RNA SEQUENCE ASSEMBLY USING BASE PAIR BOND STRUCTURE

by

Nabil Fannoush

Bachelor of Computer Science, University of Garyounis, 2001

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of

Master of Computer Science

in the Graduate Academic Unit of Computer Science

Supervisor: Patricia Evans, PhD, Computer Science

Examining Board: David Bremner, PhD, Computer Science, Chair
Virendrakumar Bhavsar, PhD, Computer Science
René Malenfant, PhD, Biology

This thesis is accepted by the
Dean of Graduate Studies

THE UNIVERSITY OF NEW BRUNSWICK

July, 2016

ABSTRACT

The process of determining nucleic acid (DNA and RNA) sequences is affected by limitations in technology that limit the length of sequence that can be read at once. Longer sequences are thus read in pieces, and so an important part of the process is to re-assemble these pieces, using overlapping pieces from multiple copies. Sequence assembly is a complex computational problem, particularly if the target sequence is one that is hitherto unknown and so there is no reference that helps to assemble the sequence pieces correctly. The advent of a new generation of high throughput technology that can quickly read shorter sequences makes the assembly an even more critical step because of the increased likelihood of missing regions in the final re-assembled sequence. Current sequence assembly algorithms are designed to seek a best result, from a series of characters perspective, in the form of a shortest possible 'superstring'. However, DNA, RNA and other biological sequences are more than just strings, and so a shortest possible string does not equate to how the pieces were originally ordered, which is ultimately the goal of sequence assembly. This work proposes an alternative approach of including known structure properties and conditions that govern the building and design of biological sequences to help the re-assembly of RNA sequences, and includes tests whose results show noticeable increase in accuracy of the sequence assembly result in comparison to past approaches and that using structure also enables some shorter sequences to be assembled from a single copy.

DEDICATION

To little Seena.

ACKNOWLEDGEMENTS

I would like to thank Dr. Patricia Evans, whose patience, understanding and care made this work possible.

Table of Contents

ABSTRACT ii
DEDICATION iii
ACKNOWLEDGEMENTS iv
Table of Contents v
List of Tables vii
List of Figures viii
CHAPTER ONE: INTRODUCTION 1
1.1. Objectives 1
1.2. Organization of this work 2
CHAPTER TWO: BACKGROUND 4
2.1. Genomic sequencing: DNA & RNA 4
2.2. Sequence assembly 5
2.2.1. Next Generation Sequencing (NGS): 6
2.3. The de Bruijn Graph 9
2.4. De Bruijn graph assembly and Eulerian paths 12
2.5. RNA Structure 13
CHAPTER THREE: THE PROPOSED METHOD 17
3.A. The problem overview 17
3.A.1. Assembly: Optimal vs. Real 17
3.A.2. Computation considerations: 19
3.A.3. Graph k-mer size 20
3.1. Method overview: 22
3.1.1. Reverse complement pair stems: 23

3.1.2. Selection of ‘startup’ pieces stem 26

3.1.3. Creation of k-mers and graph 26

3.2. Fragment assembly 27

3.3. The algorithm 27

3.4. Assembly using a single cover 31

3.4.1. Single cover with short sequences 32

3.4.2. Short single-cover algorithm variant 33

3.4.3. k-mer size selection 33

3.5. Assembly using multiple covers 35

3.5.1. How to use multiple covers 36

3.5.2. Multi-cover algorithm variant 36

3.5.3. *k*-mer size selection 37

CHAPTER FOUR: EXPERIMENTS AND TESTING RESULTS 39
4.1. The testing samples 39

4.2. Testing results 42

CHAPTER FIVE: Conclusions and Future Work 50
5.1. Conclusions 50

5.2. Possible improvements and future work 52

Bibliography 54

Appendix A 56

Curriculum Vitae

List of Tables

Table 1: Scoring of base pair bonds 15

Table 2: Scoring of character pairings [26] 31

Table 3: Tabulations of the results of the respective ***k***-mer sizes for the yeast sample 34

Table 4: Correct fragment groupings for 70S Ribosome-tRNA in *Thermus thermophilus* 38

Table 5: Sample 1 - Yeast initiator tRNA details 40

Table 6: Sample 2 - Zinc finger in *Xenopus laevis* details 40

Table 7: Sample 3 - Group I ribozyme domain 40

Table 8: Sample 4 - *Thermus thermophilus* 41

Table 9: Sample 5 - 70S Ribosome-tRNA Complex in *Thermus thermophilus* 41

Table 10: Sample 6 - Human immunodeficiency virus 1 42

Table 11: Sample 7 - SARS coronavirus 42

Table 12 : Sample 1 assembly results 43

Table 13: Sample 2 assembly results 44

Table 14: Sample 3 assembly results 44

Table 15: Sample 4 assembly results 46

Table 16: Sample 5 assembly results 46

Table 17: Correct fragment groupings in 70S Ribosome-tRNA assembly 48

Table 18: Sample 6 assembly results 48

Table 19: Sample 7 assembly results: 48

Table 20: Correct fragment groupings for HIV sample 49

List of Figures

Fig. 1: Example consensus sequence from pieces overlap 6

Fig. 2: De Bruijn Graph for GGCGAUUCAUCGA..... 11

Fig. 3: Example of basic RNA secondary structure motifs as found in the Leadzyme Ribozyme. Image courtesy of the RNA STRAND project..... 14

Fig. 4: Possible structure for sequence AUACGCGCGUAU 14

Fig. 5: 3-mer and 4-mer De Bruijn Graphs for GCACGCAU..... 22

Fig. 6: Implementation modules for the proposed algorithm 27

CHAPTER ONE: INTRODUCTION

1.1. Objectives

Sequencing of DNA and RNA is the determination of the precise order of nucleotides, the acids or 'bases' that form the actual molecular chain. The demand for computational sequencing of DNA, RNA and other nucleic acids and classifying their functions has grown with the advancement of technology that enables the reading of the sequences, because higher throughput and larger number of shorter pieces requires a greater level of automation [17]. In practice, limitations in current technology impose restriction in the length of a strand of bases being read at once to no more than two hundred a lot of the time [5], and so the strand is instead read in pieces, after which the sub-process of sequence assembly is to reattach the pieces into the original form.

Though the sequencing process is in itself a complex problem, it is exacerbated by the assembly sub-process because this requires multiple iterations of sequencing. Pieces are assembled by aligning where parts of pieces overlap, and to produce such overlaps the strand is sequenced multiple times. In practice, the number of sequencing iterations needed to produce a resulting sequence of acceptable quality can be billions of bases long in total, in some cases 20 copies of the sequence [12].

Computing an accurate assembly of the sequence pieces is a critical issue for sequencing, as by definition the output of sequence assembly should be a representation of the original sequence before the fragmentation, and errors in the assembly will compromise use of the

sequence. A higher number of pieces that are of even shorter lengths increases the likelihood of it being a source of failure.

The potential for incorrect assembly is too high in current implementations, and that is due to what can be considered an incomplete computational representation of DNA, RNA and other nucleic acids as purely a sequence of characters. The optimal result is thus a chain of characters with the minimum possible length, a characterization of the problem that is incomplete and, as the experiments in this work will show, produces wrong results. The goal of this work is therefore to introduce the concept of incorporating actual physical properties of molecules into the computation algorithms, to produce more accurate re-building from the pieces and reduce the amount of sequencing needed. The work will focus on RNAs, as they have useful physical properties.

1.2. Organization of this work

The content is organized as follows:

Chapter 2: Foundational materials and history necessary for understanding the goals of the work and the conclusions are introduced. These include relevant genomic definitions and description of RNA structure, a brief history and description of genomic sequencing and an overview of the sequence assembly problem, how it developed computationally and what difficulties and solutions are associated with it.

Chapter 3: This chapter focuses on how the sequence assembly process in its current state is needs to be improved, and how its association with certain string and sequence computation problems has contributed to the difficulties and pitfalls in resolving and even defining 'correct' assembly of pieces. This chapter also discusses the fundamentals of RNA structure and the basis of the graph-based approach to assembly that is used especially in the case of a higher throughput of shorter reads or pieces. It then introduces the new alternative algorithm for RNA sequence assembly that uses the structural properties of RNA, including variants of the algorithm to be followed in the cases of single and multiple iterations of sequencing. The former of the cases, though it has limitations for practical use, is important as it exemplifies how the expense of additional sequencing for producing overlaps can be greatly reduced.

Chapter 4: An implementation of the proposed algorithm is tested on a set of test RNA samples. The results are then compared with the assembly results of the previous sequence assembly approach, with respect to their respective level of similarity to the original RNA sequences.

Chapter 5: Last but not least, the limitations of the technique and areas for possible future improvement on the approach that arose during research and testing will be highlighted.

CHAPTER TWO: BACKGROUND

2.1. Genomic sequencing: DNA & RNA

DNA sequencing is determining how a DNA molecule's chain of nucleotides (Adenine, Guanine, Cytosine, Thymine) are ordered, as this defines the instructions for genetic inheritance [5]. RNA chains have Uracil instead of Thymine.

Frederick Sanger introduced his method to sequence DNA in 1975, determining sequences in DNA by primed synthesis with DNA polymerase and in the following years, pioneer works for DNA sequencing were introduced, namely Sanger's enzymatic dideoxy DNA sequencing technique based on the chain- terminating dideoxynucleotide analogues and the chemical degradation DNA sequencing technique in which DNA pieces were chemically cleaved at specific bases and separated by gel electrophoresis [12][5]. Two prominent laboratories were responsible for the introduction of the first automated DNA sequencers led by Caltech, which was then commercialized by industrial concerns interested in sequencing. This refinement and commercialization of the sequencing method led to a broad dissemination in the research community [17]. ABI's Prism 3700 was the first automated DNA sequencer to successfully sequence the human genome, after ten years of research and work in the human genome project with an estimated cost of nearly three billion dollars [17]. This was followed by the notable achievement of sequencing the DNA of the first small phage genome and sequencing of the human genome [12].

Regarding RNAs, the genomic sequencing problem was historically not considered as a separate problem in its own right, i.e. there is no RNA sequencing problem, only the sequencing problem when it is applied to an RNA sequence. In this work, we can consider

RNA sequencing as a separate matter of study given that RNAs as structures, not just as sequences, have features that help unlock their functionality, among other properties of interest.

2.2. Sequence assembly

Sequence assembly refers to the process of aligning and merging of genomic sequence pieces to reconstruct the longer original sequence. This process is a necessary step to determine the correct order of nucleotides in large strings in a genome. It is also the subject of focus in this work.

Due to limitations in the current state of technology and genome sequencing factors, the nucleotide sequence representing a genome is not read in at once. It is read in as separate pieces that need to be re-assembled to the original sequence.

Computationally, the sequence assembly problem is broken down into steps that deal with sub- problems to reach the solution [20] [6]:

1. Preprocessing and eliminating noise.
2. Finding the overlaps between the pieces.
3. Creating the consensus region of the pieces, represented by the full set of overlaps.
4. Computing different alignments for the reads in the consensus region.
5. Finally a ‘consensus’ sequence is computed, classically by a simple and direct majority vote. Fig. 1 below shows an example.

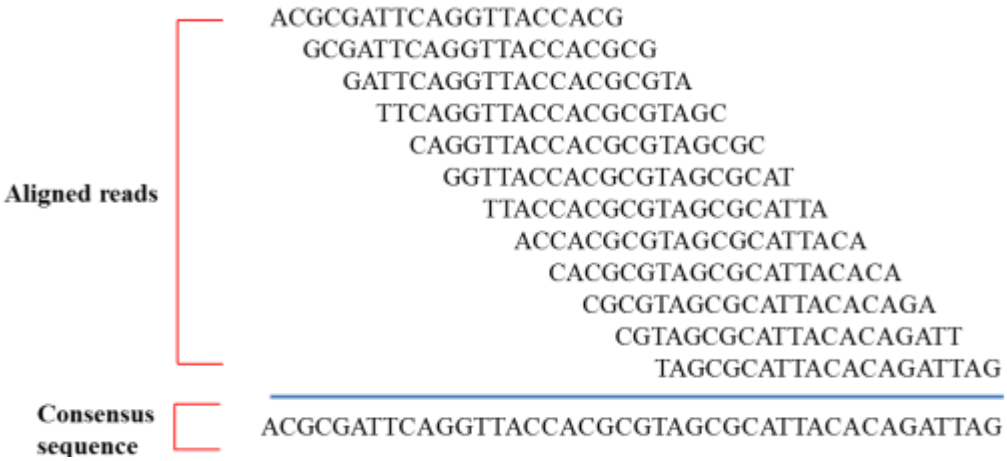

Fig. 1: Example consensus sequence from fragments overlap

2.2.1. Next Generation Sequencing (NGS)

The first Next Generation Sequencing (NGS) platform commercially available on the market, the GS 20, was developed by 454 Life Sciences. The developed technique was validated by combining single-molecule emulsion of the entire 580069-long molecule of the *Mycoplasma genitalia* genome at 96% coverage and 99.96% accuracy. This was followed by further development of NGS, such as Roche's GS FLX titanium [17]. These techniques differ from their predecessors by taking advantage of more recent technology advances and produce higher amounts of reads that are shorter in length (100 – 400 base pairs in length) [24].

This led to progress and further use of two tools: The first is *de novo* assembly using Overlap Layout Consensus [6], which differs from previous reference-based assemblies, because it does not map the reads produced on a reference genome, and is thus not practical in the case of short reads. The second is the application of the ***k***-mer based de Bruijn graph approach as an alternative to all vs. all Overlap Layout Consensus techniques, which do not work

efficiently with shorter reads [12]. A major issue with the assembly process of NGS reads is that because the reads are many, short and so inevitably high in similarity there are more repeats and overlaps that are extremely ambiguous. There is an inverse relationship between a read's length and how much coverage is needed: Sanger-era reads of genomes required a minimum of 3x coverage, whereas the much shorter reads produced by NGS can need 30x, an average of a tenfold higher coverage that needs to be computationally accounted for [6]. In addition, there is also the higher coverage needed to correct the base errors in the reads to be assembled, which can reach 1000% or even higher than previously in single or long reads[7].

In practice, even with extensive coverage there will still be identical subsequences, and this is especially true in the case of shorter reads. Because in this case identical subsequences cannot be resolved without specially paired reads, creating long sequence overlaps became very computationally difficult [6]. Therefore even though a high throughput of short read datasets became more practical, their assembly with prior techniques and whole genome assembly became very impractical.

The sequence assembly problem has long been a crucial issue in the context of the sequencing task, as how the problem is tackled affects how the task is conducted. Tackling the problem has been influenced by how it is - at least in broad terms - analogous to the more general level of string alignment and assembly, and so algorithms used on this level were a foundation for also solving sequence assembly in bioinformatics.

If we describe a genome sequence assembly task as follows:

Finding a target sequence Q over an alphabet {A,C,G,T}, from a given set of n pieces of reads S = {f1, f2, f3 .. fn} over an alphabet {A,C,G,T}, Such that Q= Shortest possible string where {f1, f2, f3 .. fn} have minimal mismatches to substrings of Q.

A problem arises that we cannot know that Q above is 'correct', or even what 'correct' means. Most assembly attempts are for finding unknown sequences, so there are no clear criteria. Also, even if the goal is to match to a pre-existing sequence, the fact that one is dealing with long lengths of a very short alphabet gives a risk of having irrelevant matches, and there is also the risk of heavy bias to the criterion being matched against. Such requirements and their targeting sets genome sequence assembly apart from other alignment and assembly problems.

A more fundamental issue presents itself: The above definition is basically that of the Shortest Common Superstring [9], i.e. the problem of finding the shortest common denominator superstring in a set of strings, thus identifying the sequence assembly problem as one of sequence compactness, i.e. the goal is to find the shortest possible string. The question that needs to be asked is: Are the two problems actually the same? Or more succinctly, is the goal indeed the shortest possible sequence? After all, the sequences to be assembled are not meant to be strings in themselves, but a representation of cellular entities subject to the laws of physics, chemistry and nature; and so the goal surely should be an assembly result that closest reflects the entity's natural state rather than the most compact string possible. It is on this premise that this work focuses on to provide an alternative solution to the sequence assembly problem.

Below are difficulties particular to sequence assembly in bioinformatics:

1. Because of the random sampling in the fragmenting process, some parts of the target

sequence may not be adequately represented or covered, and at least theoretically there is a calculable probability ratio of zero (e.g. not being covered at all) [19].

2. The computational complexity of assemblage is increased because a string read may represent a strand sequence in the genome and can also be its complementing sequence counterpart on the opposing strand, and so the number of reads is doubled [19].
3. Computational expense also includes time complexity in dealing with large sequences: When searching for overlaps in sequence string reads of lengths x and y respectively in greedy assembly methods, it would take the quadratic running time of $O(xy)$[15].
4. Repeated sequence strings in different places are naturally common, and especially in longer reads, and it can be difficult not to assemble them by mistake [14].
5. A variant of the above point is that, because of the phenomenon of polymorphism and mutation in a genome [6], highly similar repeats that represent actual different versions or features are hard to identify, especially when the computation in the algorithm allows for a margin of read error [14].

2.3. The de Bruijn Graph

The de Bruijn graph is one of two graph based path/edge approaches introduced to the assembly problem [18], particularly with the advent of short length/high throughput Next Generation Sequencing, the other being Overlap Layout Consensus graph used mostly in greedy methods focusing on the layout phase [13]. The term *de Bruijn graph* in this context refers to a special case of de Bruijn graphs, namely the fixed subsequence length $\boldsymbol{k}$-mer graph [18].

De Bruijn graphs deal with the complication of assembling a flood of less meaningful short

reads and their scaffolding computation by limiting the length ***k*** of sequences represented by the nodes, i.e. ***k***-mers, instead of having to resolve and process whole explicit overlaps. However, this resulted in creating a new problem in dealing with the assembly, that information would get lost if the repeat's length is greater than ***k***. This in turn led to the problem of calculating the appropriate trade-off between computation cost when using shorter ***k*** lengths vs. the costs of runtime memory and time complexity when using longer ***k*** lengths [15].

In graph theory, a De Bruijn graph is a graph representing overlaps between sequences of symbols. Its dimensions are determined by the number ***x*** of symbols used and a fixed length ***k*** of the sequences that build the graph. These symbol sequences are called ***k***-mers [15].

It therefore can have up to x^{k} edges, consisting of all possible length ***k*** sequences of the given symbols or ***k***-mers; the same symbol may appear multiple times in a sequence. Each edge connects two vertices representing the ***k***-mer's two length (***k***-1) substrings.

An edge can be propagated to a second edge by shifting all its symbols by one place to the left and adding a new symbol at the end of this second edge, thus the two share a connecting vertex. Existing vertices are used to connect other edges, and so the process continues until all ***k***-mers are present in the graph [2].

If we take as an example the sequence S = GGCGAUUCAUCGA, all possible ***k***-mers of size 3 are calculated as

GGC
GCG
CGA

GAU

AUU

UUC

UCA

CAU

AUC

UCG

For k-mer (3-mer) or edge GGC, its left vertex (k-1-mer = 2-mer) is GG, and its right vertex is GC. Applying this to the other k-mers, we get the following vertices:

GG,GC,CG,GA,AU,UU,UC,CA

From the above, the final De Bruijn graph can be constructed as seen in Fig. 2:

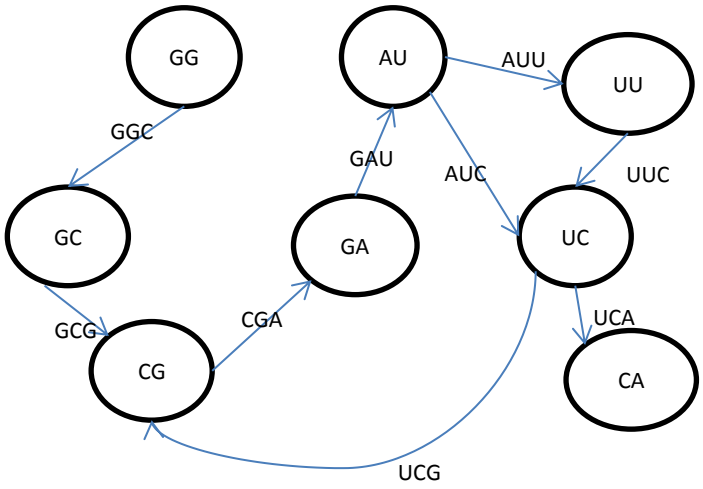

Fig. 2: De Bruijn Graph for GGCGAUUCAUCGA

2.4. De Bruijn graph assembly and Eulerian paths

Once a graph is built, the question is how to use it to find the re-assembled sequence. Not all RNA sequences are small, and consequently any graph made by its pieces would not be as simple as the illustrative example in the previous section. The complex problem of finding the assembly path in the graph is known as the Eulerian path or Eulerian walk problem [18]. In the context of a de Bruijn graph representing a genomic sequence, an Eulerian path is a path that follows all edges in a graph exactly once, and whose corresponding ***k***-mers formed the final assembled sequence.

