**Biological Sequence Alignment using Dynamic Programming**

**Kareith Dyce**+1**, James Ann**+1**, and Xavier Verdad** +1

1Auburn University, COMP, Auburn, 36849, USA \* kzd0030@auburn.edu

+these authors contributed equally to this work

**ABSTRACT**

In Biology the similarity between two sequences tells quite a bit about the two. The more similar their sequences are the closer they are related. Knowing this it can be easier to fight one virus if it is closely related to another one with a known cure. Using normal comparisons however to find the similarities would take forever as most of these sequences are over 100 characters in length. This is where using two Dynamic Programming strategies the Needleman-Wunsch algorithm and the Smith-Waterman algorithm vastly reduces the time spent calculating the alignment and quickly allows for finding the similarities between the sequences.

**Introduction**

Just like many things protein sequences can also undergo evolution. This means that there are many different protein sequences that came from a common original sequence. This similar means that if we know how to combat / treat one virus if we find that another one is closely related, we could combat it through similar means.

However, finding is not especially easy as these sequences evolve in three different ways, Substitution, Deletion and Insertion. As the names would suggest Insertion is adding to the sequence, Deletion is removing form the sequence and Substitution is changing letters in the sequence. Each of these evolutions can change more than just one letter which can make the alignment process harder. Also, both Insertion and Deletion would change a sequences length so, just because two sequences have the same length does not mean that there are closely related. Also, the opposite is true so that just because the sequences vary in length does not make it impossible for them to be closely related. These small evolutions over time has led to huge diversions from the common ancestral sequence.

Now being able to check every alignment possible between the two sequences to find the best global alignment (how the two full sequences compare to each other) would take forever with normal programming practices to just to find an alignment with a length of 100. These sequences could get very long so this is where we would utilize dynamic programming. Here we would be able to find the best global alignments using the Needleman-Wunsch algorithm. Modifying this method in to the Smith-Waterman algorithm allows us to find the local alignment (the highest similar sequence within the sequence) to the find the common parts of the sequence which helps finds viabilities in the protein.

**Methods**

Without Dynamic Programming to find the best alignment of two sequences would be to make all the possible to alignments and score each of them and return the highest. However, analyzing this shows how long this would take. If the two sequences are of length n and we are allowed r spaces, then putting if we use all r spaces and with the n letters we have or combinations for sequence one with r spaces. For sequence two since we cannot have space matches we have combinations with r spaces. Since we can have few than r spaces the we must sum the product of these two combinations which gives us To give an example of how many alignments this is having two sequences of a size of just ten gives about 8,097,453 possible alignments. This a lot of alignments for just 10 characters and for protein sequences which can could easily have a length over 100 means this will take forever. Using Dynamic Programming strategies, we can cut down the work needed to find the best global alignment and the best local alignment. For the best global alignment, we use an algorithm called Needleman-Wunsch and for the best local alignment we use modified Needleman-Wunsch version of called the Smith-Waterman algorithm.

**Needleman-Wunsch**

The Needleman-Wunsch can be used to find the best global alignment much faster than searching through all the possible alignments. Instead the algorithm makes a grid with the first sequence on the horizontal and the second sequence on the vertical. The grid is then filled in by taking the maximum of three calculations, the square to the left’s score subtracted by 1, the square above’s score minus 1 and the scores to upper left diagonal added by a given score by the BLOSUM matrix or subtracted. This is repeated until the lower right corner is reached. Then the path is backtracked by selecting the maximum for the squares above, left and upper left. Going horizontal in the best path means that a space is added to the second string and going vertical means that a space is added to the first string. A diagonal means that the was a comparison between the two sequences. Using these rules and backtracking allows the best path to be found by only considering m\*n squares where n is the length of sequence one and m is the length of sequence two. This brings down the work needed to analyze two sequences of length 10 from 8,097,453 to just 100.

***Smith-Waterman***

In the Smith-Waterman algorithm we essentially do the Needleman-Wunsch except we keep track of the where the maximum value is and negative numbers are not allowed, they are kept at zero. Backtracking is then done from the maximum square until the first zero in the grid is reached.

