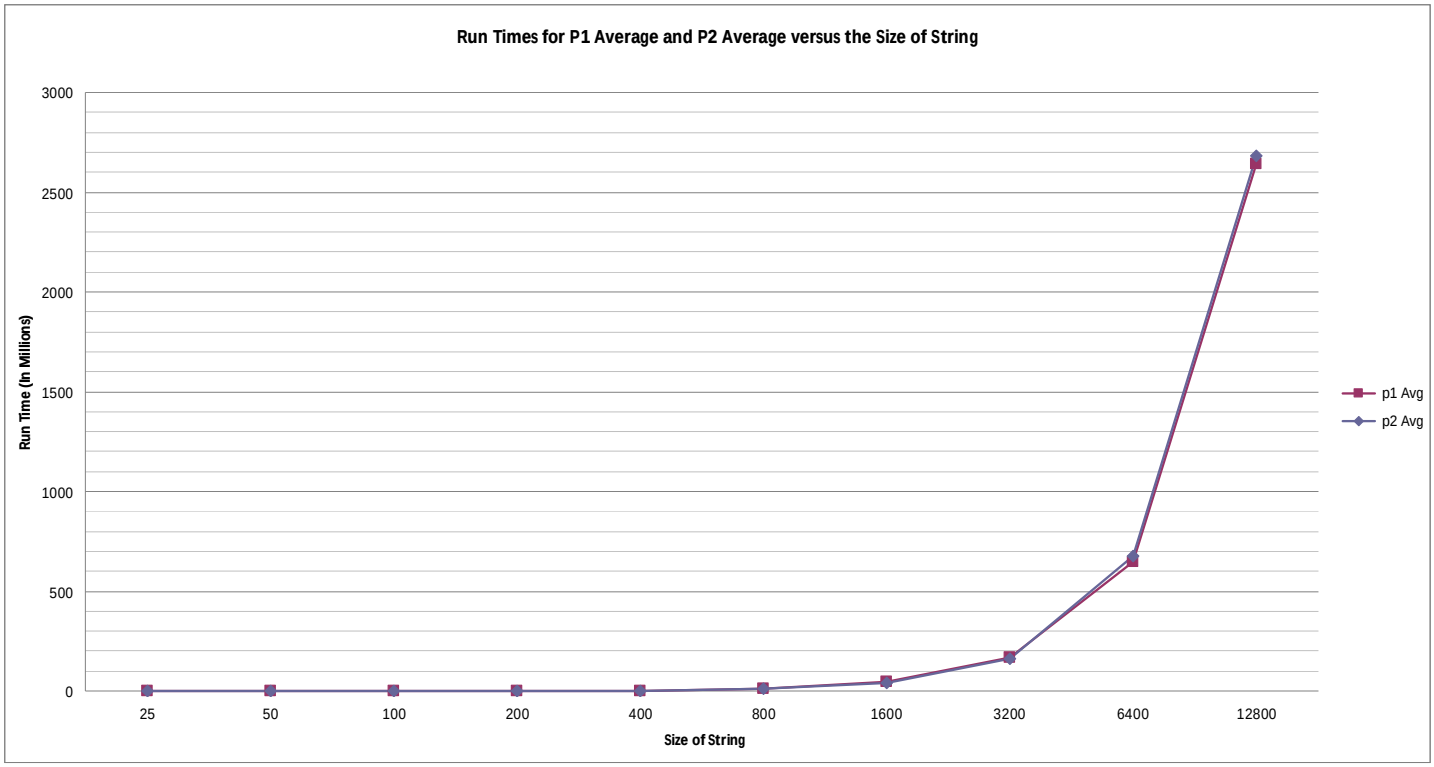
Tyler Holland
Spencer Ellsworth
CPE 349
Lab 6

Longest Common Substring

Here we have a graph showing the average run time of comparing a string to itself (p1avg) versus the size of the string and the average run time of comparing two random strings of equal length (p2avg) versus the size of the strings.
Note: The Run Time is in Millions of Nanoseconds.



