Designing Nanostructures with DNA

Connor Uhlman
Computer Science
Simpson College
Indianola, IA 50125
connor.uhlman@my.simpson.edu

Tony Clark
Computer Science
Simpson College
Indianola, IA 50125
tony.clark@my.simpson.edu

Jamie Ethington
Computer Science
Simpson College
Indianola, IA 50125
jamie.ethington@my.simpson.edu

Abstract

The program we are working on is part of a project to design DNA nanostructures. The idea is to form triangular structures, which can then be used to create 3D or 2D shapes. Since DNA strands have significant storage potential, use of these nanostructures could be the next big gain in terms of storage capacity of computers in relation to technology size. In addition to computing applications, DNA nanostructures could be used for the purpose of drug delivery, which would allow doctors to transport medicine to a specific part of a patient's body. Whatever application they may be used for, the application of DNA nanostructures is a subject well worth looking into. For the past 3-4 years, our chemistry department has done research with DNA origami. Several computational tasks have been identified that are currently solved manually.

Our task was to develop a program that could test two sequences of DNA and determine if they matched. If there is a match detected, the program should then change the sequence in a way that the new sequence doesn't match any existing sequences. In order to match, the two sequences need to have corresponding nucleotides along the length of the sequence. Our program's inputs are two sequences of DNA. We then reverse one of the sequences so that they have matching orientation and then change each nucleotide along the sequence to its corresponding nucleotide. Our program then uses a hash table to search for matching sequences of consecutive nucleotides of specified length. When we find matches, we need to change the bases to ensure that they no longer match. The reason we are doing this is for structural integrity. One way to think of this problem is in terms of different construction components. The step of changing matching bases helps ensure that, during the building process, "bolts" and "screws" go in the spots they need to go, while "staples" or "nails" only are used where they are specified.

Our program is able to find matches. Unfortunately, when we test it on large sequences, it runs slowly. Originally, we used a brute force algorithm, but we changed our method and used a hash table to find matches, which was faster. That being said, our program still runs slowly in its current stage. In the future, our program will be combined with algorithms that perform other processes for construction of DNA nanostructures, such as setting the melting temperature of our structures. These additions will complete our program, which will be used by students in our school's chemistry department.

Initial Approaches

The original method that was written to check for matches was a brute force method that used linear searches. It would take as arguments two strings that represented two strands of DNA. It would then use loops to perform a linear search that checked if the two strings had 6 of the same characters in a sequence. For small strings such as our original sequences of twelve characters, this method worked fine and ran relatively fast. When tested with longer strings the run time began to increase by very large amounts. It was decided that this would be problematic while expanding the program, as if matches were found they would need to be changed and the search would need to be executed again to examine if there were still any matches.

Hashing

The decision made to rectify this problem was to execute the search using a hash table. The new method took in the same strings as arguments that the brute force method used. It would take the first string and enter all sequences of six characters into a hash table. It would then check all sequences of six of the second string against the entries in the hash table to check for matches. This proved to be much faster, but still fairly slow when used on very large strings.

Experiments

To compare the two methods, tests were ran using identical strings of increasing length for both methods, and including the output of total runtime. The results proved to be very significant. They were run on lengths of 10, 100, 1,000, 10,000, 100,000, and 1,000,000. The results of our testing are shown here with time in seconds:

	10	100	1000	10000	100000	1000000
Hashing	0	0	0	31	79	1393
Linear	0	0	16	219	21469	2149437

Table 1: Results

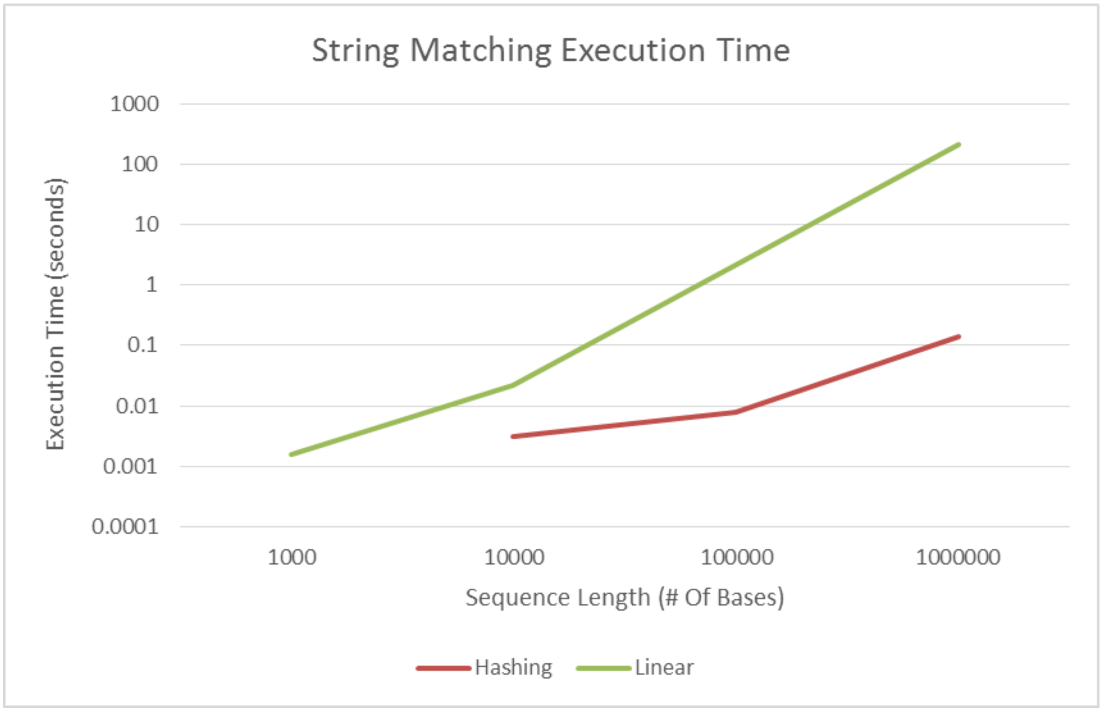

Figure 1: Results

Upon reaching lengths of 1,000,000 the hashing remained just over a second in run time, while increased by such a large amount that for the graph to display the comparison had to have a logarithmic scale for time.

Future Work

In the future this program will be combined with other done by members of Simpson College to change any matching sequences while not changing the melting temperature of the DNA strand. The final program will be used to create a software application that will generate strand to give the user specific shapes.