**Results**

***Dengue virus vs Zika virus***

Needleman-Wunsch

Score: 228

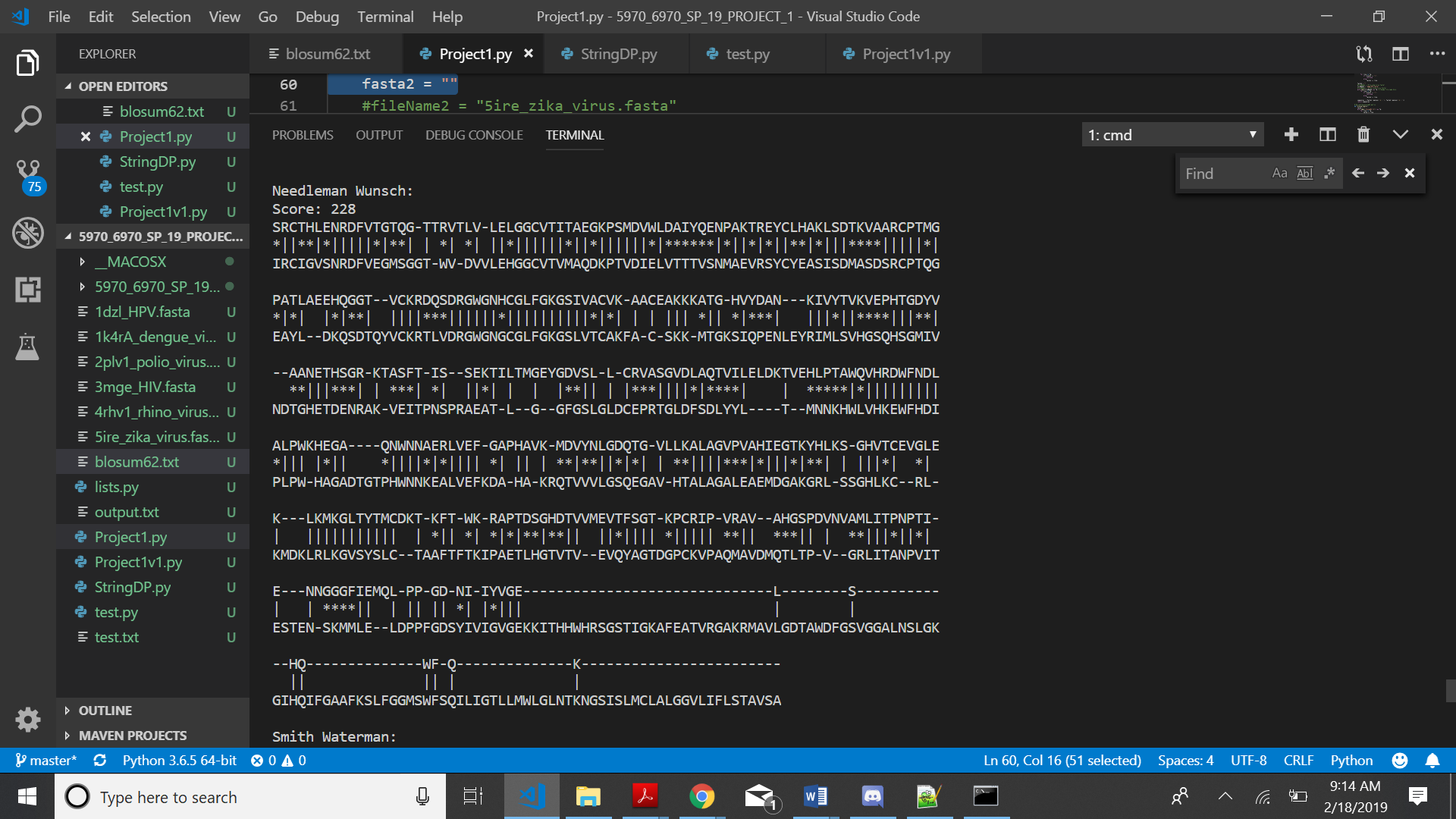


Figure 1.1: Needleman-Wunsch result for Pair 1

Smith-Waterman

Score: 582

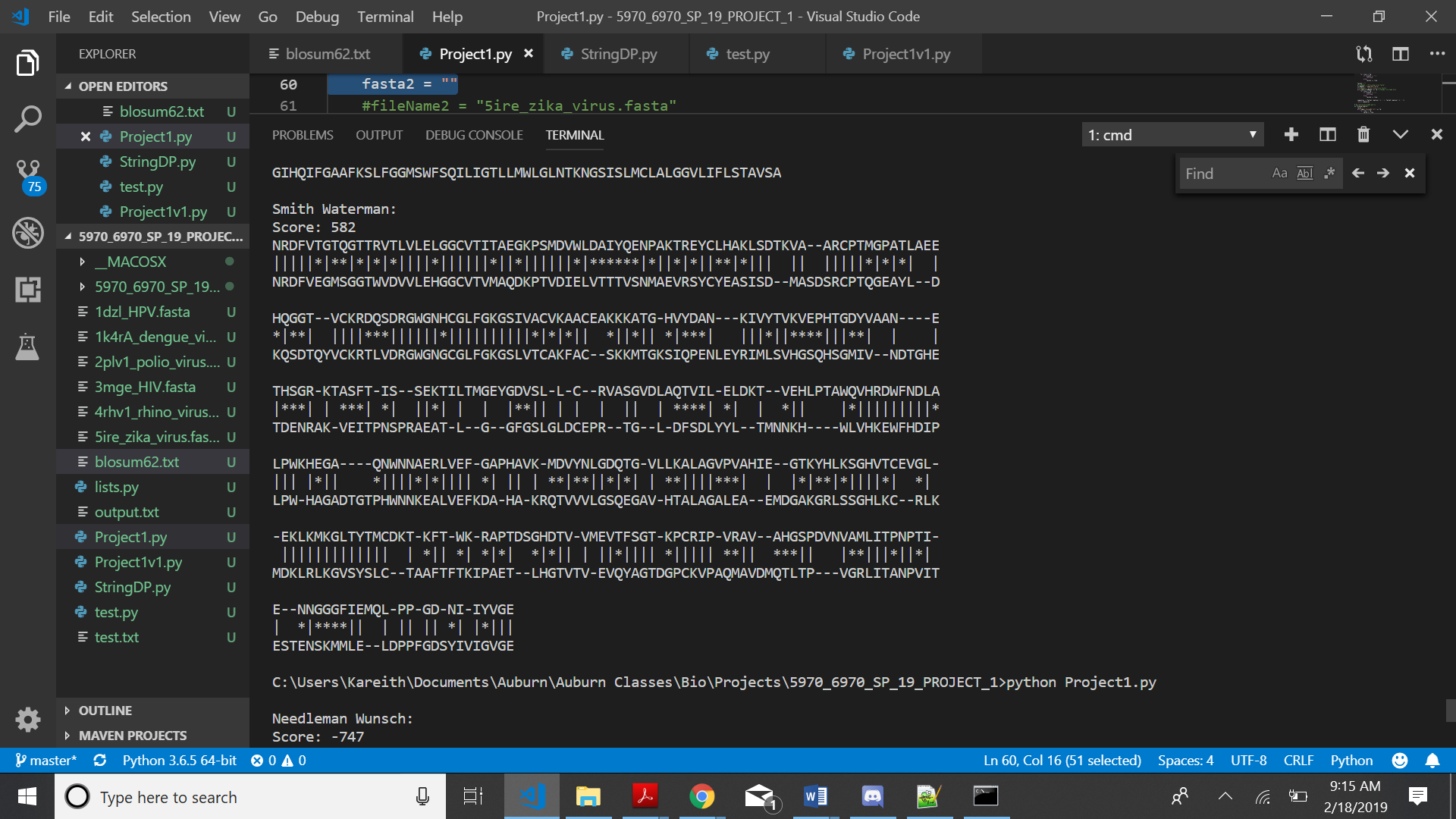


Figure 1.2: Smith-Waterman alignment for Pair 1

BLAST

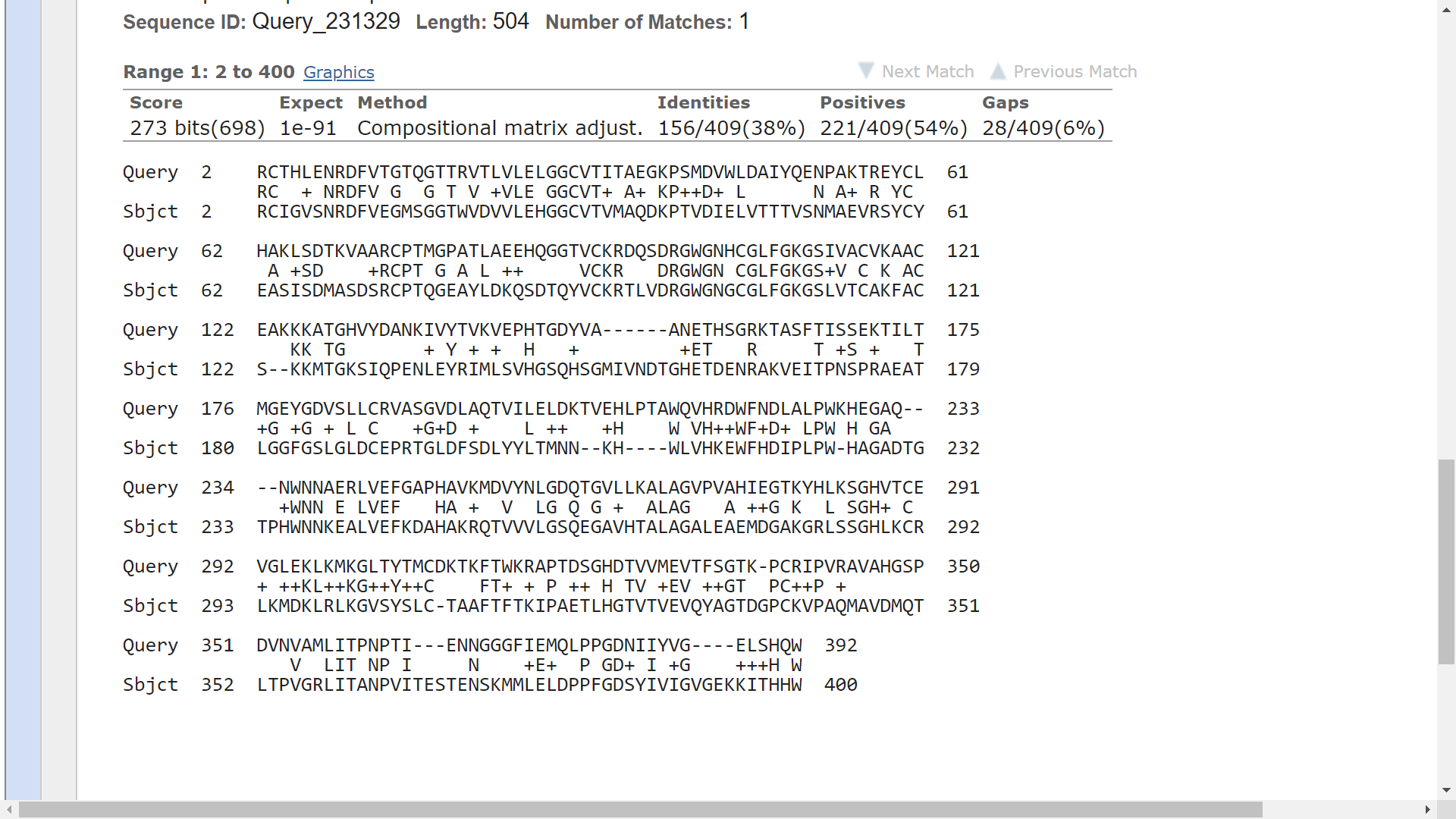


Figure 1.3: BLAST alignment for Pair 1

***HPV vs HIV***

Needleman-Wunsch

