Go Project Report - PCLT 21/22
\- - -
Transcription and Translation of DNA

Luis Almas, Nº61726
Ricardo Barqueira, Nº53607

1. **Correctness:** The server answers correctly on both the transcription and the translation of the DNA. These processes are selected by the client.

   The server also enables for multiple client requests at the same time.

   If the client selects the transcription process, the server will retrieve the request sent by the client with the DNA molecule. The server will then process the message (change the t's by u's) and send back the mRNA to the client as a response.

   If the client selects the translation process, the server will retrieve the request by the client with the DNA molecule. The server will proceed to do the transcription process (the process mentioned above) and then the mRNA will be split in groups of 3 that will be mapped into an amino acid. The sequence of amino acids will then be validated to form a valid protein. To form a valid protein, the first amino acid must be "Met", and the last codon must be one of the three "UAA", "UAG" or "UGA". Finally, the response will be sent to the client with the resulting amino acid chain and the validation flag (with values true or false).

   The server is also able to interrupt its work in the case of disconnection from the part of the client.

2. **Worker Creation:** Workers are created at various instances in the server. The first worker pool is created for the transcription process. If a translation is required, a second worker pool is created. These workers are created using goroutines in a cycle that runs according to the DNA size.

3. **Worker Management:** First the size of the DNA is divided in 3 (which corresponds to a codon) and for each one of those divisions a worker will be created to transcribe that DNA slice and will also receive an index to preserve order (a DnaSlice struct is created). After a worker is done handling the transcription, the result will be passed to the second pool of workers (with each worker waiting for a signal that transcription was completed) that will have the mRNA slice translated to an amino acid. There is a goroutine that's receiving the worker results as they come and adds them to a slice according to their indexes until the slice is filled. The solution doesn't limit parallelism potential since the number of goroutines created will depend on the request by the client, so many goroutines could be running at the same time processing each codon.

4. **Data Management:** We ensure there are no data races using buffered channels (with buffer size depending on the size of the DNA) and unbuffered (when not related to workers). Each worker will then have a buffered channel that is used to pass the DNA slice (struct DnaSlice) and another channel to write the result. The goroutine that builds the sequence and listening to the result channel will then add them by order into a slice. Finally, that slice will be passed by an unbuffered channel with the resulting sequence.

5. **Fairness and Responsiveness:** The server aims to minimize mean response time by executing the tasks in a parallel manner (using the workers mentioned before) using as the only synchronization point the unbuffered channel that tells when the amino acid sequence was completed.