Cody Kesler
CS 312 – 2
Lab 4 Write up

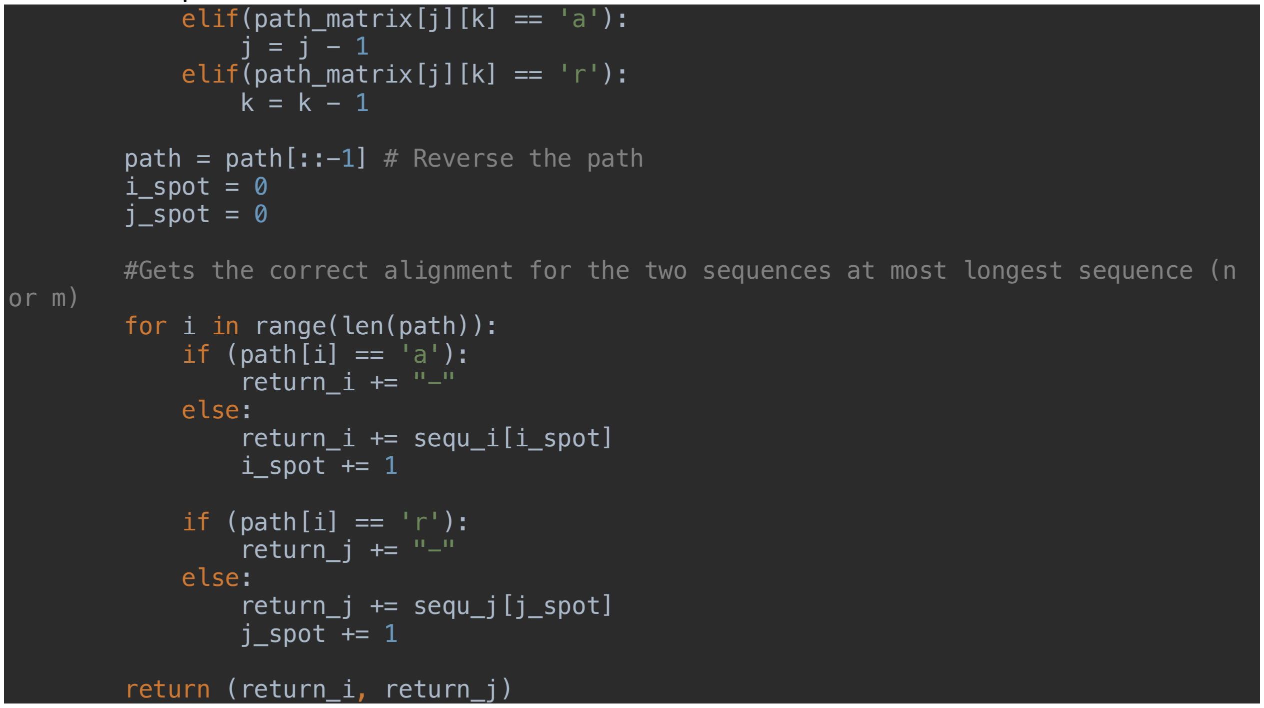
Code:

```
# O(100n*m) = O(n*m)
    def align_all( self, sequences, banded, align_length ):
        results = []
        #This loop runs 10 times
        for i in range(len(sequences)):
            jresults = []
            sequ_i = sequences[i]
            sequ_i_len = len(sequ_i)
            #This loop runs 10 times as well
            for j in range(0, len(sequences)):
                sequ_j = sequences[j]
                sequ_j_len = len(sequ_j)
                if(i == j):
                    s = {'align_cost': max(-3*align_length,-3*sequ_i_len),
                         'seqi_first100': 'Self Comparison',
                         'seqj_first100': 'Self Comparison'}
                elif((i == 0 and j != 1) or (i == 1 and j != 0)):
                    s = {'align_cost': float('inf'),
                         'seqi_first100': 'No Alignment Possible',
                         'seqj_first100': 'No Alignment Possible'}
                elif(i > j):
                    s = {'align_cost': 0,
                         'seqi_first100': 'Dummy',
                         'seqj_first100': 'Dummy' }
                else:
                    # 2 Matricies of size n*m are made here O(2n*m) = O(n*m)
                    if(align_length > sequ_i_len):
                        if(align_length > sequ_j_len):
                            dist_matrix = [[0 for col in range(sequ_i_len + 1)]
                                          for row in range(sequ_j_len + 1)]
                            path_matrix = [[("") for col in range(sequ_i_len + 1)]
                                          for row in range(sequ_j_len + 1)]
                    else:
                        dist_matrix = [[0 for col in range(align_length + 1)]
                                          for row in range(align_length + 1)]
                        path_matrix = [[("") for col in range(align_length + 1)]
                                          for row in range(align_length + 1)]

                    for m in range(len(dist_matrix[0])):
                        if(m > 3 and banded):
                            dist_matrix[0][m] = float('inf')
                        else:
                            dist_matrix[0][m] = m * 5
                        path_matrix[0][m] = 'r'

                    for m in range(len(dist_matrix)):
                        if (m > 3 and banded):
                            dist_matrix[m][0] = float('inf')
                        else:
                            dist_matrix[m][0] = m * 5
                        path_matrix[m][0] = 'a'

                    path_matrix[0][0] = ""

                 #For loop loops through each of the sequences for each of the
sequences. At most O(n*m)
                    for k in range(1, len(dist_matrix)):
                        # This code puts the bound on the rows for the constraint 3
                        if (banded):
```

