Jason McGough

Thomas Wynn

Michael Wilder

Waid Wrege

Introduction to Bioinformatics

Project 1: PCR Simulation

Project Report

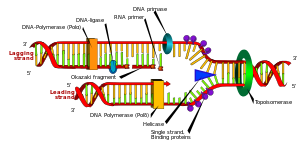
**-Description of PCR (Polymerase Chain Reaction)**

PCR is a biochemical technology technique that is used to rapidly duplicate a segment of DNA a great many number of time (billions, if required). The technique has numerous uses, such as identifying viruses, bacteria, diseases, and even connecting criminals to crime scenes. The whole process takes place in a tube that contains DNA, DNA polymerase, primers and nucleotides.



The process starts by heating the tube of DNA to at least 90 degrees Celsius. This causes the double strands of DNA to separate. Having the DNA in two separate strands is what facilitates the next part of the process to work.

The PCR process relies heavily on primers. Primers are short segments of DNA that are created in a lab in a deliberately specific way. They are designed to match to the segment of DNA that is being copied. Two primers are used. One attaches to the top strand of the segment that is being copied, while the other primer attaches to the opposite end of the bottom strand. Attaching a primer at each end marks the segment of DNA that is desired to be copied. Once the primers are in the correct position, the tube is cooled back down (below 60 degrees Celsius) in order to allow the primers to bind.



Once the primers have bonded, the temperature of the tube is then increased again, (to 72 degrees Celsius). Starting at the segment the primers have marked, the DNA polymerase binds nucleotides from the solution in the tube to the DNA strand. Once this process is done, each individual strand has become its own double strand. At this point, the number of DNA double strands has been doubled.

Due to the fact that each segment of DNA doubles every time a PCR cycle completes, the number of DNA copies grows exponentially as PCR cycles continue. This allows the total number of DNA copies to increase extremely rapidly. Note that after 10 cycles, the number of DNA copies will have reached 1,024. 10 more cycles after that though (20 cycles total) the number of copies will have reached 1,048,576. Once 30 PCR cycles have been completed, the number of copies will be 1,073, 741,824.

Note: Due to the fact that the primers will eventually fall off, the numbers listed above are only theoretical. Using 2^n (with n being the number of cycles) gives you the total number of copies if no primers have fallen off. Though in reality some primers will fall off and the number of copies will be less, the number of copies still explodes very rapidly. This makes PCR an extremely useful and cost effective method for working with DNA.

**-Data Structures/Algorithms**

Our project has been implemented in C++11, mostly due to our familiarity with it and some of the functionality that the latest version offers.

The sequence template for running PCR on is stored in a vector of characters that is read in from an input file. The name of the input file is the first command-line argument for our program. The second and third arguments are the starting and ending positions, respectively. The fourth argument is the number of cycles to run PCR for.

Once the sequence vector is filled, a configuration file is created for use with the primer3 program that we use to select our primers. This file contains the sequence template, the region we are interested in copying, the number of primers we want, and other information required by the primer3 program. Once this configuration file is complete, we run the primer3 program and write its output to another file. We then read through this file until we reach the lines containing the left and right primers and we store them as strings. Since we used the primer3 program to select our primers rather than creating our own, we do not have to check the sequence to make sure the primers are good.

The data representing our DNA strands is kept in a map which uses the length of the strand as the key and a tuple for the value. The tuple contains the counts for strands of the keys length going both directions, stored separately so that we know which way each strand needs to be copied. Depending on where the primers are located on the strand, this could result in a lot of wasted space.

Our PCR simulation algorithm is fairly straight-forward. A copy is made of the map containing the current number of stands for the cycle. The copy is then iterated over starting from the longest strand, and each strand length is checked for the current number of strands of that length. If the current number of strands is zero, or the length of the strand is less than the target gene’s length, it is skipped over. If there are strands of the current length, then we check for polymerase fall-off for each strand. The length of the strand, fail or not, is recorded and the count is updated in the original map.

**-Difficulties**

One difficulty that led to some other ones was the fact that no one in the group has a background in biology. We had to do some research on DNA and PCR before getting started. Even after this research though, we still ran into some misunderstandings that caused us to incorrectly simulate the process at first.

Our original implementation was oversimplified. We didn’t realize this until we looked at our results. The original results had an incorrect distribution. We were getting strands equally distributed across all lengths of segments, rather than focused around the targeted length. Also, after making some changes, we realized that our normal and prime counts were inverted.

One of the last problems we had resulted from us not taking into account that when you copy a prime, you get a normal, and when you copy a normal, you get a prime. We were copying normals from the starting position all the way to the end. Primes were being copied from the end, all the way to the beginning. We had to change the code so that any copies made from the full-length strand got their own special case so that they would copy from the primer to the end of the strand. All of our other strands would copy from the start position to the end position, and visa versa.

Another difficulty we faced was us sometimes trying to resolve issues that didn’t actually exist. We would spend time trying to find out why the simulation wasn’t working accurately, only to then realize that it was actually working correctly.

**-Results**

In essence, our output resembles the kind of results you would expect to see in the gel. In the gel, the segments end up being in order by length. Our output also puts the segments in order by length. In both the gel and our output, you can see that certain areas are concentrated with more segments than others. In our implementation, the main area of concentration is near the target length of the gene.

Anything larger than the target length should have had very few strands produced. Anything less than the target length was considered a failure, and fell within our 5% failure rate. Overall, our results ended up being what we expected.