Hello, my name is zhirong zhang and In this video I’m going to show you my coursework 3 code explanation.

Firstly, those are my imported library and the instance variables I would use and I’ll cover those contents little by little in the rest of the video.

So now I’m now presenting you the constructor of the class, which use a buffered reader and a while loop to read from the csv file and store each line’s data as a string array, and then add those arrays to the instance variable list. After all that, we close the bufferedreader. And this whole thing is surrounded by a try and catch block where we might get an IOexception. The total running time of the constructor will be O(kn), where n represents the total number of lines from the csv file and k stands for the length of each line.

And then we move on to the method DNA reader. The structure is quite similar to reading from csv. we also use bufferedreader to read from the text and then store the original dna string to the str, and polish the str a little bit by removing all of its whitespaces. The total running time will be O(t+r), where t is the total number of rows of the text and r be the length of the string str.

Then it’s the third method profile checker. if the dna hasn’t been set in the first place, which means the dna string is null, then we throw an illeagalargumentexception. And for the next part I’d like to introduce a model here. Suppose this is our database csv file, we can make a coordinate system about this, from left to the right we have short tandem repeats or its counts, and vertically we have names for different persons, and we take out the data by list.get I j.

And we look at the code here. Let me zoom in a little bit. Ok. So I use two for loops and a while loop to choose the specific person and to count the numbers of the specific short tandem repeats. We have an exceptional case which may cause indexerror And if everything pass the test I designed, then we return the person’s name. and we have running time Oz here where z is the list size we have. And oa where a is the length of string array we stored. Od/s where d is the length of the dna string and s be the length of the short tandem repeats we’re looking for.

Finally, if we’ve gone through all the data and didn’t find the person, we return no match. And the total running time of this method would be o(azd2/s).

For the method Huntington’s disease test we first store all the keys and values regarding to the test results to the hashmap. Hashmap is actually unnecessary and probably unproductive here because those could be replaced by a bunch of if statements followed by return statement, but I used hashmap anyway because this project is small.

And the second part is essentially the same as the counting part in the profile checker we have just mentioned, and I just changed some of the variables. And Eventually, we return the corresponding results. And the running time for the whole method would be od2 where d is the length of dna string.

Thank you for your time and attention, have a good day.