Emily Morrow

1. Explain the time and space complexity of your algorithm by showing and summing up the complexity of each subsection of your code.
    a. [10 points] Your analysis should show that your unrestricted algorithm is at most $O(nm)$ time and space.

Where n and m are lengths of the two sequences

In the unrestricted algorithm I fill in row 0 and column 0 which runs in n and m time.

```
for i in range(1, ySize):
   matrix[i][0] = matrix[i - 1][0] + INDEL

for i in range(1, xSize):
   matrix[0][i] = matrix[0][i - 1] + INDEL
```

I then go through each cell in the matrix using two (nested) for loops which will take n*m time. (the first part of this double for loop is shown below)

```
for i in range(1, len(seq2) + 1):
   for j in range(1, len(seq1) + 1):
      min = None

      # get values for if you're extracting from diag, left, or above
      if seq1[j-1] == seq2[i-1]:
         diag = matrix[i - 1][j - 1] + MATCH
      else:
         diag = matrix[i - 1][j - 1] + SUB

      left = matrix[i][j-1] + INDEL
      above = matrix[i - 1][j] + INDEL
```

After this the traceback loop goes through and runs in m + 1 time.

```
while i > 0 and j > 0:
   if traceMatrix[j][i] == 1:     # if it came from diagonal
      align1 = seq1[i - 1] + align1
      align2 = seq2[j - 1] + align2
      i -= 1
      j -= 1
   elif traceMatrix[j][i] == 2:   # if it came from left
      align1 = seq1[i - 1] + align1
      align2 = "-" + align2
      i -= 1
   elif traceMatrix[j][i] == 3:   # if it came from above
      align1 = "-" + align1
      align2 = seq2[j - 1] + align2
      j -= 1
```

This results in n+m+n*m+m+1 time or O(nm) time. The space of this algorithm is n*m from the score matrix and n*m from the traceback matrix so the space would be n*m + n*m or O(nm) space.

    b. [10 points] Your analysis should show that your banded algorithm is at most $O(kn)$ time and space.

Where k is the bandwith and n is the length of the shorter sequence.

Just like the unbanded algorithm we fill in row 0 and column 0 which takes n + m time.

```
for i in range(1, 4):
   matrix[i][0] = matrix[i - 1][0] + INDEL
for i in range(1, 4):
   matrix[0][i] = matrix[0][i - 1] + INDEL
```

In the double for loop that fills in the score and traceback matrix the outer loop runs n times and the inner loop runs a max of k times. This results in kn time.

```
minI = 1
maxI = 5
down = False

# fill the rest of the score and traceback matrix
for i in range(1, len(seq2) + 1):
   for j in range(minI, maxI):
      min = None

      # get values for if you're extracting from diag, left, or above
      if seq1[j-1] == seq2[i-1]:
         diag = matrix[i - 1][j - 1] + MATCH
      else:
         diag = matrix[i - 1][j - 1] + SUB

      left = matrix[i][j-1] + INDEL
      above = matrix[i - 1][j] + INDEL

      # determine the minimum value and keep track of where it came from
      if diag < left:
         min = diag
         cameFrom = 1   # from diagonal
      else:
         min = left
         cameFrom = 2   # from left

      if above < min:
         min = above
         cameFrom = 3   # from above

      matrix[i][j] = min
      traceMatrix[i][j] = cameFrom

   # set the indices so banded never gets larger than 7 cells horizontal or diagonal
   if maxI - minI == 7:
      minI += 1
      down = True
   elif down:
      minI += 1
   if maxI < len(seq1) + 1:
      maxI += 1
```