One of the most widely used algorithms for determining the Eulerian path in a graph, used for evaluation in this work, is Fleury's algorithm [2], the description and definition of which is as follows:

Let *G* be a connected graph.

Let a *bridge* be an edge which, if removed, produces a disconnected graph.

1. Choose any vertex *v* of *G* and set current vertex equal to *v* and current path equal to the empty sequence of edges.
2. Select any edge *e* incident with the current vertex but choosing a bridge only if there is no alternative.
3. Add *e* to the current path and set the current vertex equal to the vertex at the other end of *e*. *(If e is a loop, the current vertex will not move).*
4. Delete *e* from the graph. Delete any isolated vertices. Repeat steps 2 – 4 until all edges have been deleted from *G*. The final current path is an Eulerian path in *G*.

Ultimately, algorithms using the de Bruijn graph seek an 'optimal' solution from a set of paths (***k***-mers); and even if an *optimal* solution (i.e. a sequence S ∈ {A,C,G,U} resulting from the best path of ***k***-mers) is the same as the *real* solution (i.e. the pieces reassembled to their original state) - and that is not necessarily true - there is also the fact that repeated regions can be lost because de Bruijn graphs do not maintain that information (e.g. a single ***k***-mer might represent multiple occurrences of the substring, but an Eulerian path would cross that ***k***-mer only once).

2.5. RNA Structure

RNA molecules are formed by a single strand of nucleotides that folds in on itself, as opposed to the double helix that forms the basic structure of DNA. The nucleotides then bond and pair, forming the physical features and shapes (e.g. bulges, hairpin, loops, pseudoknots) of individual RNAs are formed, as illustrated in Fig. 3. The bonds in the secondary structure of RNA (primary structure is the RNA sequence) are formed by G-C, A-U and G-U bonds, in that order of strength; the greater the overall strength of the bonds from the pairings in a structure, the more *stable* that structure and genome is [26].

A key feature used in this work is the reverse complement stack or duplex, forming a stem. It is the foundation on which the RNA structure exists, is its most stable component, and pieces with bases that have a high degree of pairing between them are more likely in nature to form a duplex. In fact, the simplest RNA structures are for the most part a single reverse complement stack or duplex. This unique property helps determine the placing of pieces in the sequence because pieces that form a duplex would have to be on opposite sides of the RNA fold [26][1].

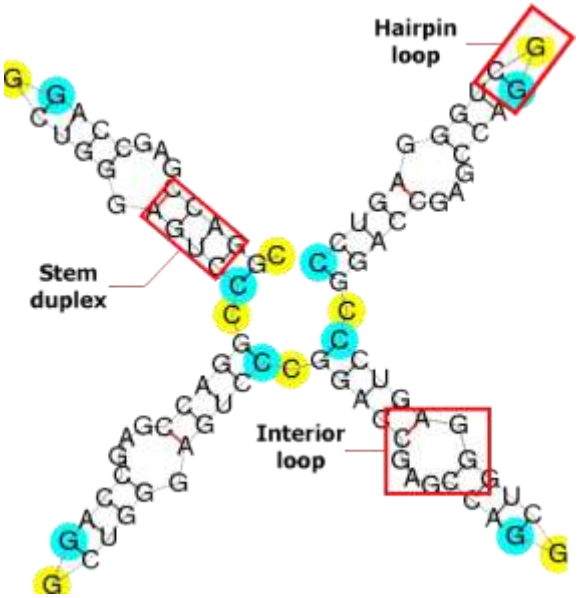

Fig. 3: Example of basic RNA secondary structure motifs as found in the Leadzyme Ribozyme. Image courtesy of the RNA STRAND project [27]

Stability is a critical factor in this work, and the reason is that it is tied to the likelihood of existence: the more stable the construct is, the more likely that it is the reality. For example, for the sequence AUACGCGCGUAU, Fig. 4 shows two examples of possible structures it can form. Even though both are possible, (b) would be considered the most likely to reflect its physical form because (a) is too unstable a structure.

(a) **(b)**

Fig. 4: Possible structure for sequence AUACGCGCGUAU

In this example, the sub-sequences AUACG and CGUAU are referred to as reverse complements of one another, and (b) shows how such reverse complements are at the basic building blocks of RNA structure. It is logical to view motifs and complex structures as a result of how the various subsequences form reverse complement stacks and the position of 'free' nucleotides in the two subsequences forming the stack; the proposed method uses this in assembly by finding the best possible reverse complements among the pieces to be assembled. An important factor in finding or selecting reverse complements is the type of bonds involved, and scoring of reverse complement candidates is therefore calculated by how many of each of these bonds form the stack. The bonds in this work are scored in the following table based on prior authoritative work that addresses the scoring of bonds [26] according to the amount of energy released by folding the RNA molecule, or free energy, and the lower the amount is the more stable is the corresponding bond. C-G pairs have the least minimum free energy, consisting of three hydrogen bonds as opposed to the two of U-A pairing, making it stronger and more stable. G-U is given the lowest score value because they are the weakest and least likely to occur.

Table 1: Scoring of base pair bonds

pair	score
C-G	8.4
U-A	8.0
G-U	1.0

The properties governing how RNA are formed are useful because they can help simplify the sequence assembly problem by reducing the number of pieces to be assembled and the number of possible assembly combinations, which would not be possible when considering the pieces as merely character sequences. Additionally, the increase in information that considering

structure provides means that there is less need for repeat sequencing to generate string overlaps used to connect the pieces together. The next chapter will show how these properties are used to construct the proposed assembly algorithm.

CHAPTER THREE: THE PROPOSED METHOD

3.1. Problem overview

3.1.1. Assembly: Optimal vs. Real

RNA is more than a string of characters. It is a construct whose existence is subject to laws of physics and forces of nature, and it is therefore reasonable to consider such factors when attempting to assemble this construct from the pieces represented by sequenced pieces. Ultimately the goal is to have a true representation –or as close to true as possible- of the genome so that features and patterns can be found, rather than an optimized one. In this work the intent is to modify the classic approach to sequence assembly so that it takes advantage of physical structure to help find that true representation.

As alluded to earlier, a key issue that affected the search for the solution is modeling the sequence assembly problem with more abstract computational problems, namely the Shortest Common Superstring problem, particularly when an assembly is basically considered as an overlap of a joint substring when stacking read alignments [8]. While the definition described in the previous section can be roughly used interchangeably for both problems, difficulties emerge when applied in sequence assembly, mainly as a result of the problem's scope.

This class of assembly generally follows a whole-genome approach: Pieces and fixed-size reads are combined to produce a vertical alignment of pieces by overlapping consensus regions, which are then 'scaffolded' by creating read-pair mates (i.e. a splitting of the sequence segment into two at both its ends and a separating gap of fixed length between them). After that, the scaffolds are mapped to the genome by sequence tagged site (STS) [20] reference sequence approximately 200-400 base pairs [23] in length of a known base and

location that occurs only once [12].

Dealing with repetitive sequences complicates sequence assembly because different sequence pieces can share the same repeat sequence originating from different genomic locations. The pieces are assembled by searching for overlap matches, and therefore repeats can be put together erroneously. In this class of assembly methods there is an inherent problem of higher error rates, which are caused by

- Overrepresented repeat regions, which have a direct relationship with the length of the read. A sequence represented $\boldsymbol{n}$ times would require $\boldsymbol{n^2}$ comparisons.
- Erroneous sequencing because the method identifies a base or bases incorrectly. Additional coverage (i.e. the average number of reads representing a given base in the reassembled sequence) to deal with sequencing errors would add to computation costs.
- The existence of polymorphisms (i.e. mutations) adds the complication of identifying the polymorphisms as errors rather than a natural variation.

It is therefore beneficial to include structure considerations in the assembly process: it would provide a context that lessens the ambiguity of repeat regions and differentiating between them; and also allow for a limited margin of error in sequencing or polymorphisms when considering the RNA from the perspective of its structure as a whole.

From the start, computational solution efforts required a sufficiently large amount of high quality sequences and the ability to handle large data sets. Also, they needed to avoid 'incorrect' assemblies due to the presence of large repetitive regions and redundant sequences. This led to the inevitable need for algorithms that handle large input data sets with

as small a use of computer time and memory as possible.

3.1.2. Computation considerations

On the outset, genome assembly gave a comparatively low throughput of long reads and this made finding the shortest common superstring a very costly process and hindered the sequencing of more complex genomes – such as mammalian genomes – for a long time, even though the advances of Sanger and what was referred to as Second Generation Sequencing have helped lower the cost at least in time complexity [6]. Problems arise from using the Shortest Common Superstring approach upon which earlier assembly approaches were based (ergo, classifying the sequence assembly problem as NP-hard), and from complications in processing repeats because aggregate repeats are folded or collapsed when building the superstring. These issues prompted research efforts to create more efficient approximation algorithms [10].

Computation of overlaps is generally a major cost in assembly. When we consider the case of whole genome sequencing with longer reads (e.g. over 100 bases, though in some Next Generation Sequencing technologies used reads of up to 500 bases are considered short length [24]) this cost becomes even more prominent, particularly when we consider the computation time in the worst case scenario, where each read must be compared to all other reads (i.e. $\binom{n}{2}$ calculations where *n* is the number of total reads) [19].

Coverage is also a factor in the level of computation. In this context, coverage is the average number of reads of the target sequence in the re-assembled pieces. Or:

Given that

The total length of assembled pieces is *L*,

The total number of reads is *R*,

The average length of reads is *A*,

Coverage $C = R*A/L$.

Coverage affects complexity of computation and quality of results. For example, if a genome has a low coverage rate, then the large number of gaps makes the accuracy of the resulting assembled sequence uncertain. On the other hand, even if it means better accuracy of the assembly of pieces, if a genome has a high rate of coverage – unless the absence of errors is guaranteed, another potential complexity in computation – it would lead to the possibility of repeated sequences that are actually distinct but 'covered' as the same sequence [25].

3.1.3. Graph *k*-mer size

In graph-based assembly solutions, finding the suitable size of length ***k*** of sequences represented by the nodes is a complex problem in itself. Since the alphabet to form the sequences is quite small {A, U, G, C}, the shorter a sequence is the less meaningful it would be in context of sequences of length of whole RNAs: in an RNA of length 200, the sequence ACG is not meaningful as it has a high likelihood of occurring multiple times. Smaller ***k*** sizes also affect the amount of coverage needed, particularly with the advent of NGS and the introduction of de Bruijn graphs, because as discussed it is the nature of graph-based assembly to absorb equal ***k***-mers into one edge, leading to loss of information and repeat areas in an RNA sequence; therefore coverage is increased in an attempt to ensure that as much of the RNA is represented as possible.

On the other hand, technology limitations affect how big a ***k*** can be used. In NGS based

sequencing tools, the maximum read length, therefore maximum ***k***, is roughly 500 nucleotides long [23]. While ***k***-mers of that length, or indeed of much lesser lengths such as 100 or even 50, are by no means unimportant, in theory at least the chances of their re-occurrence are still higher when considering complex and large genomic entities. Some RNAs can be tens of thousands of nucleotides long, and unrepresented repeat areas are by consequence more substantial and meaningful.

The above illustrates why ***k***-mer size selection is a parameter that is difficult to threshold appropriately. If too small, the low cost of computing the graph would be offset or even nullified by the need of including higher coverage; while if too large, graph computation cost would be higher and might even be impractical. Therefore, instead of one standard measure for ***k***, there should be multiple options as to how to find ***k*** depending on size and complexity of the RNA in question.

The example illustrated in Fig. 5 shows how size of ***k*** affects repeat regions. Given the sequence GCACGCAU, with size ***k***=3 the generated 3-mers are GCA, CAC, ACG, CGC, GCA and CAU; when building the corresponding de Bruijn graph, only one edge representing GCA will be in the graph and therefore the assembly result from the graph would be missing one of the two GCA regions. With the more suitable size ***k***=4 though, there is sufficient distinction to include all regions of the sequence in the graph.

Due to this impact, consideration is given in the scope of this work for a mechanism to select an adequate size k as part of the sequence assembly problem as a whole.

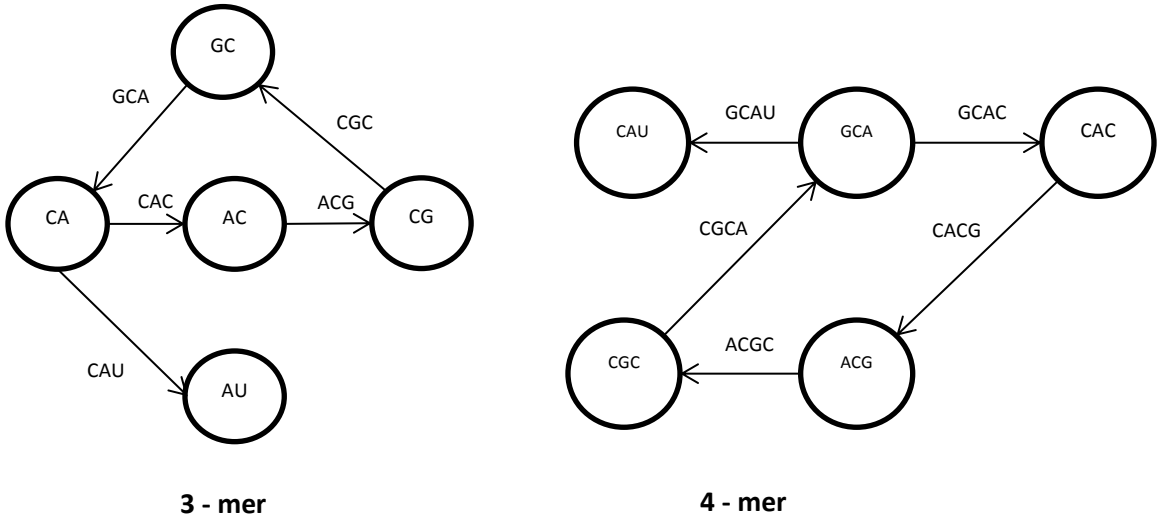

Fig. 5: 3-mer and 4-mer De Bruijn Graphs for GCACGCAU

3.2. Method overview:

The new approach proposed in this work to solve the sequence assembly problem does so in two ways. Firstly, it uses the structure of RNA to determine correct assembly; secondly it breaks the problem down into steps that incorporate structure considerations into the assembly process. These steps consist of finding reverse complement pairings between pieces, scoring the pairings, selecting the 'startup' pairing from which assembly expands, creating the graph and then following the path in the graph to create the assembled sequence using the pairings as a guide. The consideration of structure led to modifications to prior concepts used previously in sequence assembly that are used also in this approach, such as the de Bruijn graph algorithm, which will be highlighted as the steps are described in detail.

3.2.1. Reverse complement pair stems:

The initial step in the proposed assembly process is to find how the given pieces can pair with each other, effectively forming stems. The rationale behind finding these stems is that they are stabilizing factors in RNA structures, and a stable structure is one that is more likely to exist. Stability is measured by a minimum score total of base pair bonds between two pieces $S(\alpha, \beta)$ where α and β are the two bases, with G-C bonds given a score of -8.4, A-U a score of -8.0 and G-U a score of -1. Accordingly, the goal is then to find the fragment reverse complement pairings that give the lowest possible scores.

This goal is reached by automating the process of comparing the fragment set against the reverse complement set using dynamic programming algorithms similar to sequence alignment algorithms, such as the Needleman-Wunsch and Smith-Waterman algorithms [16][21], briefly described below.

Given character sequence x of length n, character sequence y of length m, and a gap penalty d for gaps in the two sequences:

1. Construct the alignment matrix F of size $(n \cdot m)$, where F[i, j] contains the best alignment score for matching x[1..i] to y[1..j]

    1.1. for i = 0 to n

    $$F[i, 0] = d \cdot i$$

    1.2. for j=0 to m

    $$F[0, j] = d \cdot j$$

1.3. for i=1 to length of x

for j=1 to length of y

Match = F[i-1,j-1] + S(x[i], y[j])

Delete = F[i-1, j] + d

Insert = F[i, j-1] + d

F[i, j] = max(Match, Insert, Delete)

2. Trace back and generating final global alignment

2.1. Alignment_x = " ", i = length of x

2.2. Alignment_y = " ", j = length of y

2.3. while (i > 0 and j > 0)

Score = F[i, j]

ScoreDiag = F[i - 1, j – 1]

ScoreUp = F[i, j – 1]

ScoreLeft = F[i - 1, j]

if (Score = ScoreDiag + S(x[i], y[j])) then

Alignment_x = x[i] + Alignment_x

Alignment_y = y[j] + Alignment_y

Decrease i by 1

Decrease j by 1

else if (Score = ScoreLeft + d)

Alignment_x = x[i] + Alignment_x

Alignment_y = "-" + Alignment_y

Decrease i by 1

otherwise (Score = ScoreUp + d)

Alignment_x = "-" + Alignment_x

Alignment_y = y[j] + Alignment_y

Decrease j by 1

while (i > 0)

Alignment_x = x[i] + Alignment_x

Alignment_y = "-" + Alignment_y

Decrease i by 1

while (j > 0)

Alignment_x = "-" + Alignment_x

Alignment_y = y[j] + Alignment_y

Decrease j by 1

In step 2.3 above, the traceback step adds the next character in the output string describing the final global alignment, and the character "-" indicating where a gap in the alignment occurred.

The difference between the proposed computation and the alignment algorithms is that the scoring is not based on whether or not two nucleotides in the two pieces' sequences match, but whether they are a G-C, A-U or G-U, with the corresponding weights used.

3.2.2. Selection of 'startup' pieces stem

Once the reverse complement pairings between pieces have been computed, the next step is to select the initial starting point of the assembling the fragment sequences, i.e. the reverse complement fragment pair that includes the starting fragment. The criteria for the selection can be described as follows:

- Find the highest scoring pair that is also completely paired on one end of the pair stack (one end has no free nucleotides).
- If no such pair is found, choose highest pair score with the least number of free nucleotides at either end of the pair stack.

The importance of this step is that it helps define a starting point in the graph that is to be created to assemble the pieces.

3.2.3. Creation of *k*-mers and graph

In this work, the calculation of k-mer size is calculated as follows:

Given that RNA length N = $\sum$ **all fragment lengths**

If $N \leq 1000$ then $\boldsymbol{k} = \lceil \mathbf{N} \cdot \mathbf{P} \rceil$

where **P =** 0.23, as determined by experimentation (see section 3.5.3)

If $N > 1000$ then $\boldsymbol{k} = 250$

A de Bruijn graph is then constructed from these ***k***-mers, but with a variation: equal ***k***-mers are not merged into one, but kept separate as a possible representation of an actual repeated region. In other words, the resulting graph might contain some edges with identical labels.

3.3. Fragment assembly

In the generated graph, an edge or ***k***-mer that is equal to the 1..***k*** substring of the selected 'startup' fragment is sought. If more than one is found, the edge that best completes the 'startup' fragment is selected; if no such edge is found, then the first one is selected by default. From there, the path is followed, adding every ***k***-mer or edge crossed to the output, until every fragment string is included in the path.

3.4. The algorithm

The algorithm accepts a list of strings representing pieces and outputs a string sequence representing the sequence assembly result, implemented according to the modules described in Fig.6.

