Pan Zhenyang(A0078075X)

Summary

This paper is about using dynamic programming in computing the similarities of the amino acid sequences of two proteins. The sequences are modeled as character strings. To solve the problem in the dynamic programming way, every state is treated as a pair of length, say L1 and L2. L1 and L2 are the lengths of two strings which initially are empty. Through several operations (insertion, deletion, match), at different states, the similarity of the strings will be different. They use a two-dimensional array to store the maximum similarity of states. In this way, the re-computation of every state will be avoided.

To make it more accurate, they introduce the concept of the cell values and the weighting factors. Cell values are assigned according to how much similarity two strings are. In the experiments on two kinds of proteins, they assign the weight 1 to identical pairs which have a maximum of three corresponding bases codons and weight 0 to those having no possible corresponding base codons. For the pairs having two or one corresponding bases, they change the weight for them in every data set for comparison. On the other hand, they also introduce a penalty weight to cut off the number of operations. In the experiments, they set it as 0, 1.03, 1.05, and 25. When the penalty is larger, the number of operations such as deletion and insertion will be fewer. In conclusion, dynamic programming can solve this problem effectively.

Comments:

After reading through the paper, I remember the past year paper for cs2309. One of the questions is to use the dynamic programming to compute the edit distance of two sentences. In that question, the words are the character in the protein’s string. It is the same idea. Then the java string class method compareTo comes into my mind. It will return a positive value if this string is lexicographically greater than another string and vice versa. I wonder can we change the string compare algorithm with the Needleman and Wunsch’s algorithm. If two strings are not the same, it will return the percentage of similarity of the two strings. If two strings are the same, it will return 1.This method can be used in searching the text where there is no match of the given key word but will have one or many highest similarity percentage words. It can also be made into a java build in method to compare two sentences. To separate the words, we just use the blank space and punctuation marks. To compute the similarity weight percentage, we just call compareTo for every word in the sentence with permutation of the words positions. If we do not want to change the original compareTo method, we can build new methods for it. That will be more powerful for the java language.