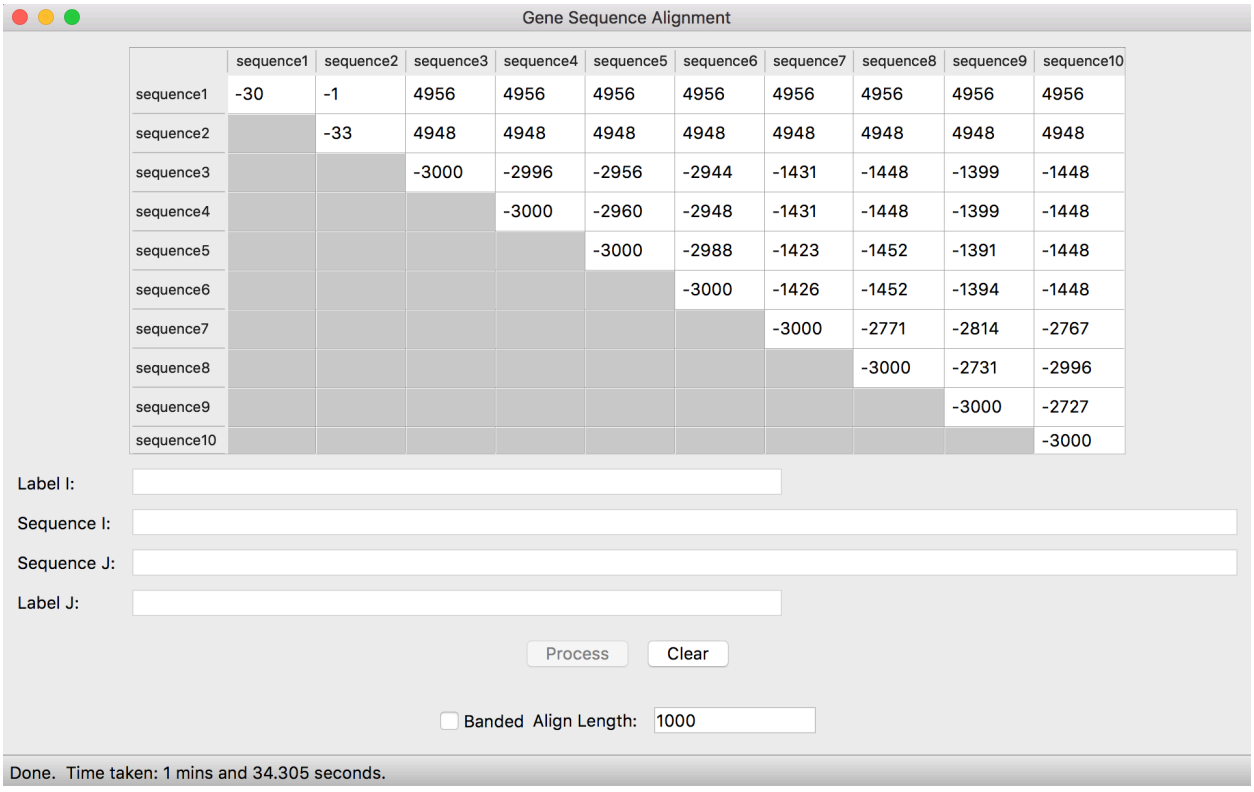
The traceback loop is the same as in the unbanded algorithm and also runs in n time. This results in n + m + kn + n time or O(kn) time. The space however, is still O(nm) since the two matrices are both n*m.

Emily Morrow

2. [10 points] Write a paragraph that explains how your alignment extraction algorithm works, including the backtrace

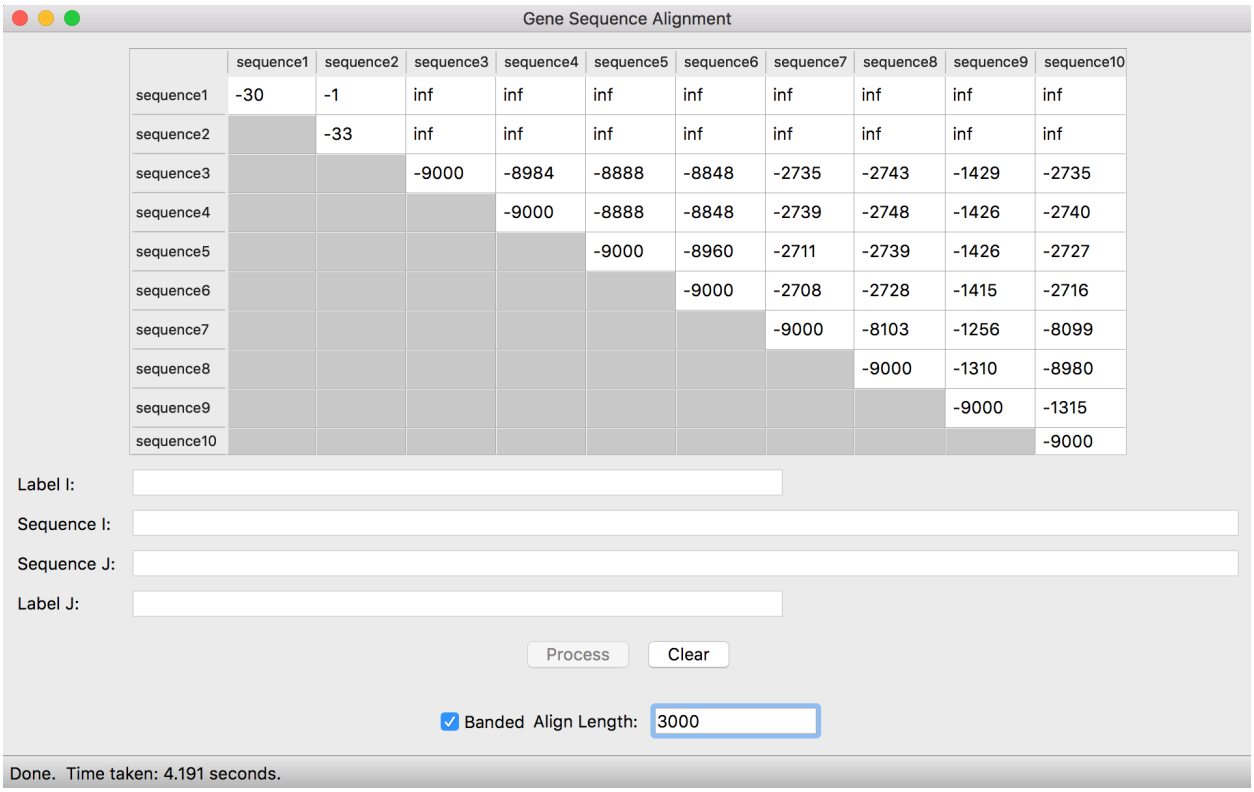
When the score matrix is being put together I also put together a traceback matrix. When figuring out which score to put in the matrix the algorithm keeps track of whether this score was extracted from the left, diagonal, or from above. This then results in a matrix with the same dimentions as the score matrix but it's filled with 1's, 2's, and 3's where 1 = diagonal, 2 = left, 3 = above. When I go to extract the alignment I start in the bottom right corner of the traceback matrix and I see if it came from diagonal, left, or above. If it came from diagonal I add the char that lines up with that value in the matrix from the 1st sequence and the 2nd sequence to the front of the 1st alignment and the 2nd alignment and move onto the cell diagonal from the current cell. If it's from left I add a '-' to the second alignment and go to the cell to the left of the current cell. If it's from above I add a '-' to the first alignment and move to the cell above the current cell. This process ends when I get to cell [0][0].