Score: -747

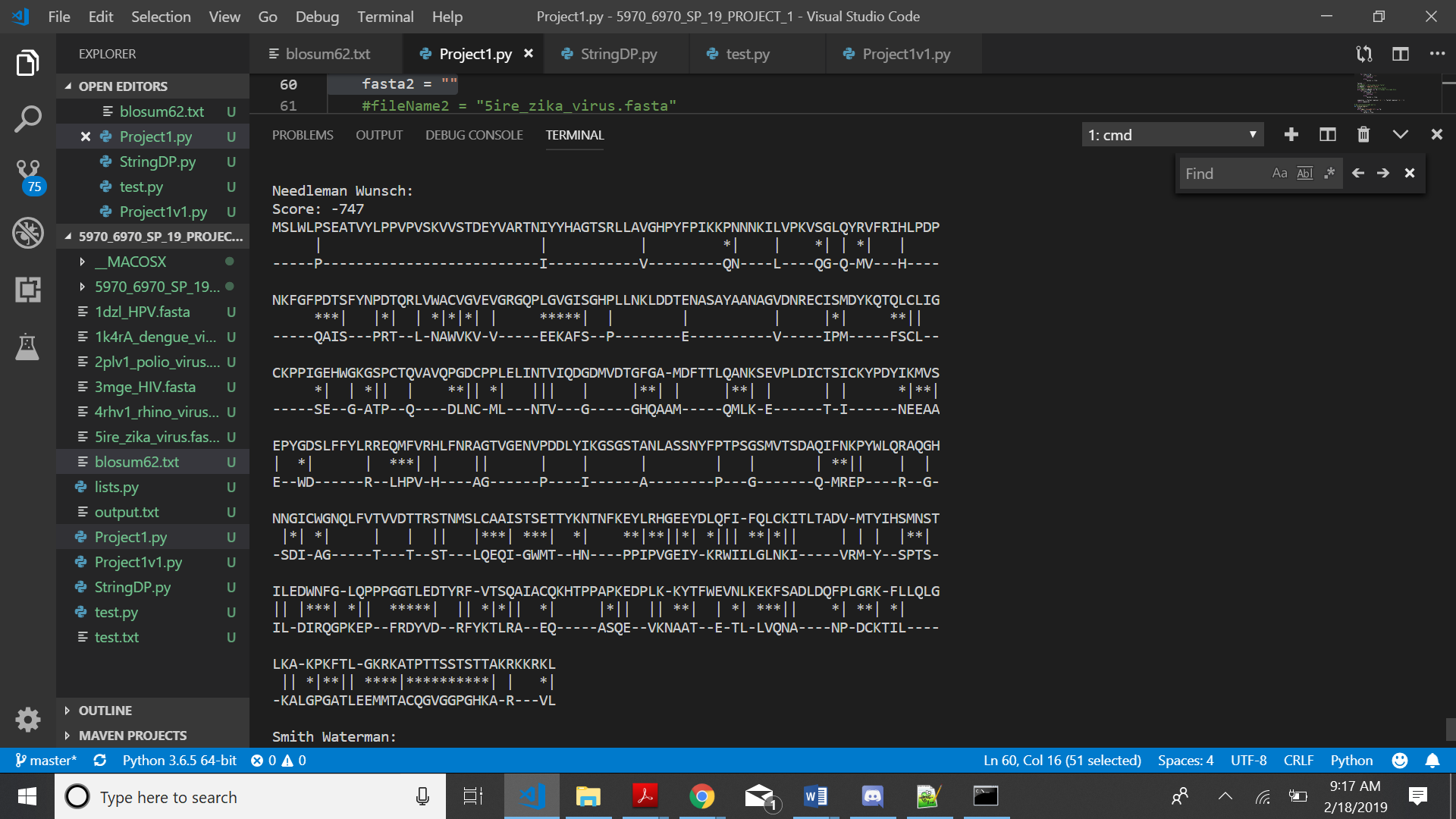


Figure 2.1: Needleman-Wunsch alignment of Pair 2

Smith-Waterman

Score: 14

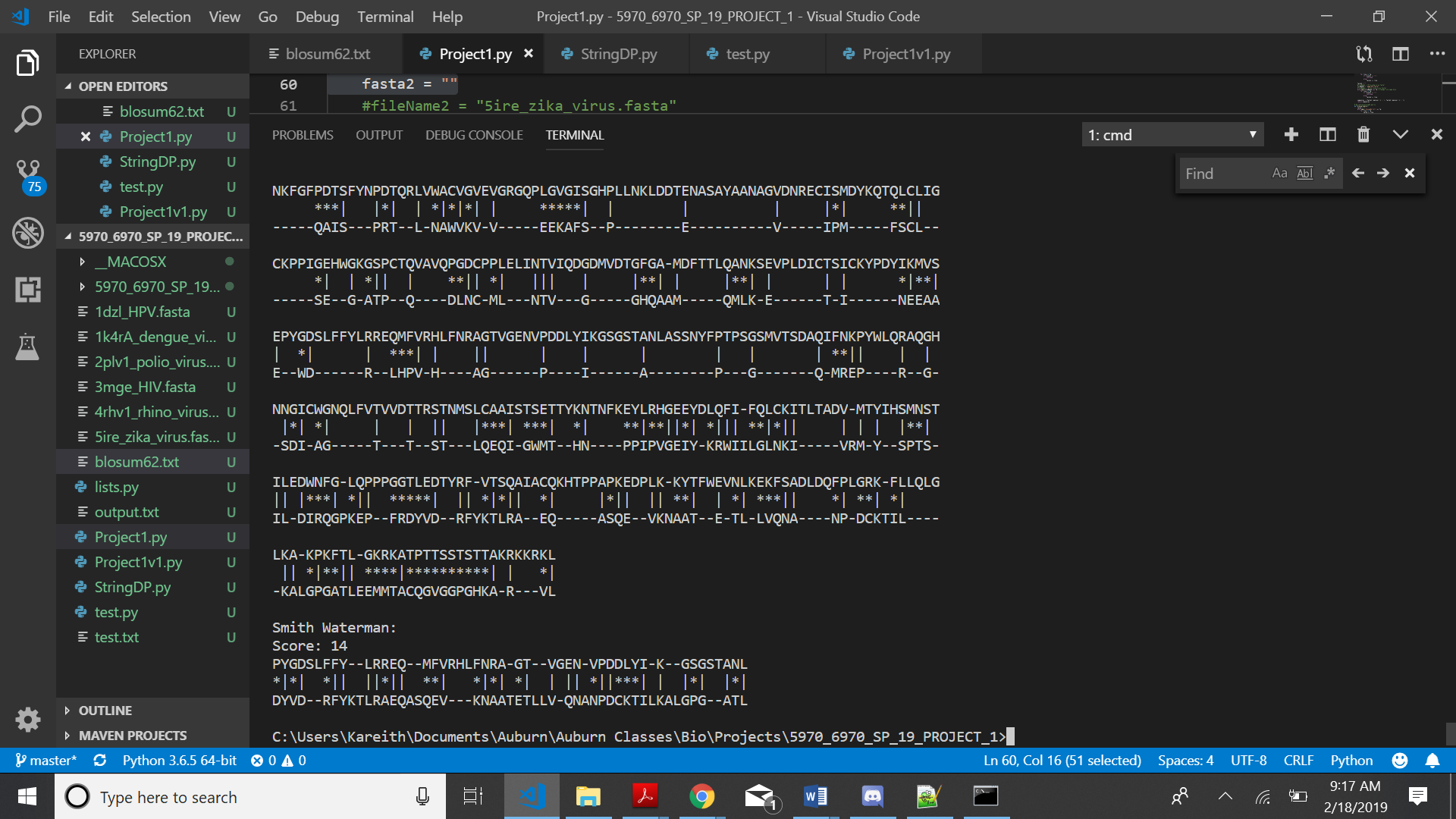


Figure 2.2: Smith-Waterman alignment of Pair 2

BLAST

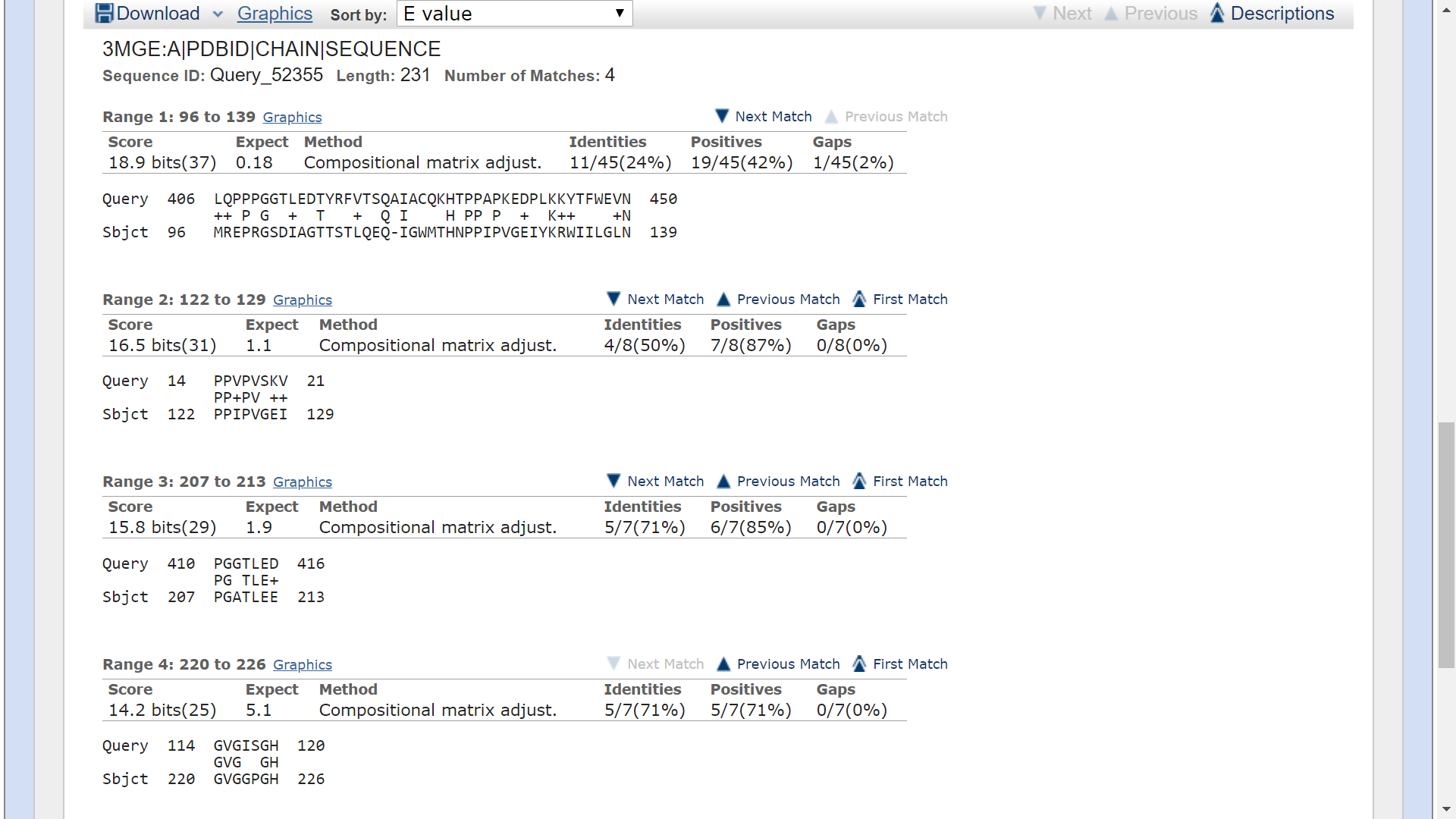


Figure 2.3: BLAST alignment of Pair 2

***Polio virus vs Rhino virus***

Needleman-Wunsch

Score: 516

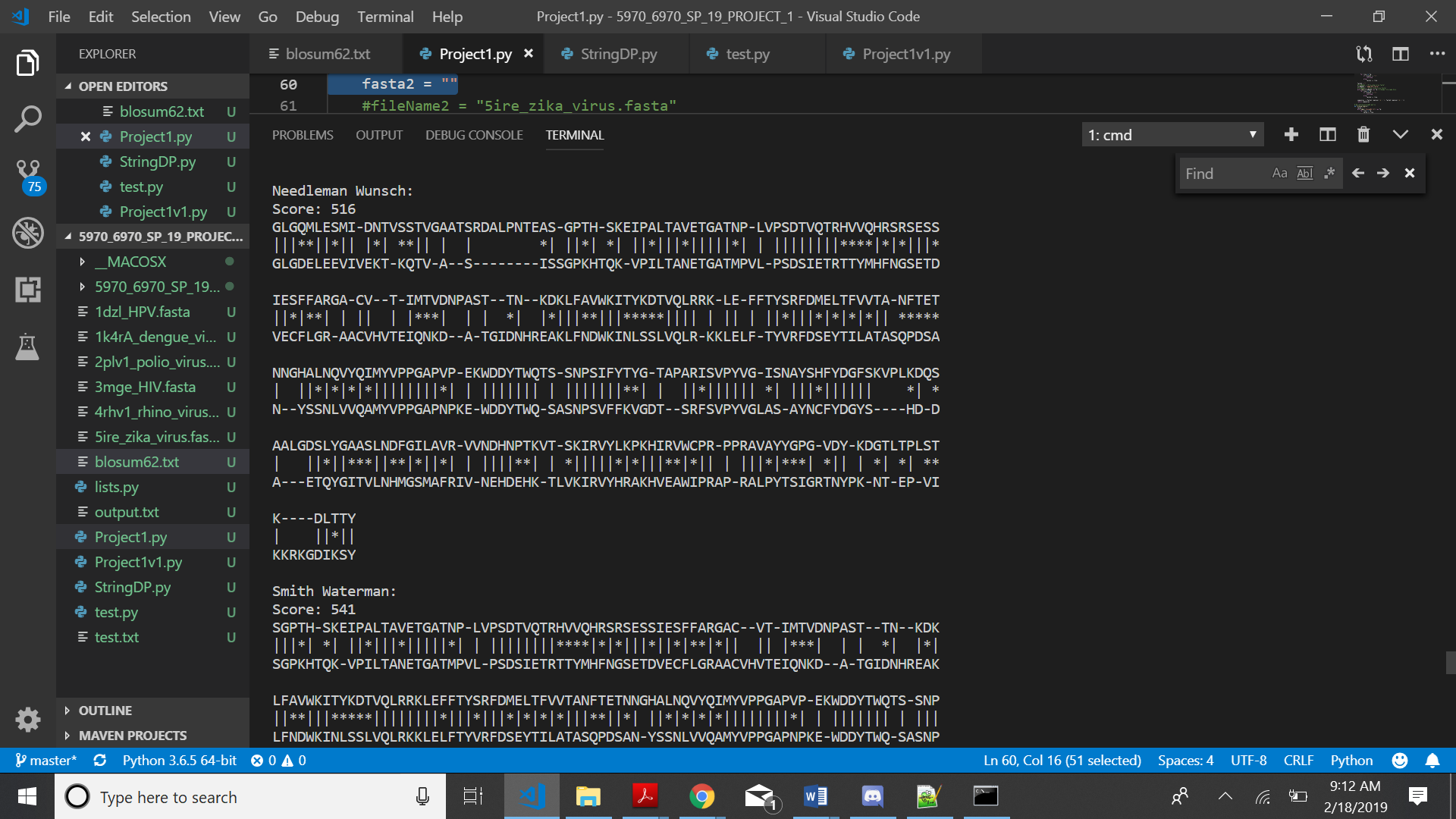


