Name: Sang Hun Kim
PUID: 0025808690
4/14/2017

Programming Assignment 5

From this assignment, we were asked to find all the possible connections that could exist from the given input binary numbers. There were two approaches that I have tried to solve this problem. One of them was to create structs of my own that contains integer values, integers to be sorted in a fashion that is in ascending order, and another integer value that which position did the input integers fall into the ascendingly ordered integer inside the Graph struct. Another struct I created was an adjacency list struct that has adjacency integers, a notification shown by integer whether that node has been visited, and pointer to the next adjacency node. The way that I was going to work was sort the given input integers in a multidimensional array that each integer in each row tend to appear in which position of the multidimensional array, in which having the very first row of the matrix be the basis for the other lines. This process will had been executed with O(n^2) because it has to go through every single node by rows and columns and sort them so that it is easier to be compared. Then, given with the information of where each given numbers fell, create an another multidimensional array with having adjacency nodes pointing next to each other. In this way, it would have been generated successfully with the time complexity of O(n^3).

Another approach that I have tried was somewhat similar, however, instead of using my own structs, I instead used the integer multidimensional array only. This method first creates two multidimensional arrays, one of them has all the array indexes filled with 1, and the other one being adjacency matrix. The matrix multidimensional array filled with 1s purposefully does its job to locate where the input integers, and adjacency matrix checks whether the nodes are connected to each other or not. This process takes quite a time to figure out which nodes are having adjacent node, which it figures to be O(n^3) just for figuring out.

Both approaches utilize depth first search method to figure out the longest conserved gene sequence, in which its time complexity is O(nlogn). If I were to finish, I would have continued the very first approach because it consumes less time.