Sequence Alignment Program with Mutation Location Features

Michael Kang and Michael Jenkins

CS 123

San José State University

Table of Contents

Problem and Purpose of Program:

Gene mutation is an essential aspect of researching the functions of certain proteins in an organism. Although there are many methods that are used to reduce the randomness of mutation sites, the verification of such mutations through sequencing is essential for experimental validation. In addition, locating mutations in already existing sequences is important for finding more targets for gene manipulation. These tasks are typically performed through sequence alignment. Locating differences in sequences by hand can be very tedious.

Existing sequence alignment programs, such as BLAST, perform well when aligning genomes. However, these programs do not output mutation locations when first viewed. Extra steps must be taken to look for mutation locations. In addition, these programs cannot be used when there is no internet connection available. The program described in this paper seeks to help streamline the process by outputting mutation type and location with the sequence alignment itself. It specifies the nucleotides replaced, along with the nucleotides that they were replaced with. The output is meant to be easily understood so that more time can be saved during research. Finally, the program is lightweight and can be used to compare smaller sequences without relying on anything other than the machine used to run it.

Design Documentation:

The program is an object-oriented dynamic programming approach to sequence alignment and data parsing. It will utilize the Needleman-Wunsch algorithm, which is a global alignment-based algorithm designed to divide problems into a series of smaller problems, then traceback until an optimal solution is reached. The way it will do this is by constructing a matrix where it will place the results of comparison score between each nucleotide from each sequence for each possible situation. The score is predetermined by the program to be that matches are 1 point, mismatches (or substitutions) are -1 point, and gaps (insert/delete) are -2 points. Once these results are calculated, the program will then start from the bottom most right positions and traceback towards the topmost left position, using the produced scores to determine the most optimal sequence. A more detailed explanation of this process will be covered during the sequence alignment module description.

The main method to initiate the program first requires the input of two text files provided within the project package itself, named seq1.txt and seq2.txt. These input files can be overwritten however it is important to keep the naming conventions so the program can load them successfully. An important note is that both files must contain **only** nucleotide sequence data, nothing else. The maximum allowed characters before the program runs into heap and memory issues is roughly 8200 characters, or 230 lines (if data is taken from sources such as NCBI). Once the proper sequences are placed into both files, the program will begin sequencing the data and will display the appropriate information.

The main class will contain the necessary functions required to parse the information provided by the sequence alignment module. It will contain a simple read file function that will encode the file into string format. The functions, displayMutations() and parseInfo(), will skim through each sequence string, compare them at each index, and produce results. For displayMutations(), it will be the in-between connection in the final result that shows a visual representation of where matches, gaps, and substitutions are done. For parseInfo(), it will take those same results and divide them into sections for easy reading.

The node class will be the object used within the scoring matrix and will contain the necessary information to help the process run smoothly. The node will represent a single nucleotide in a sequence. Within the class, there will be variables to contain the row and column in which the node will reside, as well as the score it was given and the most optimal (i.e. highest scored) previous node that it is connected to. These variables are there to help make the process of sequencing easier during traceback.

The sequence alignment class contains the algorithm and functions to produce the alignment. The constructor method will take the two strings taken from the sequence files and will instantiate a new scoring matrix based on the lengths of the two strings. It will then initialize the first row and first column of the matrix with scores based on the design of the algorithm. Needleman-Wunsch states that if the optimal path moves either horizontal or vertical, then the result is a gap, so the first row and first column are by default filled with the gap penalty. If the path is diagonal, then that is representative to a substitution, or mismatch. The function, setInitialInfo(), is responsible for initializing the first row and column with the appropriate gap penalty and will place a zero into all other nodes. The function, setNodeInfo(), is the workhorse behind the algorithm and is the primarily responsible for comparing both nucleotides and determining what score to place. It will determine this by comparing the diagonal, top, and left nodes to see which has the highest score after addition. It will then apply the highest score to the current node and continue to the next.

