COMP 204 - Assignment 2

Yue Li

Important instructions:

- Release date: February 5, 2019 at 12:00 AM
- Due date: **February 19, 2019 at 23:59**
- Submit *one* Python programs on MyCourses, which should contain all your functions
- Write your name and student ID at the top of each program
- For each question, start off from the Python file given to you. *Do not change its name.*
- Do not use any modules or any pieces of code you did not write yourself
- For each question, your program will be *automatically* tested on a variety of test cases, which will account for 75% of the mark. To be considered correct, your program should print out *exactly and only* what the question specifies. Do not add extra text, extra spaces or extra punctuation. Do not ask the user to enter any other information than what is needed.
- For each question, 25% of the mark will be assigned by the TA based on (i) your appropriate naming of variables; (ii) commenting of your program; (iii) simplicity of your program (simpler = better).
- **Please make sure you understand the Needleman-Wunsch sequence alignment algorithm as it is the most important part of this assignment. Feel free to talk to the instructor and TAs at the office hours or senior students at the help desk.**

Sequence alignment

Sequence alignment is a way of arranging the sequences of DNA, RNA, or protein to identify regions of similarity that may be a consequence of functional, structural, or evolutionary relationships between the sequences. Aligned sequences of nucleotide or amino acid residues are typically represented as rows within a matrix. Gaps are inserted between the residues so that identical or similar characters are aligned in successive columns. A general alignment technique is the Needleman-Wunsch algorithm[1]. The algorithm essentially divides a large problem (e.g. the full sequence) into a series of smaller problems and uses the solutions to the smaller problems to reconstruct a solution to the larger problem. The Needleman-Wunsch algorithm is still widely used for optimal global alignment, particularly when the quality of the global alignment is of the utmost importance. The following content is adapted from Wikipedia: `https://en.wikipedia.org/wiki/Needleman-Wunsch_algorithm`.

1 Constructing the alignment scoring grid

Suppose we want to align the following two DNA sequences:

```
seq1:  GATTACA
seq2:  GCATGCT
```

Note that the sequences do not need to be equal in length in practice. We would like to find out the *optimal alignment* between the two sequences. For example, one possible alignment is:

```
G-ATTACA
GCA-TGCT
```

The letters are matched, mismatched, or matched to a gap "-" (i.e., a deletion or insertion (indel)):

- Match: the two letters at the current index are the same
- Mismatch: the two letters at the current index are different
- Indel: one letter is matched against a gap in the other sequence

To evaluate the overall alignment quality, we will need to define the scores for match, mismatch, and Indel. In this assignment, for simplicity, we will use the following scores for both DNA and protein sequence alignment (without considering the biochemical similarity among nucleic acids or amino acids):

- Match: +2
- Mismatch: -2
- Indel: -1

Therefore, the above alignment will have score 2:

```
G-ATTACA
GCA-TGCT
```

$$2 - 1 + 2 - 1 + 2 - 2 + 2 - 2 = 2$$

To compute the optimal alignment, we first construct a grid as shown below:

	-	G	C	A	T	G	C	T
-	0							
G								
A								
T								
T								
A								
C								
A								

Here the row names are the letters of the seq1 and the column names are the letters in seq2. We also place gap '-' in front of each sequence, which will become obvious in a bit.

We will fill out each cell in the grid by moving in one of the three directions:

- move diagonal: match/mismatch depending on whether the letters in the two sequences are the same or different
- move right: align a gap in seq1 to the letter in seq2 (i.e., a deletion in seq1)
- move down: align a gap in seq2 to the letter in seq1 (i.e., a deletion in seq2)

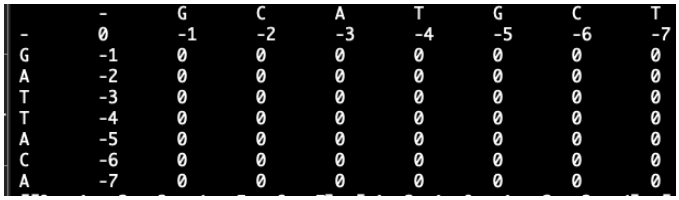
Move through the cells row by row, calculating the score for each cell. The score is calculated by comparing the scores of the cells neighboring to the left, top or top-left (diagonal) of the cell and adding the appropriate score for match, mismatch or indel. The resulting score for the cell is the highest of the three candidate scores.