Cody Kesler
CS 312 – 2
Lab 4 Write up

```
                            if ((k - 3) > 0):
                                l_start = k - 3
                            else:
                                l_start = 0
                            if ((k + 4) < len(dist_matrix[0])):
                                l_end = k + 4
                            else:
                                l_end = len(dist_matrix[0])
                        else:
                            l_start = 1
                            l_end = len(dist_matrix[0])

                        for l in range(l_start, l_end):
                            right_of = dist_matrix[k - 1][l] + 5
                            above = dist_matrix[k][l - 1] + 5
                            diagonal = dist_matrix[k - 1][l - 1]
                                  + self.diff(sequ_i, sequ_j, k-1, l-1)

                            mininum = min(right_of, above, diagonal)

                            dist_matrix[k][l] = mininum
                            if(mininum == right_of):
                               path_matrix[k][l] = ("r")
                            elif(mininum == above):
                                path_matrix[k][l] = ( "a")
                            elif(mininum == diagonal):
                                path_matrix[k][l] = ("d")

                    # Function is O(n*m)
                    alignment = self.extract_align(path_matrix, sequ_i, sequ_j)

                    s = {'align_cost': dist_matrix[-1][-1],
                         'seqi_first100': alignment[0],
                         'seqj_first100': alignment[1] }

                jresults.append(s)

            results.append(jresults)
        return results

  #O(3)
    def diff(self, sequ_i, sequ_j, k, l):
        if(sequ_i[l] == sequ_j[k]):
            return -3
        else:
            return 1

  #O(n*m)
    def extract_align(self, path_matrix, sequ_i, sequ_j):
        return_i = ""
        return_j = ""
        path = ""
        j = -1
        k = -1

       #get path from the bottom corner at most repeated the longer sequence (n or m)
        while(path_matrix[j][k] != ''):
            path += path_matrix[j][k]
            if(path_matrix[j][k] == 'd'):
                j = j - 1
                k = k - 1
```

Cody Kesler
CS 312 – 2
Lab 4 Write up



```
            elif(path_matrix[j][k] == 'a'):
                j = j - 1
            elif(path_matrix[j][k] == 'r'):
                k = k - 1

        path = path[::-1] # Reverse the path
        i_spot = 0
        j_spot = 0

        #Gets the correct alignment for the two sequences at most longest sequence (n
or m)
        for i in range(len(path)):
            if (path[i] == 'a'):
                return_i += "-"
            else:
                return_i += sequ_i[i_spot]
                i_spot += 1

            if (path[i] == 'r'):
                return_j += "-"
            else:
                return_j += sequ_j[j_spot]
                j_spot += 1

        return (return_i, return_j)
```

Complexity:

Time:

Time complexity is O(n*m) for each sequence. There are 10 sequences run at 10 times each, so it comes out to O(100*n*m) = O(n*m)