Figure 3.1: Needleman-Wunsch alignment of Pair 3

Smith-Waterman

Score: 541

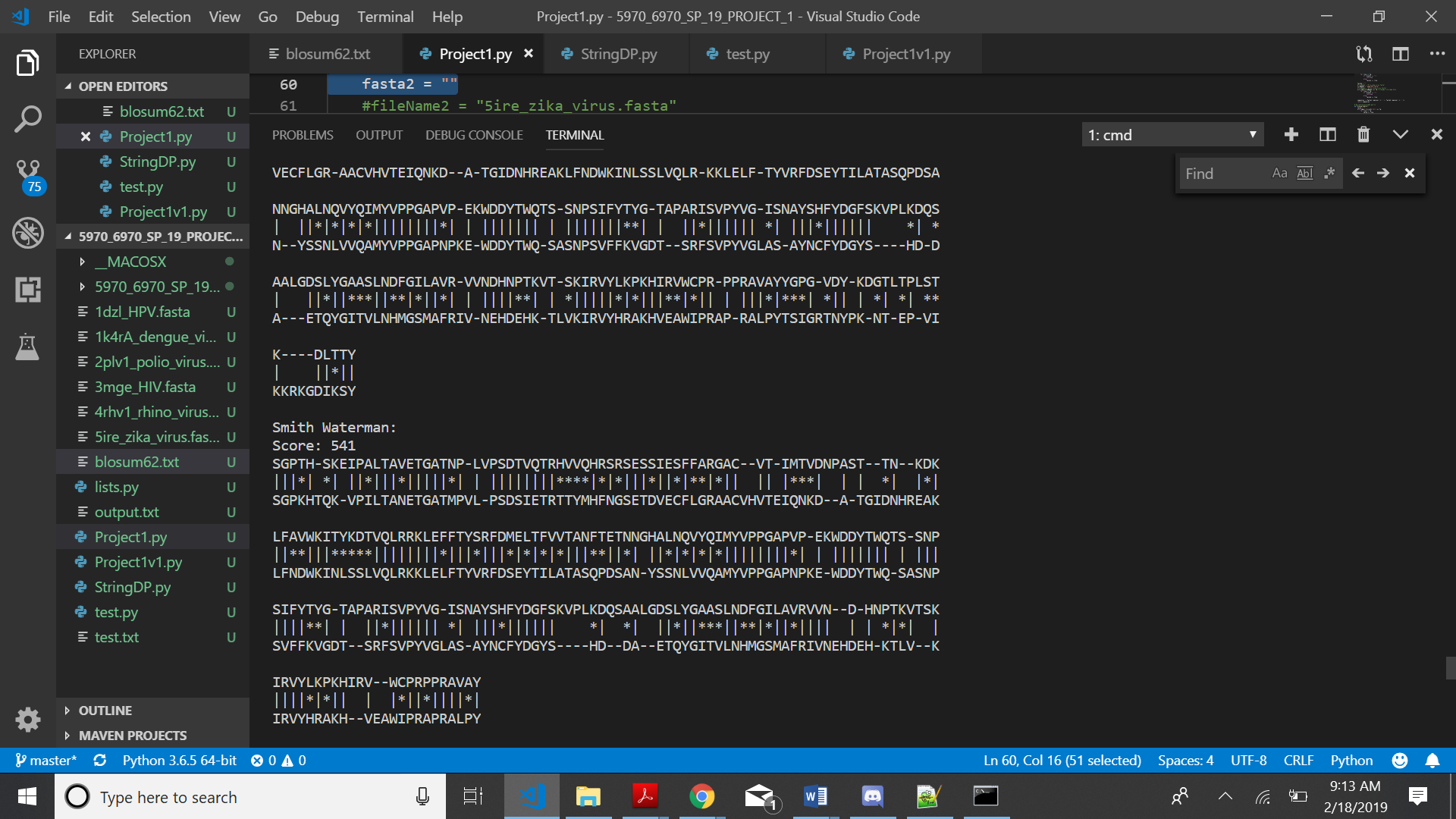


Figure 3.2: Smith-Waterman alignment of Pair 3

BLAST

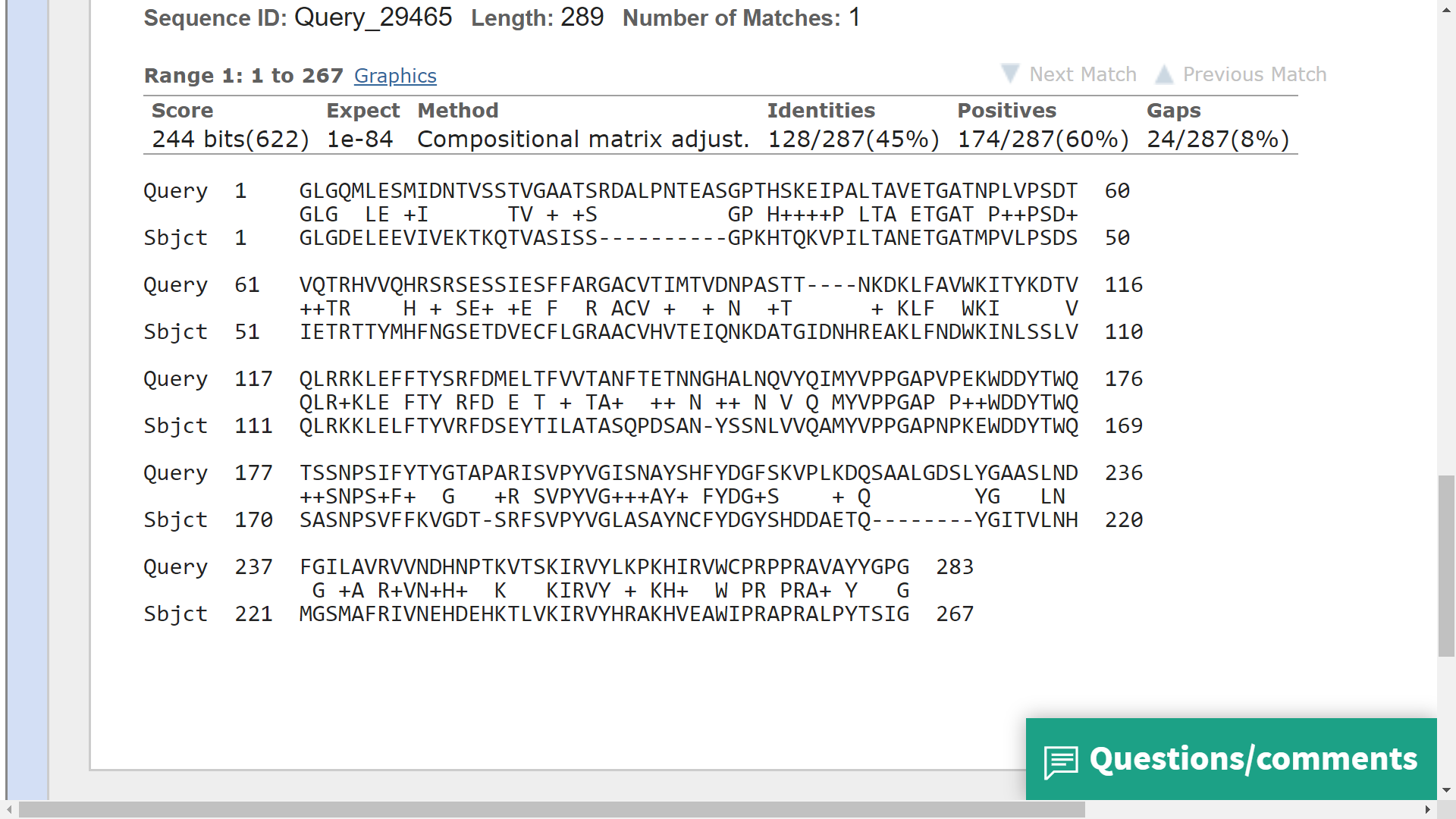


Figure 3.3: BLAST alignment of Pair 3

**Discussion**

Looking at the results the first thing that stands out is that the BLAST website is calculating local alignments and not global alignment. This