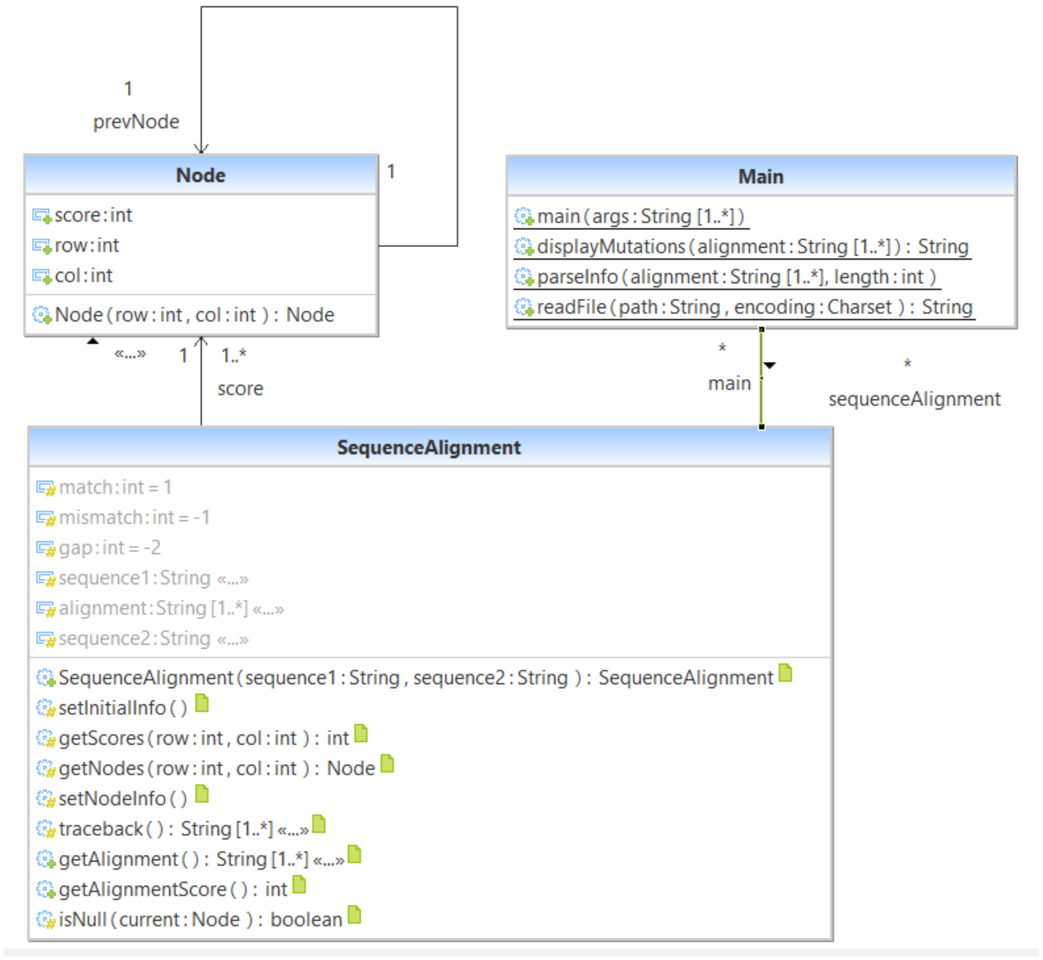
The function, traceback(), will return an array of two strings that will contain the produced sequence alignment. The method will start from the bottom most right node, and will check the scores of the top, left, and diagonal nodes around it. It will utilize the Node class variables for previous nodes and scores in order to achieve this. The result strings are instantiated as StringBuffers in order to allow insertions at index 0 of the string, or at the beginning of the string, since the traceback is going backwards. The function, alignmentScores(), simply runs through the produced alignment from traceback and compares each nucleotide at each index and tallies up a score for easy visibility.

The final output should initially show the matched sequence alignment with corresponding mutation symbols and the beginning and end indexes for each row of results. Followed by the alignment score and the parsed data divided into sections.

Instructions:

The program runs on Java 10 JDK or higher and was created specifically on the Eclipse IDE. Input files are provided within the project folder itself, named seq1.txt and seq2.txt. The text files should only contain sequence data, which means no additional labels or names. A maximum of 8200 characters is allowed for this program before memory and heap restrictions. Sequences taken from GenBank can be taken as is with labels removed and can be pasted directly into the seq1.txt or seq2.txt. The output is produced onto the console of the IDE. Once the appropriate sequence data is written onto the two text files, simply run the program from the Main class and it will begin sequencing. No additional options are available.

UML Diagram:



1
prevNode
Node
1
score : int
row : int
col : int
Node (row : int , col : int) : Node
Main
main (args : String [1..*])
displayMutations (alignment : String [1..*]) : String
parseInfo (alignment : String [1..*], length : int)
readFile (path : String , encoding : Charset) : String
«...»
1
1..*
score
*
main
*
sequenceAlignment
SequenceAlignment
match : int = 1
mismatch : int = -1
gap : int = -2
sequence1 : String «...»
alignment : String [1..*] «...»
sequence2 : String «...»
SequenceAlignment (sequence1 : String , sequence2 : String) : SequenceAlignment
setInitialInfo ( )
getScores (row : int , col : int) : int
getNodes (row : int , col : int) : Node
setNodeInfo ( )
traceback ( ) : String [1..*] «...»
getAlignment ( ) : String [1..*] «...»
getAlignmentScore ( ) : int
isNull (current : Node) : boolean

Test Data:

Tests were conducted using *E. coli* strains K-12 and O157:H7 str. Sakai.

