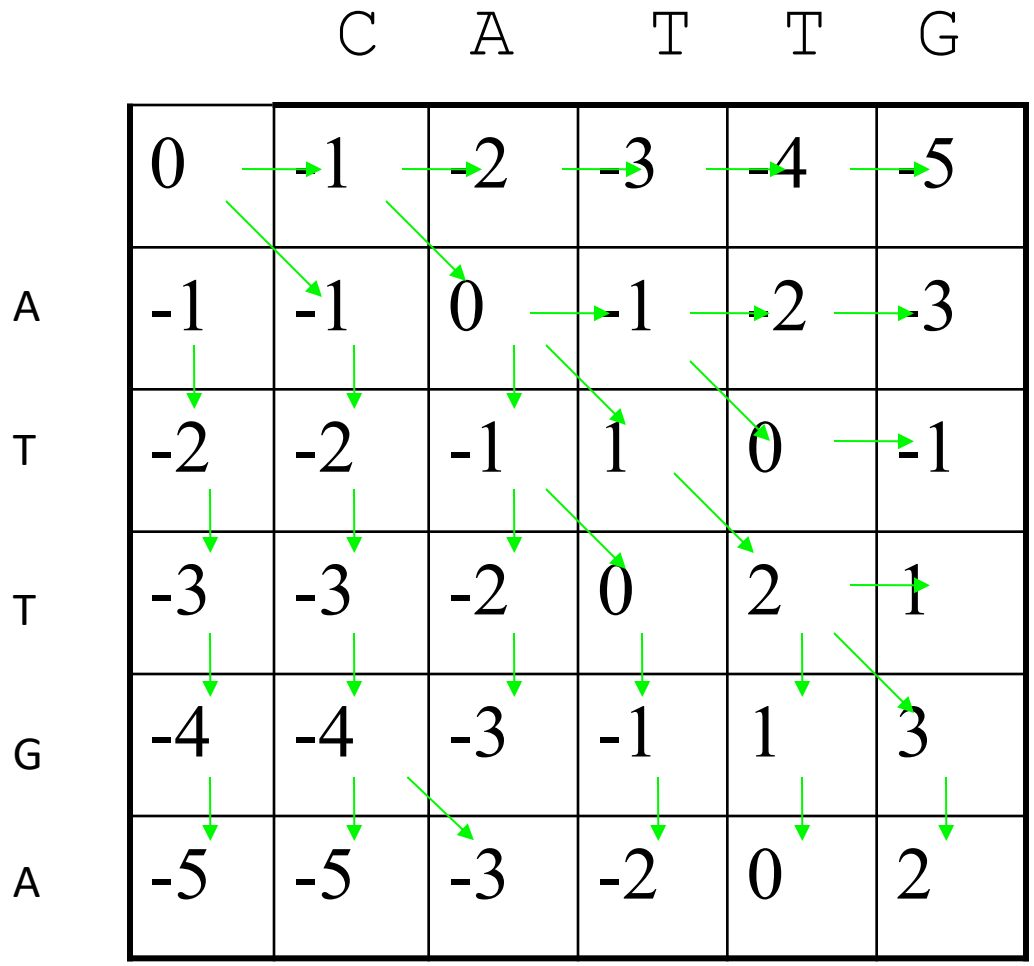
$$D[i,j] = \max \begin{cases} D[i-1, j-1] + f(S[i], T[j]); \\ D[i-1, j] + f(S[i], -); \\ D[i, j-1] + f(-, T[j]); \end{cases}$$

Output D[m,n];