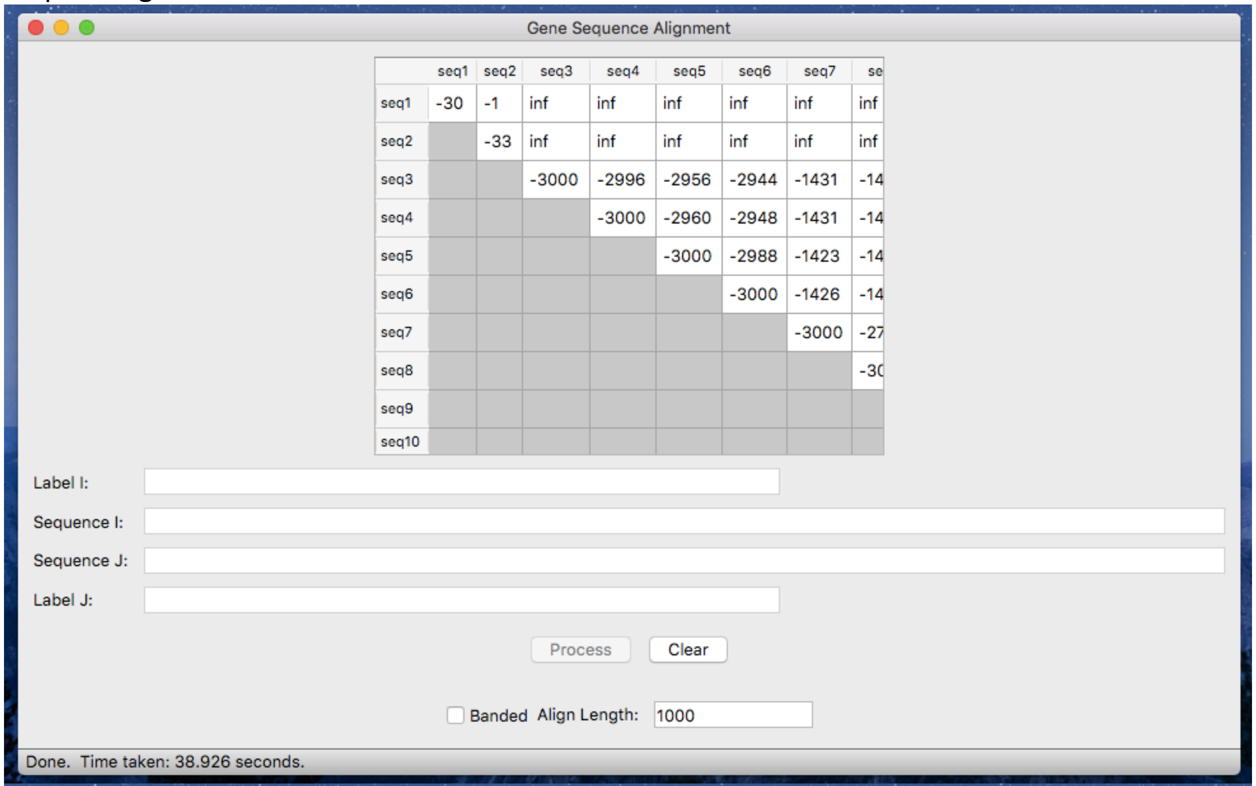
Space:

There are 2 matrices of size n*m made making the space O(2*n*m) = O(n*m)

Alignment Extraction:

For the alignment extraction I store a matrix the same size as the 2 sequences require. I store a string letter in each spot depending on if the lowest cost came from the diagonal, the right, or from above. I then go back through the matrix starting at the bottom corner (-1, -1) and moving diagonal, right, or up depending on the letter and add that letter to the path. I move up the matrix until I hit the top left corner and then stop. The path is then reversed, and the alignments are made. To make the alignments I put a dash in the output string if it comes from above and if not just add the letter of the sequence to the output string and then increment the spot in the sequence. I do the same for the second sequence but if it comes from the left (moves right) then I put a dash in the output string.