3. [20 points] Include a "results" section showing both a screen-shot of your 10x10 score matrix for the unrestricted algorithm with align length k = 1000 and a screen-shot of your 10x10 score matrix for the banded algorithm with align length k = 3000.



	sequence1	sequence2	sequence3	sequence4	sequence5	sequence6	sequence7	sequence8	sequence9	sequence10
sequence1	-30	-1	4956	4956	4956	4956	4956	4956	4956	4956
sequence2		-33	4948	4948	4948	4948	4948	4948	4948	4948
sequence3			-3000	-2996	-2956	-2944	-1431	-1448	-1399	-1448
sequence4				-3000	-2960	-2948	-1431	-1448	-1399	-1448
sequence5					-3000	-2988	-1423	-1452	-1391	-1448
sequence6						-3000	-1426	-1452	-1394	-1448
sequence7							-3000	-2771	-2814	-2767
sequence8								-3000	-2731	-2996
sequence9									-3000	-2727
sequence10										-3000

Emily Morrow



4. [10 points] Include in the "results" section the extracted alignment for the first 100 characters of sequences #3 and #10 (counting from 1), computed using the unrestricted algorithm with k = 1000. Display the sequences in a side-by-side fashion in such a way that matches, substitutions, and insertions/deletions are clearly discernible as shown above in the To Do section. Also include the extracted alignment for the same pair of sequences when computed using the banded algorithm and k = 3000.

Unrestricted
Sequence 3 (on top) Sequence 10 (on bottom) <- this is for sure right but I would double check the next
attgcgagcgatttgcgtgcgtgcat-ccc--gcttcact-gatctcttgttagatcttttcataatctaaactttataaaaacatccactccctgt-ag
ataa-gagtgattggcgtccgtacgtacccttttctactctcaaactcttgttagtttaaatc-taatctaaactttataaa--cggc-acttcctgtgtg

Banded
Sequence 3 (on top) Sequence 10 (on bottom)
attgcgagcgatttgcgtgcgtgcat-ccc--gcttcact-gatctcttgttagatcttttcataatctaaactttataaaaacatccactccctgt-ag
ataa-gagtgattggcgtccgtacgtacccttttctactctcaaactcttgttagtttaaatc-taatctaaactttataaa--cggc-acttcctgtgtg

