BMEG 3105 Fall 2024

Data analytics for personalized genomics and precision medicine

Lecturer: Yu LI (李煜) from CSE

Lecture 3: Sequence data and DP (13/9/2024)

Sequence Data:

1. Data Types
2. Python Programming
3. Sequence Data
4. Sequence Comparison and Alignment Score

1. Data Types

All data is stored as different formats, here in this lecture, 8 types of data were introduced with simple examples of each corresponding types.

1.1 Sequential Data

Definition:

➔ Sequential data is data arranged in sequences where order matters.

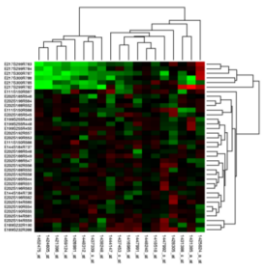
Examples:

1.2 Data Matrix

Definition:

- As tables or charts.
- Consists of a collection of records, each of which consists of a fixed set of attributes.
- Can be represented by an n by m matrix:
  n rows = n objects, m columns = m attribute.

Examples:



Person	Height (m)	Weight (kg)
P1	1.79	75
P2	1.64	54
P3	1.70	63
P4	1.88	78

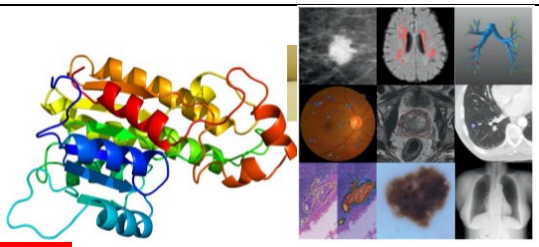
**Data stored WON'T be affected if the entire row or column is shuffled. **

1.3 Spatial Data

Definition:

- As pictures or images.
- Geographic locations and spatial information (related to where things are).

Examples:



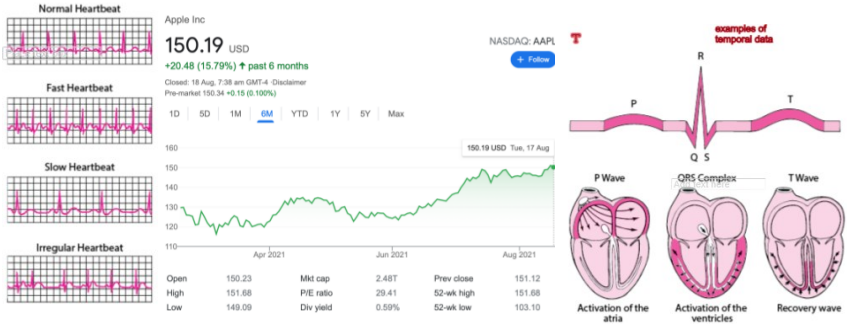
**Data stored WILL be affected if the entire row or column is shuffled. **

1.4 Temporal Data

Definition:

- As graphs or other data representations involving time.

Examples:

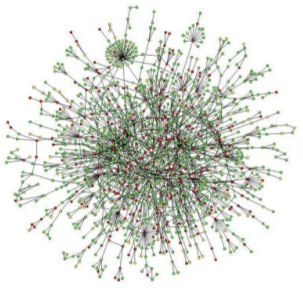


1.5 Graph or Networks

Definition:

- Involving objects and connections
- Social network and PPI network (protein-protein interaction network)

Examples:



1.6 Text

Definition:

- As sentences (both long and shotr) or text documents

Examples:

Results

Generation of a Lentiviral Vector System and Stable 4T1-luc2 Cell Lines

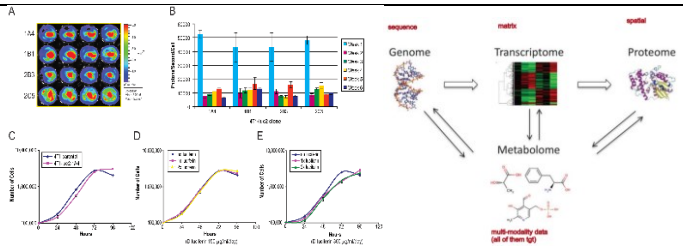
A lentiviral vector containing the firefly *luc2* gene conjugated to a human ubiquitin C promoter was constructed to generate stable bioluminescent cancer cell lines[16], [17], [18]. Mouse mammary tumor 4T1 cells were then transfected with the lentiviral vector and stable clones were selected using puromycin (4T1-luc2). Eight clones were chosen for further analyses and their luciferase activities were monitored for four weeks without selection marker. Although there were variations among the clones, the majority of them clones emitted more than 3,000 photons/sec/cell of light. Considering that most cell lines labeled with previous generations of luciferase emit less than 250 photons/sec/cell, our luc2 clones exhibited considerably higher level of light emission[20]. Surprisingly, one clone (C26) initially emitted as much as 52,000 photons/sec/cell but its light emission decreased to 6,400 photons/sec/cell after four weeks (Figure S1A). To confirm that no alteration of cellular physiology occurred during the labeling/cloning process, we compared the clones to the original parental 4T1 cells on several different levels. First, we examined growth patterns. From the eight initially selected clones, we chose two lines (C27 and C38) and compared their growth patterns to the parental 4T1 cells. Both lines had similar doubling times to the parental cells (12.6 hour doubling time for both clones, versus 12.0 hours for the original 4T1 cells).

1.7 Multi-modality Data

Definition:

- As a mixture of different types of data

Examples:



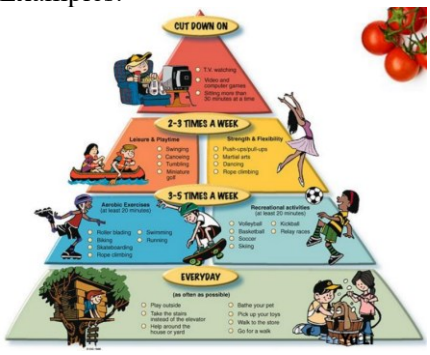
- Video: Temporal images + audio + transcript
- Electronic health records: Data matrix (blood test) + images + text (personal statement / diagnosis)
- Spatial transcriptomics: Spatial data + sequence + data matrix

1.8 Unknown Data Types

Definition:

- The data have not been shown to us
- Data that is yet to be digitalised

Examples:



Diet and Exercise

2. Python Programming

Simple definition: A thing that is used to communicate with computers with specific "languages" so that the computers can executes outputs as our desires and commands. Think of a translator between me (a living thing) and a computer (a non-living thing).

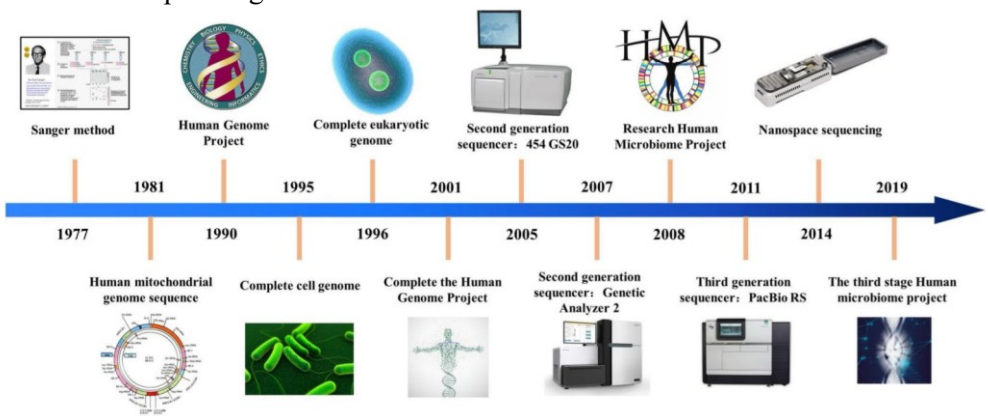
Some simple functions of Python coding			
"Numpy"	• Additional plug-in function for actions related to numbers and values. • Needs to be loaded once before use → "import numpy"	import numpy a = [1, 2, 3] numpy.mean(a) numpy.std(a) numpy.median(a) numpy.max(a) numpy.min(a)	*(load the plug-in)* *(define variable)* *(find mean)* *(find standard dv.)* *(find median)* *(find max)* *(find min)*
"Print"	• For printing out something • No need to be loaded, but type each time to use → "print( )"	print("a", b, numpy.mean(a)) *Use "" if want to get exactly what's inside the "". *Use the variable only if want to print out what's inside the variable *Can print out executions (numpy.mean(a) as an example here)	