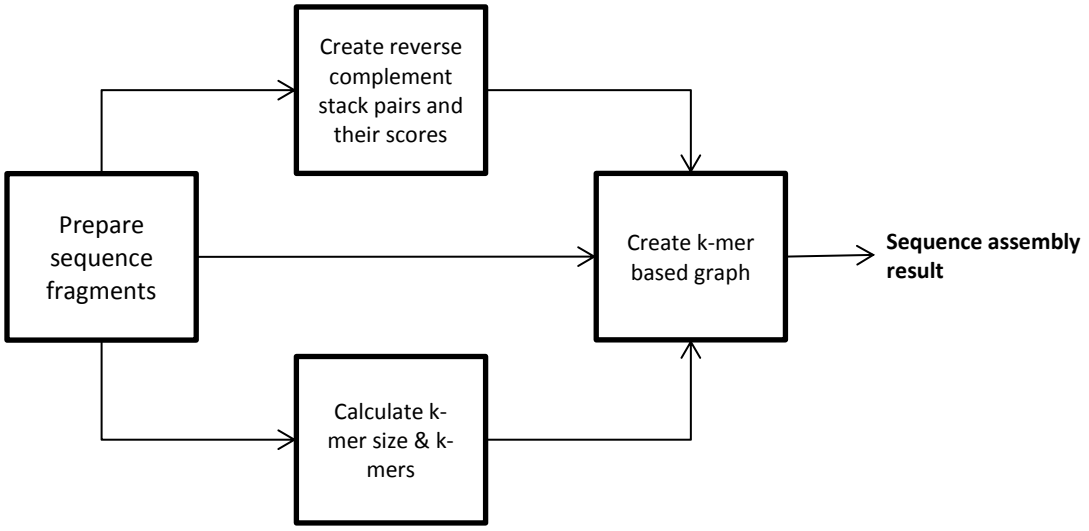

Fig. 6: Implementation modules for the proposed algorithm

Below is the main algorithm, detailing the description provided in Section 3.2, which calculates the score of possible fragment stem pairings, creating the graph and returning the result of traversing it.

Main algorithm:

1. Get score for reverse complement pairs

   INPUT: matrix containing sequence pieces to assemble

   OUTPUT: score matrix

   1.1. For all possible pairs of fragment pairs in the fragment matrix

      1.1.1. Initialize similarity matrix simil[n, m] :

      All first row cells of simil[ ] = 0

      All first column cells of simil[ ] = 0

      1.1.2. Fill the rest of similarity matrix simil[ ] and keep track of maximum score maxval from pairwise alignment of two pieces (string1, string2) using given weights for (G,C), (A,U), and (G,U) pairs, with other pairs scored as 0.0:

      for i=1 to length of string1 do

      for j=1 to length of string2 do

      If string1 [i-1] and string2 [j-1]

      match any combination in table1 below,

      score = corresponding value from table 1.

      else score=0

      simil[i, j] = max(simil[i-1, j]-1,

      simil[I, j-1]-1 and simil[i-1][j-1]+(-1 · score))

      if simil[i, j] > maxval then maxval = simil[i ,j]

      RETURN maxval

1.1.3. Store returned *maxval* value in corresponding cell in score matrix.

1.2. OUTPUT: score matrix

2. Get best candidate fragment pair for start of sequence

INPUT: Fragment matrix, reverse complements matrix, score matrix;

OUTPUT: Best candidate fragment pair

INIT: Best pair = null, high score = -1;

2.1. For every pair and its score:

If current pair score is greater than best pair high score AND pair is attached at either tip then best pair = current pair

If Best pair still = null then

If current pair score is greater than best pair high score AND pair is attached at either tip then

best pair = current pair

OUTPUT: best pair

3. Create modified de Bruijn graph

INPUT: pieces, total length, best pair;

OUTPUT: directed graph (kmers [ ], kmers entry vertices [ ], kmers exit vertices[ ])

3.1. Initialize arrays kmers [ ], kmers entry vertices [ ], kmers exit vertices[ ]

3.2. Get kmer length:

If total length is greater than or equal to 1000 then

kmer length = round (total length · 0.23)

Else kmer length = **15**

3.3. For all pieces where the best pair are considered first and last

3.3.1. Divide each fragment into kmers that are less than or equal to kmer length

3.3.2. If kmers[] contains new kmer at kmers[j] then

If new kmer cover is equal to kmers[j] cover then

Add new kmer to kmers[ ]

Else ignore new kmer

Else Add new kmer to kmers[ ]

3.3.3. If kmers entry vertices [ ] does NOT contain new kmer

then add new kmer's substring (0..length-1) to kmers entry vertices[ ].

3.3.4. If kmers exit vertices [ ] does NOT contain new kmer

then add new kmer's substring (1..length) to kmers exit vertices[ ].

3.4. RETURN graph ([ ]kmers, [ ] kmers Left vertices, [ ] kmers right vertices))

4. Find sequence

INPUT: Pieces[], graph, best pair ;

OUTPUT: sequenceResult

4.1. Start at first ***k***-mer whose sequence best matches the sequence of the start fragment of the best pair, follow path in graph using vertices.

4.2. sequenceResult = sequenceResult + first ***k***-mer

4.2.1. If more than one outgoing ***k***-mer found, pick one such that ***k***-mer's first character best concatenates to an actual fragment from which ***k***-mers are derived.

4.2.2. If no optimal outgoing ***k***-mer found, take first one.

4.2.3. sequenceResult = sequenceResult + selected ***k***-mer.

4.2.4. If sequenceResult contains a fragment then remove from Pieces[ ]

4.2.5. If Pieces[ ] = empty then stop

Else go to 4.2.1

1st Character	2nd Character	Score
G	C	-8.4
C	G	-8.4
A	U	-8.0
U	A	-8.0
G	U	-1.0
U	G	-1.0

Table 2: Scoring of character pairings. These values were selected based on previous work [26]

3.5. Assembly using a single cover

Coverage is a crucial issue when it comes to sequence assembly because of its cost, particularly when sequencing larger genomes and additional covers are needed for increased accuracy. So this work explores the possibility of using the structural properties to compensate for coverage. Moreover, it explores the possibility of using a single cover, meaning a single sequencing iteration, to assemble pieces.

The method described applies de Bruijn graphs with a variation. De Bruijn graphs are typically compressed by only inserting exactly the first occurrence of ***k***-mer as an edge in the graph, with all following occurrences considered to be copies. In our method, repeat ***k***-mers in a single specific cover are inserted as separate edges leading to different paths when the specific cover's ***k***-mers are created from its pieces, or alternatively are represented using edge weight values

on the ***k***-mer edge that are adjusted by -1 each time the edge is traversed, becoming impassable when the weight becomes 0.

3.5.1. Single cover with short sequences

In the case of RNA with length of 1000 or less, an approach to dealing with the fragment order problem using pieces from a single cover uses the following variation:

1. Derive the possible combinations of fragment orders and run the process on each one that begins with the ‘startup’ fragment. Since the ***k***-mer size is going to be around 23% of the pieces' total length of the RNA, the number of combinations to test is small. This is illustrated in step 3.2 in the main algorithm.
2. Once the 'candidate' assembly results are obtained from the derived combinations, the final assembly result is selected by finding a candidate result whose length equals the total fragment length and which contains single, non-overlapping copies of all the pieces.

The combinations derived in the above steps are necessary because the ***k***-mers needed for the graphs are created from the pieces themselves and thus the order in which the pieces are read in affects the result. Since our goal is to find the correct fragment order, we cannot know in advance which order to use.

This variation however cannot be considered to RNA of sizes larger than 1000, since the maximum ***k***-mer size is 250 and the computational cost of processing multiple combinations of pieces would be too high.

The following subsection details the modifications and additions to the proposed algorithm to implement this variation.

3.5.2. Short single-cover algorithm variant

Below is the variation of the algorithm for using order combinations of single cover pieces, as discussed in the previous subsection. It creates all the permutations of the cover pieces array, and it would be placed before step 3 in the algorithm:

VarA. permutationsAssembly

INPUT: Array of pieces Pieces[ ]

1. Create all permutations of contents in Pieces[ ]

2. For every permutation of Pieces[ ]

    2.1. go to 3 in main algorithm with permutation as input

    2.2. If length of sequenceResult = total of lengths of pieces in permutation AND sequenceResult contains all pieces then

        OUTPUT: sequenceResult

3.5.3. *k*-mer size selection

Historically, the selection of optimum k-mer length for graph-based assemblers has been subject to much speculation and discussion and in practice it is mostly determined via anecdotal evidence or trial and error [3]. That is to be expected, since intuitively the effect of size ***k*** is directly relevant to the overall sequence of RNAs, whose variance in size alone without even factoring in other elements of practice and industry (e.g. how much coverage was done to produce the pieces) makes it difficult to pre-determine a ‘good’ universal ***k***-mer size. This work thus allows for adjustments in the k-mer size by making it equal to a given ratio of the entire length (rounded to the nearest integer) of the individual RNA sequence. In the interest

of time and scope of work, this ratio was determined by a simple experiment involving a small RNA sample described below:

1. As a test subject, the crystal structure of yeast initiator tRNA sample with length 75 used in this work (see Appendix A) was selected and fragmented randomly.
2. An open k-mer size version of the implementation of the proposed algorithm was run using the pieces, beginning at k-mer size 4 (the minimum possible including all letters {A,C,G,U}) and the result was compared with the original sequence.
3. Step 2 was repeated, incrementing k and comparing result with original with each iteration.
4. At the iteration where ***k***=17, or approximately 23% of the overall sample tRNA sequence length, the assembly returned a result identical to the original sequence. This ratio was therefore taken as a possible good working candidate. Below is a tabulation of the results corresponding to the respective ***k***-mer sizes from which they were obtained.

Table 3: Tabulations of the results corresponding to the respective k-mer sizes for the yeast sample

Size *k*	Result
4	AGCGCCGUGGCGCAGUGGCGCGCAGGGCGCAGUGGAAACCA
5	AGCGCCGUGGCGCAGUGGCAGGGCUCAUAACCGAGCGGCGCAGUGGAAGCGCCGUGGCGCUACCA
6	AGCGCCGUGGCGCAGUGGUGGCGCAGGGCUCAUAACCCUGAUGUCCUCGGAUCGAAACCGAGCGGCGCUACCA
7	AGCGCCGUGGCGCAGUGUGGCGCAGGGCUCAUAACCCUGAUGUCCUCGGAUCGAAACCGAGCGGCGCUACCA
8	AGCGCCGUGGCGCAGUGGGCGCAGUGGAAGCGCGCAGGGCUCAUAACCCUGAUGUCCUCGGAUCGAAACCGAGCGGCGCUACCA
9	AGCGCCGUGGCGCAGUGGCGCAGUGGAAGCGCGCAGGGCUCAUAACCCUGAUGUCCUCGGAUCGAAACCGAGCGGCGCUACCA
10	AGCGCCGUGGCGCAGUGCGCAGUGGAAGCGCGCAGGGCUCAUAACCCUGAUGUCCUCGGAUCGAAACCGAGCGGCGCUACCA
11	AGCGCCGUGGCGCAGUGCGCAGUGGAAGCGCGCAGGGCUCAUAACCCUGAUGUCCUCGGACGAAACCGAGCGGCGCUACCA
12	AGCGCCGUGGCGCAGUGCAGUGGAAGCGCGCAGGGCUCAUAACCCUGAUGUCCUCGGAUCGAAACCGAGCGGCGCUACCA
13	AGCGCCGUGGCGCAGUGGUGGAAGCGCGCAGGGCUCAUAACCCUGAUGUCCUCGG

	AUCGAAACCGAGCGGCGCUACCA
14	AGCGCCGUGGCGCAGUGGUGGAAGCGCGCAGGGCUCAUAACCCUGAUGUCCUCGG AUCGAAACCGAGCGGCGCUACCA
15	AGCGCCGUGGCGCAGUGUGGAAGCGCGCAGGGCUCAUAACCCUGAUGUCCUCGGA UCGAAACCGAGCGGCGCUACCA
16	AGCGCCGUGGCGCAGUGGGAAGCGCGCAGGGCUCAUAACCCUGAUGUCCUCGGAU CGAAACCGAGCGGCGCUACCA
17	AGCGCCGUGGCGCAGUGGAAGCGCGCAGGGCUCAUAACCCUGAUGUCCUCGGAUC GAAACCGAGCGGCGCUACCA
Original sequence AGCGCCGUGGCGCAGUGGAAGCGCGCAGGGCUCAUAACCCUGAUGUCCUCGGAUCGAAACCGAG CGGCGCUACCA ***k*** = 17 gives an exact match	

5. The ratio was then tested on other RNA samples used in this work of relatively short lengths of less than 500, which again provided a replica of the original version.

Based on the result of the above simple test, 23% of the overall sum of lengths of pieces in a single cover was selected as a calculation of the ***k***-mer size used in the algorithm according to the input pieces. This test was not intended to find an optimum length of k, but one that has a good possibility of yielding good results in practice.

3.6. Assembly using multiple covers

Although it is an economically attractive proposition to use the one cover for sequence assembly, computation starts to become problematic and expensive as far as the method described in this work is concerned when assembling RNA that are of lengths in the thousands. At these lengths, the limitation on the maximum possible fragment size leads to more and more pieces used to generate permutations, raising the computation cost exponentially. For this reason, a variant is considered that uses additional coverage in order to incorporate as much information as possible regarding the structure.

3.6.1. How to use multiple covers

In terms of the graph itself, the variation of de Bruijn graph used allowing for repeat ***k***-mers results in more multiple paths outgoing from a vertex, making choosing the outgoing path a more complex problem. Without any distinguishing characteristic (e.g. only one outgoing path matches a fragment sequence) any path is valid and the first one is selected. It is for this reason that in the algorithm, rather than trying to find the 'best path' in the graph, edges are continuously traversed until all pieces are accounted for in the final result while maintaining a record of to which cover each fragment belongs. This approach leads to extra 'noise' nucleotides as well, for which an additional step of using pieces from a single cover as a guide to eliminate them. In the testing phase detailed later in this work, coverage from two to six was used to test the algorithm's performance.

3.6.2. Multi-cover algorithm variant

The following pseudocode describes the variation for using multiple covers. The purpose is to remove 'noise' data that occur because of extra paths and to prevent the repeat of ***k***-mer information of the same structure region, using the pieces of a single cover as a guide. This variation takes place at the end of the main algorithm.

VarB. MultiCoverResult

INPUT: sequenceResult, GuideCoverPieces[ ]

OUPUT: sequenceResult

1. Initialize a holding array FragFoundHolder [ ] that contains the pieces that have been found.

2. For i = 1 to length of GuideCoverPieces [ ]

    2.1. If sequenceResult contains GuideCoverPieces [i] then

        FragFoundHolder [position in sequenceResult where GuideCoverPieces [i] is found] = Pieces [i]

3. Re-initialize sequenceResult = ” ”

4. For j = 1 to length of FragFoundHolder[ ]

    4.1. add FragFoundHolder[j] at end of sequenceResult

5. OUTPUT sequenceResult

3.6.3. *k*-mer size selection

A ***k***-mer size needs to be selected for RNA that is too large to apply the 23% ratio to due to sequencing technology limitations. By incrementally testing k-mer sizes on the three largest samples and examining the resulting assembly, it was determined that k = 15 gave the best result when considering all three, as shown for the 70S Ribosome-tRNA Complex in Thermus Thermophilus in the table below. Again, the intent is not to find the ideal ***k***-mer size, but a workable one that helps test the hypothesis. Additionally, larger ***k***-mer sizes (e.g. the maximum of 250) increase ‘noise’ data to be identified and discarded. This is owing to the fact that the larger k-mers are, the less likely they are to overlap and this consequently leads to a longer sequence to traverse; this is compounded by the allowance of repeat k-mers in the cover discussed earlier in this work.

Table 4: Correct fragment groupings for 70S Ribosome-tRNA Complex in *Thermus thermophilus*

	Correct fragment groupings for 70S Ribosome-tRNA Complex in *Thermus thermophilus* (Cover = 5)
***k*-mer size = 15**	(1),(8-5-6),(17-10-18),(7-9),(13-12),2,3,4,11,14,15
***k*-mer size = 25**	(1),(6-2),(10-18),(9-13),3,4,5,7,8,11,12,14,15,16,17
***k*-mer size = 100**	(1),(6-2),(7-9-13-12),(16-17),3,4,5,8,11,14,15,18

CHAPTER FOUR: EXPERIMENTS AND TESTING RESULTS

4.1. The testing samples

A challenge that was faced in the course of this work was identifying a suitable candidate to test the proposed assembly process on. Much of the RNA sequence information available in the public domain does not include details as to how the sequences were obtained, which makes selecting a candidate to test a sequence assembly algorithm problematic: In order to conclude if the algorithm is viable or not, it has to be certain that the original sequence that the assembly result is going to be compared against is a true representation of the actual physical RNA structure. However, a lot of RNA sequences available for study are themselves product of earlier sequencing projects and shortest common superstring based assembly techniques, and a large amount do not have the sequence assembly details specified at all.

In this work, to have a realistic benchmark for assembly result quality, it was desirable that the test sample RNAs would be structures that are verified by either one of two methods:

1. Nuclear magnetic resonance (NMR) spectroscopy: Atomic level physical and structural features of molecules are derived in this method using the phenomenon of resonance frequency caused by the magnetic fields of atoms, thus providing physical evidence of the accuracy of the assembly process [11].
2. X-ray crystallography: A laboratory technique in which x-rays are used to derive crystal structure and atomic positions in RNAs and other molecules, and as in the previous method this provides a proven sequence that can be used to verify the accuracy of the assembly process [22].

RNA structures fitting this criterion were found in the RCSB Protein Data Bank project (http://www.rcsb.org), a repository for genomic data resources, and five were chosen for testing with the proposed method, described below. The sample set chosen is of gradual increase in size and complexity order to help show the effect of complexity on the application.

Table 5: Sample 1 - Yeast initiator tRNA details

DESCRIPTION	Yeast initiator tRNA (obtained from crystal structure)
TYPE	tRNA
LENGTH	75
REFERENCE	http://www.rcsb.org/pdb/explore/explore.do?structureId=1YFG

Table 6: Sample 2 - Zinc finger in *Xenopus laevis* details

DESCRIPTION	Zinc finger in *Xenopus laevis* (obtained from crystal structure)
TYPE	5S Ribosomal RNA
LENGTH	122
REFERENCE	http://www.rcsb.org/pdb/explore/explore.do?structureId=1UN6

Table 7: Sample 3 - Group I ribozyme domain

DESCRIPTION	Group I ribozyme domain (obtained from crystal structure)
TYPE	Ribozyme RNA
LENGTH	316
REFERENCE	http://www.rcsb.org/pdb/explore/explore.do?structureId=1GID

Table 8: Sample 4 - 16S Ribosomal RNA in *Thermus thermophilus*

DESCRIPTION	*Thermus thermophilus* (Structure of functionally activated small ribosomal subunit)
TYPE	16S Ribosomal RNA
LENGTH	1174
REFERENCE	http://www.rcsb.org/pdb/explore/explore.do?structureId=1FKA

Table 9: Sample 5 - 70S Ribosome-tRNA Complex in *Thermus thermophilus*

DESCRIPTION	70S Ribosome-tRNA Complex in *Thermus thermophilus* (obtained from crystal structure)
TYPE	23S Ribosomal RNA
LENGTH	3009
REFERENCE	http://www.rcsb.org/pdb/explore/explore.do?structureId=4V4I

Additionally, even though no x-ray crystallography or NMR spectroscopy of their structure was found for them, it was considered worthwhile to include two viral RNAs. They are more complex organisms and are a class of RNA that is subject to much interest in biomedical research.

Table 10: Sample 6 - Human immunodeficiency virus 1

DESCRIPTION	Human immunodeficiency virus 1
TYPE	Viral RNA
LENGTH	10340
REFERENCE	http://www.ncbi.nlm.nih.gov/nuccore/114430851?report=fasta

Table 11: Sample 7 - SARS coronavirus

DESCRIPTION	SARS coronavirus
TYPE	Viral RNA
LENGTH	29751
REFERENCE	http://www.ncbi.nlm.nih.gov/nuccore/30271926?report=fasta

The actual representative sequences for the samples can be found in appendix A.