5. [30 points] Attach your commented source code for both your unrestricted and banded algorithms.

```
def notBanded(self, seq1, seq2):
    ySize = len(seq2) + 1
    xSize = len(seq1) + 1
    cameFrom = 0

    # initialize the score matrix and traceback matrix
    matrix = np.zeros((ySize, xSize), dtype=int)
    traceMatrix = np.zeros((ySize, xSize), dtype=int)

    # fill the first row and first column of the score matrix
```

```
    for i in range(1, ySize):
        matrix[i][0] = matrix[i - 1][0] + INDEL

    for i in range(1, xSize):
        matrix[0][i] = matrix[0][i - 1] + INDEL

    # fill the rest of the score and traceback matrix
    for i in range(1, len(seq2) + 1):
        for j in range(1, len(seq1) + 1):
            min = None

            # get values for if you're extracting from diag, left, or above
            if seq1[j-1] == seq2[i-1]:
                diag = matrix[i - 1][j - 1] + MATCH
            else:
                diag = matrix[i - 1][j - 1] + SUB

            left = matrix[i][j-1] + INDEL
            above = matrix[i - 1][j] + INDEL

            # determine the minimum value and keep track of where it came from
            if diag < left:
                min = diag
                cameFrom = 1   # from diagonal
            else:
                min = left
                cameFrom = 2   # from left

            if above < min:
                min = above
                cameFrom = 3   # from above

            matrix[i][j] = min
            traceMatrix[i][j] = cameFrom

    align1 = ""
    align2 = ""

    i = len(seq1)
    j = len(seq2)
    score = matrix[j][i]

    # traceback through the matrix to get the alignments
    while i > 0 and j > 0:
        if traceMatrix[j][i] == 1:     # if it came from diagonal
            align1 = seq1[i - 1] + align1
            align2 = seq2[j - 1] + align2
            i -= 1
            j -= 1
        elif traceMatrix[j][i] == 2:   # if it came from left
            align1 = seq1[i - 1] + align1
            align2 = "_" + align2
            i -= 1
        elif traceMatrix[j][i] == 3:   # if it came from above
            align1 = "_" + align1
            align2 = seq2[j - 1] + align2
            j -= 1

    return {'a1':align1, 'a2':align2, 'score':score}
```

Emily Morrow

```
def bandedFill(self, seq1, seq2):
   ySize = len(seq2) + 1
   xSize = len(seq1) + 1
   cameFrom = 0

   # initialize the score matrix and traceback matrix
   matrix = np.zeros((ySize, xSize), dtype=int)
   traceMatrix = np.zeros((ySize, xSize), dtype=int)

   # fill the matrix with a large number so if it isn't in the band
   # it won't interfere with the scores
   matrix.fill(9000)
   matrix[0][0] = 0

   # fill the first row and first column of the score matrix
   for i in range(1, 4):
      matrix[i][0] = matrix[i - 1][0] + INDEL
   for i in range(1, 4):
      matrix[0][i] = matrix[0][i - 1] + INDEL

   # set the indices where we should fill in the banded matrix
   minI = 1
   maxI = 5
   down = False

   # fill the rest of the score and traceback matrix
   for i in range(1, len(seq2) + 1):
      for j in range(minI, maxI):
         min = None

         # get values for if you're extracting from diag, left, or above
         if seq1[j-1] == seq2[i-1]:
            diag = matrix[i - 1][j - 1] + MATCH
         else:
            diag = matrix[i - 1][j - 1] + SUB

         left = matrix[i][j-1] + INDEL
         above = matrix[i - 1][j] + INDEL

         # determine the minimum value and keep track of where it came from
         if diag < left:
            min = diag
            cameFrom = 1   # from diagonal
         else:
            min = left
            cameFrom = 2   # from left

         if above < min:
            min = above
            cameFrom = 3   # from above

         matrix[i][j] = min
         traceMatrix[i][j] = cameFrom

      # set the indices so banded never gets larger than 7 cells horizontal or
diagonal
      if maxI - minI == 7:
         minI += 1
         down = True
      elif down:
         minI += 1
```

Emily Morrow

```
        if maxI < len(seq1) + 1:
            maxI += 1

    # traceback through the matrix to get the alignments
    align1 = ""
    align2 = ""

    i = len(seq1)
    j = len(seq2)
    score = matrix[j][i]
    while i > 0 and j > 0:
        if traceMatrix[j][i] == 1:     # if it came from diagonal
            align1 = seq1[i - 1] + align1
            align2 = seq2[j - 1] + align2
            i -= 1
            j -= 1
        elif traceMatrix[j][i] == 2:   # if it came from left
            align1 = seq1[i - 1] + align1
            align2 = "-" + align2
            i -= 1
        elif traceMatrix[j][i] == 3:   # if it came from above
            align1 = "-" + align1
            align2 = seq2[j - 1] + align2
            j -= 1
        else:  # if the alignment can't be complete break out of loop
            i = 0
            j = 0


    return {'a1': align1, 'a2': align2, 'score': score}
```