Escherichia coli str. K-12 substr. MG1655K12, Accession: U00096

Obtained from https://www.ncbi.nlm.nih.gov/nuccore/U00096.3

Escherichia coli O157:H7 str. Sakai DNA, Accession: BA000007

Obtained from https://www.ncbi.nlm.nih.gov/nuccore/BA000007

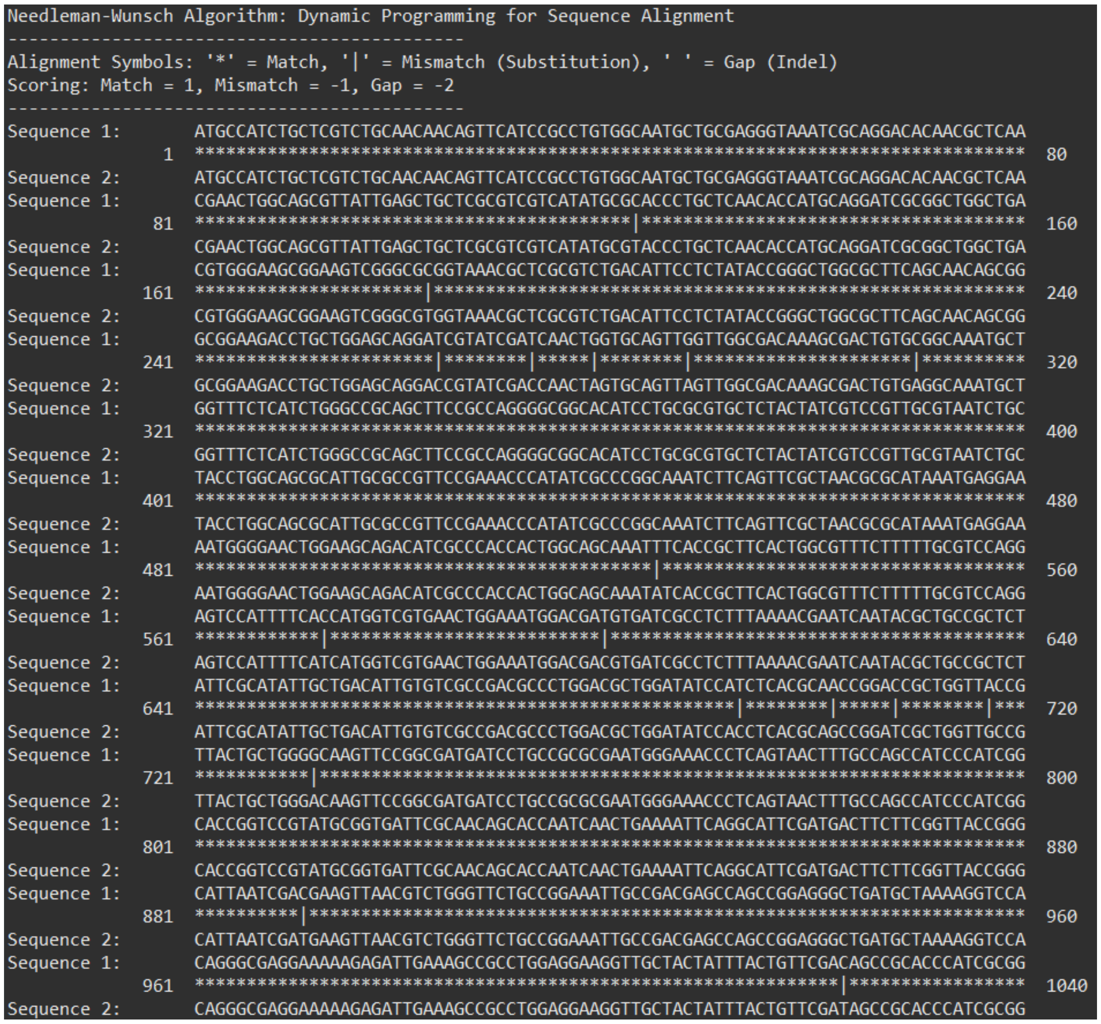
These strains were taken from NCBI's GenBank. *E. coli* str. K-12 is considered normal flora in the human gut and is considered non-pathogenic. *E. coli* O157:H7 str. Sakai is a shiga toxin-producing strain that is pathogenic. A BLAST showed that these two strains are 98.68% identical. Thus, while these two organisms are of the same species, their differences were significant. These were compared because they are known to have similar-length sequences and specific nucleotide differences when the same protein-encoding regions are used. In this case, sequences encoding sgrR in both organisms were selected from their complete genome sequences for testing. Entire genomic sequences could not be tested due to memory limitations. The tests carried out demonstrate the ability of the program to accurately determine the locations of mutation, the mutation type, and the number of those types of mutations. It also tested the program's ability to score the alignment and show the types of nucleotides that are different between the two sequences.

The test concluded that the program was operating appropriately and with consistent results. Several other genes found in the two strains were tested in order to sufficiently test the program, and all tests appeared sound and complete.