To test the proposed sequence assembly method, the sample pieces are also assembled using an implementation of classical de Bruijn graph sequence assembly based on Fleury algorithm Eulerian path selection [2], and the results of the two approaches are compared. The ***k***-mer size used in the latter approach is the same as the former to maintain input parity in the two approaches, and the same number of covers was used in both methods as well.

4.2. Testing results

Below are the results of the tests on the samples as described in the previous section. In the case of samples where the length is less 1200, results for the proposed methods have two versions: One is the result based on multi-coverage (specifically, two covers) to deal with cases

where there is no clear structure preference in outgoing edges in the graph as previously discussed, while the other is the fragment order combination variant discussed in chapter three in section 3.5. For the larger samples only the multi-coverage based assembly was tested. In the result tables, the 'length' column can be used to determine how much information is lost in the case of the classic de Bruijn graph based method assembly result when compared with the length of the actual original sequence. The red highlighted regions in the de Bruijn graph based method assembly result are regions found also in the original sequence, but not in the same position.

Table 12 : Sample 1 assembly results

***k*-mer size = 17**

	Length	**Sequence**
Original sequence	75	AGCGCCGUGGCGCAGUGGAAGCGCGCAGGGCUCAUAACCCUGAUGUCCUCGGAUCGAAACCGAGCGGCGCUACCA
Proposed method/Single cover-fragment order combinations	75	AGCGCCGUGGCGCAGUGGAAGCGCGCAGGGCUCAUAACCCUGAUGUCCUCGGAUCGAAACCGAGCGGCGCUACCA
Proposed method/Multiple coverage(Cover =2)	75	AGCGCCGUGGCGCAGUGGAAGCGCGCAGGGCUCAUAACCCUGAUGUCCUCGGAUCGAAACCGAGCGGCGCUACCA
Classic deBruijn Graph based method	36	GAAGCGCGCAGGGCUCAUAAUAAAGCGCCGUGGCGCAGU

Table 13: Sample 2 assembly results

***k*-mer size = 28**

	Length	Sequence
Original sequence	122	GCCGGCCACACCUACGGGGCCUGGUUAGUACCUGGGAAACCU GGGAAUACCAGGUGCCGGCGCCGGCCACACCUACGGGGCCUG GUUAGUACCUGGGAAACCUGGGAAUACCAGGUGCCGGC
Proposed method/Single cover-fragment order combinations	122	GCCGGCCACACCUACGGGGCCUGGUUAGUACCUGGGAAACCU GGGAAUACCAGGUGCCGGCGCCGGCCACACCUACGGGGCCUG GUUAGUACCUGGGAAACCUGGGAAUACCAGGUGCCGGC
Proposed method/Multiple coverage(Cover =2)	122	GCCGGCCACACCUACGGGGCCUGGUUAGUACCUGGGAAACCU GGGAAUACCAGGUGCCGGCGCCGGCCACACCUACGGGGCCUG GUUAGUACCUGGGAAACCUGGGAAUACCAGGUGCCGGC
Classic deBruijn Graph based method	67	AAACCUGGGAAUACCAGGUGCCGGCGCCGGCCGGCCACAGAA ACCUGGGAAUACCAGGUGCCGGCGC

Table 14: Sample 3 assembly results

***k*-mer size = 72**

	Length	Sequence
Original sequence	319	GAAUUGCGGGAAAGGGGUCAACAGCCGUUCAGUACCAAGUCU CAGGGGAAACUUUGAGAUGGCCUUGCAAAG GGUAUGGU AAUAAGCUGACGGACAUGGUCCUAACCACGCAGCCAAGUCCU AAGUCAACAGAUCUUCUGUUGAUAUGGAUGCAGUUCGAAUU GCGGGAAAGGGGUCAACAGCCGUUCAGUACCAAGUCUCAGGG GAAACUUUGAGAUGGCCUUGCAAAGGGUAUGGUAAUAAGCU GACGGACAUGGUCCUAACCACGCAGCCAAGUCCUAAGUCAAC AGAUCUUCUGUUGAUAUGGAUGCAGUUC

Proposed method/Single cover-fragment order combinations	319	GAAUUGCGGGAAAGGGGUCAACAGCCGUUCAGUACCAAGUCU CAGGGGAAACUUUGAGAUGGCCUUGCAAAGGGUAUGGUAAU AAGCUGACGGACAUGGUCCUAACCACGCAGCCAAGUCCUAAG UCAACAGAUCUUCUGUUGAUAUGGAUGCAGUUCGAAUUGCG GGAAAGGGGUCAACAGCCGUUCAGUACCAAGUCUCAGGGGAA ACUUUGAGAUGGCCUUGCAAAGGGUAUGGUAAUAAGCUGAC GGACAUGGUCCUAACCACGCAGCCAAGUCCUAAGUCAACAGA UCUUCUGUUGAUAUGGAUGCAGUUC
Proposed method/Multiple coverage(Cover =2)	319	GAAUUGCGGGAAAGGGGUCAACAGCCGUUCAGUACCAAGUCU CAGGGGAAACUUUGAGAUGGCCUUGCAAAGGGUAUGGUAAU AAGCUGACGGACAUGGUCCUAACCACGCAGCCAAGUCCUAAG UCAACAGAUCUUCUGUUGAUAUGGAUGCAGUUCGAAUUGCG GGAAAGGGGUCAACAGCCGUUCAGUACCAAGUCUCAGGGGAA ACUUUGAGAUGGCCUUGCAAAGGGUAUGGUAAUAAGCUGAC GGACAUGGUCCUAACCACGCAGCCAAGUCCUAAGUCAACAGA UCUUCUGUUGAUAUGGAUGCAGUUC
Classic deBruijn Graph based method	304	GGUAUGGUAAUAAGCUGACGGACAUGGUCCUAACCACGCAGC CAAGUCCUAAGUCAACAGAUCUUCUGUUGAUAUGGAUGCAGU UCGAAUUGCGGGAAAGGGGUCAACAGCCGUUCAGUACCAAGU CUCAGGGGAAACUUUGAG GAAUUGCGGGAAAGGGGUCAA CAGCCGUUCAGUACCAAGUCUCAGGGGAAACUUUGAGAUGGC CUUGCAAAGAUGGCCUUGCAAAGGGUAUGGUAAUAAGCUGA CGGACAUGGUCCUAACCACGCAGCCAAGUCCUAAGUCAACAG AUCUUCUGUUG

The above samples are small enough to enable an easy analysis of the assembly results of the approaches. Even in the case of RNAs of such small sizes, the traditional method has resulted in a sequence with meaningful regions missing, and the regions present are not in the right

order. That does not mean that the method only gives wrong results, but there is a high probability that the assembly would be incorrect and without the benefit of a reference sequence it is not possible to determine whether the assembly result is correct or not. The proposed method, both in the case of single cover and multiple cover versions, the assembly result matches exactly the original sequence.

As they are far too large, the remainder of the results' tables does not include the actual sequences, which are detailed in the appendix. However the results for sample 5 will be discussed in detail to illustrate the algorithm's behavior in the case of larger RNA.

Table 15: Sample 4 assembly results

k-mer size = 250

Length of original sequence	1174
Length of proposed method result	1174
Length of Classic deBruijn Graph based method result	499

Table 16: Sample 5 assembly results

k-mer size = 15

Length of original sequence	3009
Length of proposed method result	3009
Length of Classic deBruijn Graph based method result	3010[1]

[1] Even though the length of the result is close to the original, the sequence was not the same, nor did it contain all the reference fragments.

In this case, the ***k***-mer length is 0.4% of the RNA's size. As obtainable ***k***-mer length becomes less meaningful with respect to the size of the RNA, it becomes more difficult to guarantee following the relevant path in the created graph of growing complexity and similar branching. This means that even though the sequence information is largely intact, it does not mean it will be necessarily assembled in the order in which it was originally discovered, especially without backtracking which was discarded because it would raise computation cost.

However, as the assembly in this work is considered from a structure perspective, the ordering is still useful because it tends to cluster related or neighbouring pieces together. So, even though the representative string sequence is not strictly identical to what it was originally in its entirety, valuable information on how the RNA is built is maintained which steadily increases with additional coverage. Although it is beyond the scope of this work, there is indication that there would be a finite number of covers for an RNA that would eventually group the pieces into the one cluster whose sequence would be the same as the original state. This correlation can be deduced from the tabulations below showing how the algorithm correctly clustered the RNA regions for the 70S Ribosome-tRNA when compared to the original sequence as the coverage increases.

Table 17: Correct fragment groupings in 70S Ribosome-tRNA assembly

	Correct fragment groupings (18 pieces from selected test cover)
Coverage =2	(1),(16-17),2,3,4,5,6,7,8,9,10,11,12,13,14,15,18
Coverage =3	(1),(2-7),(8-5-6),(14,15),3,4,9,10,11,12,13,16,17,18
Coverage =4	(1) , (9-13), (17-10-18),3,4,5,6,7,8,9,11,12,14,15,16
Coverage =5	(1) , (13-2) , (8-5-6) , (7-9) , (17-10-18), 3,4,11,12,14,15,16
Coverage =6	(1) , (13-2-7-9) , (4-8-5-6) , (17-10-18) , 3, 11,12,14,15,16

Table 18: Sample 6 assembly results

k-mer size = 15

Length of original sequence	10340
Length of proposed method result	10340
Length of Classic deBruijn Graph based method result	249

Table 19: Sample 7 assembly results:

k-mer size = 15

Length of original sequence	29751
Length of proposed method result	29751
Length of Classic deBruijn Graph based method result	18498

From the results, several observations can be made. Firstly, there were no regions lost in the assembly result of the proposed method implementation for all the samples. Secondly, the implementation of the classic deBruijn graph assembly resulted in missing regions, even with relaxing the conditions of Eulerian paths in Fleury's algorithm. Thirdly**,** multicover assembly

result improves with increased coverage, and the improvement happens with a relatively low number of covers considering the size of the RNA.

The success of the assembly can be illustrated by the multicover results of sample 6 –that of the HIV genome- shown in the table below. The largest clustering of pieces is actually the ENV protein, forming the viral envelope that enables the virus to target and attach to specific cell types and infiltrate their membranes [4].

Table 20: Correct fragment groupings for HIV sample

	Correct fragment groupings for HIV sample(46 pieces from selected test cover)
Coverage =6	**(13-3) , (12-8) , (31-9) , (25-33) , (36-30) , (7-39-32-24) , (14-10-18-17) , (37-11-22) , (45-20-26)**

CHAPTER FIVE: Conclusions and Future Work

5.1. Conclusions

The tests and observations in the previous chapter show that in assembling RNA sequences, the method and algorithm proposed in this work is an improvement in the sense of minimizing the exclusion of sub-sequences resembling actual regions of the RNA in the assembly result. While further tests with larger RNAs with more automated indexing are needed to further establish correctness, this technique also shows some success in determining the correct order of the pieces in the assembly result as well.

This technique is also an improvement with respect to the coverage required to assemble. For the results obtained, the number of multiple covers was at the minimum of 2 or 3, and it has proven that in many cases a single cover is sufficient to assemble the sequence correctly. This is highly indicative, as how much coverage is used has a direct effect on how costly the sequence assembly process is, and higher amounts of coverage were needed in overlap and graph based sequence assembly. Previous non-structure based techniques rely on overlaps and thus cannot assemble from pieces of a single copy of the sequence.

The above is a substantial argument for the use of RNA structure information together with the sequence data to solve the problem of sequence assembly. More fundamentally, it shows that is worth considering RNA characteristics across multiple disciplines including physical and chemical properties and how they are represented from a computational perspective. In the

context of this work, structural information refers specifically to stems, although it might prove worth investigating more sophisticated and complex structure features.

During the course of research and testing, issues and difficulties with this approach became apparent that should be opened for discussion and development. They are more concerning implementation technique and efficiency of performance than the concept itself, which yielded positive results even if with room for improvement in the implementation and programming.

- Determining the starting fragment: In the proposed algorithm the starting ***k***-mer or edge in the graph is selected by finding the pieces that form the reverse complement pair with the highest bond score. Then the path in the graph is followed from the ***k***-mer or edge that corresponds with the top or pre-bend fragment sequence. While this is a reasonable starting hypothesis, it has certain problems: It is possible that the beginning and end of RNA are not part of a stabilizing or 'spine' stem, and such a case the assembly result would have at least two pieces in the wrong place or order. Another problem is that to counter under-representation of repeated regions in the sequence, the graph was modified to contain repeat ***k***-mers or edges, so if a ***k***-mer or edge that corresponds with the beginning fragment has more than one version in the graph, on what basis should one be chosen over the others? The relatively large size of ***k*** helps in minimizing the likelihood of this occurrence, but it can still happen. In the testing implementation in this work, simply the first ***k***-mer found is the one selected.
    - In a different type of graph path selection problem, 'trivial' path selection occurs when there are two or more outgoing ***k***-mers or edges from the current vertex in the graph and there is nothing in the fragment or stem information that helps pick one outgoing edge over the others and so any outgoing edge is chosen. Evaluating the

consequences of taking each path would require backtracking, a potentially exponential computation that is outside the scope of this work. Again, the test implementation randomly takes the first one if no fragment-based preference is found.

The problems become more pronounced when assembling larger RNA and ***k***-mer size becomes less meaningful by virtue of technology limitations. It is worth noting that these problems are related to the sequence path selection in the generated graph.

5.2. Possible improvements and future work

While the proposed method and algorithm, and the test implementation as proof of concept, shows that RNA structure information has high potential as part of the solution to the sequence assembly problem, there is still room for improvement in implementation and concept research. Indeed, while it proposes a practical alternative approach, this work is also intended to motivate further examination of the sequence assembly problem and considering the different feature spaces and vectors that RNA and other types of sequences in their different aspects and viewpoints have to offer in solving the problem. It would be worthwhile to follow up the effort by considering the following avenues for development and research:

- As the proposed method proved viable in assembling RNAs, there is potential for expanding or modifying it to specialize in RNA subtypes as well as other sequences such as DNAs and proteins. A successful expansion would be beneficial in applying classification of sequences projects, and would also allow for researching the relationship between the structure of the molecules and their possible functions.

- A key issue in the proposed method is dealing with the situation where a node in the graph has multiple outgoing edges, or ***k***-mers, that the algorithm cannot determine the best among them. This leads to the creation of additional noise sub-sequences that need to be identified and removed. It is worth exploring more efficient criteria for selecting the most appropriate outgoing edge from multiple edges the proposed algorithm in its current state cannot determine the best among them.
- Another aspect of this work that would merit further investigation is further use to fit stems found into schematics of typical RNA structures, as that might in turn open the door to incorporating other computation techniques and algorithms. As an example, how a stem fits in the structure can be an indicator as to its position in the overall sequence; that would then lead to the possibility of introducing the use of computation algorithms concerned with similar problems.

This work showed that it is reasonable to consider structure when conducting sequence assembly. More importantly, it showed that considering structure is also quite practical, and practicality is an issue that frequently rises in applied bioinformatics. The accuracy of sequence assembly is a factor that is impacting advances in genetic and molecular research, and thus using structural information to improve sequence assembly would contribute greatly to further works and discoveries in molecular biology and genomic sequence classification.

Bibliography

[1] Auffinger P., Westhoff E., ***Hydration of RNA Base Pairs.*** . Journal of Biomolecular Structure & Dynamics 16:693-707, 1998

[2] de Bruijn, N. G., ***A Combinatorial Problem****, Koninklijke Nederlandse Akademie v. Wetenschappen 49: 758–764, 1946*

[3] ***De Novo Assembly using Illumina Reads,*** Illumina products technical notes: Sequencing, Page 7 https://www.illumina.com/Documents/products/technotes/technote_denovo_assembly_ecoli.pdf

[4] GenBank repository, accession number AB253704.1, HIV 1 Env protein region http://www.ncbi.nlm.nih.gov/nuccore/114430851?from=6906&to=9506&sat=3&sat_key=58732

[5] Gilbert, W. ***DNA sequencing and gene structure***, Bioscience Reports 1: 353-375, 1981

[6] Henson J., Tischler G., Ning Z., ***Next-generation sequencing and large genome assemblies***, Pharmacogenomics 13:901-915, 2012

[7] Hossain S., Azimi N., Skiena S., ***Crystallizing short-read assemblies around lone Sanger reads,*** BMC Bioinformatics 10:1-12, 2009

[8] Huson D., Bioinformatics I course notes:9-10, Tubingen University, 2009

[9] Jiang T.**,** Li M., ***On the Approximation of Shortest Common Supersequences and Longest Common Subsequences,*** SIAM Journal on Computing 24: 1122–1139, 1995

[10] Kaplan H., Shafrir N., ***The greedy algorithm for shortest superstrings***, Information Processing Letters 93:13-17, 2005

[11] Keeler J., ***Understanding NMR Spectroscopy: Chapter 2: NMR and energy levels (reprinted at University of Cambridge).*** University of California, Irvine. https://www.researchgate.net/publication/267218200_Understanding_NMR_Spectroscopy

[12] Morozova,O. and Marra, M. A. ***Applications of next-generation sequencing technologies in functional genomics***. Genomics 92: 255–264, 2008

[13] Myers E., ***Toward Simplifying and Accurately Formulating Fragment Assembly,*** Journal of Computational Biology**,** 2:275-290, 1995

[14] Myers E., Kececioglu J. ***Exact and approximate algorithms for the sequence reconstruction problem.*** Algorithmica 13: 7-51, 1995

[15] Nagarajan N., Pop M., ***Parametric Complexity of Sequence Assembly: Theory and Applications to Next Generation Sequencing,*** Journal of Computational Biology 16: 897-908, 2009

[16] Needleman S. B., Wunsch C. D., ***A general method applicable to the search for similarities in the amino acid sequence of two proteins***, Journal of Molecular Biology 48:443-453, 1970

[17] Pareek, C. S., Smoczynski R., Tretyn A., ***Sequencing technologies and genome sequencing,*** Journal of Applied Genetics 52:353-375, 2011

[18] Pevzner P.A., Tang H., Waterman M.S., ***An Eulerian path approach to DNA fragment assembly,*** PMC 98: 9748-9753, 2001

[19] Pop M., ***Genome assembly reborn: recent computational challenges***, Briefings in Bioinformatics 10: 354-366, 2009

[20] Scheibye-Alsing K., Hoffmann S., Frankel A.M., Jensen P., Stadler P.F., ***Sequence Assembly,*** Computational Biology and Chemistry 33:121-136, 2009

[21] Smith, T. F. and Waterman, M., ***Identification of common molecular subsequences***, Journal of Molecular Biology 147:195-197, 1981

[22] Smyth M.S., Martin J.H.J., ***x Ray crystallography***, Molecular Pathology 53:8-14, 2000

[23] Venter, C. *et al.*, ***The sequence of the human genome***, Science 291:1304– 1351, 2001

[24] Wommack KE, Bhavsar J, Ravel J, ***Metagenomics: Read Length Matters***. Applied and Environmental Microbiology 74:1453–1463, 2008

[25] Ye C, Ma Z.S., Cannon C.H., Pop M., Yu D.W., ***Exploring Sparseness in De Novo Genome Assembly,*** BMC Bioinformatics 13:1-8, 2012

[26] Zuker M., and Stiegler P., ***Optimal computer folding of lare RNA sequences using thermodynamics and auxiliary information,*** Nucleic Acids Research 9:138-144, 1981

[27] *http://www.rnasoft.ca/strand/show_results.php?molecule_ID=PDB_00176*
Mirela Andronescu, Rosalia Aguirre-Hernandez, Anne Condon, and Holger H. Hoos: ***RNAsoft: a suite of RNA secondary structure prediction and design software tools***, Nucleic Acids Research, 31: 3416-3422, 2003
Wedekind J.E., McKay D.B., ***Crystal Structure of the Leadzyme at 1.8 a Resoltion: Metal Ion Binding and the Implications for Catalytic Mechanism and Allo Site Ion Regulation***, Biochemistry 42: 9554-9563, 2003

Appendix A

Description of the FASTA format:

As the original nucleotide sequences of the samples used in the tests are provided in the FASTA format, it is worth briefly describing this format and its file content representing RNA.

