This homework is due next Friday, April 13. You should turn in the written part in class and turn in your program by 3:00 PM on that day.

A note about the take-home part of the final exam: We have decided that the take-home part of the final exam will actually be a project that you choose, with my approval. The project can be a research paper or a programming assignment. For one type of research paper, you might use some existing algorithm implementation and report on what it can do and how you used it. Group projects are possible, if they are sufficiently ambitious. You might take inspiration from the large catalog of algorithms in the second part of the textbook. In any case, you should meet with me no later than Friday, April 20, to get approval for your idea. I strongly advise you to meet with me earlier than that! The project itself is due on the last day of classes, Monday, April 30.

1. Given an undirected graph G and a positive integer k, a k-coloring of G means that every vertex of G is assigned one of the integers 1 through k in such a way that for every edge in the graph, the two vertices of the edge are assigned different values. (Think of the integers 1 through k as representing k different colors. We are trying to color the vertices so that no edge connects two vertices that have the same color.)

a) Outline an algorithm for searching for a k-coloring of a graph. The input to the algorithm is a graph G and the integer k. You can use a recursive algorithm that implements an exhaustive search for a solution (with lots of pruning).

b) An exhaustive search might not be feasible for a large graph. Pick a heuristic search technique, either simulated annealing or genetic algorithms, and discuss in general terms how it could be applied to this particular problem.

2. (*Problem 8-7 in the Textbook.*) In the United States, coins have denominations 1, 5, 10, 25, and 50 cents. Now, consider a country whose coins have denominations d_1, d_2, ..., d_k. Let $C(n)$ be the number of distinct ways for coins to add up to the value n. We want to compute $C(n)$. For example, in a country whose denominations are $\{1, 6, 10\}$, $C(5)$ is 1, $C(6)$ through $C(9)$ are 2. $C(10)$ is 3, and $C(12)$ is 4.

a) What is $C(20)$, if the denominations are $\{1, 6, 10\}$? (Show your work.)

b) Give an efficient [dynamic programming] algorithm to compute $C(n)$. The input to the algorithm is the set of denominations (d_1, d_2, ..., d_k) as well as the value of n. (Hint: Think in terms of computing $C(n, j)$, the number of ways that coins of just the first j denominations (d_1, d_2, ..., d_j) can add up to n.)

3. (*From Problem 8-1 in the Textbook.*) The simple edit distance algorithm assigns a cost of 1 to an insertion and to a deletion. But another type of edit is to transpose (or swap) two items that are next to each other in the original sequence. A transposition has a cost of 2 in the original algorithm, a deletion followed by an insertion. However, it can make sense for a transposition to have a cost of 1. Explain how to modify the simple edit distance algorithm to make the cost of a transposition equal to 1 instead of 2.

(*Continued on reverse!*)

4. This is a fairly short programming assignment. Your program will implement the edit distance algorithm and apply it to strings read from a file. Furthermore, the version that you implement should handle transpostions as discussed in the previous problem. That is, insertion, deletion, and transposition should all have cost equal to 1. The program should read lines from a file. The file can be specified as a command line argument, or the program can ask the user to input the path to the file. Each line from the file should be converted to an array of chars (such as by using "`char[] seq = in.nextLine().toCharArray()`"); you can store the char arrays in an `ArrayList<char[]>`. Then, for *each pair* of char arrays, you should output the edit distance for the two arrays.

Your program must be named *DNATest.java*, and it must be submitted to your homework folder in /classes/cs327/homework.

You should run your program on the file */classes/cs327/fake-dna.txt*. This file contains 20 long sequences that could be similar DNA sequences from several organisms. (That is, they are random sequences of A, T, G, and C). There are differences between the sequences that might be due to mutations that occur in the course of evolution. In fact, the data has been manipulated to make it look like the DNA samples come from several groups of organisms, where the organisms in a group are more closely related to each other than to the organisms in the other groups. Can you identify the groups from the edit distance data? **Add a comment to your program saying how many groups there are and which sequences belong to each group, and why.**