Additionally, we have here included the data collected, as the chart does not show much of a noticeable difference, but the difference is clear when viewed in the collected data.
Note: The Run Time is in Nanoseconds.

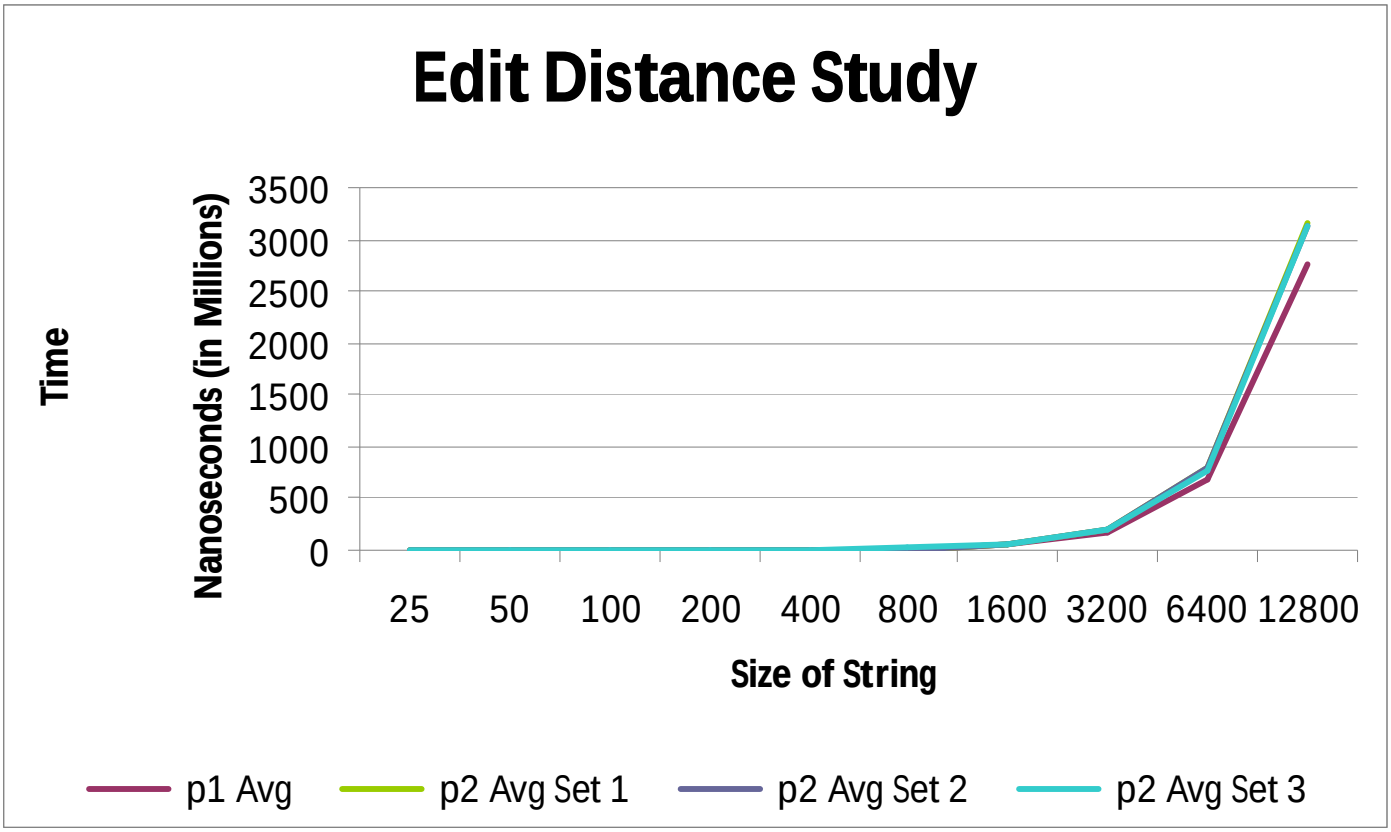
Size	p1 Avg	p2 Avg
25	71192	56127
50	44050	43476
100	172185	169195
200	853013	614225
400	2531567	2634642
800	9620734	12169480
1600	43813391	42692756
3200	167368841	164265317
6400	646083134	675335508
12800	2640330686	2684214908

We very quickly began to run out of stack heap space, so we increased the space (through Eclipse) to 1.5 gigabytes. At 1.5 GB of stack heap space, we were able to go up to strings of length 12800. At this size, we had hit our “serious limitation spot.” (As a side note, this was tested on a computer with an octicore processor and twelve gigabytes of RAM.)