Below is an example of an RNA sequence in FASTA format. The first line contains identification and description information, and the second line is RNA the sequence itself.

> File PDB_00004.ct. RCSB Protein Data Bank 1A51, number of molecules: 1.

GGCCGAUGGUAGUGUGGGGUCUCCCCAUGCGAGAGUAGGCC

Sample 1: 3 A crystal structure of yeast initiator tRNA

Obtained from the RCSB Protein Databank

http://www.rcsb.org/pdb/explore/explore.do?structureId=1YFG

> File PDB_00229.ct. RNA SSTRAND database. External source: RCSB Protein Data Bank 1YFG, number of molecules: 1. The secondary structure annotation was obtained with RNAview.

AGCGCCGUGGCGCAGUGGAAGCGCGCAGGGCUCAUAACCCUGAUGUCCUCGGA

UCGAAACCGAGCGGCGCUACCA

Sample 2: Crystal structure of Zinc finger in Xenopus Laevis

Obtained from the RCSB Protein Databank

http://www.rcsb.org/pdb/explore/explore.do?structureId=1UN6

> File PDB_00752.ct. RNA SSTRAND database. External source: RCSB Protein Data Bank 1UN6, number of molecules: 2.The secondary structure annotation was obtained with RNAview.

GCCGGCCACACCUACGGGGCCUGGUUAGUACCUGGGAAACCUGGGAAUACCAG

GUGCCGGCGCCGGCCACACCUACGGGGCCUGGUUAGUACCUGGGAAACCUGGG

AAUACCAGGUGCCGGC

Sample 3: Crystal structure of a group I ribozyme domain

Obtained from the RCSB Protein Databank

http://www.rcsb.org/pdb/explore/explore.do?structureId=1GID

> File PDB_00078.ct. RNA SSTRAND database. External source: RCSB Protein Data Bank 1GID, number of molecules: 2. The secondary structure annotation was obtained with RNAview.

GAAUUGCGGGAAAGGGGUCAACAGCCGUUCAGUACCAAGUCUCAGGGGAAAC

UUUGAGAUGGCCUUGCAAAGGGUAUGGUAAUAAGCUGACGGACAUGGUCCUA

ACCACGCAGCCAAGUCCUAAGUCAACAGAUCUUCUGUUGAUAUGGAUGCAGU

UCGAAUUGCGGGAAAGGGGUCAACAGCCGUUCAGUACCAAGUCUCAGGGGAA

ACUUUGAGAUGGCCUUGCAAAGGGUAUGGUAAUAAGCUGACGGACAUGGUCC

UAACCACGCAGCCAAGUCCUAAGUCAACAGAUCUUCUGUUGAUAUGGAUGCA

GUUC

Sample 4: Structure of functionally activated small ribosomal subunit in Thermus Thermophilus

Obtained from the RCSB Protein Databank

http://www.rcsb.org/pdb/explore/explore.do?structureId=1FKA

> File PDB_00409.ct. RNA SSTRAND database. External source: RCSB Protein Data Bank 1FKA, number of molecules: 1. the secondary structure annotation was obtained with RNAview.

GAGUUGAUCCUGGCUCAGGGUAACGCUGGCGGCGUGCGACAUGCAGUCGUGC

GGGCCGCGGGGUACUCCGUGGUCAGCGGCGGUGAGCGUGGGUCCCCGGAAGA

GGGGGACAACCCGGGGAAACUCGGGCCCCCCAUGGACCCGCCCCUUGGGGUGU

GUCGGGCUUUGCCCGCUUCCGGAUGGGCCCGCGUCCCGCUUUGGUGGGGUGGC

CCACCAAGGCGGGUAGCCGGUCUGAGAGGAUGCCGGCCACGGGGCACUGAGAC

CGGGCCCCACUCCUGGAGAGCAGUUAGGAAUCUUCCGCAAUGGGCGCCUGCGG

AGCGACGCCGCUGGAGGAAGCCCUUCGGGGAAACUCCUGAACCCGGGCGAAAC

CCCCGACGAGGGGACUGACACCGGGGUGCGCCGGCCUCCGUCCAGCGCCGCGG

CGGAGGGCGCAGCGUUACCCGCGUAGGCGUGUAGGCGGCCUGGGGCGUCCCAU

GUGAAAGACCACGGCUCAACCGUGGGGGAGCGUGGGAUACGCUCAGGCUAGA

CGGUGGGAGAGGGUGGUGGAAUUCCCGGAGUAGCGGUGAAAUGCCCGGGAGG

AACGCCGAGGCGAAGGCAGCCACCUGGUCCACCCGUGGCUGAGGCGCAGCGUG

GGGCAAACCGGAUUCCGGGUAGUCCACGCCCUAACGCGCUGUCUCUGGGUCCU

GGGGGCCGAAGCCAGCGCGCCUGGGGCGGCUGAAACUCAAGGAAUUGACGGG

GCCCGAGCGGUGAGCAUGUGGUUUUCGCGAAGACCAGGCCUGCUAGGACCCGG

GUGGCCUGGGUGCCCGCGAGGGAGCCCUAGCCAGGGCUCCGUCGGCUCGUGCC

GGUUGGGUUCCCGACGAGCCCCCGCCGUUACCAGCGGCCGGGCUCUAACGGGG

CCCGCGAAAGCGGGAGGAAGGGGGACGUCUGGUCAGCAUGGCCCGCCUGGCAC

GUGCUAUGCCCUAAGCGAGCCACCCGGCAACGGGGAGCCAAUGGGCCGGAUUG

GGGUCUGAAGACCCCAUCCGAUCGCUUAAUCGCGGGCCGCGGUGAAUACGGGC

CUUGUACACGCCCGUCACGCCAUGGGAGCGGGCUCUACCAAGUCGCCGCUGGC

AGGCGCCGAGGGUAGGGCCCGUGACUGGGGCGAAGUCGUAAAGCUGUACCGG

AAGGUGCGGCUGGA

Sample 5: Crystal Structure of a 70S Ribosome-tRNA Complex in Thermus Thermophilus

Obtained from the RCSB Protein Databank

http:/www.rcsb.org/pdb/explore/explore.do?structureId=4V4I

> File PDB_00795.ct. RNA SSTRAND database. External source: RCSB Protein Data Bank 1VSA, number of molecules: 2. The secondary structure annotation was obtained with RNAview.

GGUCAAGAUGGUAAGGGCCCACGGUGGAUGCCUCGGCACCCGAGCCGAUGAA

GGACGUGGCUACCUGCGAUAAGCCAGGGGGAGCCGGUAGCGGGCGUGGAUCC

CUGGAUGUCCGAAUGGGGGAACCCGGCCGGCGGGAACGCCGGUCACCGCGCUU

UUUGCGCGGGGGGAACCUGGGGAACUGAAACAUCUCAGUACCCAGAGGAGAG

GAAAGAGAAAUCGACUCCCUGAGUAGCGGCGAGCGAAAGGGGACCAGCCUAA

ACCGUCCGGCUUGUCCGGGCGGGGUCGUGGGGCCCUCGGACACCGAAUCCCCA

GCCUAGCCGAAGCUGUUGGGAAGCAGCGCCAGAGAGGGUGAAAGCCCCGUAG

GCGAAAGGUGGGGGGAUAGGUGAGGGUACCCGAGUACCCCGUGGUUCGUGGA

GCCAUGGGGGAAUCUGGGCGGACCACCGCCUAAGGCUAAGUACUCCGGGUGAC

CGAUAGCGCACCAGUACCGUGAGGGAAAGGUGAAAAGAACCCCGGGAGGGGA

GUGAAAUAGAGCCUGAAACCGUGGGCUUACAAGCAGUCACGGCCCCGCAAGG

GGUUGUGGCGUGCCUAUUGAAGCAUGAGCCGGCGACUCACGGUCGUGGGCGA

GCUUAAGCCGUUGAGGCGGAGGCGUAGGGAAACCGAGUCCGAACAGGGCGUC

UAGUCCGCGGCCGUGGACCCGAAACCGGGCGAGCUAGCCCUGGCCAGGGUGAA

GCUGGGGUGAGACCCAGUGGAGGCCCGAACCGGUGGGGGAUGCAAACCCCUCG

GAUGAGCUGGGGCUAGGAGUGAAAAGCUAACCGAGCCCGGAGAUAGCUGGUU

CUCCCCGAAAUGACUUUAGGGUCAGCCUCAGGCGCUGACUGGGGCCUGUAGAG

CACUGAUAGGGCUAGGGGGCCCACCAGCCUACCAAACCCUGUCAAACUCCGAA

GGGUCCCAGGUGGAGCCUGGGAGUGAGGGCGCGAGCGAUAACGUCCGCGUCC

GAGCGCGGGAACAACCGAGACCGCCAGCUAAGGCCCCCAAGUCUGGGCUAAGU

GGUAAAGGAUGUGGCGCCGCGAAGACAGCCAGGAGGUUGGCUUAGAAGCAGC

CAUCCUUUAAAGAGUGCGUAAUAGCUCACUGGUCGAGUGGCGCCGCGCCGAA

AAUGAUCGGGGCUCAAGCCCAGCGCCGAAGCUGCGGGUCUGGGGGAUGACCCC

AGGCGGUAGGGGAGCGUUCCCGAUGCCGAUGAAGGCCGACCCGCGAGGGCGGC

UGGAGGUAAGGGAAGUGCGAAUGCCGGCAUGAGUAACGAUAAAGAGGGUGAG

AAUCCCUCUCGCCGUAAGCCCAAGGGUUCCUACGCAAUGGUCGUCAGCGUAGG

GUUAGGCGGGACCUAAGGUGAAGCCGAAAGGCGUAGCCGAAGGGCAGCCGGU

UAAUAUUCCGGCCCUUCCCGCAGGUGCGAUGGGGGGACGCUCUAGGCUAGGG

GGACCGGAGCCAUGGACGAGCCCGGCCAGAAGCGCAGGGUGGGAGGUAGGCA

AAUCCGCCUCCCAAAAGCUCUGCGUGGUGGGGAAGCCCGUACGGGUGACAACC

CCCCGAAGCCAGGGAGCCAAGAAAAGCCUCUAAGCACAACCUGCGGGAACCCG

UACCGCAAACCGACACAGGUGGGCGGGUGCAAGAGCACUCAGGCGCGCGGGAG

AACCCUCGCCAAGGAACUCUGCAAGUUGGCCCCGUAACUUCGGGAGAAGGGGU

GCUCCCUGGGGUGAUGAGCUCCGGGGAGCCGCAGUGAACAGGCUCUGGCGACU

GUUUACCAAAAACACAGCUCUCUGCGAACUCGUAAGAGGAGGUAUAGGGAGC

GACGCUUGCCCGGUGCCGGAAGGUCAAGGGGAGGGGUGCAAGCCCCGAACCGA

AGCCCCGGUGAACGGCGGCCGUAACUAUAACGGUCCUAAGGUAGCGAAAUUCC

UUGUCGGGUAAGUUCCGACCUGCACGAAAAGCGUAACGACCGGAGCGCUGUC

UCGGCGAGGGACCCGGUGAAAUUGAACUGGCCGUGAAGAUGCGGCCUACCCG

UGGCAGGACGAAAAGACCCCGUGGAGCUUUACUGCAGCCUGGUGUUGGCUCU

UGGUCGCGCCUGCGUAGGAUAGGUGGGAGCCUGUGAACCCCCGCCUCCGGGUG

GGGGGGAGGCGCCGGUGAAAUACCACCCUGGCGCGGCUGGGGGCCUAACCCUC

GGAUGGGGGGACAGCGCUUGGCGGGCAGUUUGACUGGGGCGGUCGCCUCCUA

AAAGGUAACGGAGGCGCCCAAAGGUCCCCUCAGGCGGGACGGAAAUCCGCCGG

AGAGCGCAAGGGUAGAAGGGGGCCUGACUGCGAGGCCUGCAAGCCGAGCAGG

GGCGAAAGCCGGGCCUAGUGAACCGGUGGUCCCGUGUGGAAGGGCCAUCGAU

CAACGGAUAAAAGUUACCCCGGGGAUAACAGGCUGAUCUCCCCCGAGCGUCCA

CAGCGGCGGGGAGGUUUGGCACCUCGAUGUCGGCUCGUCGCAUCCUGGGGCUG

AAGAAGGUCCCAAGGGUUGGGCUGUUCGCCCAUUAAAGCGGCACGCGAGCUG

GGUUCAGAACGUCGUGAGACAGUUCGGUCUCUAUCCGCCACGGGCGCAGGAG

GCUUGAGGGGGGCUCUUCCUAGUACGAGAGGACCGGAAGGGACGCACCUCUG

GUUUCCCAGCUGUCCCUCCAGGGGCAUAAGCUGGGUAGCCAUGUGCGGAAGG

GAUAACCGCUGAAAGCAUCUAAGCGGGAAGCCCGCCCCAAGAUGAGGCCUCCC

ACGGCGUCAAGCCGGUAAGGACCCGGGAAGACGACCCGGUGGAUGGGCCGGG

GGUGUAAGCGCCGCGAGGCGUUGAGCCGACCGGUCCCAAUCGUCCGAGGUCUU

GACCUCCCCCGUGCCCAUAGCGGCGUGGAACCACCCGUUCCCAUUCCGAACAC

GGAAGUGAAACGCGCCAGCGCCGAUGGUACUGGGCGGGCGACCGCCUGGGAG

AGUAGGUCGGUGCGGGGGA

Sample 6: Human immunodeficiency virus 1

Obtained from the NCBI Databank

http://www.ncbi.nlm.nih.gov/nuccore/114430851?report=fasta

>gi|114430851|dbj|AB253704.1| Human immunodeficiency virus 1 proviral DNA, complete genome, clone: pJPDR2192AE08