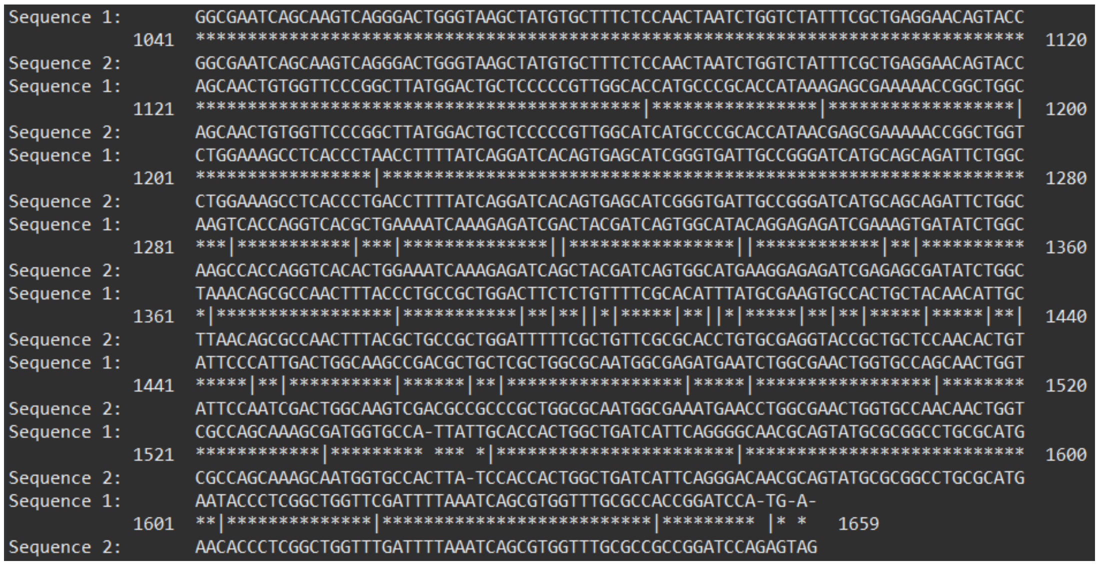
Comparison with NCBI Blast's Needleman-Wunsch Sequence Alignment:

Fairly similar results are present in both applications. NCBI's scoring for match/mismatch/gaps is slightly different than this project's implementation. For NCBI's Blast, the scoring for gaps was handled differently, which broke it into two categories: existence and extension. However, the results appeared the same barring an additional gap. The project implementation reported 1593 matches with 5 gaps while NCBI's implementation reported 1593 matches and 4 gaps.