Based on observations we made from the data we collected, it appears that there is very little difference between the length of time necessary to find an LCS of two random strings and the length of time necessary to find an LCS of a string with itself. However, there is a slight difference, and through this difference we observed that it mostly takes less time to find an LCS of two random strings.

Edit Distance

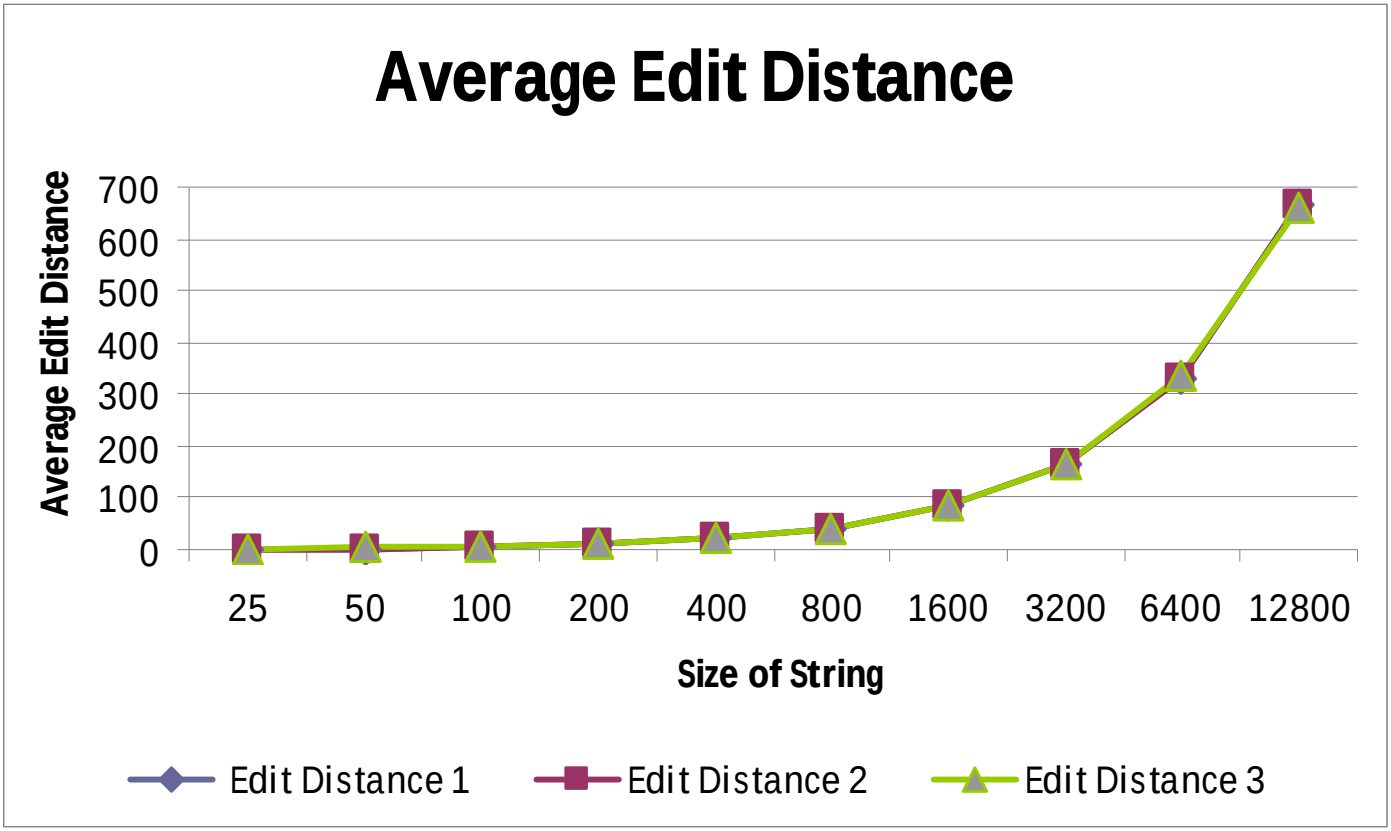
Here we show a graph demonstrating the result of our study for finding the edit distance.