UGGAUGGGCUAAUUUACUCCAAGAAAAGACAAGAGAUCCUUGACUUAUGGGU

CUAUAAUACACAAGGCUUCUUCCCUGAUUGGCAAAACUACACAGAAGGGCCA

GGGACCAGAUUCCCACUGUGCUUUGGAUGGUGCUUCAAGCUAGUACCAGUUG

ACCCAAGAGAAGUAGAGGAGAACAACAAAGGAGAAAACAGCUGCCUGUUACA

CCCCAUGAGCCAGCAUGGAAUAGAGGACGAAGAUAGAGAAGUGCUGAUGUGG

AAGUUUGACAGCUCCCUAGCACGAAGACACAUAGCCCGAGAAAAGCACCCAGA

GUUCUAUAAAGACUGCUGACAAAGAAGUUUCUAACUGGGACUUCCGCUGGGG

ACUUUCCAGGGGAGGUGUGGCCGGGGCGGAGUUGGGGAGUGGUUAACCCUCA

GAUGCUGCAUAAAAGCAGCCGCUUUUCGCUUGUACUGGGUCUCUCUUGUUAG

ACCAGGUCGAGCCCGGGAGCUCUCUGGCUAGCAAGGGAACCCACUGCUUAAAG

CCUCAAUAAAGCUUGCCUUGAGUGCUUAAAGUGGUGUGUGCCCGUCUGUGUU

AGGACUCUGGUAACUAGAGAUCCCUCAGACCACUCUAGACUAAGUAAAAAUC

UCUAGCAGUGGCGCCCGAACAGGGACUUGAAAGUGAAAGUUAAUAGGGACUC

GAAAGCGGAAGGUUCCAGAGAAGUUCUCUCGACGCAGGACUCGGCUUGCUGA

GGUGCACACAGCAAGAGGCGAGAGCGGCGACUGGUGAGUACGCCAAAUUUUG

ACUAGCGGUGGCUAGAAGGAGAGAAGAUGGGUGCGAGAGCGUCAGUAUUAAG

CGGGGGAAAAUUAGAUGCAUGGGAAAAAAUUCGGUUACGGCCAGGGGGGAAG

AAAAAAUAUAGGAUGAAACAUUUAGUAUGGGCAAGCAGAGAGUUGGAAAGAU

UCGCAAUUAACCCUGGCCUUUUAGAAACAGCAGAAGGAUGUCAACAAAUACU

AGAACAGUUACAGACAACUCUCAAGACAGGAUCAGAAGAACUUAAAUCAUUA

UUUAAUACAAUAGCAACCCUCUGGUGCGUGCACCAGAGGAUAGAGGUAAGAG

ACACCAAGGAAGCUUUAGAUAAAAUAGAGGAAGUACAAAAUAAGAGCCAGCA

AAAGACACAGCAGGGAGCAGCUGGCACAGGAAGCAGCAGCAAAGUCAGCCAA

AAUUAUCCUAUAGUGCAAAAUGCACAAGGGCAAAUGGUACAUCAGCCUUUAU

CACCUAGAACCUUGAAUGCAUGGGUGAAAGUAAUAGAAGAAAAGGGUUUUAA

CCCAGAAGUAAUACCCAUGUUCACAGCAUUAUCAGAGGGAGCCACCCCACAAG

AUUUAAAUAUGAUGCUAAAUAUAGUGGGGGGACACCAGGCAGCAAUGCAAAU

GUUAAAAGAAACCAUCAAUGAGGAAGCUGCAGAAUGGGAUAGGGUACACCCA

GUACAUGCAGGGCCUAUUCCACCAGGCCAGAUGAGGGAACCAAGGGGAAGUG

ACAUAGCAGGAUCUACUAGUACCCUUCAAGAACAAAUAGGAUGGAUGACAAG

CAAUCCUCCUAUCCCAGUGGGAGACAUCUAUAAAAGGUGGAUAAUCCUGGGA

UUAAAUAAAAUAGUAAGAAUGUAUAGCCCUGUUAGCAUUUUGGACAUAAGAC

AAGGGCCAAAAGAACCCUUCAGAGACUAUGUAGAUAGGUUCUAUAAAACUCU

CAGAGCAGAACAAGCUACACAGGAGGUAAAAAACUGGAUGACAGAAACCUUG

CUAGUCCAAAAUGCGAAUCCAGACUGUAAGUCCAUUUUAAAAGCAUUAGGAA

CAGGAGCUACAUUAGAAGAAAUGAUGACAGCAUGCCAGGGAGUAGGAGGACC

UGGCCAUAAAGCAAGGGUUUUGGCUGAGGCAAUGAGCCAGGUACAACAUGCA

AAGAUAAUGAUGCAGAGAGGCAAUUUUAAGGGCCAAAGAAGCAUUAAGUGCU

UCAACUGUGGCAGAGAAGGACACCUAGCCAGAAAUUGCAGGGCCCCCAGAAA

AAAGGGCUGUUGGAAAUGUGGAAAGGAAGGACAUCAAAUGAAAGACUGCACU

GAGAGACAGGCUAAUUUUUUAGGGAAAAUUUGGCCUUCCCACAAGGGAAGGC

CAGGAAAUUUUCCUCAGAGCAGACCAGAGCCAUCAGCCCCACCAGCAGAAAGC

UGGGAGAUGAGGGAAGAGACAACCUCCUUACUGAAGCAGGAGCAGAAAGACA

AGGAACAUCCUCCUCCUUUAGCUUCCCUCAAAUCACUCUUUGGCAACGACCCA

UUGUCACAGUAAAAAUAGAAGGACAGCUGAUAGAAGCUCUGUUAGAUACAGG

AGCAGAUGAUACAGUAUUAGAAAAUAUAAAUUUGCAAGGAAAAUGGAAACCA

GUAAUGAUAGGGGGAAUUGGAGGUUUUGUCAAAGUAAGGCAAUAUGAUCAGG

UACCUAUAGAAAUUUGUGGAAAAAAGGCUAUAGGUACAGUGUUAGUAGGACC

UACACCGUUCAACAUAAUUGGGCGAAAUAUGAUGACCCAGAUUGGUUGUACU

UUAAAUUUCCCAAUUAGUCCUAUUGACACUGUACCAGUAAAAUUAAAGCCAG

GAAUGGAUGGACCAAAGGUUAAACAGUGGCCAUUGACAGAAGAAAAAAUAAA

AGCAUUAACAGAAAUUUGUAAAGAGAUGGAAGAGGAAGGAAAAAUCUCAAAA

AUUGGGCCUGAAAAUCCAUACAAUACUCCAGUAUUUGCUAUAAAGAAAAAGA

ACAGCGACAGAUGGAGAAAAUUAGUAGAUUUCAGAGAGCUCAAUAAAAGAAC

UCAGGACUUUUGGGAAGUUCAAUUAGGAAUACCGCAUCCAGCAGGUUUAAAA

AAGAAAAAAUCAGUAACAAUACUAGAUGUGGGAGAUGCAUAUUUUUCAGUUC

CUUUAGAUGAAAACUUUAGAAAGUAUACUGCAUUCACCAUACCUAGUCUAAA

CAAUGAGACACCAGGAAUCAGAUAUCAGUACAAUGUGCUGCCACAGGGAUGG

AAAGGAUCACCGGCAAUAUUCCAGUGUAGCAUGACAAAAAUCUUAGAGCCCU

UUAAAAUAAAAAAUCCAGAAAUGGUUAUCUGUCAAUACAUGGAUGACUUGUA

UGUAGGAUCUGAUUUAGAAAUAGGGCAGCACAGAGCAAAAAUAGAGGAGCUA

AGAGCUCAUCUAUGGAGCUGGGGAUUUUUUACACCAGACUGGAAGCAUCAGA

AGGAACCUCCAUUCCAUUGGAUGGGAUAUGAACUCCAUCCUGACAAAUGGAC

AGUCCAGCCUAUAGAACUGCCAGAAAAAGACAGCUGGACUGUCAAUGAUAUA

CAGAAAUUAGUGGGAAAACUAAAUUGGGCAAGUCAAAUUUAUGCAGGGAUUA

AGGUAAAGCAACUGUGUAAACUCCUCAGGGGAGCUAAAGCACUAACAGACAU

AGUACCACUGACUGAAGAAGCAGAAUUAGAAUUGGCAGAGAACAGGGAGAUU

CUAAAAACCCCUGUGCAUGGAGUAUAUUAUGACCCAUCAAAAGACUUAGUAG

CAGAAGUACAGAAACAAGGGCAGGACCAAUGGACAUAUCAAAUUUUUCAAGA

GCCAUUUAAAAAUCUAAAAACAGGAAAAUAUGCUAGAAAAAGGUCUGCUCAC

ACUAAUGAUGUAAAACAAUUAACAGAAGUGGUACAAAAAAUAGCCACAGAAA

GCAUAGUAAUAUGGGGAAAGACCCCUAAAUUUAGACUACCCCUACAAAAAGA

AACAUGGGAAACAUGGUGGACGGAGUAUUGGCAGGCUACCUGGAUUCCUGAA

UGGGAGUUUGUUAAUACCCCUCCUCUAGUAAAAUUAUGGUACCAAUUAGAAA

AGGAACCCAUAGUAGGAGCAGAGACUUUCUAUGUAGAUGGGGCAGCUAGUAG

GGAGACUAAGCUAGGAAAAGCAGGGUAUGUCACUAACAGAGGAAGACAAAAG

GUAGUUUCCCUAACUGAGACAACAAAUCAAAAAAACUGAGUUACAUGCGAUCU

ACUUAGCCUUGCAGGAUUCAGGACCAGAAGUAAAUAUAGUAACAGACUCACA

AUAUGCAUUAGGAAUCAUUCAGGCACAACCAGACAGGAGUGACUCAGAAAUA

GUCAACCAAAUAAUAGAGGAGCUAAUAAAAAAGGAAAAGGUCUACCUGUCAU

GGGUACCAGCACACAAGGGGAUUGGAGGAAAUGAACAAGUAGAUAAAUUAGU

CAGUUCAGGAAUCAGAAGAGUGCUAUUUUUAGAUGGGAUAGAUAAGGCUCAA

GAAGAACAUGAAAGAUAUCACAGCAAUUGGAGAACAAUGGUUAGUGAUUUUA

AUUUGCCACCUAUAGUAGCAAAGGAAAUAGUAGCCAACUGUGAUAAAUGUCA

ACUAAAAGGGGAAGCUAUACAUGGACAAGUGGACUGUAGUCCAGGAAUAUGG

CAAUUAGAUUGCACACAUCUAGAAGGAAAAGUCAUCCUGGUAGCAGUCCACG

UGGCCAGUGGAUAUAUAGAAGCAGAAGUUAUCCCAGCAGAAACAGGACAGGA

GACAGCAUACUUUCUGCUAAAAUUAGCAGGAAGAUGGCCAGUAAAAGUAAUA

CACACAGACAAUGGUAGCAAUUUCACCAGCGCUGCAGUUAAAGCAGCCUGUU

GGUGGGCCAAUGUCAGACAGGAAUUUGGGAUCCCCUACAAUCCACAAAGUCA

AGGAGUAGUAGAAUCCAUGAAUAAGGAAUUAAAGAAAAUCAUAGGGCAAGUA

AGAGAGCAAGCUGAGCACCUUAAGACAGCAGUACAAAUGGCAGUAUUCAUUC

ACAAUUUUAAAAGAAAAGGGGGAAUUGGGGGGUACAGUGCAGGGGAAAGAAU

AAUAGACAUAAUAGCAACAGACAUACAAACUAAAGAAUUACAAAAACAAAUU

ACAAAAAUUCAAAAUUUUCGGGUUUAUUACAGGGACAGCAGAGACCCAAUUU

GGAAAGGACCAGCAAAACUGCUCUGGAAAGGUGAAGGGGCAGUAGUAAUACA

AGACAAUAGUGAUAUAAAAGUAGUACCAAGAAGAAAAGCAAAGAUCAUUAGA

GAUUAUGGAAAACAGAUGGCAGGUGAUGAUUGUGUGGCAAGUAGACAGGAUG

AGGAUCAGAACAUGGAAUAGUUUAGUAAAACAUCACAUGUAUGUCUCAAAGA

AAGCUAAAAAUUGGUUUUAUAGACAUCAUUAUGAAAGCCAGCAUCCAAAGGU

GAGUUCAGAAGUACAUAUCCCACUAGGAGAGGCUAAAUUAGUAAUAAGAACA

UAUUGGGGUCUGCAGACAGGAGAAAAGGACUGGCAAUUGGGCCAUGGUACGC

CUGGGGGAUCGAAUUCGGGAUCCCCUACAAUCCACAAAGUCAAGGAGUAGUA

GAAUCUAUGAAUAAGGAAUUAAAGAAAAUCAUAGGGCAAGUAAGAGAGCAAG

CUGAACACCUUAAGACAGCAGUACAAAUGGCAGUAUUCAUUCACAAUUUUAA

AAGAAAAGGGGGGAUUGGGGGGUACAGUGCAGGGGAAAGAAUAAUAGACAUA

AUAGCAACAGACAUACAAACUAAAGAAUUACAAAAACAAAUUACAAAAAUUC

AAAAUUUUCGGGUUUAUUACAGGGACAGCAGAGACCCAAUUUGGAAAGGACC

AGCAAAACUGCUCUGGAAAGGUGAAGGGGCAGUAGUAAUACAAGACAAUAGU

GAUAUAAAAGUAGUACCAAGAAGAAAAGCAAAGAUCAUUAGAGAUUAUGGAA

AACAGAUGGCAGGUGAUGAUUGUGUGGCAAGUAGACAGGAUGAGGAUCAGAA

CAUGGAAUAGUUUAGUAAAACAUCACAUGUAUGUCUCAAAGAAAGCUAAAAA

UUGGUUUUAUAGACAUCAUUAUGAAAGCCAGCAUCCAAAGGUGAGUUCAGAA

GUACAUAUCCCACUAGGAGAGGCUAAAUUAGUAAUAAGAACAUAUUGGGGUC

UGCAGACAGGAGAAAAGGACUGGCAAUUGGGUCAUGGAGUCUCCAUAGAAUG

GAGGCAGAGAACAUAUAGCACACAAAUAGAUCCUGACCUAGCAGACCAACUG

AUUCAUCUACAAUAUUUUGACUGUUUUUCAGACUCUGCCAUAAGGAAAGCCA

UAUUAGGACAAGUAGUUAGAUAUAGGUGUGAAUAUCCAUCAGGACAUAACCA

GGUAGGAUCCCUACAAUAUUUGGCACUGAAAGCAUUAACAACACCAAAAAGG

ACAAGGCCACCUCUUCCUAGUGUUAAGAAAUUAACAGAAGAUAGAUGGAACA

AGCCCCAGAAGAUCAGGGGCCACAGAGAGAACCCCACAAUGAAUGGACAUUA

GAACUGUUAGAGGAGCUUAAAAAUGAAGCUGUUAGACAUUUUCCUAGGCCCU

GGCUCCUUGGCCUAGGACAGUACAUCUAUGACACUUAUGGGGAUACUUGGGA

AGGGGUUGAAGCUAUAAUAAGAAUUUUGCAACAACUACUGUUUGUUCAUUAC

AGAAUUGGGUGUCAGCAUAGCAGAAUAGGCAUUAUACCAGGGAGAAGAGGCA

GGAAUGGAGCCAGUAGAUCCUAACCUAGAGCCCUGGAAUCAUCCGGGAAGUC

AGCCUACAACUGCUUGUAGCAAGUGUUACUGUAAAAGAUGUUGCUGGCAUUG

CCAACUAUGCUUUCUGAACAAAGGCUUAGGCAUCUCCUAUGGCAGGAAGAAG

CGGAAGCACCGACGAAGAACUCCUCAGAGCAGUAAGGAUCAUCAACAUACUA

UACCAGAGCAGUAAGUAUUAGUAUCUGUAAUGUCACCUUUGGAAAUUAGUGC

AAUAGUAGGACUGAUAGUAGCACUAAUCUUUGCAAUAGUAGUGUGGACUAUA

AUAGCUAUAGAAUAUAAGAAAAUACUAAAGCAAAGAAAAAUAGACAGGUUAA

UUAAAAGAAUAAGAGAAAGAGAAGAAGAUAGUGGAAAUGAGAGUGAAGGAG

ACACAGAUGAUUUGGCCAAACUUGUGGAAAUGGGGAACUUUGAUCCUUGGAU

UGGUAAUAAUUUGUAGUGCCUCAGACAACUUGUGGGUUACAGUUUAUUAUGG

GGUUCCUGUGUGGAGAGAUGCAGAUACCACCCUAUUUUGUGCAUCAGAUGCC

AAAGCACAUGAGACAGAAGUGCACAAUAUCUGGGCCACACAUGCCUGUGUAC

CCACAGACCCCAACCCACAAGAAAUACGCCUGGAAAAUGUAACAGAAAAUUUU

AACAUGUGGAAAAAUAACAUGGUAGAGCAGAUGCAGGAGGAUGUAAUCAGUU

UAUGGGAUCAAAGUCUACAGCCAUGUGUAAAGUUAACUCCUCUCUGCGUUAC

UUUAAAUUGUACCAAUGCUACGUUGAAUGCUAAUUUGACCUAUGUCAAUAAC

AUAACUAAUGGCCAUAACACAAUAGGAAAUAUAACAGAUGAAGUAAAAAACU

GUUCUUUUAAGAUGACCACAGAACUAAGAGAUAAGAGGAAGAAGGUCCAUGC

ACUUUUUUAUAAGCUUGAUAUAGUACAACUUAAAGGUAAUAAAAAUAAGAAU

AGUAAUUUUAGUCAGUAUAGGUUAAUAAGUUGUAAUACUUCAGUCAUUAAGC

AGGCUUGUCCAAAGAUAUCCUUUGAUCCAAUUCCUAUACAUUAUUGUACUCC

AGCUGGUUAUGCGAUUUUAAAGUGUAAUGAUAAGAAUUUCACUGGGACAGGG

CCCUGUAAAAAUGUCAGCUCAGUACAAUGCACACAUGGAAUCAAGCCAGUGG

UAUCAACUCAAUUGCUGUUAAAUGGCAGUCUAGCAGAAGAAGAGAUAAUAAU

CAGAUCUGAAAAUCUCACAAACAAUGCCAAAACCAUAAUAGUGCACCUUAAU

AAAUCUGUAGAAAUCAAUUGUACCAGACCCUCCAACAAUACAAGAACAAGUA

CACAUAUAGGACCAGGACAAGUAUUCUAUAGAACAGGAGACAUAAUUGGAGA

UAUAAGGAAAGCAUAUUGUGAAAUUAAUGGAACAAAAUGGAAUGAAACUUUA

AAACAGGUAGCUGAAAAAUUAAAAGAGCAUUUUAAUAAUAAGACAAUAAUCU

UUCAACCACCCUCAGGAGGAGAUCUAGAAGUUACAAUGCAUCAUUUUAAUUG

UAGAGGGGAAUUUUUCUAUUGCAAUACAUCAAAAAUGUUUAAUAGUACUUGG

AAAGAAAAUGAAACAAUGGGGGGGCUUAAUGGCACUAUCAUACUUCCAUGCA

GGAUAAAGCAAAUUAUAAACAUGUGGCAGGGAGUAGGACAAGCAAUGUAUGC

UCCUCCCAUCAGUGGAAGAAUUAAUUGUGUAUCAAAUAUUACAGGAAUACUA

UUGACAAGAGAUGGUGGUGCUAAUAAUGCGACUAAUGAGACCUUUAGACCUG

GAGGAGGAAAUAUAAAGGACAAUUGGAGAAGUGAAUUAUAUAAAUAUAAAGU

AGUACAAAUUGAACCUCUAGGAAUAGCACCCACCAGGGCAAAGAGAAGAGUG

GUGGAGAGAGAAAAAAGAGCAGUGGGAAUAGGAGCUAUGAUCUUUGGGUUCU

UGGGAGCAGCAGGAAGCACUAUGGGCGCAGCGUCAAUAACGCUGACGGUACA

GGCCAGACAAUUAUUGUCUGGUAUAGUGCAACAGCAAAGCAAUUUGCUGAGG

GCUAUAGAGGCGCAGCAGCAUCUGUUGCAACUCACAGUCUGGGGCAUUAAAC

AGCUCCAGGCAAGAGUCCUGGCUGUGGAAAGAUACCUAAAUGAUCAAAAGUU

CCUAGGACUUUGGGGCUGCUCUGGAAAAAUCAUCUGCACCACUGCUGUGCCCU

GGAACUCCACUUGGAGUAAUAAAUCUUAUGAAGAGAUUUGGAACAACAUGAC

AUGGGUAGAAUGGGAGAGAGAAAUUAGCAAUUACACAAACCAAAUAUAUGAC

AUACUUACAGAAUCACAGAACCAGCAGGACAAAAAUGAAAAGGAUUUAUUGG

AAUUGGAUAAAUGGGCAAAUCUGUGGAAUUGGUUUGGCAUAACAAAGUGGCU

GUGGUAUAUAAAAAUAUUUAUAAUGAUAGUAGGAGGUUUAAUAGGUUUAAG

AAUAGUUUUUGCUGUGCUUUCUAUAGUAAAUAGAGUUAGGCAGGGAUACUCA

CCUUUGUCUUUCCAGACCCCCUCCCAUCACCAGAGGGAAACCGACAGACCCGA

AGGAAUCGAAGAAGAAGGUGGCGAGCAAGGCAGAGACAGAUCCGUGCGCUUA

GUGAGCGGAUUCUUAGCUCUUGUCUGGGACGAUUUACGGAGCCUGUGCCUCU

UCAGCUACCACCGCUUGAGAGACUUCAUCUUGAUUGCAGCGAGGACUCUGGA

AAUUCUGGGACACAGCAGUCUCAAGGGACUGAGACGGGGGUGGGAAGUCCUC

AAAUAUCUGGGGAAUCUUCUGUCAUAUUGGGGCCAGGAACUAAAAACUAGUG

CUAUUUCUUUGCUUAAUGCUACAGCAAUAGCAGUAGCGGGGUGGACAGACAG

GGUUAUAGAAGUAGCACAAAGAGCUUGGAGAGCUUUUCUCCACAUACCUAGG

AGAAUCAGACAGGGCUUUGAAAGGGCUUUGCUAUAACAUGGGAGGCAAGUGG