Cody Kesler
CS 312 – 2
Lab 4 Write up
Sequencing for un-banded at 1000:

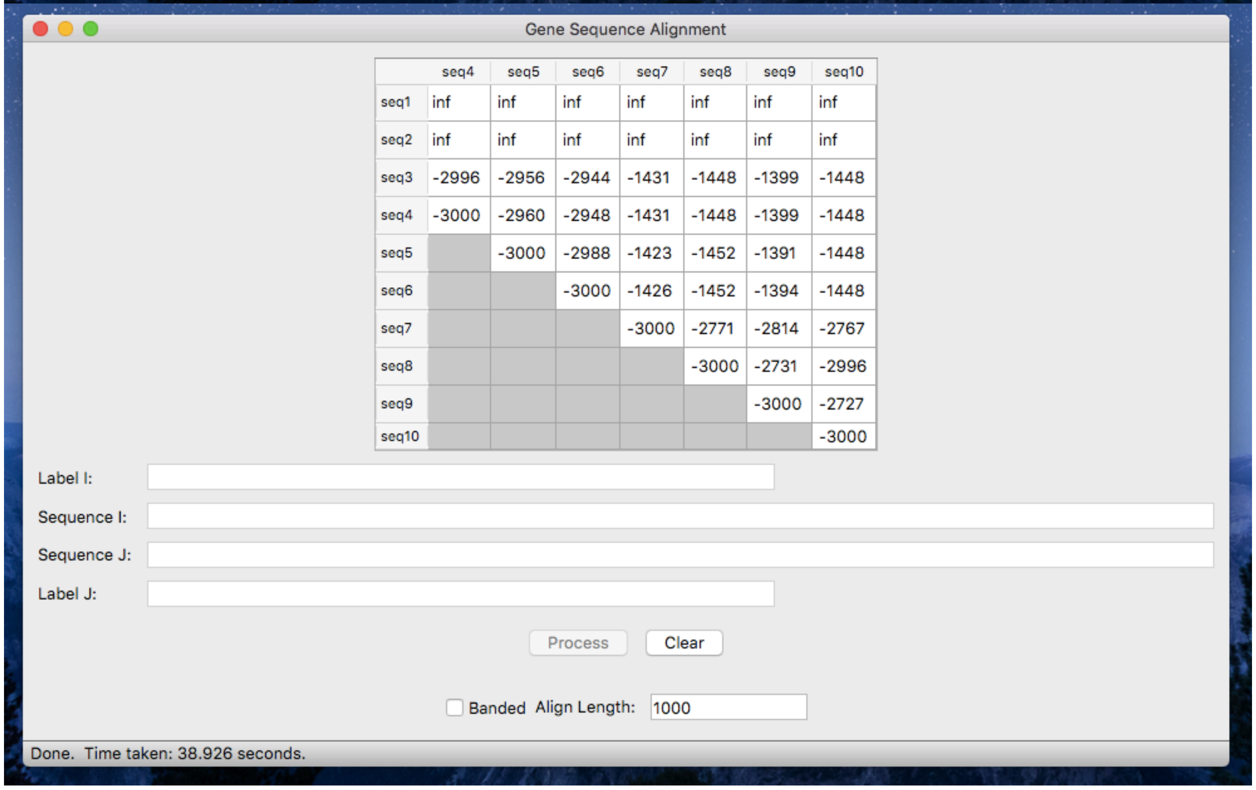


	seq1	seq2	seq3	seq4	seq5	seq6	seq7	se
seq1	-30	-1	inf	inf	inf	inf	inf	inf
seq2		-33	inf	inf	inf	inf	inf	inf
seq3			-3000	-2996	-2956	-2944	-1431	-14
seq4				-3000	-2960	-2948	-1431	-14
seq5					-3000	-2988	-1423	-14
seq6						-3000	-1426	-14
seq7							-3000	-27
seq8								-30
seq9								
seq10								

Label I:
Sequence I:
Sequence J:
Label J:

Banded Align Length: 1000

Done. Time taken: 38.926 seconds.



