Assignment 1- Global Alignment

Xinyuan Min
950829573070

1. **Using seq1 and seq2, can you reproduce the alignments from figure 5.9, 5.11, 5.12? Put the output in your report.**

   Gap penalty = 8: `('THISLINE_', 'ISALIGNED')`

   Gap penalty = 4: `('THIS_LI_NE_', '__ISALIGNED')`

   Gap penalty = 4, end gap not penalized: `('THIS_LI_NE_', '__ISALIGNED')`

2. **Describe your traceback strategy in words. In case of equal scores from multiple cells, which direction is preferred?**

   I created a Class Cell( ) *to store all the information related to a single cell, including the alignment origin.* self.origin *will reutn a string, e.g. 'diagonal' or 'left' to indicate the origin of each cell.*

   Under traceback( ) *function, the traceback starts from the bottom right cell in the matrix. First initiate two empty string* aligned_seq1 & aligned_seq2 *to store the residues alignment result. Next, get the indexes of bottom right cell , then the* self.origin *attribute of that cell is obtained. In case of 'diagonal',* self.res1 & self.res2 *are added to the beginning of* aligned_seq1 & aligned_seq2 r*espectively, and decrease both indexes by 1. Similarly, the residues alignment result under situation of 'left' and 'above' origin are obtained and added to* aligned_seq1 & aligned_seq2

   In this algorithm, although I recorded multiple alignment origins in self.prepath *attribute, I didn't distinguish multiple alignment path (I tied to distinguish multiple paths but failed, it's beyond my capability) .*

3. **When you align seq1 and seq2 using different linear gap penalties, ranging from 1 to 20 (and no separate end gap penalty), how many different alignments do you get? List the different alignments. Explain the differences, given the settings.**

1, 2, 3, 4	THIS_LI_NE_ _ _ISALIGNED
5, 6, 7,	THISLI_NE_ IS_ALIGNED
8 - 20	THISLINE_ ISALIGNED

When a small gap penalty is used, gaps are more likely to be introduced. Instead if we set a high gap penalty, a gap is less likely to be present in the optimal alignment.
In the above example, when the gap penalty is set in the range of 1 -4, the algorithm gives an alignment with 5 gaps. But when the gap penalty is higher than 7, the algorithms tries to obtain an output with minimum gap, thus only 1 gap is introduced at the end of the sequence

which has a shorter length.

4. **Align two related proteins, seq3 and seq4, and report the alignment score (right bottom in the matrix) using 2 different settings: linear gap penalty = 5 & end gap = 1 linear gap penalty = 5 & end gap = 10**
   linear gap penalty = 5 & end gap = 1: 1505
   linear gap penalty = 5 & end gap = 10: 1505

5. **What is the percentage of identity between the aligned sequences (seq3 and seq4) for both settings above?**
   linear gap penalty = 5 & end gap = 1: 74.15%
   linear gap penalty = 5 & end gap = 10: 74.15%

6. **Can you reproduce your results using the EMBOSS needle program?** **http://www.ebi.ac.uk/Tools/psa/emboss_needle/ Note the settings under "more options". Print the needle output.**
   The output of this algorithm can produce the same results, except for the percentage of identity between seq3 and seq4 is slightly different compared with EMBOSS program (74.3% vs.74.15%) under settings [linear gap penalty = 5 & end gap = 1 or 10]

7. **What is the criterion that the algorithm optimizes?**
   The algorithm tries to maximize the alignment score. And Dynamic Programming can be applied to solve this problem, i.e. the best alignment is obtained through calculate the best score of the sequences up to that point, plus the best score for the two additional symbols.
8. **Is the algorithm exact or approximate?**
   This is an exact algorithm, it uses a quantitative scoring function and will always return the best alignment with the highest score.
9. **What are the time and space complexity of the algorithm you implemented?**
   *Time complexity: O(m*n): the score of m*n cells need to be calculated, thus the number of steps in this algorithm is proportional to the product of the sequence length.*
   *The running can be decreased by divide the problem into several sub-problems. For example, if the dimension of filled_matrix is 100*100, theoretically the score of 10,000 cells need to be calculated. But if we first only fill in a 10*10 sub-matrix, and then take the terminal cell of this sub-matrix as the start cell of next sub-matrix, then in total, only 10*10*10 cells need to be calculated, the number of steps is only 1/10 of the original algorithm.*

   Space complexity: this algorithm stores the alignment score and alignment option information in two matrixes, the size of matrixes depends on the length of the sequences.

10. **Describe how you could decrease running time by limiting the number of gaps allowed in the alignment.**
    Set gap_num *= 0, each time when a gap is inserted, (while* self.prepath == 'left' *or* 'above'), *increase* gap_num *by 1. When gap_num reached the pre-defined limit, the score is no longer obtained by getting the optimal score of three paths, but becomes (the score of diagonal cell – similarity score in the bolsum62 matrix)*