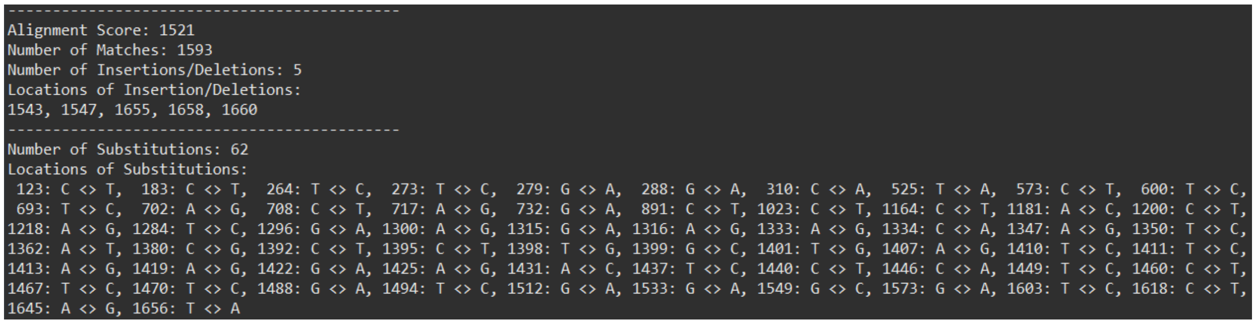
Output of Test Data:



```
Needleman-Wunsch Algorithm: Dynamic Programming for Sequence Alignment
--------------------------------------------
Alignment Symbols: '*' = Match, '|' = Mismatch (Substitution), ' ' = Gap (Indel)
Scoring: Match = 1, Mismatch = -1, Gap = -2
--------------------------------------------
Sequence 1:       ATGCCATCTGCTCGTCTGCAACAACAGTTCATCCGCCTGTGGCAATGCTGCGAGGGTAAATCGCAGGACACAACGCTCAA
                1 ********************************************************************************  80
Sequence 2:       ATGCCATCTGCTCGTCTGCAACAACAGTTCATCCGCCTGTGGCAATGCTGCGAGGGTAAATCGCAGGACACAACGCTCAA
Sequence 1:       CGAACTGGCAGCGTTATTGAGCTGCTCGCGTCGTCATATGCGCACCCTGCTCAACACCATGCAGGATCGCGGCTGGCTGA
               81 ******************************************|*************************************  160
Sequence 2:       CGAACTGGCAGCGTTATTGAGCTGCTCGCGTCGTCATATGCGTACCCTGCTCAACACCATGCAGGATCGCGGCTGGCTGA
Sequence 1:       CGTGGGAAGCGGAAGTCGGGCGCGGTAAACGCTCGCGTCTGACATTCCTCTATACCGGGCTGGCGCTTCAGCAACAGCGG
              161 **********************|*********************************************************  240
Sequence 2:       CGTGGGAAGCGGAAGTCGGGCGTGGTAAACGCTCGCGTCTGACATTCCTCTATACCGGGCTGGCGCTTCAGCAACAGCGG
Sequence 1:       GCGGAAGACCTGCTGGAGCAGGATCGTATCGATCAACTGGTGCAGTTGGTTGGCGACAAAGCGACTGTGCGGCAAATGCT
              241 ***********************|********|*****|********|*********************|**********  320
Sequence 2:       GCGGAAGACCTGCTGGAGCAGGACCGTATCGACCAACTAGTGCAGTTAGTTGGCGACAAAGCGACTGTGAGGCAAATGCT
Sequence 1:       GGTTTCTCATCTGGGCCGCAGCTTCCGCCAGGGGCGGCACATCCTGCGCGTGCTCTACTATCGTCCGTTGCGTAATCTGC
              321 ********************************************************************************  400
Sequence 2:       GGTTTCTCATCTGGGCCGCAGCTTCCGCCAGGGGCGGCACATCCTGCGCGTGCTCTACTATCGTCCGTTGCGTAATCTGC
Sequence 1:       TACCTGGCAGCGCATTGCGCCGTTCCGAAACCCATATCGCCCGGCAAATCTTCAGTTCGCTAACGCGCATAAATGAGGAA
              401 ********************************************************************************  480
Sequence 2:       TACCTGGCAGCGCATTGCGCCGTTCCGAAACCCATATCGCCCGGCAAATCTTCAGTTCGCTAACGCGCATAAATGAGGAA
Sequence 1:       AATGGGGAACTGGAAGCAGACATCGCCCACCACTGGCAGCAAATTTCACCGCTTCACTGGCGTTTCTTTTTGCGTCCAGG
              481 ********************************************|***********************************  560
Sequence 2:       AATGGGGAACTGGAAGCAGACATCGCCCACCACTGGCAGCAAATATCACCGCTTCACTGGCGTTTCTTTTTGCGTCCAGG
Sequence 1:       AGTCCATTTTCACCATGGTCGTGAACTGGAAATGGACGATGTGATCGCCTCTTTAAAACGAATCAATACGCTGCCGCTCT
              561 ************|**************************|****************************************  640
Sequence 2:       AGTCCATTTTCATCATGGTCGTGAACTGGAAATGGACGACGTGATCGCCTCTTTAAAACGAATCAATACGCTGCCGCTCT
Sequence 1:       ATTCGCATATTGCTGACATTGTGTCGCCGACGCCCTGGACGCTGGATATCCATCTCACGCAACCGGACCGCTGGTTACCG
              641 ****************************************************|********|*****|********|***  720
Sequence 2:       ATTCGCATATTGCTGACATTGTGTCGCCGACGCCCTGGACGCTGGATATCCACCTCACGCAGCCGGATCGCTGGTTGCCG
Sequence 1:       TTACTGCTGGGGCAAGTTCCGGCGATGATCCTGCCGCGCGAATGGGAAACCCTCAGTAACTTTGCCAGCCATCCCATCGG
              721 ***********|********************************************************************  800
Sequence 2:       TTACTGCTGGGACAAGTTCCGGCGATGATCCTGCCGCGCGAATGGGAAACCCTCAGTAACTTTGCCAGCCATCCCATCGG
Sequence 1:       CACCGGTCCGTATGCGGTGATTCGCAACAGCACCAATCAACTGAAAATTCAGGCATTCGATGACTTCTTCGGTTACCGGG
              801 ********************************************************************************  880
Sequence 2:       CACCGGTCCGTATGCGGTGATTCGCAACAGCACCAATCAACTGAAAATTCAGGCATTCGATGACTTCTTCGGTTACCGGG
Sequence 1:       CATTAATCGACGAAGTTAACGTCTGGGTTCTGCCGGAAATTGCCGACGAGCCAGCCGGAGGGCTGATGCTAAAAGGTCCA
              881 **********|*********************************************************************  960
Sequence 2:       CATTAATCGATGAAGTTAACGTCTGGGTTCTGCCGGAAATTGCCGACGAGCCAGCCGGAGGGCTGATGCTAAAAGGTCCA
Sequence 1:       CAGGGCGAGGAAAAAGAGATTGAAAGCCGCCTGGAGGAAGGTTGCTACTATTTACTGTTCGACAGCCGCACCCATCGCGG
              961 **************************************************************|*****************  1040
Sequence 2:       CAGGGCGAGGAAAAAGAGATTGAAAGCCGCCTGGAGGAAGGTTGCTACTATTTACTGTTCGATAGCCGCACCCATCGCGG
```