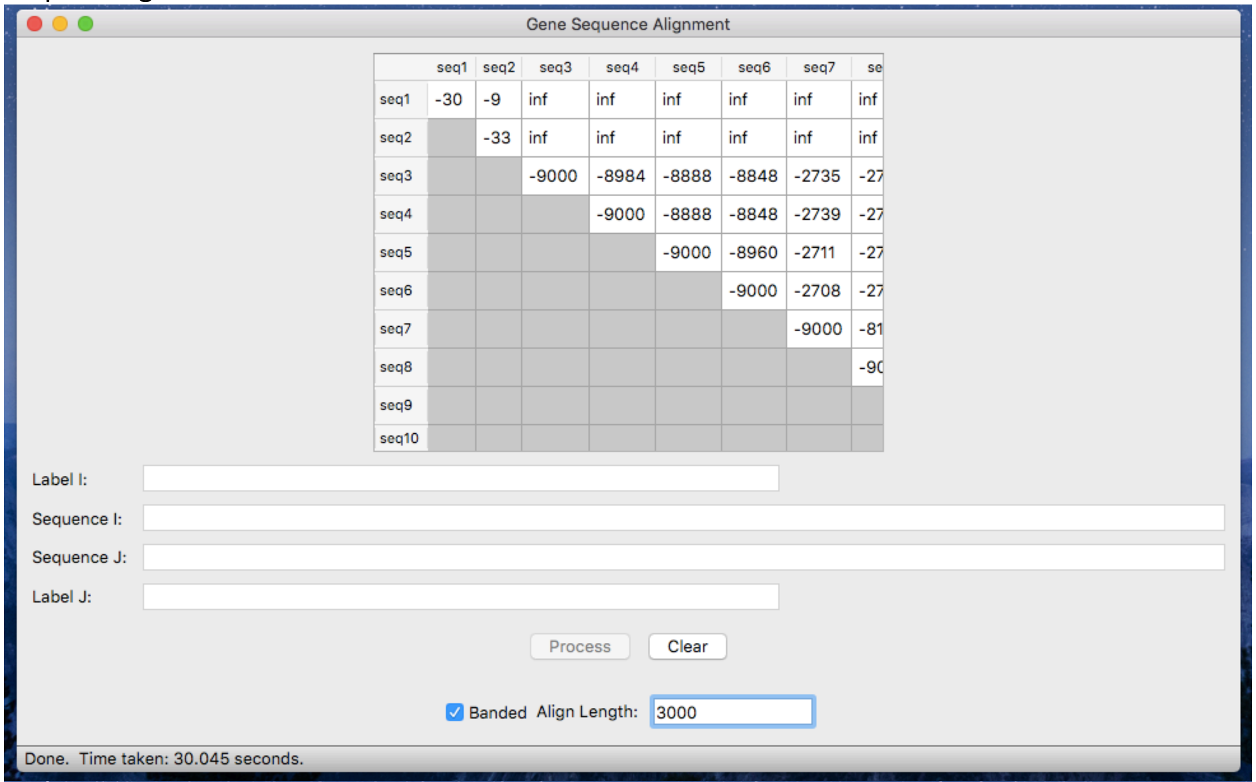
	seq4	seq5	seq6	seq7	seq8	seq9	seq10
seq1	inf	inf	inf	inf	inf	inf	inf
seq2	inf	inf	inf	inf	inf	inf	inf
seq3	-2996	-2956	-2944	-1431	-1448	-1399	-1448
seq4	-3000	-2960	-2948	-1431	-1448	-1399	-1448
seq5		-3000	-2988	-1423	-1452	-1391	-1448
seq6			-3000	-1426	-1452	-1394	-1448
seq7				-3000	-2771	-2814	-2767
seq8					-3000	-2731	-2996
seq9						-3000	-2727
seq10							-3000

Label I:
Sequence I:
Sequence J:
Label J:

Banded Align Length: 1000

Done. Time taken: 38.926 seconds.

Cody Kesler
CS 312 – 2
Lab 4 Write up
Sequencing for banded at 3000:



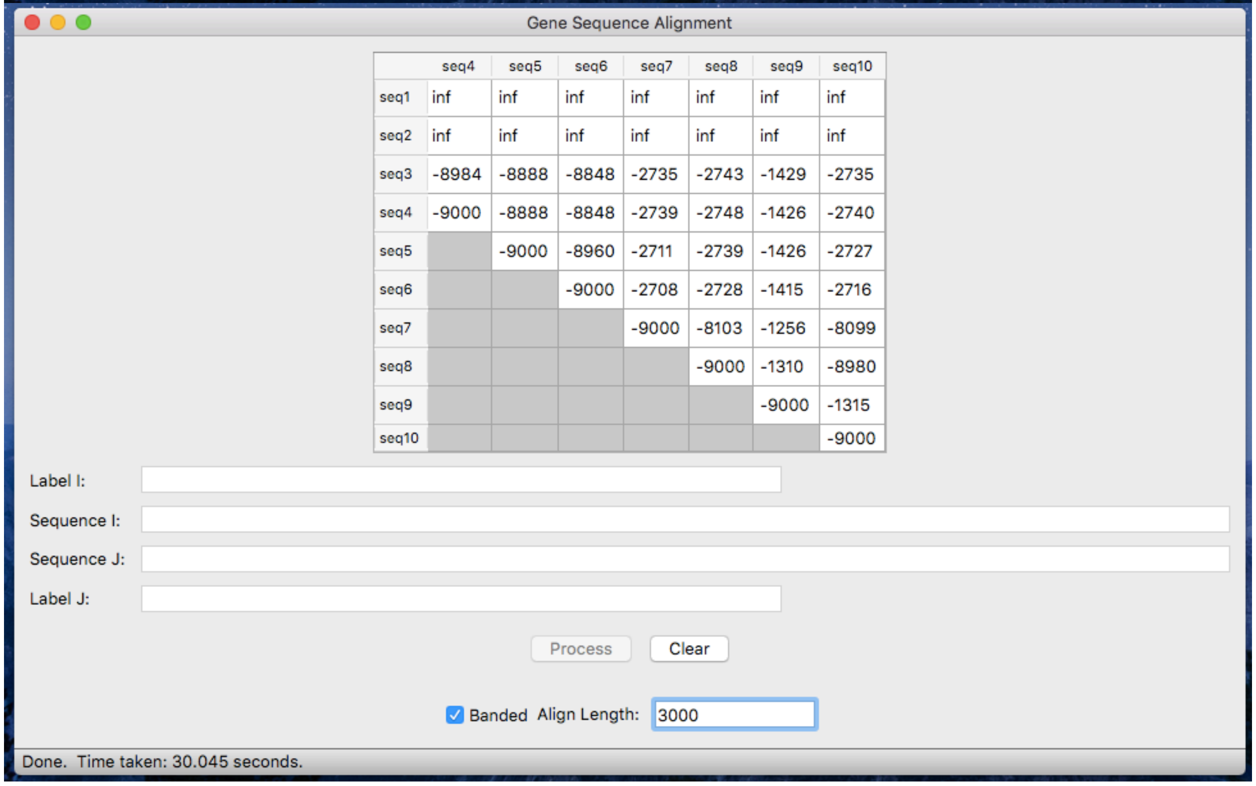
Gene Sequence Alignment

	seq1	seq2	seq3	seq4	seq5	seq6	seq7	se
seq1	-30	-9	inf	inf	inf	inf	inf	inf
seq2		-33	inf	inf	inf	inf	inf	inf
seq3			-9000	-8984	-8888	-8848	-2735	-27
seq4				-9000	-8888	-8848	-2739	-27
seq5					-9000	-8960	-2711	-27
seq6						-9000	-2708	-27
seq7							-9000	-81
seq8								-90
seq9								
seq10								

Label I:
Sequence I:
Sequence J:
Label J:

Banded [✓] Align Length: 3000

Done. Time taken: 30.045 seconds.