3. Sequence Data

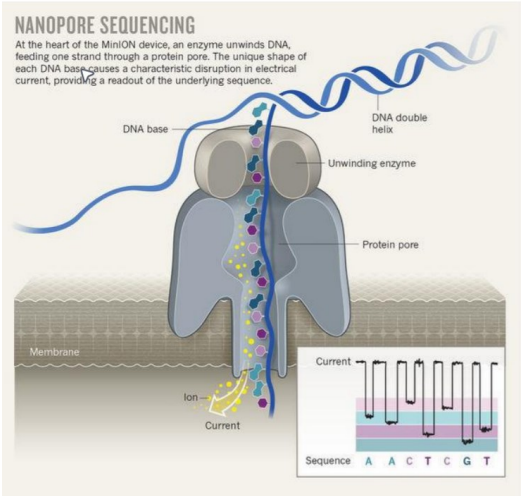
1. Central dogma
2. Genetic information in DNA sequences
3. Phenotypes = genotype + environment
4. DNA sequences (ATCG)
5. RNA sequences (AUCG)
6. Protein sequence (20 amino acids)

Sequences Obtaining

- DNA/RNA sequencing

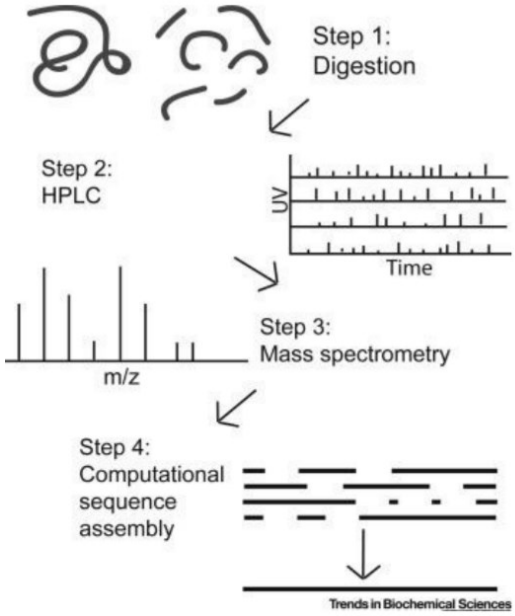


  - Nanopore sequencing



DNA strands pass through a chemical pore, different bases (ATCG) then generate changes in electrical current. Sequencing is done by detecting the regarding changes in current.
Capable for sequencing very long DNA sequences (up tp 3Mb) but with a high error rate (5%)

- Protein sequencing – mass spectrometry (MS)



Protein sequences are broken down into shorter pieces, then each piece of sequence is determined by the weight differences by MS. After that, the short pieces are assembled into raw sequences.

Raw sequence processing

- DNA sequences:
  Quality control
  Map reads to reference genome
  variant calling
  Phenotype associated variant

- Protein sequences:
  Sequence comparison: similar sequence → similar structure → similar function (Biomolecular function and property prediction)
  Multiple sequence alignment: Homology (Possible common ancestor) (evolution, identifying conservative region, investigating mechanism)

4. Sequence Comparison and Alignment Score

To determine the similarities between sequences and identifying regions of similarity by detecting the alignment score through sequence comparison. Pairwise sequence alignment is used for sequence comparison, that is, by arranging 2 sequences to maximise the similarity between them. Before starting the alignment, similarity between bases needs to be maximised by inserting gaps.

ATCG _ _ _ _
_ _ _ _ ATCG ← these 2 sequences are identical, but such original arrangement makes them very different if we calculate the alignment score directly based on this directly → leads to wrong conclusions!

Alignment score is determined by the combination (the total) of the 2 compared sequences, and such score is calculated differently under 3 conditions:

Match (identical bases)	Mismatch (Substitution)	Gap (Insertion / deletion)
A-A, T-T, C-C, G-G	A-T, T-G, C-A etc…	A-_, T-_, C-_, G-_

Sequence alignment score and scoring matrix:

Scoring matrix:

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

Gap penalty = -10

gap is the most different, therefore assign that as -10

Alignment score 1 = 2 + (-7) + 2 + 2 + (-10) + 2
= **-9**

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

Alignment score 2 = 2 + (-7) + 2 + 2 + (-7) + (-10)
= **-18**

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

➔ Higher the alignment score = higher the similarity between the comparing sequences.