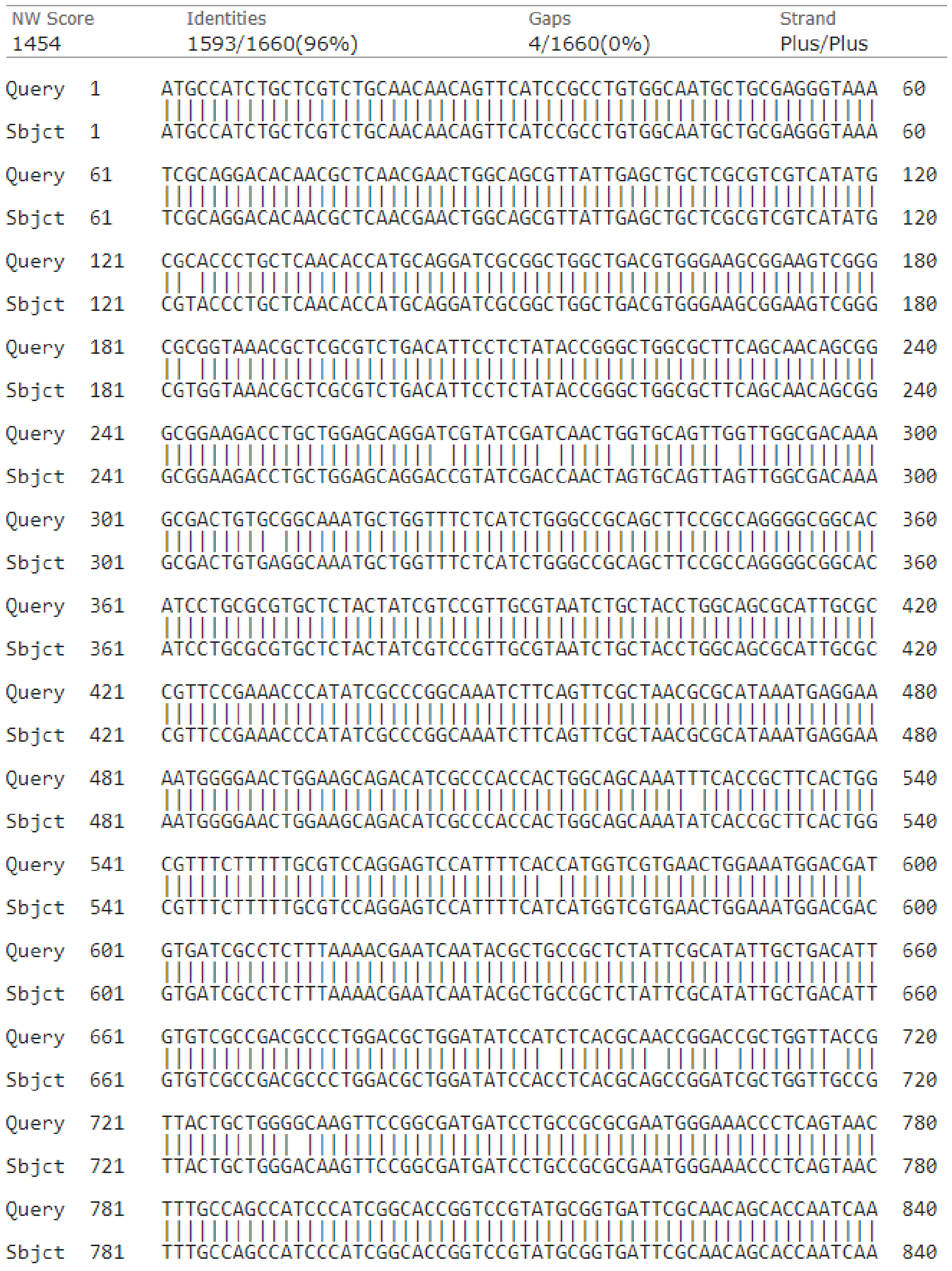


```
Sequence 1:       GGCGAATCAGCAAGTCAGGGACTGGGTAAGCTATGTGCTTTCTCCAACTAATCTGGTCTATTTCGCTGAGGAACAGTACC
             1041 ********************************************************************************  1120
Sequence 2:       GGCGAATCAGCAAGTCAGGGACTGGGTAAGCTATGTGCTTTCTCCAACTAATCTGGTCTATTTCGCTGAGGAACAGTACC
Sequence 1:       AGCAACTGTGGTTCCCGGCTTATGGACTGCTCCCCCGTTGGCACCATGCCCGCACCATAAAGAGCGAAAAACCGGCTGGC
             1121 *******************************************|****************|******************|  1200
Sequence 2:       AGCAACTGTGGTTCCCGGCTTATGGACTGCTCCCCCGTTGGCATCATGCCCGCACCATAACGAGCGAAAAACCGGCTGGT
Sequence 1:       CTGGAAAGCCTCACCCTAACCTTTTATCAGGATCACAGTGAGCATCGGGTGATTGCCGGGATCATGCAGCAGATTCTGGC
             1201 *****************|**************************************************************  1280
Sequence 2:       CTGGAAAGCCTCACCCTGACCTTTTATCAGGATCACAGTGAGCATCGGGTGATTGCCGGGATCATGCAGCAGATTCTGGC
Sequence 1:       AAGTCACCAGGTCACGCTGAAAATCAAAGAGATCGACTACGATCAGTGGCATACAGGAGAGATCGAAAGTGATATCTGGC
             1281 ***|***********|***|**************||****************||************|**|**********  1360
Sequence 2:       AAGCCACCAGGTCACACTGGAAATCAAAGAGATCAGCTACGATCAGTGGCATGAAGGAGAGATCGAGAGCGATATCTGGC
Sequence 1:       TAAACAGCGCCAACTTTACCCTGCCGCTGGACTTCTCTGTTTTCGCACATTTATGCGAAGTGCCACTGCTACAACATTGC
             1361 *|*****************|***********|**|**||*|*****|**||*|*****|**|**|*****|*****|**|  1440
Sequence 2:       TTAACAGCGCCAACTTTACGCTGCCGCTGGATTTTTCGCTGTTCGCGCACCTGTGCGAGGTACCGCTGCTCCAACACTGT
Sequence 1:       ATTCCCATTGACTGGCAAGCCGACGCTGCTCGCTGGCGCAATGGCGAGATGAATCTGGCGAACTGGTGCCAGCAACTGGT
             1441 *****|**|**********|******|**|*****************|*****|*****************|********  1520
Sequence 2:       ATTCCAATCGACTGGCAAGTCGACGCCGCCCGCTGGCGCAATGGCGAAATGAACCTGGCGAACTGGTGCCAACAACTGGT
Sequence 1:       CGCCAGCAAAGCGATGGTGCCA-TTATTGCACCACTGGCTGATCATTCAGGGGCAACGCAGTATGCGCGGCCTGCGCATG
             1521 ************|********* *** *|***********************|***************************  1600
Sequence 2:       CGCCAGCAAAGCAATGGTGCCACTTA-TCCACCACTGGCTGATCATTCAGGGACAACGCAGTATGCGCGGCCTGCGCATG
Sequence 1:       AATACCCTCGGCTGGTTCGATTTTAAATCAGCGTGGTTTGCGCCACCGGATCCA-TG-A-
             1601 **|**************|*************************|********* |* *   1659
Sequence 2:       AACACCCTCGGCTGGTTTGATTTTAAATCAGCGTGGTTTGCGCCGCCGGATCCAGAGTAG
```



```
--------------------------------------------
Alignment Score: 1521
Number of Matches: 1593
Number of Insertions/Deletions: 5
Locations of Insertion/Deletions:
1543, 1547, 1655, 1658, 1660
--------------------------------------------
Number of Substitutions: 62
Locations of Substitutions:
 123: C <> T,  183: C <> T,  264: T <> C,  273: T <> C,  279: G <> A,  288: G <> A,  310: C <> A,  525: T <> A,  573: C <> T,  600: T <> C,
 693: T <> C,  702: A <> G,  708: C <> T,  717: A <> G,  732: G <> A,  891: C <> T, 1023: C <> T, 1164: C <> T, 1181: A <> C, 1200: C <> T,
1218: A <> G, 1284: T <> C, 1296: G <> A, 1300: A <> G, 1315: G <> A, 1316: A <> G, 1333: A <> G, 1334: C <> A, 1347: A <> G, 1350: T <> C,
1362: A <> T, 1380: C <> G, 1392: C <> T, 1395: C <> T, 1398: T <> G, 1399: G <> C, 1401: T <> G, 1407: A <> G, 1410: T <> C, 1411: T <> C,
1413: A <> G, 1419: A <> G, 1422: G <> A, 1425: A <> G, 1431: A <> C, 1437: T <> C, 1440: C <> T, 1446: C <> A, 1449: T <> C, 1460: C <> T,
1467: T <> C, 1470: T <> C, 1488: G <> A, 1494: T <> C, 1512: G <> A, 1533: G <> A, 1549: G <> C, 1573: G <> A, 1603: T <> C, 1618: C <> T,
1645: A <> G, 1656: T <> A
```