Gene Sequence Alignment

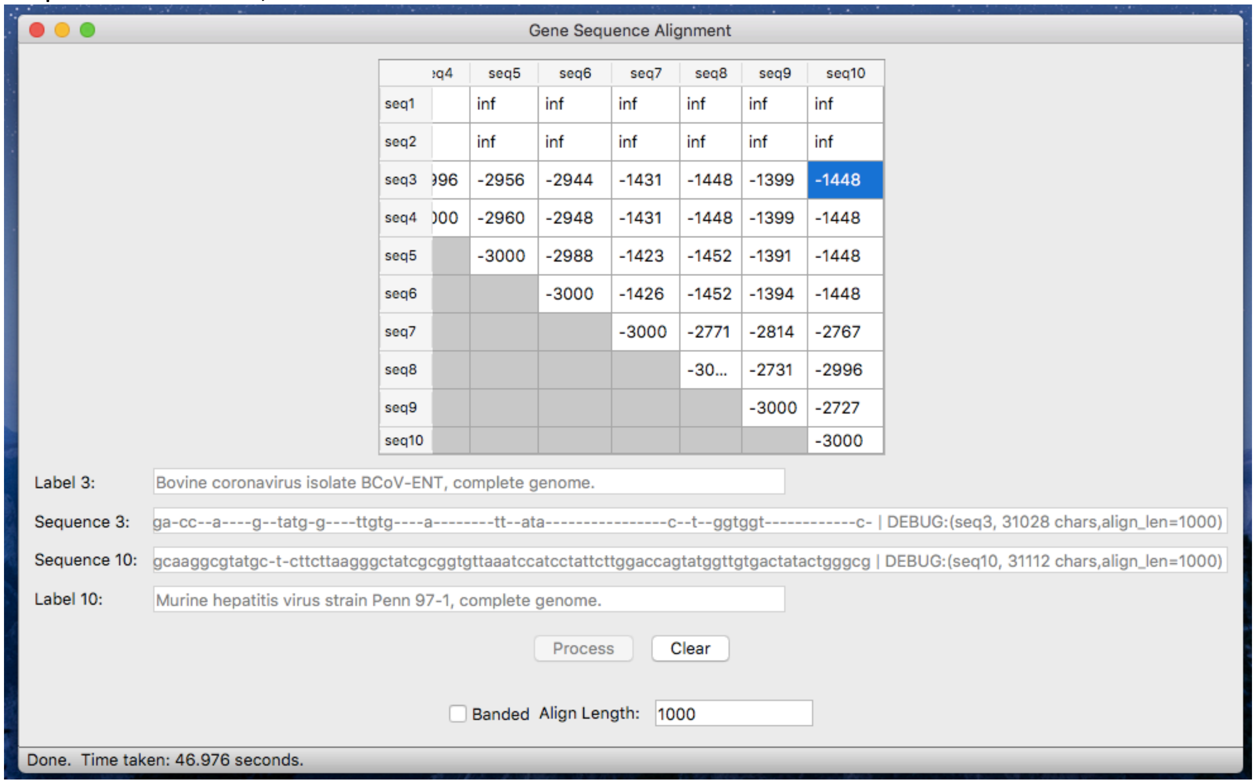
	seq4	seq5	seq6	seq7	seq8	seq9	seq10
seq1	inf	inf	inf	inf	inf	inf	inf
seq2	inf	inf	inf	inf	inf	inf	inf
seq3	-8984	-8888	-8848	-2735	-2743	-1429	-2735
seq4	-9000	-8888	-8848	-2739	-2748	-1426	-2740
seq5		-9000	-8960	-2711	-2739	-1426	-2727
seq6			-9000	-2708	-2728	-1415	-2716
seq7				-9000	-8103	-1256	-8099
seq8					-9000	-1310	-8980
seq9						-9000	-1315
seq10							-9000

Label I:
Sequence I:
Sequence J:
Label J:

Banded [✓] Align Length: 3000

Done. Time taken: 30.045 seconds.

Cody Kesler
CS 312 – 2
Lab 4 Write up
Sequences 3 and 10, un-banded:



Gene Sequence Alignment

	:q4	seq5	seq6	seq7	seq8	seq9	seq10
seq1		inf	inf	inf	inf	inf	inf
seq2		inf	inf	inf	inf	inf	inf
seq3	)96	-2956	-2944	-1431	-1448	-1399	-1448
seq4	)00	-2960	-2948	-1431	-1448	-1399	-1448
seq5		-3000	-2988	-1423	-1452	-1391	-1448
seq6			-3000	-1426	-1452	-1394	-1448
seq7				-3000	-2771	-2814	-2767
seq8					-30...	-2731	-2996
seq9						-3000	-2727
seq10							-3000

Label 3: Bovine coronavirus isolate BCoV-ENT, complete genome.

Sequence 3: ga-cc--a----g--tatg-g----ttgtg----a--------tt--ata---------------c--t--ggtggt-----------c- | DEBUG:(seq3, 31028 chars,align_len=1000)

Sequence 10: gcaaggcgtatgc-t-cttcttaagggctatcgcggtgttaaatccatcctattcttggaccagtatggttgtgactatactgggcg | DEBUG:(seq10, 31112 chars,align_len=1000)