UCAAAAAGGAGCAGAGAGGGAUGGCCUCAGGUCAGGGAAAAAAUAAAGCAAA

CUCCUCCAGCAGCAGCAGAAGGAGUAGGAGCAGUAUCUCAAGAUCUAGAUAA

ACAUGGAGCAAUAACAAGUAGUAAUAUGAAUAAUGCUGAUUGUGUCUGGCUG

GAAGCACAAGAGGAUGAGGAGGUAGGCUUUCCAGUCACGCCGCAGGUACCUC

UAAGACCAAUGACUUUUAAGGGAGCUUUUGAUCUUAGCUUCUUUUUAAAAGA

AAAGGGGGGACUGGAUGGGCUAAUUUACUCCAAGAAAAGACAAGAGAUCCUU

GACUUAUGGGUCUAUAAUACACAAGGCUUCUUCCCUGAUUGGCAAAACUACA

CAGAGGGGCCAGGGACCAGAUUCCCACUGUGCUUUGGAUGGUGCUUCAAGCU

AGUACCAGUUGACCCAAGAGAAGUAGAGGAGAACAACAAAGGAGAAAACAGC

UGCCUGUUACACCCCAUGAGCCAGCAUGGAAUAGAGGACGAAGAUAGAGAAG

UGCUGAUGUGGAAGUUUGACAGCUCCCUAGCACGAAGACACAUAGCCCGAGA

AAAGCACCCAGAGUUCUAUAAAGACUGCUGACAAACAAGUUUCUAACUGGGA

CUUCCGCUGGGGACUUUCCAGGGGAGGUGUGGCCGGGGCGGAGUUGGGGAGU

GGUUAACCCUCAGAUGCUGCAUAAAAGCAGCCGCUUUUCGCUUGUACUGGGU

CUCUCUUGUUAGACCAGGUCGAGCCCGGGAGCUCUCUGGCUAGCAAGGGAACC

CACUGCUUAAAGCCUCAAUAAAGCUUGCCUUGAGUGC

Sample 7: SARS coronavirus

Obtained from the NCBI Databank

http://www.ncbi.nlm.nih.gov/nuccore/30271926?report=fasta

>gi|30271926|ref|NC_004718.3| SARS coronavirus, complete genome

AUAUUAGGUUUUUACCUACCCAGGAAAAGCCAACCAACCUCGAUCUCUUGUA

GAUCUGUUCUCUAAACGAACUUUAAAAUCUGUGUAGCUGUCGCUCGGCUGCA

UGCCUAGUGCACCUACGCAGUAUAAACAAUAAUAAAUUUUACUGUCGUUGAC

AAGAAACGAGUAACUCGUCCCUCUUCUGCAGACUGCUUACGGUUUCGUCCGUG

UUGCAGUCGAUCAUCAGCAUACCUAGGUUUCGUCCGGGUGUGACCGAAAGGU

AAGAUGGAGAGCCUUGUUCUUGGUGUCAACGAGAAAACACACGUCCAACUCA

GUUUGCCUGUCCUUCAGGUUAGAGACGUGCUAGUGCGUGGCUUCGGGGACUC

UGUGGAAGAGGCCCUAUCGGAGGCACGUGAACACCUCAAAAAUGGCACUUGU

GGUCUAGUAGAGCUGGAAAAAGGCGUACUGCCCCAGCUUGAACAGCCCUAUG

UGUUCAUUAAACGUUCUGAUGCCUUAAGCACCAAUCACGGCCACAAGGUCGU

UGAGCUGGUUGCAGAAAUGGACGGCAUUCAGUACGGUCGUAGCGGUAUAACA

CUGGGAGUACUCGUGCCACAUGUGGGCGAAACCCCAAUUGCAUACCGCAAUGU

UCUUCUUCGUAAGAACGGUAAUAAGGGAGCCGGUGGUCAUAGCUAUGGCAUC

GAUCUAAAGUCUUAUGACUUAGGUGACGAGCUUGGCACUGAUCCCAUUGAAG

AUUAUGAACAAAACUGGAACACUAAGCAUGGCAGUGGUGCACUCCGUGAACU

CACUCGUGAGCUCAAUGGAGGUGCAGUCACUCGCUAUGUCGACAACAAUUUC

UGUGGCCCAGAUGGGUACCCUCUUGAUUGCAUCAAAGAUUUUCUCGCACGCGC

GGGCAAGUCAAUGUGCACUCUUUCCGAACAACUUGAUUACAUCGAGUCGAAG

AGAGGUGUCUACUGCUGCCGUGACCAUGAGCAUGAAAUUGCCUGGUUCACUG

AGCGCUCUGAUAAGAGCUACGAGCACCAGACACCCUUCGAAAUUAAGAGUGCC

AAGAAAUUUGACACUUUCAAAGGGGAAUGCCCAAAGUUUGUGUUUCCUCUUA

ACUCAAAAGUCAAAGUCAUUCAACCACGUGUUGAAAAGAAAAAGACUGAGGG

UUUCAUGGGGCGUAUACGCUCUGUGUACCCUGUUGCAUCUCCACAGGAGUGU

AACAAUAUGCACUUGUCUACCUUGAUGAAAUGUAAUCAUUGCGAUGAAGUUU

CAUGGCAGACGUGCGACUUUCUGAAAGCCACUUGUGAACAUUGUGGCACUGA

AAAUUUAGUUAUUGAAGGACCUACUACAUGUGGGUACCUACCUACUAAUGCU

GUAGUGAAAAUGCCAUGUCCUGCCUGUCAAGACCCAGAGAUUGGACCUGAGC

AUAGUGUUGCAGAUUAUCACAACCACUCAAACAUUGAAACUCGACUCCGCAA

GGGAGGUAGGACUAGAUGUUUUGGAGGCUGUGUGUUUGCCUAUGUUGGCUGC

UAUAAUAAGCGUGCCUACUGGGUUCCUCGUGCUAGUGCUGAUAUUGGCUCAG

GCCAUACUGGCAUUACUGGUGACAAUGUGGAGACCUUGAAUGAGGAUCUCCU

UGAGAUACUGAGUCGUGAACGUGUUAACAUUAACAUUGUUGGCGAUUUUCAU

UUGAAUGAAGAGGUUGCCAUCAUUUUGGCAUCUUUCUCUGCUUCUACAAGUG

CCUUUAUUGACACUAUAAAGAGUCUUGAUUACAAGUCUUUCAAAACCAUUGU

UGAGUCCUGCGGUAACUAUAAAGUUACCAAGGGAAAGCCCGUAAAAGGUGCU

UGGAACAUUGGACAACAGAGAUCAGUUUUAACACCACUGUGUGGUUUUCCCU

CACAGGCUGCUGGUGUUAUCAGAUCAAUUUUUGCGCGCACACUUGAUGCAGC

AAACCACUCAAUUCCUGAUUUGCAAAGAGCAGCUGUCACCAUACUUGAUGGU

AUUUCUGAACAGUCAUUACGUCUUGUCGACGCCAUGGUUUAUACUUCAGACC

UGCUCACCAACAGUGUCAUUAUUAUGGCAUAUGUAACUGGUGGUCUUGUACA

ACAGACUUCUCAGUGGUUGUCUAAUCUUUUGGGCACUACUGUUGAAAAACUC

AGGCCUAUCUUUGAAUGGAUUGAGGCGAAACUUAGUGCAGGAGUUGAAUUUC

UCAAGGAUGCUUGGGAGAUUCUCAAAUUUCUCAUUACAGGUGUUUUUGACAU

CGUCAAGGGUCAAAUACAGGUUGCUUCAGAUAACAUCAAGGAUUGUGUAAAA

UGCUUCAUUGAUGUUGUUAACAAGGCACUCGAAAUGUGCAUUGAUCAAGUCA

CUAUCGCUGGCGCAAAGUUGCGAUCACUCAACUUAGGUGAAGUCUUCAUCGC

UCAAAGCAAGGGACUUUACCGUCAGUGUAUACGUGGCAAGGAGCAGCUGCAA

CUACUCAUGCCUCUUAAGGCACCAAAAGAAGUAACCUUUCUUGAAGGUGAUU

CACAUGACACAGUACUUACCUCUGAGGAGGUUGUUCUCAAGAACGGUGAACU

CGAAGCACUCGAGACGCCCGUUGAUAGCUUCACAAAUGGAGCUAUCGUUGGC

ACACCAGUCUGUGUAAAUGGCCUCAUGCUCUUAGAGAUUAAGGACAAAGAAC

AAUACUGCGCAUUGUCUCCUGGUUUACUGGCUACAAACAAUGUCUUUCGCUU

AAAAGGGGGUGCACCAAUUAAAGGUGUAACCUUUGGAGAAGAUACUGUUUGG

GAAGUUCAAGGUUACAAGAAUGUGAGAAUCACAUUUGAGCUUGAUGAACGUG

UUGACAAAGUGCUUAAUGAAAAGUGCUCUGUCUACACUGUUGAAUCCGGUAC

CGAAGUUACUGAGUUUGCAUGUGUUGUAGCAGAGGCUGUUGUGAAGACUUUA

CAACCAGUUUCUGAUCUCCUUACCAACAUGGGUAUUGAUCUUGAUGAGUGGA

GUGUAGCUACAUUCUACUUAUUUGAUGAUGCUGGUGAAGAAAACUUUUCAUC

ACGUAUGUAUUGUUCCUUUUACCCUCCAGAUGAGGAAGAAGAGGACGAUGCA

GAGUGUGAGGAAGAAGAAAUUGAUGAAACCUGUGAACAUGAGUACGGUACAG

AGGAUGAUUAUCAAGGUCUCCCUCUGGAAUUUGGUGCCUCAGCUGAAACAGU

UCGAGUUGAGGAAGAAGAAGAGGAAGACUGGCUGGAUGAUACUACUGAGCAA

UCAGAGAUUGAGCCAGAACCAGAACCUACACCUGAAGAACCAGUUAAUCAGU

UUACUGGUUAUUUAAAACUUACUGACAAUGUUGCCAUUAAAUGUGUUGACAU

CGUUAAGGAGGCACAAAGUGCUAAUCCUAUGGUGAUUGUAAAUGCUGCUAAC

AUACACCUGAAACAUGGUGGUGGUGUAGCAGGUGCACUCAACAAGGCAACCA

AUGGUGCCAUGCAAAAGGAGAGUGAUGAUUACAUUAAGCUAAAUGGCCCUCU

UACAGUAGGAGGGUCUUGUUUGCUUUCUGGACAUAAUCUUGCUAAGAAGUGU

CUGCAUGUUGUUGGACCUAACCUAAAUGCAGGUGAGGACAUCCAGCUUCUUA

AGGCAGCAUAUGAAAAUUUCAAUUCACAGGACAUCUUACUUGCACCAUUGUU

GUCAGCAGGCAUAUUUGGUGCUAAACCACUUCAGUCUUUACAAGUGUGCGUG

CAGACGGUUCGUACACAGGUUUAUAUUGCAGUCAAUGACAAAGCUCUUUAUG

AGCAGGUUGUCAUGGAUUAUCUUGAUAACCUGAAGCCUAGAGUGGAAGCACC

UAAACAAGAGGAGCCACCAAACACAGAAGAUUCCAAAACUGAGGAGAAAUCU

GUCGUACAGAAGCCUGUCGAUGUGAAGCCAAAAAUUAAGGCCUGCAUUGAUG

AGGUUACCACAACACUGGAAGAAACUAAGUUUCUUACCAAUAAGUUACUCUU

GUUUGCUGAUAUCAAUGGUAAGCUUUACCAUGAUUCUCAGAACAUGCUUAGA

GGUGAAGAUAUGUCUUUCCUUGAGAAGGAUGCACCUUACAUGGUAGGUGAUG

UUAUCACUAGUGGUGAUAUCACUUGUGUUGUAAUACCCUCCAAAAAGGCUGG

UGGCACUACUGAGAUGCUCUCAAGAGCUUUGAAGAAAGUGCCAGUUGAUGAG

UAUAUAACCACGUACCCUGGACAAGGAUGUGCUGGUUAUACACUUGAGGAAG

CUAAGACUGCUCUUAAGAAAUGCAAAUCUGCAUUUUAUGUACUACCUUCAGA

AGCACCUAAUGCUAAGGAAGAGAUUCUAGGAACUGUAUCCUGGAAUUUGAGA

GAAAUGCUUGCUCAUGCUGAAGAGACAAGAAAAUUAAUGCCUAUAUGCAUGG

AUGUUAGAGCCAUAAUGGCAACCAUCCAACGUAAGUAUAAAGGAAUUAAAAU

UCAAGAGGGCAUCGUUGACUAUGGUGUCCGAUUCUUCUUUUAUACUAGUAAA

GAGCCUGUAGCUUCUAUUAUUACGAAGCUGAACUCUCUAAAUGAGCCGCUUG

UCACAAUGCCAAUUGGUUAUGUGACACAUGGUUUUAAUCUUGAAGAGGCUGC

GCGCUGUAUGCGUUCUCUUAAAGCUCCUGCCGUAGUGUCAGUAUCAUCACCAG

AUGCUGUUACUACAUAUAAUGGAUACCUCACUUCGUCAUCAAAGACAUCUGA

GGAGCACUUUGUAGAAACAGUUUCUUUGGCUGGCUCUUACAGAGAUUGGUCC

UAUUCAGGACAGCGUACAGAGUUAGGUGUUGAAUUUCUUAAGCGUGGUGACA

AAAUUGUGUACCACACUCUGGAGAGCCCCGUCGAGUUUCAUCUUGACGGUGA

GGUUCUUUCACUUGACAAACUAAAGAGUCUCUUAUCCCUGCGGGAGGUUAAG

ACUAUAAAAGUGUUCACAACUGUGGACAACACUAAUCUCCACACACAGCUUG

UGGAUAUGUCUAUGACAUAUGGACAGCAGUUUGGUCCAACAUACUUGGAUGG

UGCUGAUGUUACAAAAAUUAAACCUCAUGUAAAUCAUGAGGGUAAGACUUUC

UUUGUACUACCUAGUGAUGACACACUACGUAGUGAAGCUUUCGAGUACUACC

AUACUCUUGAUGAGAGUUUUCUUGGUAGGUACAUGUCUGCUUUAAACCACAC

AAAGAAAUGGAAAUUUCCUCAAGUUGGUGGUUUAACUUCAAUUAAAUGGGCU

GAUAACAAUUGUUAUUUGUCUAGUGUUUUAUUAGCACUUCAACAGCUUGAAG

UCAAAUUCAAUGCACCAGCACUUCAAGAGGCUUAUUAUAGAGCCCGUGCUGG

UGAUGCUGCUAACUUUUGUGCACUCAUACUCGCUUACAGUAAUAAAACUGUU

GGCGAGCUUGGUGAUGUCAGAGAAACUAUGACCCAUCUUCUACAGCAUGCUA

AUUUGGAAUCUGCAAAGCGAGUUCUUAAUGUGGUGUGUAAACAUUGUGGUCA

GAAAACUACUACCUUAACGGGUGUAGAAGCUGUGAUGUAUAUGGGUACUCUA

UCUUAUGAUAAUCUUAAGACAGGUGUUUCCAUUCCAUGUGUGUGUGGUCGUG

AUGCUACACAAUAUCUAGUACAACAAGAGUCUUCUUUUGUUAUGAUGUCUGC

ACCACCUGCUGAGUAUAAAUUACAGCAAGGUACAUUCUUAUGUGCGAAUGAG

UACACUGGUAACUAUCAGUGUGGUCAUUACACUCAUAUAACUGCUAAGGAGA

CCCUCUAUCGUAUUGACGGAGCUCACCUUACAAAGAUGUCAGAGUACAAAGG

ACCAGUGACUGAUGUUUUCUACAAGGAAACAUCUUACACUACAACCAUCAAG

CCUGUGUCGUAUAAACUCGAUGGAGUUACUUACACAGAGAUUGAACCAAAAU

UGGAUGGGUAUUAUAAAAAGGAUAAUGCUUACUAUACAGAGCAGCCUAUAGA

CCUUGUACCAACUCAACCAUUACCAAAUGCGAGUUUUGAUAAUUUCAAACUC

ACAUGUUCUAACACAAAAUUUGCUGAUGAUUUAAAUCAAAUGACAGGCUUCA

CAAAGCCAGCUUCACGAGAGCUAUCUGUCACAUUCUUCCCAGACUUGAAUGGC

GAUGUAGUGGCUAUUGACUAUAGACACUAUUCAGCGAGUUUCAAGAAAGGUG

CUAAAUUACUGCAUAAGCCAAUUGUUUGGCACAUUAACCAGGCUACAACCAA

GACAACGUUCAAACCAAACACUUGGUGUUUACGUUGUCUUUGGAGUACAAAG

CCAGUAGAUACUUCAAAUUCAUUUGAAGUUCUGGCAGUAGAAGACACACAAG

GAAUGGACAAUCUUGCUUGUGAAAGUCAACAACCCACCUCUGAAGAAGUAGU

GGAAAAUCCUACCAUACAGAAGGAAGUCAUAGAGUGUGACGUGAAAACUACC

GAAGUUGUAGGCAAUGUCAUACUUAAACCAUCAGAUGAAGGUGUUAAAGUAA

CACAAGAGUUAGGUCAUGAGGAUCUUAUGGCUGCUUAUGUGGAAAACACAAG

CAUUACCAUUAAGAAACCUAAUGAGCUUUCACUAGCCUUAGGUUUAAAAACA

AUUGCCACUCAUGGUAUUGCUGCAAUUAAUAGUGUUCCUUGGAGUAAAAUUU

UGGCUUAUGUCAAACCAUUCUUAGGACAAGCAGCAAUUACAACAUCAAAUUG

CGCUAAGAGAUUAGCACAACGUGUGUUUAACAAUUAUAUGCCUUAUGUGUUU

ACAUUAUUGUUCCAAUUGUGUACUUUUACUAAAAGUACCAAUUCUAGAAUUA

GAGCUUCACUACCUACAACUAUUGCUAAAAAUAGUGUUAAGAGUGUUGCUAA

AUUAUGUUUGGAUGCCGGCAUUAAUUAUGUGAAGUCACCCAAAUUUUCUAAA

UUGUUCACAAUCGCUAUGUGGCUAUUGUUGUUAAGUAUUUGCUUAGGUUCUC

UAAUCUGUGUAACUGCUGCUUUUGGUGUACUCUUAUCUAAUUUUGGUGCUCC

UUCUUAUUGUAAUGGCGUUAGAGAAUUGUAUCUUAAUUCGUCUAACGUUACU

ACUAUGGAUUUCUGUGAAGGUUCUUUUCCUUGCAGCAUUUGUUUAAGUGGAU

UAGACUCCCUUGAUUCUUAUCCAGCUCUUGAAACCAUUCAGGUGACGAUUUC

AUCGUACAAGCUAGACUUGACAAUUUUAGGUCUGGCCGCUGAGUGGGUUUUG

GCAUAUAUGUUGUUCACAAAAUUCUUUUAUUUAUUAGGUCUUUCAGCUAUAA

UGCAGGUGUUCUUUGGCUAUUUUGCUAGUCAUUUCAUCAGCAAUUCUUGGCU

CAUGUGGUUUAUCAUUAGUAUUGUACAAAUGGCACCCGUUUCUGCAAUGGUU

AGGAUGUACAUCUUCUUUGCUUCUUUCUACUACAUAUGGAAGAGCUAUGUUC

AUAUCAUGGAUGGUUGCACCUCUUCGACUUGCAUGAUGUGCUAUAAGCGCAA

UCGUGCCACACGCGUUGAGUGUACAACUAUUGUUAAUGGCAUGAAGAGAUCU

UUCUAUGUCUAUGCAAAUGGAGGCCGUGGCUUCUGCAAGACUCACAAUUGGA

AUUGUCUCAAUUGUGACACAUUUUGCACUGGUAGUACAUUCAUUAGUGAUGA

AGUUGCUCGUGAUUUGUCACUCCAGUUUAAAAGACCAAUCAACCCUACUGACC

AGUCAUCGUAUAUUGUUGAUAGUGUUGCUGUGAAAAAUGGCGCGCUUCACCU

CUACUUUGACAAGGCUGGUCAAAAGACCUAUGAGAGACAUCCGCUCUCCCAUU

UUGUCAAUUUAGACAAUUUGAGAGCUAACAACACUAAAGGUUCACUGCCUAU

UAAUGUCAUAGUUUUUGAUGGCAAGUCCAAAUGCGACGAGUCUGCUUCUAAG

UCUGCUUCUGUGUACUACAGUCAGCUGAUGUGCCAACCUAUUCUGUUGCUUG

ACCAAGCUCUUGUAUCAGACGUUGGAGAUAGUACUGAAGUUUCCGUUAAGAU

GUUUGAUGCUUAUGUCGACACCUUUUCAGCAACUUUUAGUGUUCCUAUGGAA

AAACUUAAGGCACUUGUUGCUACAGCUCACAGCGAGUUAGCAAAGGGUGUAG

CUUUAGAUGGUGUCCUUUCUACAUUCGUGUCAGCUGCCCGACAAGGUGUUGU

UGAUACCGAUGUUGACACAAAGGAUGUUAUUGAAUGUCUCAAACUUUCACAU

CACUCUGACUUAGAAGUGACAGGUGACAGUUGUAACAAUUUCAUGCUCACCU

AUAAUAAGGUUGAAAACAUGACGCCCAGAGAUCUUGGCGCAUGUAUUGACUG

UAAUGCAAGGCAUAUCAAUGCCCAAGUAGCAAAAAGUCACAAUGUUUCACUC

AUCUGGAAUGUAAAAGACUACAUGUCUUUAUCUGAACAGCUGCGUAAACAAA

UUCGUAGUGCUGCCAAGAAGAACAACAUACCUUUUAGACUAACUUGUGCUAC

AACUAGACAGGUUGUCAAUGUCAUAACUACUAAAAUCUCACUCAAGGGUGGU

AAGAUUGUUAGUACUUGUUUUAAACUUAUGCUUAAGGCCACAUUAUUGUGCG