Start with 0 in index (0,0) in the grid, because the first letter is a gap, we start by filling out the first row and first column by moving right and moving down, respectively. Adding the gap score of -1 for each shift, the results in the first row and the first column is shown below:

	-	G	C	A	T	G	C	T
-	0	-1	-2	-3	-4	-5	-6	-7
G	-1							
A	-2							
T	-3							
T	-4							
A	-5							
C	-6							
A	-7							

Question 1: Complete `initializeAlignmentScoreGrid` (10 points)

Complete the function `initializeAlignmentScoreGrid(seq1, seq2, gap_score)`, that is given to you to initialize the alignment scoring grid. You can use function `printAlignmentGrid(seq1, seq2, alignmentScoreGrid)` that is given to you to check the validity of your initialized grid: `printAlignmentGrid(seq1, seq2, initializeAlignmentScoreGrid(seq1, seq2, gap_score))`. For the two test sequences, the initialized grid should look like the one shown in the screenshot below:



	-	G	C	A	T	G	C	T
-	0	-1	-2	-3	-4	-5	-6	-7
G	-1	0	0	0	0	0	0	0
A	-2	0	0	0	0	0	0	0
T	-3	0	0	0	0	0	0	0
T	-4	0	0	0	0	0	0	0
A	-5	0	0	0	0	0	0	0
C	-6	0	0	0	0	0	0	0
A	-7	0	0	0	0	0	0	0

2 Filling out the alignment scoring grid

To fill out cell in index (1,1), the neighboring cells are shown below:

	-	G
-	0	-1
G	-1	?

We have three possible candidate sums:

- move diagonal: the diagonal top-left neighbor has score 0. The pairing of G and G is a match, so add the score for match: 0+2 = 2
- move right: the left neighbor has score -1, represents an indel and produces (-1) + (-1) = -2.
- move down: the top neighbor has score -1 and moving from there represents an indel, so add the score for indel: (-1) + (-1) = (-2)

The best move is to move diagonal with the highest score equal to is 2:

	-	G	C	A	T	G	C	T
-	0	-1	-2	-3	-4	-5	-6	-7
G	-1	2						
A	-2							
T	-3							
T	-4							
A	-5							
C	-6							
A	-7							

To fill out the cell in index(1,2), the neighboring cells are shown below:

	-	G	C
-	0	-1	-2
G	-1	2	?

We have three possible candidate sums:

- move diagonal: the diagonal top-left neighbor has score 0. The pairing of G and C is a mismatch, so add the score for the mismatch: (-1) + (-2) = -3
- move right: the left neighbor has score 1, represents an indel and produces 2 + (-1) = 1.

- move down: the top neighbor has score -2 and moving from there represents an indel, so add the score for indel: (-2) + (-1) = (-3)

The best move is to move right with the highest score equal to is 1:

	-	G	C	A	T	G	C	T
-	0	-1	-2	-3	-4	-5	-6	-7
G	-1	2	1					
A	-2							
T	-3							
T	-4							
A	-5							
C	-6							
A	-7							

Let's do another one. To fill out cell in index(2,1), the neighboring cells are shown below:

	-	G
-	0	-1
G	-1	2
A	-2	?