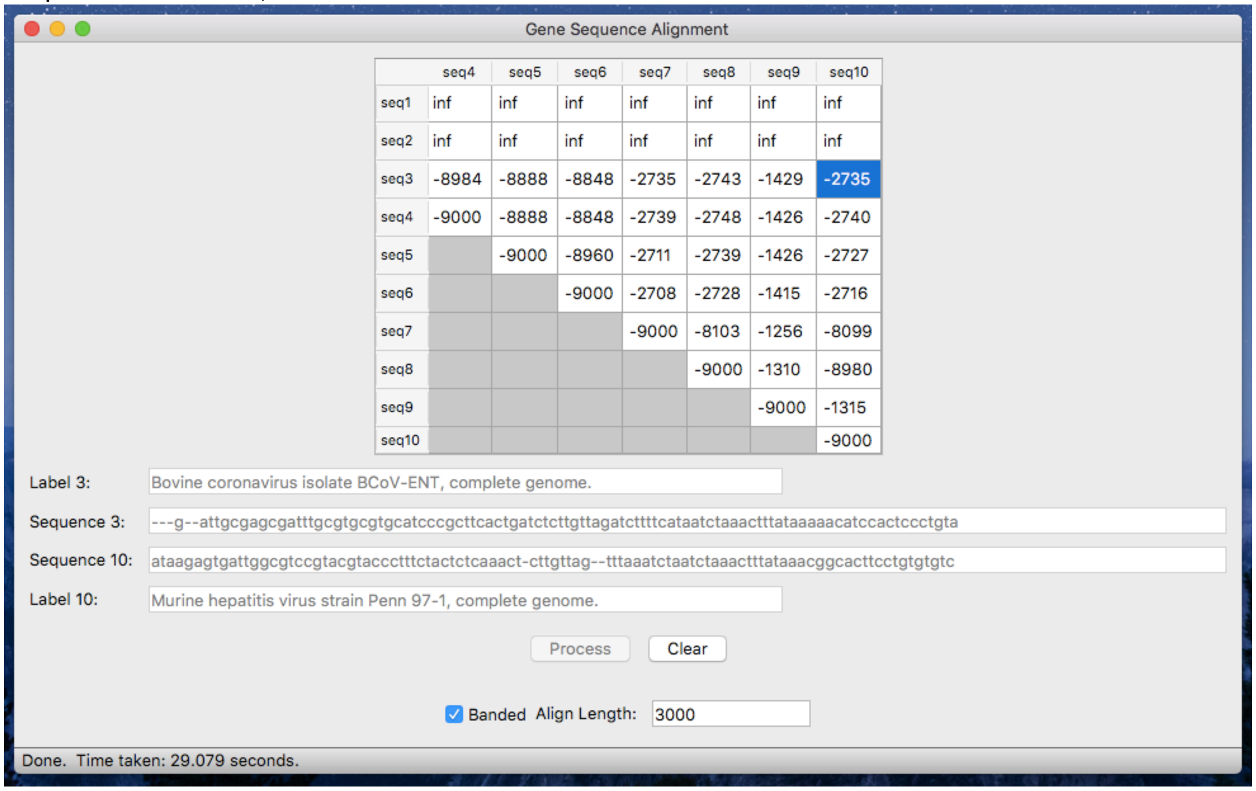
Label 10: Murine hepatitis virus strain Penn 97-1, complete genome.

Process Clear

☐ Banded Align Length: 1000

Done. Time taken: 46.976 seconds.

Sequences 3 and 10, banded:



Gene Sequence Alignment

	seq4	seq5	seq6	seq7	seq8	seq9	seq10
seq1	inf	inf	inf	inf	inf	inf	inf
seq2	inf	inf	inf	inf	inf	inf	inf
seq3	-8984	-8888	-8848	-2735	-2743	-1429	-2735
seq4	-9000	-8888	-8848	-2739	-2748	-1426	-2740
seq5		-9000	-8960	-2711	-2739	-1426	-2727
seq6			-9000	-2708	-2728	-1415	-2716
seq7				-9000	-8103	-1256	-8099
seq8					-9000	-1310	-8980
seq9						-9000	-1315
seq10							-9000

Label 3: Bovine coronavirus isolate BCoV-ENT, complete genome.

Sequence 3: ---g--attgcgagcgatttgcgtgcgtgcatcccgcttcactgatctcttgttagatcttttcataatctaaactttataaaaacatccactccctgta

Sequence 10: ataagagtgattggcgtccgtacgtaccctttctactctcaaact-cttgttag--tttaaatctaatctaaactttataaacggcacttcctgtgtgtc

Label 10: Murine hepatitis virus strain Penn 97-1, complete genome.

Process Clear

☑ Banded Align Length: 3000

Done. Time taken: 29.079 seconds.