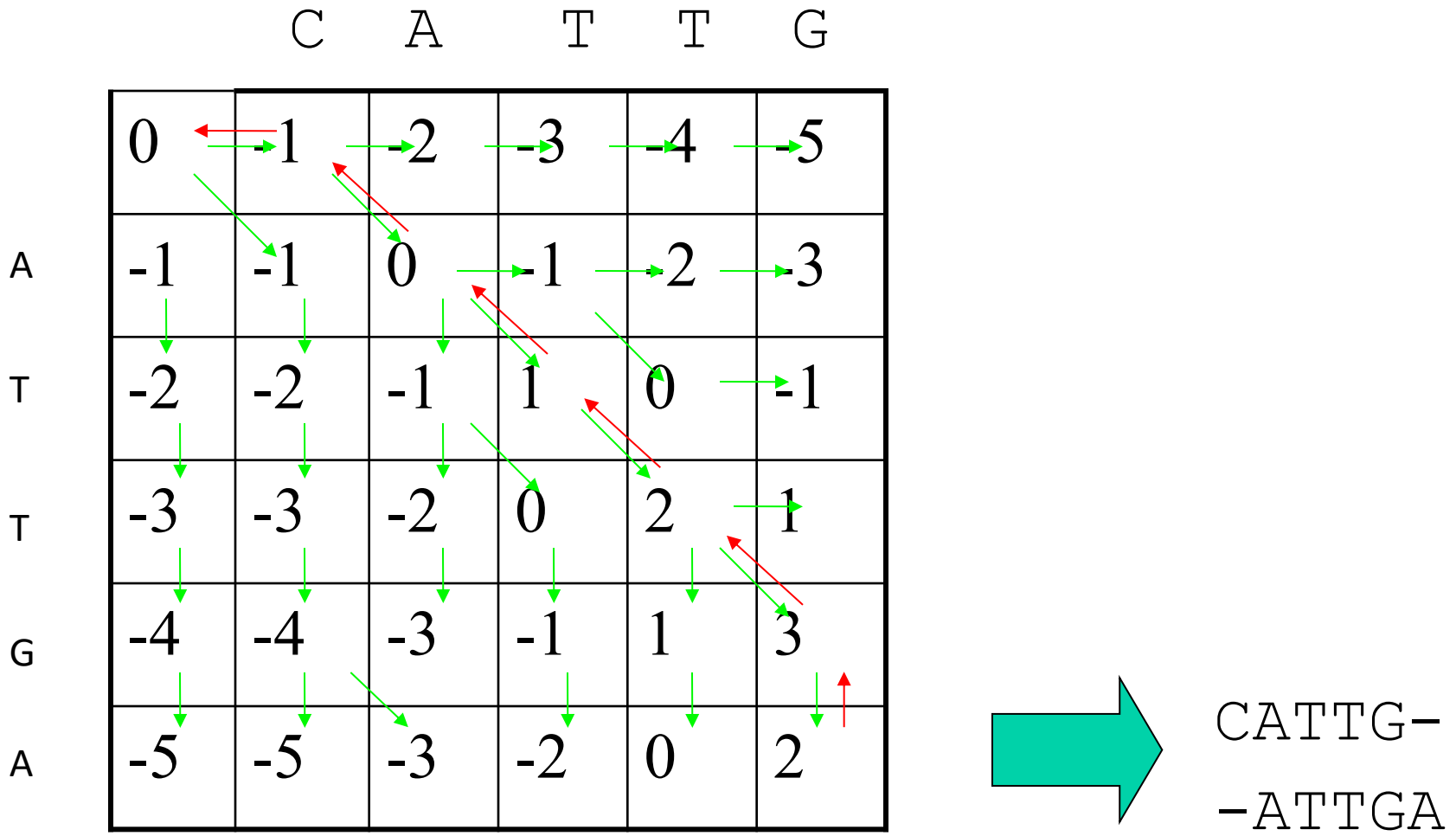
Dynamic Programming Table

		C	A	T	T	G
	0	-1	-2	-3	-4	-5
A	-1	-1	0	-1	-2	-3
T	-2	-2	-1			
T	-3					
G	-4					
A	-5					

Dynamic Programming Table



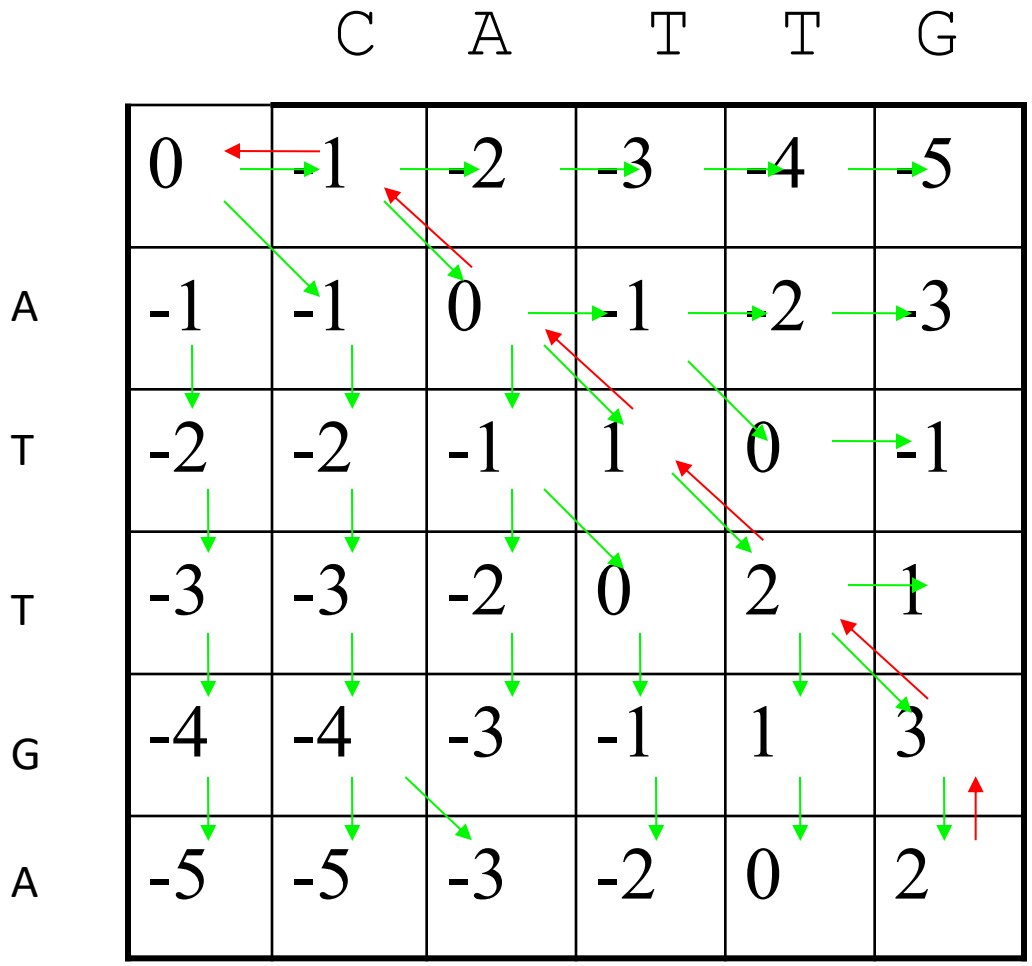
Getting the actual alignment – backtracking



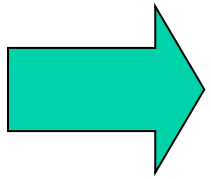
Complexity

- **Time Complexity**:
  - Filling the table takes O(nm) time: Each step requires only 3 checks to other points in the matrix.
  - How about the backtracking?
- Space Complexity:
  - O(nm)