Using strings of size 12800, our implementation took a significant amount of time. We ran this on the same machine as part 1 of this lab.

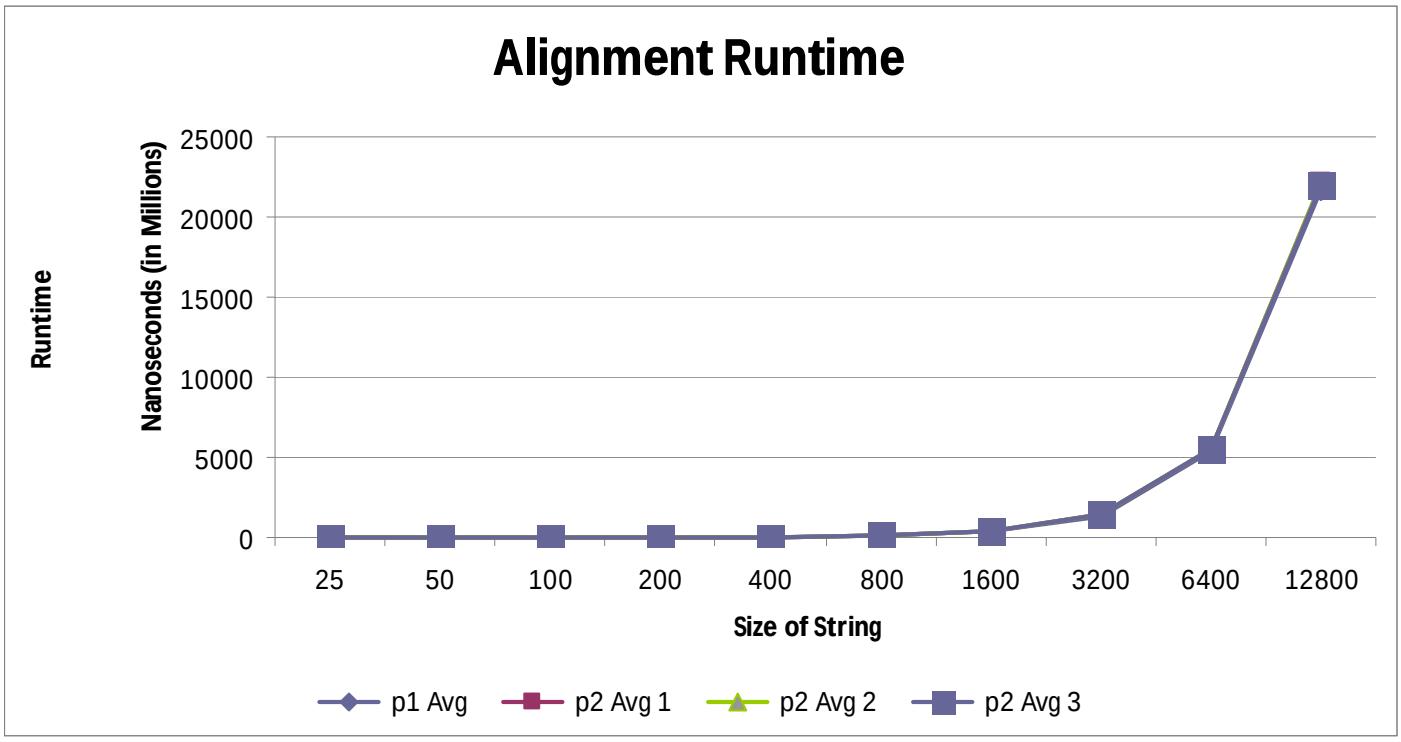
According to the data we collected, it is easier (takes less time) to find the edit distance of a string with itself than the edit distance with another string.