Dynamic Programming (DP):

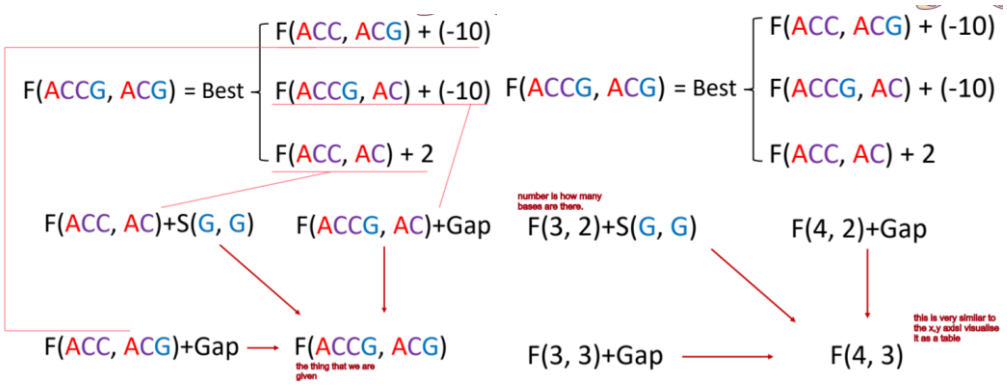
➔ Break problems into smaller sub-problems
➔ Solve sub-problems optimally and recursively
➔ Optimal solutions construct optimal solutions

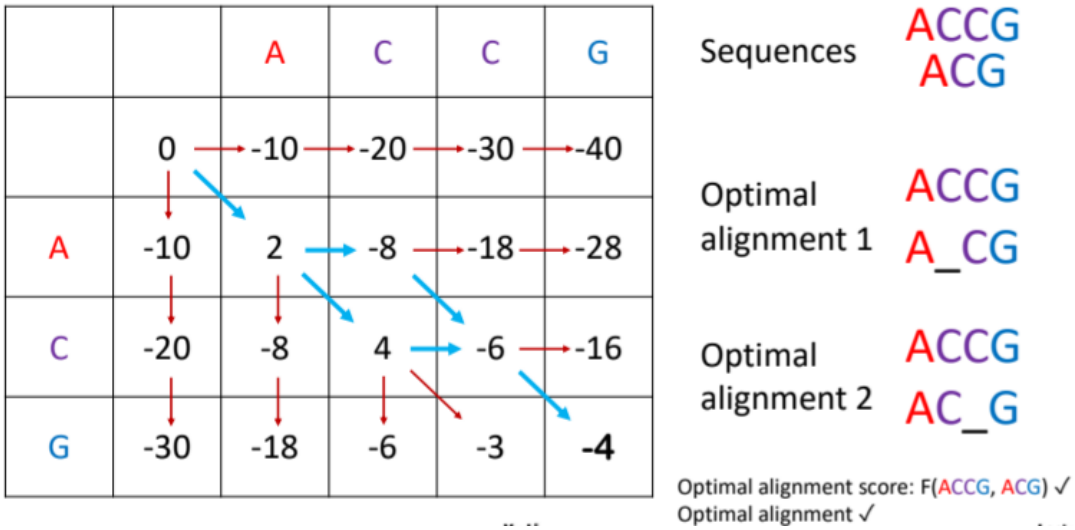
By implementing DP, the tedious procedures of enumeration can be skipped.
ie. If solve by enumeration….

➢ Too many possible alignments!!!
➢ Align two sequences with length n
➢ $\binom{2n}{n} = \frac{(2n)!}{(n!)^2}$

Sequence alignment with DP:

Logic behind: Starting from the last pair of the alignment (the pre-destination), then reduce options one by one according to the alignment score, that is, producing a recursive solution. Let's say, the 2 sequences we're comparing consist of 4 bases and 3 bases respectively: ACCG and ACG, we can represent the original set as: F(ACCG, ACG). After the recursive procedure, the original set will be ultimately reduced to F(X, X), or F(X, _), these are called the boundary case.





By filling the dynamic programming table, and working backward from the last pair of bases, the optimal alignment solution is obtained.