NCBI’s BLAST Results:



```
NW Score      Identities           Gaps           Strand
1454          1593/1660(96%)       4/1660(0%)     Plus/Plus

Query  1     ATGCCATCTGCTCGTCTGCAACAACAGTTCATCCGCCTGTGGCAATGCTGCGAGGGTAAA  60
             ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  1     ATGCCATCTGCTCGTCTGCAACAACAGTTCATCCGCCTGTGGCAATGCTGCGAGGGTAAA  60

Query  61    TCGCAGGACACAACGCTCAACGAACTGGCAGCGTTATTGAGCTGCTCGCGTCGTCATATG  120
             ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  61    TCGCAGGACACAACGCTCAACGAACTGGCAGCGTTATTGAGCTGCTCGCGTCGTCATATG  120

Query  121   CGCACCCTGCTCAACACCATGCAGGATCGCGGCTGGCTGACGTGGGAAGCGGAAGTCGGG  180
             || |||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  121   CGTACCCTGCTCAACACCATGCAGGATCGCGGCTGGCTGACGTGGGAAGCGGAAGTCGGG  180

Query  181   CGCGGTAAACGCTCGCGTCTGACATTCCTCTATACCGGGCTGGCGCTTCAGCAACAGCGG  240
             || |||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  181   CGTGGTAAACGCTCGCGTCTGACATTCCTCTATACCGGGCTGGCGCTTCAGCAACAGCGG  240

Query  241   GCGGAAGACCTGCTGGAGCAGGATCGTATCGATCAACTGGTGCAGTTGGTTGGCGACAAA  300
             ||||||||||||||||||||||| |||||||| ||||| |||||||| ||||||||||||
Sbjct  241   GCGGAAGACCTGCTGGAGCAGGACCGTATCGACCAACTAGTGCAGTTAGTTGGCGACAAA  300

Query  301   GCGACTGTGCGGCAAATGCTGGTTTCTCATCTGGGCCGCAGCTTCCGCCAGGGGCGGCAC  360
             ||||||||| ||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  301   GCGACTGTGAGGCAAATGCTGGTTTCTCATCTGGGCCGCAGCTTCCGCCAGGGGCGGCAC  360

Query  361   ATCCTGCGCGTGCTCTACTATCGTCCGTTGCGTAATCTGCTACCTGGCAGCGCATTGCGC  420
             ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  361   ATCCTGCGCGTGCTCTACTATCGTCCGTTGCGTAATCTGCTACCTGGCAGCGCATTGCGC  420

Query  421   CGTTCCGAAACCCATATCGCCCGGCAAATCTTCAGTTCGCTAACGCGCATAAATGAGGAA  480
             ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  421   CGTTCCGAAACCCATATCGCCCGGCAAATCTTCAGTTCGCTAACGCGCATAAATGAGGAA  480

Query  481   AATGGGGAACTGGAAGCAGACATCGCCCACCACTGGCAGCAAATTTCACCGCTTCACTGG  540
             |||||||||||||||||||||||||||||||||||||||||||| |||||||||||||||
Sbjct  481   AATGGGGAACTGGAAGCAGACATCGCCCACCACTGGCAGCAAATATCACCGCTTCACTGG  540

Query  541   CGTTTCTTTTTGCGTCCAGGAGTCCATTTTCACCATGGTCGTGAACTGGAAATGGACGAT  600
             |||||||||||||||||||||||||||||||| |||||||||||||||||||||||||| 
Sbjct  541   CGTTTCTTTTTGCGTCCAGGAGTCCATTTTCATCATGGTCGTGAACTGGAAATGGACGAC  600

Query  601   GTGATCGCCTCTTTAAAACGAATCAATACGCTGCCGCTCTATTCGCATATTGCTGACATT  660
             ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  601   GTGATCGCCTCTTTAAAACGAATCAATACGCTGCCGCTCTATTCGCATATTGCTGACATT  660

Query  661   GTGTCGCCGACGCCCTGGACGCTGGATATCCATCTCACGCAACCGGACCGCTGGTTACCG  720
             |||||||||||||||||||||||||||||||| |||||||| ||||| |||||||| |||
Sbjct  661   GTGTCGCCGACGCCCTGGACGCTGGATATCCACCTCACGCAGCCGGATCGCTGGTTGCCG  720

Query  721   TTACTGCTGGGGCAAGTTCCGGCGATGATCCTGCCGCGCGAATGGGAAACCCTCAGTAAC  780
             ||||||||||| ||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  721   TTACTGCTGGGACAAGTTCCGGCGATGATCCTGCCGCGCGAATGGGAAACCCTCAGTAAC  780

Query  781   TTTGCCAGCCATCCCATCGGCACCGGTCCGTATGCGGTGATTCGCAACAGCACCAATCAA  840
             ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
Sbjct  781   TTTGCCAGCCATCCCATCGGCACCGGTCCGTATGCGGTGATTCGCAACAGCACCAATCAA  840
```