is best shown in Figure 2.3 where there are four matches but, can still be seen in Figure 1.3 where the BLAST alignment starts at the second index. Comparing the local alignments for Figure 3.2 against Figure 3.3 we can see the difference in the two local alignments. For the Smith-Waterman which uses BLOSUM62 to score the alignment we see that this alignment ends fours characters earlier than its BLAST counter part meaning that there is a clear difference in scoring. In our algorithm we also see they we had much more spaces than the BLAST method hinting that the BLAST only used spaces as a last result while it was okay for our programs to have spaces. This likely means that the BLAST method penalized heavier for spaces more so than our implementation of the Smith-Waterman algorithm.

However overall there are still quite a few similarities in both approaches that shows a correlation. In Figure 2.1 we see that HIV and HPV are not closely related with a score of -747. Figures 2.2 and 2.3 back this up with Figure 2.2 giving the smallest local alignment score overall with a score of 14 and the BLAST alignment having four small sections. The first pair had the highest local alignment scores for both BLAST and our Smith-Waterman with similar alignments meaning that while the scoring was different there was still the same correlation regarding similarity. This difference could come from the difference in importance that the scoring matrices give for evolution.

**References**

1. “BLAST: Basic Local Alignment Search Tool.” *National Center for Biotechnology Information*, U.S. National Library of Medicine, blast.ncbi.nlm.nih.gov/Blast.cgi.