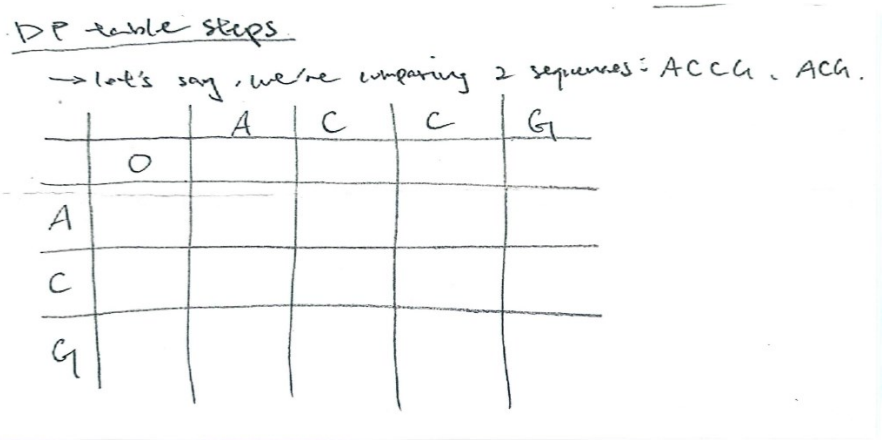
*Red arrows: progress; blue arrows: the highest alignment score
** Before finding F(4,3) (this is the optimal solution), the intermediate left, up and diagonal cells around F(4,3) needs to be found, and such score is calculated depending on the diagonal score.

Below are the detailed steps of making a DP table for finding the optimal score and alignment(s):

Scoring matrix:

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

Gap penalty = -10

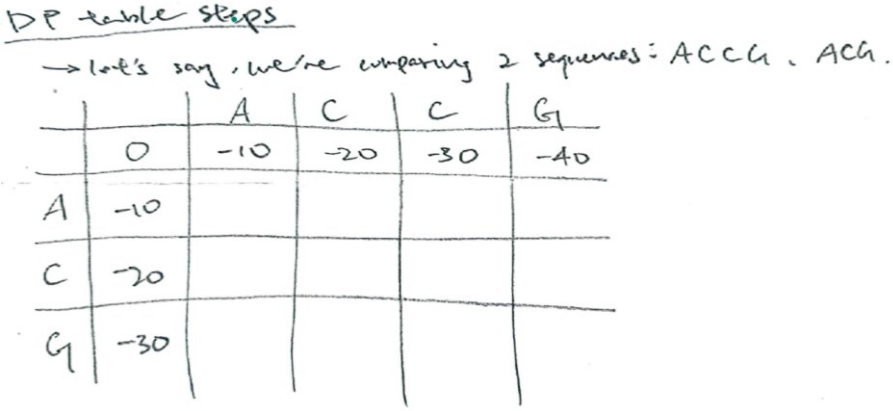


1) Draw out the DP table by arranging sequences that are going to be compared.

Scoring matrix:

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

Gap penalty = -10

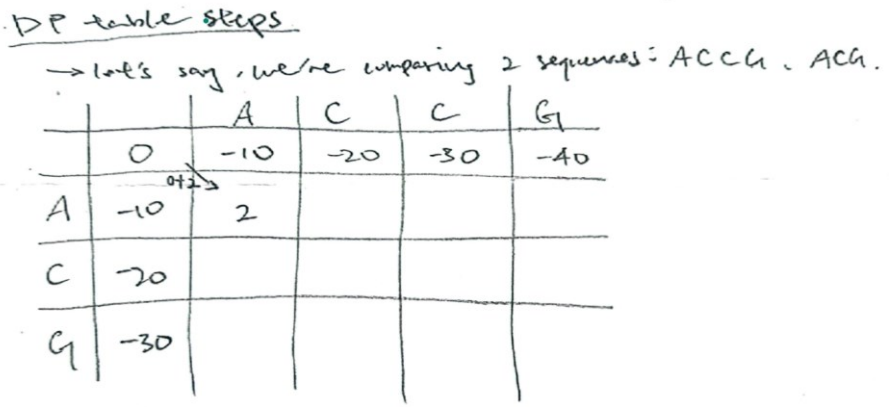


2) Each vertical / horizontal "move" result a gap, thus, receive a penalty of -10. Fill these first.

