Alex Addy

Cory Sutyak

Nicholas Thompson

STAR Alignment

STAR alignment, which falls under the progressive alignment category, is one of the many methods of aligning multiple sequences, specifically DNA sequences. When aligning two sequences a dynamic algorithm can be applied to easily align the two sequences however once a third sequence is introduced the task requires a much longer running time. For three sequences a run time becomes O(n3) where n is the length of the sequences and any further additional sequences will give you a run time of O(2knk) where k is the number of sequences. Progressive alignment algorithms aim to accomplish this through creating a profile to compare sequences. In the STAR alignment method the profile is the sequence that is most similar to all the other sequences being aligned. A star is created, like the one shown in figure 1, according to the scores of the sequences. In this example sequence three is most similar to the other five sequences. Once the center sequence is determined all other sequences are aligned to the center sequence starting with the one with the highest score. According to figure one sequence three and sequence six have the highest pairwise alignment score and will be used as the basis for the overall alignment. The remaining sequences are added according to their alignment with the center sequence. For STAR alignment once a gap has been added that gap will always be there and can never be removed. STAR alignment's complexity comes out to be O( k2n2 + k2nmax) which comes out to be far superior to O(2knk). For our project our goal was to implement the STAR alignment algorithm in C# in order to demonstrate its capabilities.

S2

Figure 1

S5

S1

S3

S4

S6

Our implementation of the STAR alignment algorithm included a Windows form GUI and code written in C#. The choice of C# was made due to group member Cory Sutyak being most familiar with that particular language as well as its ease to create GUI's. The GUI itself can read a number of files to get any number of sequences. As for file format the GUI will look for a new sequence on each new line of the file and will only take in sequence written out in all-caps. Once a file is read in the sequences contained in it are added to the list on the left. Any number of files can be read in to create a longer list of sequences. With a click of the center button the program will align the sequences and output the list of aligned sequences on the right side. The process itself is fairly straightforward. The program begins by creating a matrix for storing the scores that are created when all of the sequences are aligned with each other. For the pairwise alignment the scores are negative one for a gap, zero for a mismatch, and one for a match. The algorithm used for the pairwise alignment was pulled from our first project and recreated in C#. Once the score table is filled out the program begins to align the sequences starting with the highest scoring pair. The program also includes an option to sort the sequences when they are outputted on the right.

In conclusion our project was a success. We set out to implement the STAR alignment algorithm in C# and did exactly that. Our program works as intended and is presented in a usable format. For the future some improvements could be made. For example the inclusion of supporting FASTA format would be a big improvement. The program could also be optimized in some areas. The pairwise alignment algorithm is written in a sort of one-to-one way. It creates matrices in order to keep track of scores when the comparisons are being done when another data structure or a better algorithm all together could help improve running time.