We have three possible candidate sums:

- move diagonal: the diagonal top-left neighbor has score -1. The pairing of A and G is a mismatch, so add the score for the mismatch: (-1) + (-2) = -3
- move right: the left neighbor has score -2, and moving from there represents an indel and produces -2 + (-1) = -3.
- move down: the top neighbor has score -2 and moving from there represents an indel, so add the score for indel: (2) + (-1) = (1)

The best move is to *move down* with the highest score equal to is 1:

	-	G	C	A	T	G	C	T
-	0	-1	-2	-3	-4	-5	-6	-7
G	-1	2	1					
A	-2	1						
T	-3							
T	-4							
A	-5							
C	-6							
A	-7							

We can repeat this process until we fill out all of the cells in the grid, which gives the following completed alignment scoring matrix:

	-	G	C	A	T	G	C	T
-	0	-1	-2	-3	-4	-5	-6	-7
G	-1	2	1	0	-1	-2	-3	-4
A	-2	1	0	3	2	1	0	-1
T	-3	0	-1	2	5	4	3	2
T	-4	-1	-2	1	4	3	2	5
A	-5	-2	-3	0	3	2	1	4
C	-6	-3	0	-1	2	1	4	3
A	-7	-4	-1	2	1	0	3	2

Question 2: `completeAlignmentScoreGrid` (30 points)

Complete the body of the function: `completeAlignmentScoreGrid(seq1, seq2, alignmentScoreGrid, match_score, mismatch_score, gap_score)`

Once again, you can use function `printAlignmentGrid(seq1, seq2, alignmentScoreGrid)` that is given to you to check the validity of your initialized grid:

For the two test sequences, the completed grid should be:

```
      -    G    C    A    T    G    C    T
-     0    -1   -2   -3   -4   -5   -6   -7
G     -1   2    1    0    -1   -2   -3   -4
A     -2   1    0    3    2    1    0    -1
T     -3   0    -1   2    5    4    3    2
T     -4   -1   -2   1    4    3    2    5
A     -5   -2   -3   0    3    2    1    4
C     -6   -3   0    -1   2    1    4    3
A     -7   -4   -1   2    1    0    3    2
```

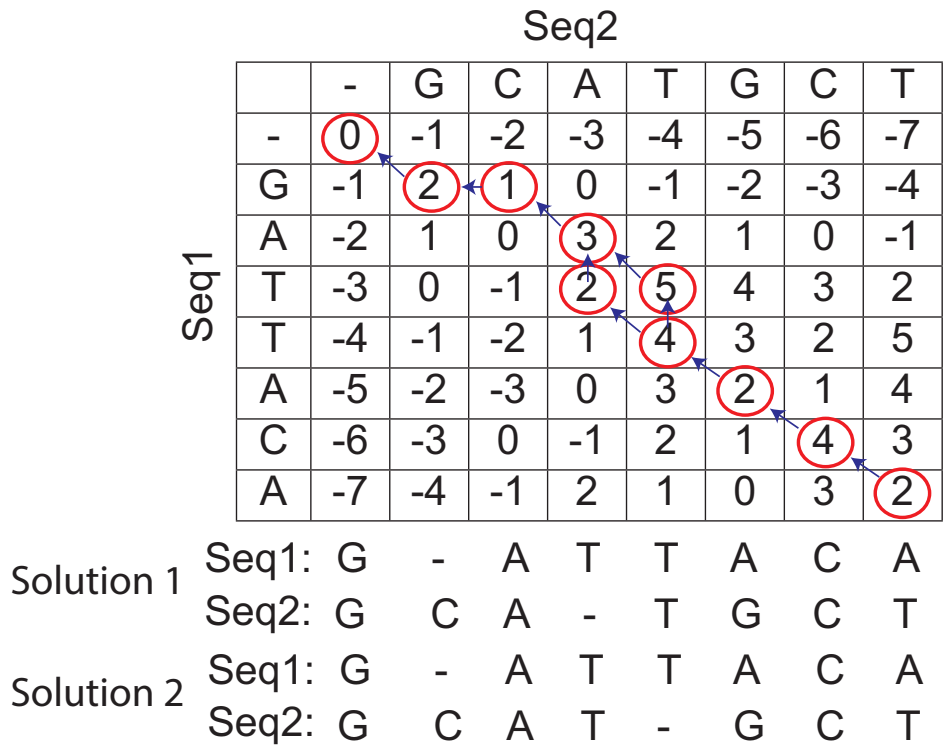
Hint: the algorithm can be summarized as follows: At any given index [i,j], we do the following to fill out the cell:

1. if seq1[i-1]==seq2[j-1]
   - move_diagonal = alignmentScoreGrid[i-1,j-1] + match_score

   else
   - move_diagonal = alignmentScoreGrid[i-1,j-1] + mismatch_score
2. move_down = alignmentScoreGrid[i-1, j] + gap_score
3. move_right = alignmentScoreGrid[i, j-1] + gap_score
4. alignmentScoreGrid[i,j] = max(move_diagonal, move_down, move_right)

3 Traceback the optimal alignment(s)

Based on the completed alignment scoring grid, we can traceback the alignments by starting from the bottom right cell and check which direction we have taken to obtain the value in that cell. From this we can read off the sequence alignment as shown below Note that we obtain two equally good solutions here.



Question 3. complete `traceback` (20 points)

Complete the function `traceback` so that it will return as tuple two strings with the first string indicating the alignment solution for seq1 and second the alignment solution for seq2. If there are multiple equivalent solutions, your program should return *exactly one of the equivalent solutions*. In the above example, `traceback` should return either (`"G-ATTACA"`, `"GCA-TGCT"`) or (`"G-ATTACA"`, `"GCAT-GCT"`) but not both.

4 Compute pairwise sequence similarity from homologous sequences

Given the above alignment program, we can read off the sequence similarity from the value in the bottom right cell. In the above example, the seq1 and seq2 have similarity score equal to 2. We can use this to evaluate pairwise sequence similarity between any pairs of DNA, RNA or protein sequences. We are given 8 amino acid sequences corresponding to the histone cluster 1 H1 family member A (*HIST1H1A*) for 8 species namely human, chimp, orangutan, mouse, rat, panda, dolphin, and tilapia.

Question 4 sequence similarity matrix (20 points)

Use the alignment program to calculate the similarity between every pair of amino acid sequences. To simplify the task, you are given a function called `align(seq1, seq2, match_score=2, gap_score=-1, mismatch_score=-2)`, which takes two sequences and returns as a tuple of two elements: (1) the alignment scoring matrix (from `completeAlignmentScoreGrid`); (2) alignment solutions as a string (from `traceback`).

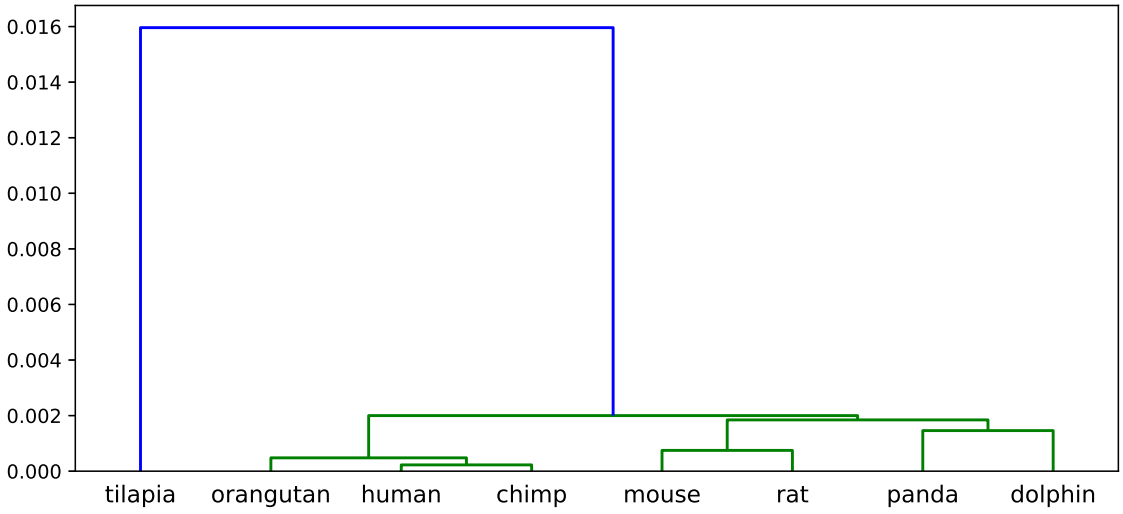
Your program should produce a 2D list or a matrix with the following values where each row is a list:

[[430, 418, 398, 260, 267, 298, 287, 114], [418, 430, 394, 268, 275, 306, 295, 114], [398, 394, 430, 252, 259, 286, 283, 118], [260, 268, 252, 426, 381, 296, 269, 104], [267, 275, 259, 381, 428, 299, 268, 111], [298, 306, 286, 296, 299, 430, 319, 110], [287, 295, 283, 269, 268, 319, 436, 119], [114, 114, 118, 104, 111, 110, 119, 446]]

The list can be formed into a symmetric matrix (upper-triangle is the same as the lower triangle values) to tell us which species are more similar to each other in terms of their HIST1H1A sequences as follows:

	human	chimp	orangutan	mouse	rat	panda	dolphin	tilapia
human	430	418	398	260	267	298	287	114
chimp	418	430	394	268	275	306	295	114
orangutan	398	394	430	252	259	286	283	118
mouse	260	268	252	426	381	296	269	104
rat	267	275	259	381	428	299	268	111
panda	298	306	286	296	299	430	319	110
dolphin	287	295	283	269	268	319	436	119
tilapia	114	114	118	104	111	110	119	446

Using the inverse of the similarity (i.e., 1/similarity) as a value for the *genetic distance*, we can apply a hierarchical clustering function (provided to you) to the distance matrix to obtain a tree cluster, where the nodes are the 8 species and the species that are more similar to each other are grouped under common branches. The species clusters are further grouped into common branches until there is only one cluster left containing all of the 8 species at the root of the tree.



We can see that the tilapia amino acid sequence is far different from all of the other 7 species. The human sequence is the most similar to the chimp sequence and second most similar to the orangutan sequence. Mouse and rat are more similar to each other than they are to other species, so as panda and dolphin. Given the large distance between tilapia and the other 7 species, we will remove tilapia from the following analyses.

5 Find consensus sequence (20 points)

Given the sequence similarity matrix computed above, we are interested in obtaining a *consensus sequence* that preserves all of the amino acids that are same among the 7 species.

Question 5 `findConsensusSequence`

Write a function named `findConsensusSequence(seqlist_new)`, which takes as a list of strings an argument named `seqlist_new`.

The program should behave as follows.

1. Find out the two pair of **distinct** sequences that are the most similar to each other
2. Align the two most similar sequences using the `align` implemented above
3. Create a consensus sequence based on the resulting alignment. The consensus is an exact consensus, i.e., only residues found at a given position in all sequences are stored, otherwise an unknown residue ('X') is recorded.
4. Remove the two sequences from the `seqlist_new` and insert the consensus sequence into the `seqlist_new`
5. Repeat 1-4 until only one consensus sequence is left

Your consensus sequence among those 7 species should look like the following sequence:

MSETXXPXXXXXAXXAXXXXXEKPXAXXKXKXXXPAKAXXXAXXXKKPXGPSVSELIVQAX
SSSKERXGVSLAALKKXLAXAGYDVEKNNSRIKLGXXSLVXKGTLVQTKGTGAXGSFKXNKKX
AXXXXXXKXXXXXXKVXXKXKXXXXSXXXKKXKKXTXXXAXKKXVKTPKXXKKPAXXXXTXXX
SKXPXKXKXXXXKXKKXXKSPAKAKAVKPKAXKXXXVTKPKTXAKPKKAAPKKK