Scoring matrix:

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

Gap penalty = -10



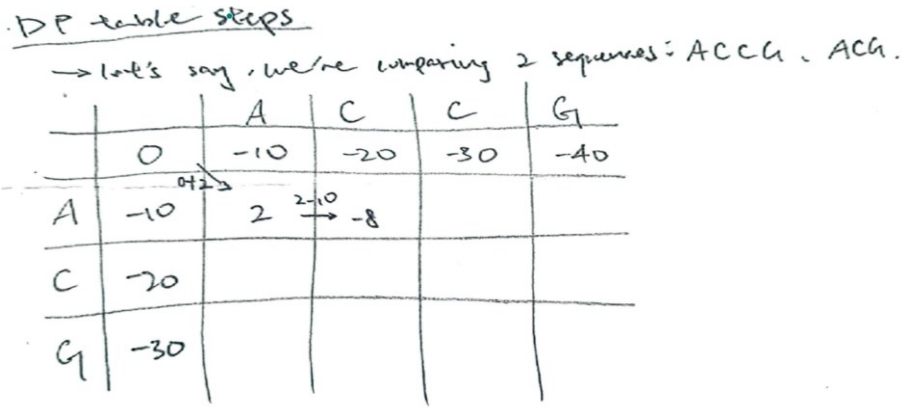
3) Each diagonal "move" results an addition of the top left corner cell plus the score being assigned to that specific cell, which is defined in the scoring matrix on the left. (ie. Since A-A results a score "2" from the scoring matrix, we calculated the score "2" in our DP table by "0+2".)

Before selecting the best score, score from all 3 directions (top, left and diagonal cells) should be first calculated. After that, the highest score will be selected, and an arrow should be drawn to indicate its best "flow". (ie. Top: -10 – 10 = -20; Left: -10-10 = -20; **Diagonal: 0+2 = 2. Since diagonal results the highest score, both the top and left paths can be eliminated and excluded from consideration.)

Scoring matrix:

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

Gap penalty = -10

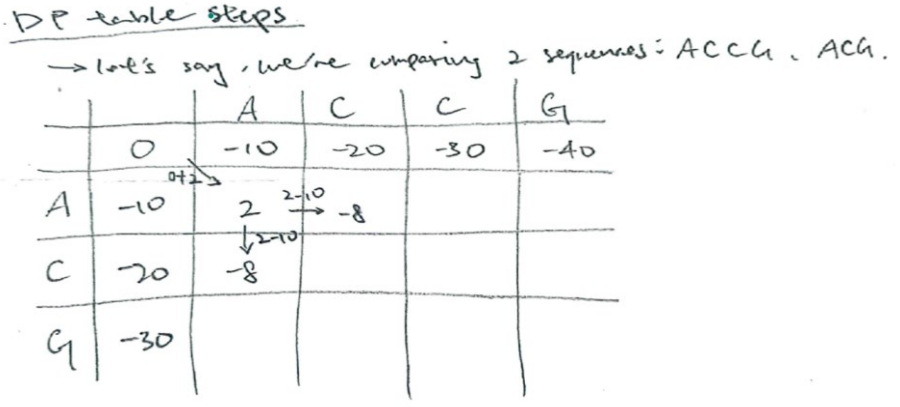


4) Then we continue to fill out the table according to the same rules and operations. (Top: -20 – 10 = -30; **Left: 2-10 = -8**; Diagonal: -10+(-7) = -17. Since left results the highest score, both the top and diagonal paths can be eliminated and excluded from consideration.)

Scoring matrix:

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

Gap penalty = -10

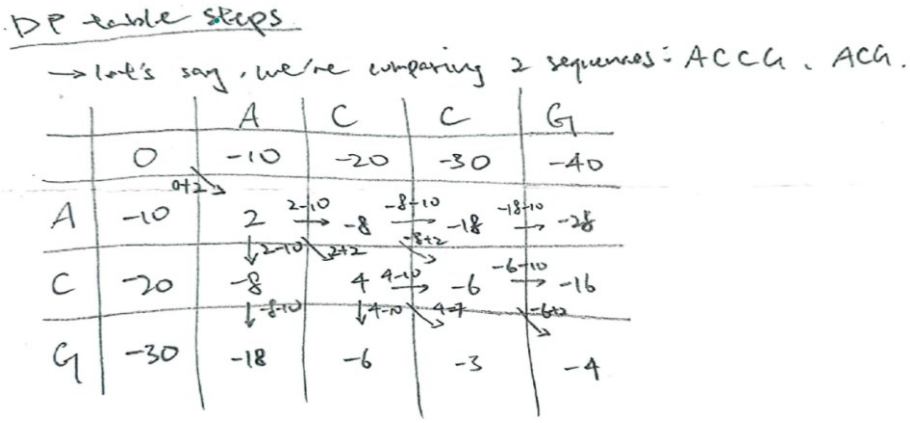


5) Then we continue to fill out the table according to the same rules and operations. (**Top: 2 – 10 = -8**; Left: -20-10 = -30; Diagonal: -10+(-7) = -17. Since top results the highest score, both the left and diagonal paths can be eliminated and excluded from consideration.)

Scoring matrix:

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

Gap penalty = -10



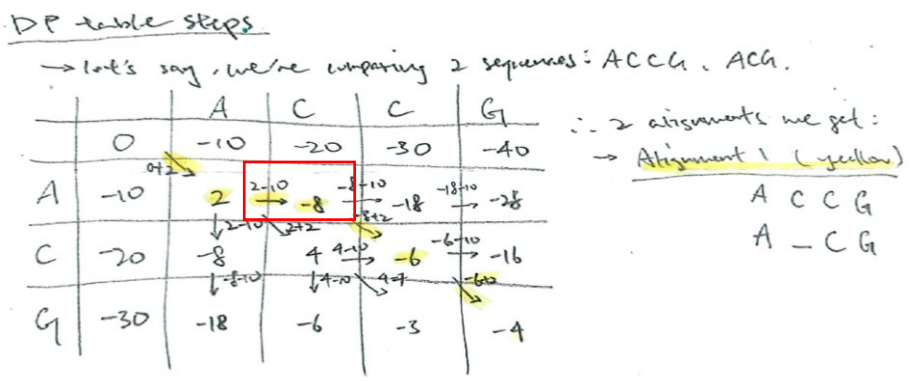
6) We continue to fill out the table until it is completed. (until the bottom right cell is filled)

** The bottom right cell is important since it tell us the last base pair of these 2 sequences. In the later steps we need to work backwards to find the optimal alignment(s) between these 2, based on the calculated best scores.

Scoring matrix:

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

Gap penalty = -10

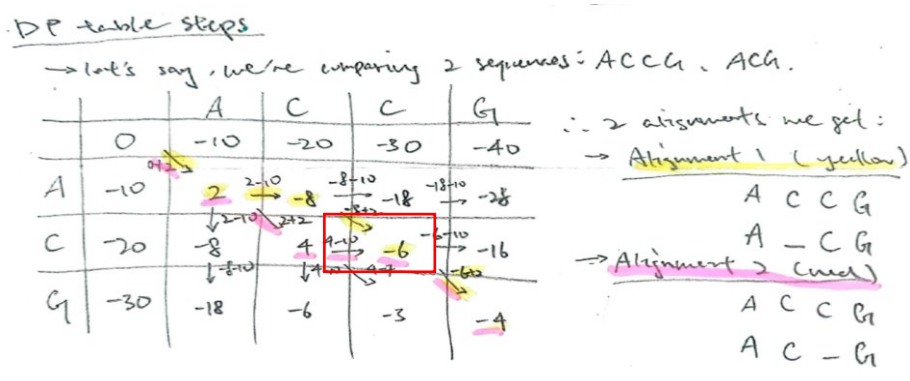


7) Matching with the table and write down the bases by walking backwards (from bottom right to top left). If arrow pointing diagonally, the base pair(s) must be the ones that the arrow is pointing, that is, the A-A, C-C, and G-G. Otherwise, that results a base – gap, that is, the C-_ (highlighted in red box).

Scoring matrix:

	A	C	G	T
A	2	-7	-5	-7
C	-7	2	-7	-5
G	-5	-7	2	-7
T	-7	-5	-7	2

Gap penalty = -10



8) Here we noticed that there are 2 possible paths to end up at the same destination. Matching with the table and write down the bases following the abovementioned logic.

➔ Final solution:

We obtained 2 optimal alignments, . . .