References

Hayashi T., Makino K., Ohnishi M., Kurokawa K., Ishii K., Yokoyama K., Han C., Ohtsubo E., Nakayama K., Murata T., Tanaka M., Tobe T., Iida T., Takami., Honda T., Sasakawa C., Ogasawara N., Yasunaga T., Kuhara S., Shiba T., Hattori M., Shinagawa H. (2001). Complete Genome Sequence of *Enterohemorrhagic Eschelichia coli* O157:H7 and Genomic Comparison with a Laboratory Strain K-12. *DNA Res*. *8*: 11-22. doi: 10.1093/dnares/8.1.11

Needleman, Saul B. & Wunsch, Christian D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J Mol Bio. 48 (3):* 443–53. doi:10.1016/0022-2836(70)90057-4.

Wikipedia contributors. (2019, November 15). Needleman–Wunsch algorithm. *In Wikipedia, The Free Encyclopedia.* Retrieved 04:17, November 28, 2019, from https://en.wikipedia.org/w/index.php?title=Needleman%E2%80%93Wunsch_algorithm&oldid=926225687

Zhang Z., Schwartz S., Wagner L., Miller W. (2000). A greedy algorithm for aligning DNA sequences. *J Comput Biol 2000, 7(1-2)*,203-214.

Zvelebil M. & Baum J O. (2000). *Understanding Bioinformatics.* New York, NY. Garland Science, Taylor and Francis Group, LLC.