A Practical Trick



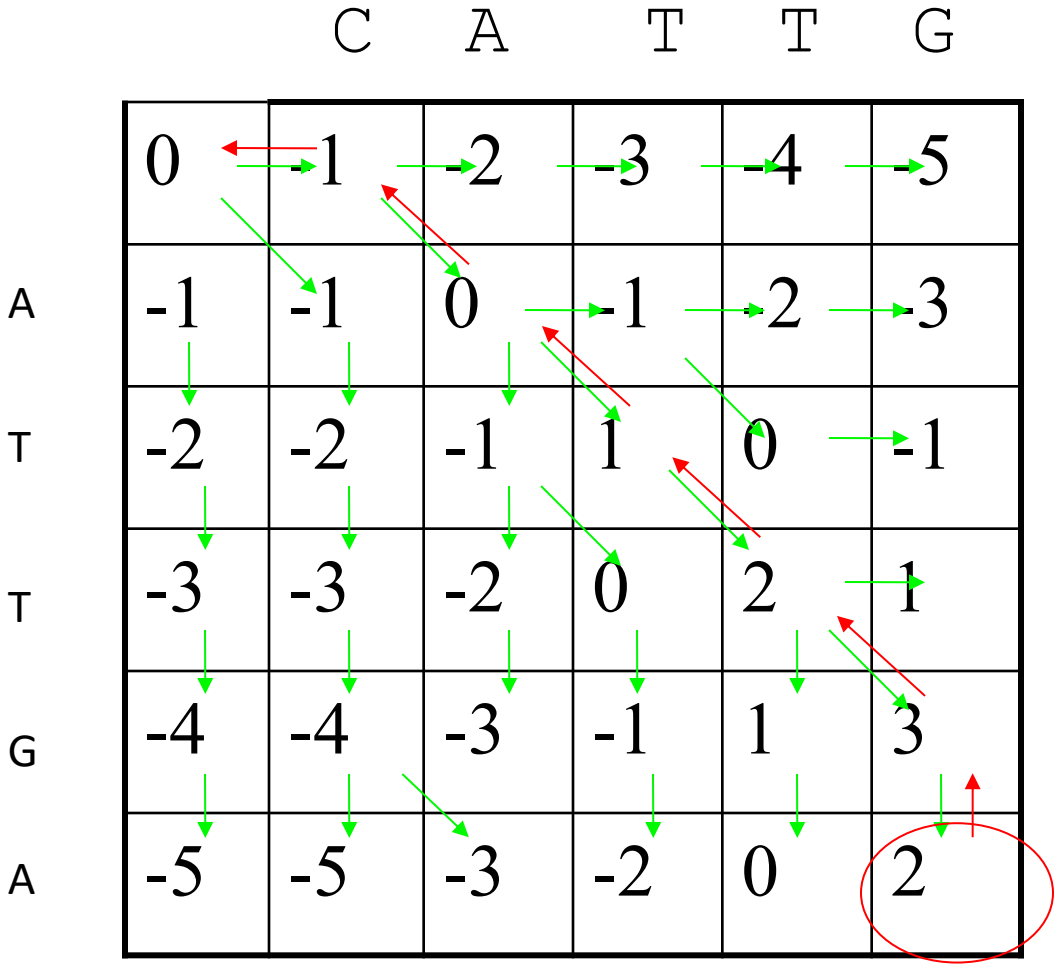
No need to physically record the green arrows. Why?



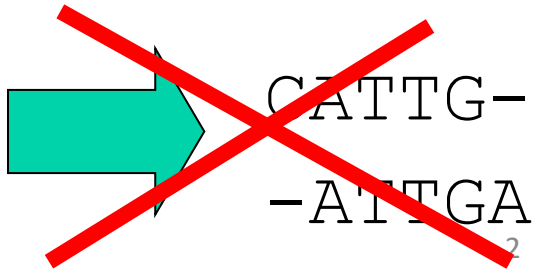
CATTG-

-ATTGA

Another Trick



If only score is needed, then space complexity can be reduced.

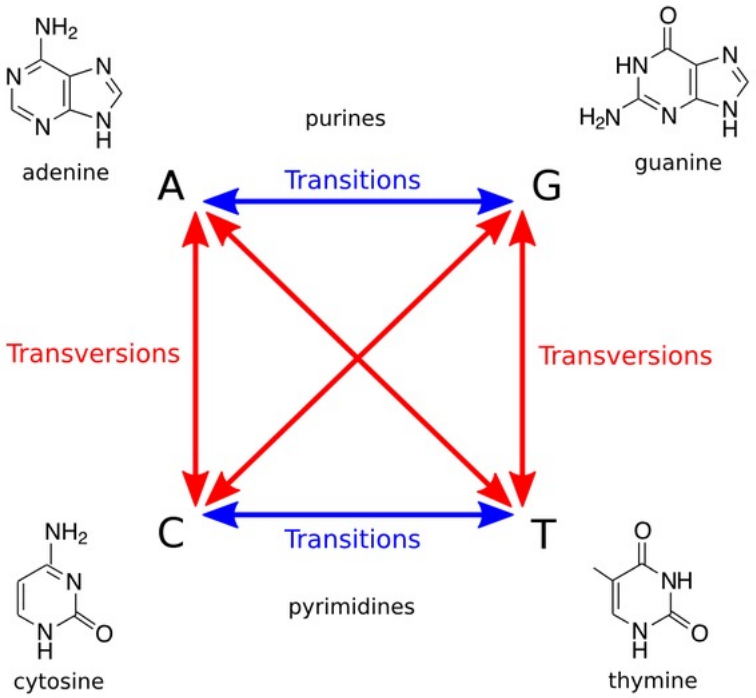


Score Function

- Now we have the algorithm for any score scheme f(x,y)
- Such separation of scoring and algorithm is a good thing. It allows us to optimize the score scheme independent to the algorithm.

> The effective exploitation of his powers of abstraction must be regarded as one of the most vital activities of a competent programmer.
>
> - Dijkstra

Transition vs. Transversion



- Transition happens more frequently 2/3 of SNPs are transitions.
- In other words, transition is easier and therefore should be less penalized.