Regarding the behaviour of the average edit distance, it increases as the size of the strings increases. This can be seen in the graph below.



String Alignment

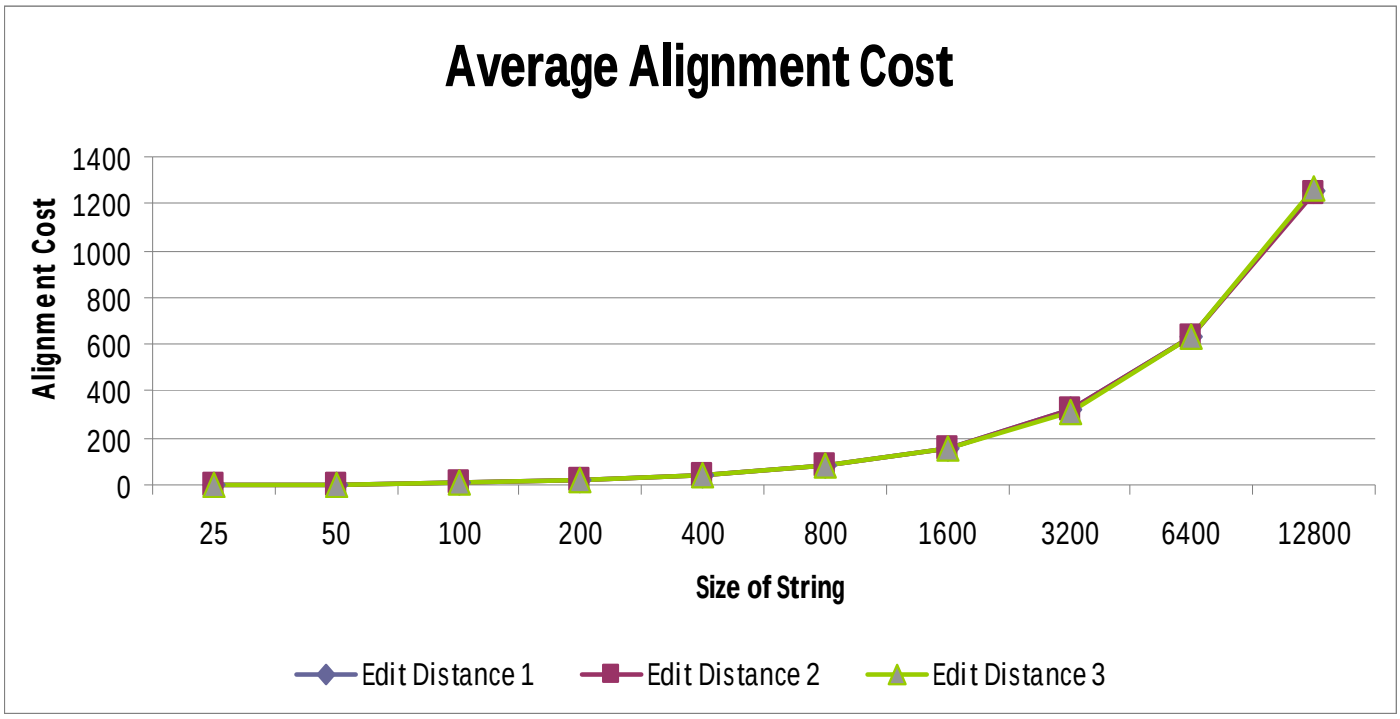
For the string alignment portion of this lab, we collected the following data, as shown in our graph.



With size of string 12800, our implementation took a considerable amount of time. Using the same computer as in the previous parts, it took approximately ten minutes to run our study program.

According to our graphs, there appears to be no difference in time between the lengths of time necessary to find the alignment of a string with itself and the alignment of two random strings.

Here we have a plot showing the behaviour of the average best alignment cost for two random strings as the size of the strings increases.



According to the observations we made while running our code, as the size of the strings increases, the best alignment cost also increases.