We can align the consensus sequence with the 7 sequences to perform a simple multiple sequence alignments [2]. This will give the following alignments:

MSET-VP----PA-PAASAAPEKPLA-GK-KAKKPAKA--AAASKKKPAGPSVSELIVQAASSSKE
RGGVSLAALKKALAAAGYDVEKNNSRIKLGIKSLVSKGTLVQTKGTGASGSFKLNKK-A-SSVE
TK--PGASKVATKTKATGAS---KKLKKAT--GASKKSVKTPKKAKKPA---ATRKSSKNP-K-KPKTV
KPKKVAKSPAKAKAVKPKAAK-ARVTKPKT-AKPKKAAPKKK # human

MSET-VP----PA-PAASAAPEKPLA-GK-KAKKPAKA--AAASKKKPAGPSVSELIVQAASSSKE
RGGVSLAALKKALAAAGYDVEKNNSRIKLGIKSLVSKGTLVQTKGTGASGSFKLNKK-A-SSVE
TK--PGASKVATKTKATGAS---KKPKKAT--GASKKSVKTPKKAKKPA---ATRKSSKNP-K-KPKIV
KPKKVAKSPAKAKAVKPKAAK-AKVTKPKT-AKPKKAAPKKK # chimp

MSET-VP----TA-PAASAAPEKPLA-GK-KAKKPAKA-VVA-SKKKPAGPSVSELIVQAASSSKE
RGGVSLAALKKALAVAGYDVEKNNSRIKLGIKSLVSKGTLVQTKGTGASGSFKLNKK-A-FSVET
K--PGASKVAAKTKATGAS---KKLKKAT--GASKKSVKTPKKAKKPA---ATRKSSKNP-K-KPKTLK
PKKVAKSPAKAKAVKPKAAK-AKVTKPKT-AKPKKAAPKKK # orangutan

MSET-AP----VA-QAASTATEKP-AAAK-KTKKPAKA---AAPRKKPAGPSVSELIVQAVSSSKE
RSGVSLAALKKSLAAAGYDVEKNNSRIKLGLKSLVNKGTLVQTKGTGAAGSFKLNKK-A----ESK
--AITTKVSVKAK---ASGAAKKPKK-TAGAAAKKTVKTPKKPKKPAVSKKT---SKSP-K-KPKVVKA
KKVAKSPAKAKAVKPKASK-AKVTKPKTPAKPKKAAPKKK # mouse

MSET-AP-VPQPASVA----PEKP-AATK-KTRKPAKA---AVPRKKPAGPSVSELIVQAVSSSKE
RSGVSLAALKKSLAAAGYDVEKNNSRIKLGLKSLVNKGTLVQTKGTGAAGSFKLNKK-A----ESK

--ASTTKVTVKAK---ASGAAKKPKK-TAGAAAKKTVKTPKKPKKPAVSKKT--SSKSP-K-KPKVVK
AKKVAKSPAKAKAVKPKAAK-VKVTKPKTPAKPKKAAPKKK # rat

MSET-AP----PA-SATSTPPEKP-AAGK-KAKRPAKA--AAAAKKKPTGPSVSELIVQAVSSSKE
RSGVSLAALKKALAAAGYDVEKNNSRIKLGLKSLVSKGTLVQTKGTGASGSFKLNKK-A-ASGE
VK--ANPTKV-VKAKVTGTS---KKPKKVT--AAVKKAVKTPKKAKKPA---VTKKSSKSP-K-KPKVV
KPKKVAKSPAKAKAVKPKAAK-AKVTKPKTAAKPKKAAPKKK # panda