E.g.:

```
AAAGCAAA        vs      AAAGCAAA
AAAT-AAA                AAA-TAAA
```

- This can be easily achieved by changing score scheme f(a,b).

Alignment v.s. LCS vs. Edit Distance

- By a properly defined score scheme, alignment can represent LCS and Edit distance, respectively.
  - match =
  - mismatch =
  - indel =

How to Build a Score Function

- First, know what you want.
- **Purpose 1**: the optimal alignment reveals the true evolutionary history.
- **Purpose 2**: high score indicates homology (derived from same ancestor).
- We want purpose 1 if possible, but purpose 2 is also useful.

```
ATGCA-TTTATTCCGAGG
|||  |  ||  |||  ||  ||
ATGTACTT-ATTACGTGG
```

Philosophy of a Score Function

- For purpose 1, right away: we might be **wrong**.
- That is, the alignment that has highest **score** may not be the one that actually matches evolutionary history.
- So you should never trust that an alignment must be right. It just optimizes the score.
- Should we give up purpose 1 at all?

Philosophy of A Score Function

- For purpose 1, the optimal alignment may be **approximately** correct **under certain conditions** in practice.
- As long as we know the limitation, we can still use it.
- For example, for the following alignment, it is “very likely” the alignment is approximately equal to the evolutionary history.
  - `ACGTATTACCGG-TTACCG`
  - `|||  ||||||||  ||||||`
  - `ACGGATTACCGGATTACCG`
- Limitation we keep in mind: when score is low, alignment itself is not too useful.

Gaps

```
AGATTTTTTTC        AGATTTTTTTTTC
AGA---TTTTC        AGA-T-T-T-T-C
```

- The left seems “simpler” than the right.
- Indeed, during evolution, indels are relatively rare. However, insertion or deletion a segment of k consecutive bases is much easier than k scattered indels.
- But our current scoring method (adding up column scores) cannot distinguish the two.
- Currently, a gap of length k costs k*indel. Thus, this is called the **linear gap penalty**.

Arbitrary gap penalty

- Consecutive insertions or deletions are called a gap. Suppose the gap penalty of a length k gap is g(k) instead of the simple c*k.
- Assume g(x)+g(y)<= g(x+y). (Otherwise does not serve the purpose of grouping indels.)
- Can the old DP still work?

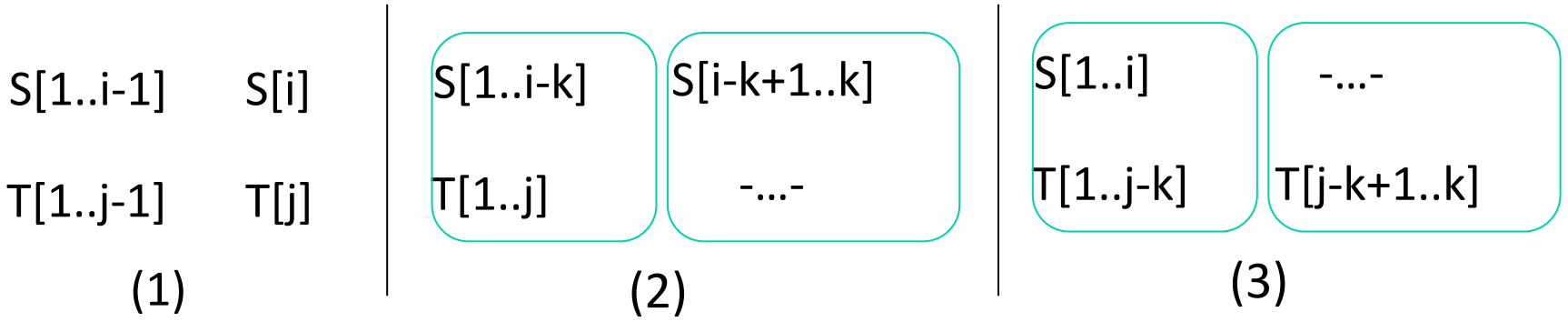
S[1..i-1]	S[i]	S[1..i-1]	S[i]	S[1..i]	-
T[1..j-1]	T[j]	T[1..j]	-	T[1..j-1]	T[j]
	(1)		(2)		(3)

Arbitrary Gap Penalty

- Old algorithm does not work anymore because we do not know the contribution of the last column to the gap penalty in the last two cases.
- The length of the gap is needed.