MSET-AP--PVPA--A-STPPEKPSAGRKAK--KPAKA--VATAKKKPAGPSVSELIVQAVSSSKE
RSGVSLAALKKALAAAGYDVEKNNSRIKLGLRSLVSKGTLVQTKGTGASGSFKFNKKVA--SVD
SK--PSATKVAAKAK--VTS-SSKKPKKATGAAAGKKGVKTPKNAKKPA---ATKKSSKSP-K-KSR
VVKPKKIGKSPAKAKAVKPKAAK-AKVTKPKTAAKPKKAAPKKK # dolphin

6 Bonus question: Calculate the conservation score in the human sequence (10 points)

Sequence conservation implies the residues in the sequence are observed in multiple species. Highly conserved residues among distant species imply their importance in biological functions.

The aligned human sequence from the above multiple sequence alignment results look like the following:

MSET-VP----PA-PAASAAPEKPLA-GK-KAKKPAKA--AAASKKKPAGPSVSELIVQAASSSKE
RGGVSLAALKKALAAAGYDVEKNNSRIKLGIKSLVSKGTLVQTKGTGASGSFKLNKK-A-SSVE
TK--PGASKVATKTKATGAS---KKLKKAT--GASKKSVKTPKKAKKPA---ATRKSSKNP-K-KPKTV
KPKKVAKSPAKAKAVKPKAAK-ARVTKPKT-AKPKKAAPKKK

Compare the non-gap residue in the aligned human sequence with the residue in the same position from the aligned sequence in each of the other 6 species (excluding tilapia). For each residue, count the number of species that have the same residue in their aligned sequences. The more species agree the more conserved that residue is.

Save the counted values in a list named `conservedCounts`, which should look like this: [6,6,6
,6,2,6,4,6,2,6,3,5,2,3,5,6,6,6,2,6,3,6,6,3,4,5,6,6,6,6,2,6,3,2,4,6,6,6,5,6,6,6,6,6,6,6,6,6,6,6,2,6,6,
6,6,6,6,2,6,6,6,6,6,6,6,6,6,4,6,6,5,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,2,5,6,6,6,4,6,6,6,6,6,6,6,6,6,6,6,
6,4,6,6,6,6,5,6,6,6,6,1,4,3,5,2,6,3,2,3,2,6,6,3,1,6,2,6,2,3,3,4,6,6,6,1,6,6,3,6,2,6,2,6,6,2,6,6,6,6,
6,5,4,6,6,6,6,3,6,2,4,5,6,6,2,6,6,6,5,5,1,5,6,4,6,6,5,5,6,6,6,6,6,6,6,6,6,6,6,6,6,5,6,5,0,6,6,6,6,6,
6,6,6,6,6,6,6,6,6,6,6,6,6]

We can visualize the conservation by printing out the human residue and the corresponding conserved counts as follows (the code is provided):

```
MSETVPPAPAASAAPEKPLAGKKAKKPAKAAAASKKKPAGPSVSELIVQA
66662646263523566626366345666626324666566666666666
ASSSKERGGVSLAALKKALAAAGYDVEKNNSRIKLGIKSLVSKGTLVQTK
26666662666666666465666666666666666625666466666666
GTGASGSFKLNKKASSVETKPGASKVATKTKATGASKKLKKATGASKKSV
66664666656666143526323266316262334666166362626626
KTPKKAKKPAATRKSSKNPKKPKTVKPKKVAKSPAKAKAVKPKAAKARVT
66665466663624566266655156466556666666666666565066
KPKTAKPKKAAPKKK
666666666666666
```

The number below each residue indicates the number of species (6 species at max, excluding tilapia) that share common residue with the human sequence.

References

[1] Saul B Needleman and Christian D Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453, 1970.

[2] Desmond G Higgins and Paul M Sharp. CLUSTAL: a package for performing multiple sequence alignment on a microcomputer. *Gene*, 73(1):237–244, 1988.