Alignment Algorithm for Arbitrary Gap Penalty

- We still use D[i,j] to denote the optimal alignment score of S[1..i] and T[1..j].
- We change cases 2 and 3 to include the last gap (not the last column).
- D[i,j] = max of the following three cases:
  - D[i-1,j-1]+f(s[i],t[j]). (s[i] v.s. t[j])
  - $\max_{1 \le k \le i} D[i-k,j] + g(k)$
  - $\max_{1 \le k \le j} D[i,j-k] + g(k)$



Time Complexity

- Cubic time complexity.
- In bioinformatics, very often we face the choice between:
  - Reality: How close it approximates the real biology
  - Simplicity: How easy it can be computed
- Now let's simplify the g(k) a little. We basically want a function that grows slower than linear.
- g(k) = a + b*k
  - a = gap open penalty
  - b = gap extension penalty
- This is called **affine gap penalty,** in contrast to linear gap penalty**.**

Affine gap penalty example

For example: match = 1; mismatch = -1; gap open = -5; gap extension = -1.

- `ATAGG--AAG`
- `|| ||  | |`
- `ATTGGCAATG`
- `6 match, 2 mismatch, 1 gap open, 2 gap extension, score = ?`
- `ATAGG-AA-G`
- `|| || || |`
- `ATTGGCAATG`

Old Algorithm Does not Work

- Consider the last column of an alignment again:

```
AT-GG-        ATGG--
||  ||        ||  |
ATTGGC        ATTGGC
```

- When the last column is an indel, the added cost depends on the previous column.
  - If previous column has a gap opened already, then
  - D[4,6] = D[4,5] + gapext
  - Else
  - D[4,6] = D[4,5] + gapopen + gapext
- How do we know the previous column's configuration?
- Because by induction we know the optimal solution for D[i,j-1], can we simple look at it and use the configuration?

Algorithm for Affine Gap

- We compute the optimal solution by limiting the last column to one of the following three configurations:

```
ATAGG      ATAGG-      ATAGGC
|| ||      || ||       || ||
ATTGG      ATTGGC      ATTGG-
```

$D_0[i,j]$ $D_1[i,j]$ $D_2[i,j]$

- We only distinguish them by the last column, there is no constraint for columns before the last column.

Recurrence Relation

$$D_0[i,j] = f(s[i], t[j]) + \max \begin{cases} D_0[i-1, j-1]; \\ D_1[i-1, j-1]; \\ D_2[i-1, j-1]; \end{cases}$$

$$D_1[i,j] = \text{gapext} + \max \begin{cases} D_0[i, j-1] + \text{gapopen}; \\ D_1[i, j-1]; \\ D_2[i, j-1] + \text{gapopen}; \end{cases}$$

$$D_2[i,j] = \text{gapext} + \max \begin{cases} D_0[i-1, j] + \text{gapopen}; \\ D_1[i-1, j] + \text{gapopen}; \\ D_2[i-1, j]; \end{cases}$$

Note the grayed cases can't be optimal so can be safely removed.

Algorithm

- No difference to the simple DP but now uses three arrays.
- Backtracking should be very careful!
- Still O(nm) time. Approximately 3 times slower.
- This is okay because the model is more expressive.
- Much faster than the general gap penalty.

Gotoh, O., 1982. An improved algorithm for matching biological sequences. *Journal of molecular biology*, *162*(3), pp.705-708.

Review: Evolution and alignment

- Two sequences always arise from a common ancestor.
- Since that ancestor lived, there have been a long number of descendants, leading up to the present time.
- A full evolutionary history would detail the mutations that happened over the course of history.
- We don’t have a time machine.
- The next best thing: alignments.
- Characterize which positions in the two sequences arose from the common ancestor.
- Between these, “indel” mutations.

Review

- DP algorithm for alignment
- Matrix entry: score of best alignment of S(1…i) to T(1…j).
- Can compute matrix entries in constant time ➔O(nm) runtime.
- Can backtrack through matrix to find optimal alignment.
- If only score is needed, then linear space.
- Scoring function important
- Some do not change DP (better scoring scheme)
- Some change (gap penalty)
- General gap penalty cubic time.
- Affined gap penalty still quadratic time.