UUCUUGCUGCAUUGGUUUGUUAUAUCGUUAUGCCAGUACAUACAUUGUCAAU

CCAUGAUGGUUACACAAAUGAAAUCAUUGGUUACAAAGCCAUUCAGGAUGGU

GUCACUCGUGACAUCAUUUCUACUGAUGAUUGUUUUGCAAAUAAACAUGCUG

GUUUUGACGCAUGGUUUAGCCAGCGUGGUGGUUCAUACAAAAAUGACAAAAG

CUGCCCUGUAGUAGCUGCUAUCAUUACAAGAGAGAUUGGUUUCAUAGUGCCU

GGCUUACCGGGUACUGUGCUGAGAGCAAUCAAUGGUGACUUCUUGCAUUUUC

UACCUCGUGUUUUUAGUGCUGUUGGCAACAUUUGCUACACACCUUCCAAACUC

AUUGAGUAUAGUGAUUUUGCUACCUCUGCUUGCGUUCUUGCUGCUGAGUGUA

CAAUUUUUAAGGAUGCUAUGGGCAAACCUGUGCCAUAUUGUUAUGACACUAA

UUUGCUAGAGGGUUCUAUUUCUUAUAGUGAGCUUCGUCCAGACACUCGUUAU

GUGCUUAUGGAUGGUUCCAUCAUACAGUUUCCUAACACUUACCUGGAGGGUU

CUGUUAGAGUAGUAACAACUUUUGAUGCUGAGUACUGUAGACAUGGUACAUG

CGAAAGGUCAGAAGUAGGUAUUUGCCUAUCUACCAGUGGUAGAUGGGUUCUU

AAUAAUGAGCAUUACAGAGCUCUAUCAGGAGUUUUCUGUGGUGUUGAUGCGA

UGAAUCUCAUAGCUAACAUCUUUACUCCUCUUGUGCAACCUGUGGGUGCUUU

AGAUGUGUCUGCUUCAGUAGUGGCUGGUGGUAUUAUUGCCAUAUUGGUGACU

UGUGCUGCCUACUACUUUAUGAAAUUCAGACGUGUUUUUGGUGAGUACAACC

AUGUUGUUGCUGCUAAUGCACUUUUGUUUUUGAUGUCUUUCACUAUACUCUG

UCUGGUACCAGCUUACAGCUUUCUGCCGGGAGUCUACUCAGUCUUUUACUUG

UACUUGACAUUCUAUUUCACCAAUGAUGUUUCAUUCUUGGCUCACCUUCAAU

GGUUUGCCAUGUUUUCUCCUAUUGUGCCUUUUUGGAUAACAGCAAUCUAUGU

AUUCUGUAUUUCUCUGAAGCACUGCCAUUGGUUCUUUAACAACUAUCUUAGG

AAAAGAGUCAUGUUUAAUGGAGUUACAUUUAGUACCUUCGAGGAGGCUGCUU

UGUGUACCUUUUUGCUCAACAAGGAAAUGUACCUAAAAUUGCGUAGCGAGAC

ACUGUUGCCACUUACACAGUAUAACAGGUAUCUUGCUCUAUAUAACAAGUAC

AAGUAUUUCAGUGGAGCCUUAGAUACUACCAGCUAUCGUGAAGCAGCUUGCU

GCCACUUAGCAAAGGCUCUAAAUGACUUUAGCAACUCAGGUGCUGAUGUUCU

CUACCAACCACCACAGACAUCAAUCACUUCUGCUGUUCUGCAGAGUGGUUUUA

GGAAAAUGGCAUUCCCGUCAGGCAAAGUUGAAGGGUGCAUGGUACAAGUAAC

CUGUGGAACUACAACUCUUAAUGGAUUGUGGUUGGAUGACACAGUAUACUGU

CCAAGACAUGUCAUUUGCACAGCAGAAGACAUGCUUAAUCCUAACUAUGAAG

AUCUGCUCAUUCGCAAAUCCAACCAUAGCUUUCUUGUUCAGGCUGGCAAUGU

UCAACUUCGUGUUAUUGGCCAUUCUAUGCAAAAUUGUCUGCUUAGGCUUAAA

GUUGAUACUUCUAACCCUAAGACACCCAAGUAUAAAUUUGUCCGUAUCCAACC

UGGUCAAACAUUUUCAGUUCUAGCAUGCUACAAUGGUUCACCAUCUGGUGUU

UAUCAGUGUGCCAUGAGACCUAAUCAUACCAUUAAAGGUUCUUUCCUUAAUG

GAUCAUGUGGUAGUGUUGGUUUUAACAUUGAUUAUGAUUGCGUGUCUUUCUG

CUAUAUGCAUCAUAUGGAGCUUCCAACAGGAGUACACGCUGGUACUGACUUA

GAAGGUAAAUUCUAUGGUCCAUUUGUUGACAGACAAACUGCACAGGCUGCAG

GUACAGACACAACCAUAACAUUAAAUGUUUUGGCAUGGCUGUAUGCUGCUGU

UAUCAAUGGUGAUAGGUGGUUUCUUAAUAGAUUCACCACUACUUUGAAUGAC

UUUAACCUUGUGGCAAUGAAGUACAACUAUGAACCUUUGACACAAGAUCAUG

UUGACAUAUUGGGACCUCUUUCUGCUCAAACAGGAAUUGCCGUCUUAGAUAU

GUGUGCUGCUUUGAAAGAGCUGCUGCAGAAUGGUAUGAAUGGUCGUACUAUC

CUUGGUAGCACUAUUUUAGAAGAUGAGUUUACACCAUUUGAUGUUGUUAGAC

AAUGCUCUGGUGUUACCUUCCAAGGUAAGUUCAAGAAAAUUGUUAAGGGCAC

UCAUCAUUGGAUGCUUUUAACUUUCUUGACAUCACUAUUGAUUCUUGUUCAA

AGUACACAGUGGUCACUGUUUUUCUUUGUUUACGAGAAUGCUUUCUUGCCAU

UUACUCUUGGUAUUAUGGCAAUUGCUGCAUGUGCUAUGCUGCUUGUUAAGCA

UAAGCACGCAUUCUUGUGCUUGUUUCUGUUACCUUCUCUUGCAACAGUUGCU

UACUUUAAUAUGGUCUACAUGCCUGCUAGCUGGGUGAUGCGUAUCAUGACAU

GGCUUGAAUUGGCUGACACUAGCUUGUCUGGUUAUAGGCUUAAGGAUUGUGU

UAUGUAUGCUUCAGCUUUAGUUUUGCUUAUUCUCAUGACAGCUCGCACUGUU

UAUGAUGAUGCUGCUAGACGUGUUUGGACACUGAUGAAUGUCAUUACACUUG

UUUACAAAGUCUACUAUGGUAAUGCUUUAGAUCAAGCUAUUUCCAUGUGGGC

CUUAGUUAUUUCUGUAACCUCUAACUAUUCUGGUGUCGUUACGACUAUCAUG

UUUUUAGCUAGAGCUAUAGUGUUUGUGUGUUGAGUAUUACCCAUUGUUAU

UUAUUACUGGCAACACCUUACAGUGUAUCAUGCUUGUUUAUUGUUUCUUAGG

CUAUUGUUGCUGCUGCUACUUUGGCCUUUUCUGUUUACUCAACCGUUACUUC

AGGCUUACUCUUGGUGUUUAUGACUACUUGGUCUCUACACAAGAAUUUAGGU

AUAUGAACUCCCAGGGGCUUUUGCCUCCUAAGAGUAGUAUUGAUGCUUUCAA

GCUUAACAUUAAGUUGUUGGGUAUUGGAGGUAAACCAUGUAUCAAGGUUGCU

ACUGUACAGUCUAAAAUGUCUGACGUAAAGUGCACAUCUGUGGUACUGCUCU

CGGUUCUUCAACAACUUAGAGUAGAGUCAUCUUCUAAAUUGUGGGCACAAUG

UGUACAACUCCACAAUGAUAUUCUUCUUGCAAAAGACACAACUGAAGCUUUC

GAGAAGAUGGUUUCUCUUUUGUCUGUUUUGCUAUCCAUGCAGGGUGCUGUAG

ACAUUAAUAGGUUGUGCGAGGAAAUGCUCGAUAACCGUGCUACUCUUCAGGC

UAUUGCUUCAGAAUUUAGUUCUUUACCAUCAUAUGCCGCUUAUGCCACUGCCC

AGGAGGCCUAUGAGCAGGCUGUAGCUAAUGGUGAUUCUGAAGUCGUUCUCAA

AAAGUUAAAGAAAUCUUUGAAUGUGGCUAAAUCUGAGUUUGACCGUGAUGCU

GCCAUGCAACGCAAGUUGGAAAAGAUGGCAGAUCAGGCUAUGACCCAAAUGU

ACAAACAGGCAAGAUCUGAGGACAAGAGGGCAAAAGUAACUAGUGCUAUGCA

AACAAUGCUCUUCACUAUGCUUAGGAAGCUUGAUAAUGAUGCACUUAACAAC

AUUAUCAACAAUGCGCGUGAUGGUUGUGUUCCACUCAACAUCAUACCAUUGA

CUACAGCAGCCAAACUCAUGGUUGUUGUCCCUGAUUAUGGUACCUACAAGAA

CACUUGUGAUGGUAACACCUUUACAUAUGCAUCUGCACUCUGGGAAAUCCAG

CAAGUUGUUGAUGCGGAUAGCAAGAUUGUUCAACUUAGUGAAAUUAACAUGG

ACAAUUCACCAAAUUUGGCUUGGCCUCUUAUUGUUACAGCUCUAAGAGCCAA

CUCAGCUGUUAAACUACAGAAUAAUGAACUGAGUCCAGUAGCACUACGACAG

AUGUCCUGUGCGGCUGGUACCACACAAACAGCUUGUACUGAUGACAAUGCAC

UUGCCUACUAUAACAAUUCGAAGGGAGGUAGGUUUGUGCUGGCAUUACUAUC

AGACCACCAAGAUCUCAAAUGGGCUAGAUUCCCUAAGAGUGAUGGUACAGGU

ACAAUUUACACAGAACUGGAACCACCUUGUAGGUUUGUUACAGACACACCAA

AAGGGCCUAAAGUGAAAUACUUGUACUUCAUCAAAGGCUUAAACAACCUAAA

UAGAGGUAUGGUGCUGGGCAGUUUAGCUGCUACAGUACGUCUUCAGGCUGGA

AAUGCUACAGAAGUACCUGCCAAUUCAACUGUGCUUUCCUUCUGUGCUUUUG

CAGUAGACCCUGCUAAAGCAUAUAAGGAUUACCUAGCAAGUGGAGGACAACC

AAUCACCAACUGUGUGAAGAUGUUGUGUACACACACUGGUACAGGACAGGCA

AUUACUGUAACACCAGAAGCUAACAUGGACCAAGAGUCCUUUGGUGGUGCUU

CAUGUUGUCUGUAUUGUAGAUGCCACAUUGACCAUCCAAAUCCUAAAGGAUU

CUGUGACUUGAAAGGUAAGUACGUCCAAAUACCUACCACUUGUGCUAAUGAC

CCAGUGGGUUUUACACUUAGAAACACAGUCUGUACCGUCUGCGGAAUGUGGA

AAGGUUAUGGCUGUAGUUGUGACCAACUCCGCGAACCCUUGAUGCAGUCUGC

GGAUGCAUCAACGUUUUUAAACGGGUUUGCGGUGUAAGUGCAGCCCGUCUUA

CACCGUGCGGCACAGGCACUAGUACUGAUGUCGUCUACAGGGCUUUUGAUAU

UUACAACGAAAAAGUUGCUGGUUUUGCAAAGUUCCUAAAAACUAAUUGCUGU

CGCUUCCAGGAGAAGGAUGAGGAAGGCAAUUUAUUAGACUCUUACUUUGUAG

UUAAGAGGCAUACUAUGUCUAACUACCAACAUGAAGAGACUAUUUAUAACUU

GGUUAAAGAUUGUCCAGCGGUUGCUGUCCAUGACUUUUUCAAGUUUAGAGUA

GAUGGUGACAUGGUACCACAUAUAUCACGUCAGCGUCUAACUAAAUACACAA

UGGCUGAUUUAGUCUAUGCUCUACGUCAUUUUGAUGAGGGUAAUUGUGAUAC

AUUAAAAGAAAUACUCGUCACAUACAAUUGCUGUGAUGAUGAUUAUUUCAAU

AAGAAGGAUUGGUAUGACUUCGUAGAGAAUCCUGACAUCUUACGCGUAUAUG

CUAACUUAGGUGAGCGUGUACGCCAAUCAUUAUUAAAGACUGUACAAUUCUG

CGAUGCUAUGCGUGAUGCAGGCAUUGUAGGCGUACUGACAUUAGAUAAUCAG

GAUCUUAAUGGGAACUGGUACGAUUUCGGUGAUUUCGUACAAGUAGCACCAG

GCUGCGGAGUUCCUAUUGUGGAUUCAUAUUACUCAUUGCUGAUGCCCAUCCU

CACUUUGACUAGGGCAUUGGCUGCUGAGUCCCAUAUGGAUGCUGAUCUCGCA

AAACCACUUAUUAAGUGGGAUUUGCUGAAAUAUGAUUUUACGGAAGAGAGAC

UUUGUCUCUUCGACCGUUAUUUUAAAUAUUGGGACCAGACAUACCAUCCCAA

UUGUAUUAACUGUUUGGAUGAUAGGUGUAUCCUUCAUUGUGCAAACUUUAAU

GUGUUAUUUUCUACUGUGUUUCCACCUACAAGUUUUGGACCACUAGUAAGAA

AAAUAUUUGUAGAUGGUGUUCCUUUUGUUGUUUCAACUGGAUACCAUUUUCG

UGAGUUAGGAGUCGUACAUAAUCAGGAUGUAAACUUACAUAGCUCGCGUCUC

AGUUUCAAGGAACUUUUAGUGUAUGCUGCUGAUCCAGCUAUGCAUGCAGCUU

CUGGCAAUUUAUUGCUAGAUAAACGCACUACAUGCUUUUCAGUAGCUGCACU

AACAAACAAUGUUGCUUUUCAAACUGUCAAACCCGGUAAUUUUAAUAAAGAC

UUUUAUGACUUUGCUGUGUCUAAAGGUUUCUUUAAGGAAGGAAGUUCUGUUG

AACUAAAACACUUCUUCUUUGCUCAGGAUGGCAACGCUGCUAUCAGUGAUUA

UGACUAUUAUCGUUAUAAUCUGCCAACAAUGUGUGAUAUCAGACAACUCCUA

UUCGUAGUUGAAGUUGUUGAUAAAUACUUUGAUUGUUACGAUGGUGGCUGUA

UUAAUGCCAACCAAGUAAUCGUUAACAAUCUGGAUAAAUCAGCUGGUUUCCC

AUUUAAUAAAUGGGGUAAGGCUAGACUUUAUUAUGACUCAAUGAGUUAUGAG

GAUCAAGAUGCACUUUUCGCGUAUACUAAGCGUAAUGUCAUCCCUACUAUAA

CUCAAAUGAAUCUUAAGUAUGCCAUUAGUGCAAAGAAUAGAGCUCGCACCGU

AGCUGGUGUCUCUAUCUGUAGUACUAUGACAAAUAGACAGUUUCAUCAGAAA

UUAUUGAAGUCAAUAGCCGCCACUAGAGGAGCUACUGUGGUAAUUGGAACAA

GCAAGUUUUACGGUGGCUGGCAUAAUAUGUUAAAAACUGUUUACAGUGAUGU

AGAAACUCCACACCUUAUGGGUUGGGAUUAUCCAAAAUGUGACAGAGCCAUG

CCUAACAUGCUUAGGAUAAUGGCCUCUCUUGUUCUUGCUCGCAAACAUAACAC

UUGCUGUAACUUAUCACACCGUUUCUACAGGUUAGCUAACGAGUGUGCGCAA

GUAUUAAGUGAGAUGGUCAUGUGUGGCGGCUCACUAUAUGUUAAACCAGGUG

GAACAUCAUCCGGUGAUGCUACAACUGCUUAUGCUAAUAGUGUCUUUAACAU

UUGUCAAGCUGUUACAGCCAAUGUAAAUGCACUUCUUUCAACUGAUGGUAAU

AAGAUAGCUGACAAGUAUGUCCGCAAUCUACAACACAGGCUCUAUGAGUGUC

UCUAUAGAAAUAGGGAUGUUGAUCAUGAAUUCGUGGAUGAGUUUUACGCUUA

CCUGCGUAAACAUUUCUCCAUGAUGAUUCUUUCUGAUGAUGCCGUUGUGUGC

UAUAACAGUAACUAUGCGGCUCAAGGUUUAGUAGCUAGCAUUAAGAACUUUA

AGGCAGUUCUUUAUUAUCAAAAUAAUGUGUUCAUGUCUGAGGCAAAAUGUUG

GACUGAGACUGACCUUACUAAAGGACCUCACGAAUUUUGCUCACAGCAUACA

AUGCUAGUUAAACAAGGAGAUGAUUACGUGUACCUGCCUUACCCAGAUCCAU

CAAGAAUAUUAGGCGCAGGCUGUUUUGUCGAUGAUAUUGUCAAAACAGAUGG

UACACUUAUGAUUGAAAGGUUCGUGUCACUGGCUAUUGAUGCUUACCCACUU

ACAAAACAUCCUAAUCAGGAGUAUGCUGAUGUCUUUCACUUGUAUUUACAAU

ACAUUAGAAAGUUACAUGAUGAGCUUACUGGCCACAUGUUGGACAUGUAUUC

CGUAAUGCUAACUAAUGAUAACACCUCACGGUACUGGGAACCUGAGUUUUAU

GAGGCUAUGUACACACCACAUACAGUCUUGCAGGCUGUAGGUGCUUGUGUAU

UGUGCAAUUCACAGACUUCACUUCGUUGCGGUGCCUGUAUUAGGAGACCAUU

CCUAUGUUGCAAGUGCUGCUAUGACCAUGUCAUUUCAACAUCACACAAAUUA

GUGUUGUCUGUUAAUCCCUAUGUUUGCAAUGCCCCAGGUUGUGAUGUCACUG

AUGUGACACAACUGUAUCUAGGAGGUAUGAGCUAUUAUUGCAAGUCACAUAA

GCCUCCCAUUAGUUUUCCAUUAUGUGCUAAUGGUCAGGUUUUUGGUUUAUAC

AAAAACACAUGUGUAGGCAGUGACAAUGUCACUGACUUCAAUGCGAUAGCAA

CAUGUGAUUGGACUAAUGCUGGCGAUUACAUACUUGCCAACACUUGUACUGA

GAGACUCAAGCUUUUCGCAGCAGAAACGCUCAAAGCCACUGAGGAAACAUUU

AAGCUGUCAUAUGGUAUUGCCACUGUACGCGAAGUACUCUCUGACAGAGAAU

UGCAUCUUUCAUGGGAGGUUGGAAAACCUAGACCACCAUUGAACAGAAACUA

UGUCUUUACUGGUUACCGUGUAACUAAAAAUAGUAAAGUACAGAUUGGAGAG

UACACCUUUGAAAAAGGUGACUAUGGUGAUGCUGUUGUGUACAGAGGUACUA

CGACAUACAAGUUGAAUGUUGGUGAUUACUUUGUGUUGACAUCUCACACUGU

AAUGCCACUUAGUGCACCUACUCUAGUGCCACAAGAGCACUAUGUGAGAAUU

ACUGGCUUGUACCCAACACUCAACAUCUCAGAUGAGUUUUCUAGCAAUGUUG

CAAAUUAUCAAAAGGUCGGCAUGCAAAAGUACUCUACACUCCAAGGACCACCU

GGUACUGGUAAGAGUCAUUUUGCCAUCGGACUUGCUCUCUAUUACCCAUCUG

CUCGCAUAGUGUAUACGGCAUGCUCUCAUGCAGCUGUUGAUGCCCUAUGUGA

AAAGGCAUUAAAAUAUUUGCCCAUAGAUAAAUGUAGUAGAAUCAUACCUGCG

CGUGCGCGCGUAGAGUGUUUUGAUAAAUUCAAAGUGAAUUCAACACUAGAAC

AGUAUGUUUUCUGCACUGUAAAUGCAUUGCCAGAAACAACUGCUGACAUUGU

AGUCUUUGAUGAAAUCUCUAUGGCUACUAAUUAUGACUUGAGUGUUGUCAAU

GCUAGACUUCGUGCAAAACACUACGUCUAUAUUGGCGAUCCUGCUCAAUUACC

AGCCCCCCCGCACAUUGCUGACUAAAGGCACACUAGAACCAGAAUAUUUUAAUU

CAGUGUGCAGACUUAUGAAAACAAUAGGUCCAGACAUGUUCCUUGGAACUUG

UCGCCGUUGUCCUGCUGAAAUUGUUGACACUGUGAGUGCUUUAGUUUAUGAC

AAUAAGCUAAAAGCACACAAGGAUAAGUCAGCUCAAUGCUUCAAAAUGUUCU

ACAAAGGUGUUAUUACACAUGAUGUUUCAUCUGCAAUCAACAGACCUCAAAU

AGGCGUUGUAAGAGAAUUUCUUACACGCAAUCCUGCUUGGAGAAAAGCUGUU

UUUAUCUCACCUUAUAAUUCACAGAACGCUGUAGCUUCAAAAAUCUUAGGAU

UGCCUACGCAGACUGUUGAUUCAUCACAGGGUUCUGAAUAUGACUAUGUCAU

AUUCACACAAACUACUGAAACAGCACACUCUUGUAAUGUCAACCGCUUCAAUG

UGGCUAUCACAAGGGCAAAAAUUGGCAUUUUGUGCAUAAUGUCUGAUAGAGA

UCUUUAUGACAAACUGCAAUUUACAAGUCUAGAAAUACCACGUCGCAAUGUG

GCUACAUUACAAGCAGAAAAUGUAACUGGACUUUUUAAGGACUGUAGUAAGA

UCAUUACUGGUCUUCAUCCUACACAGGCACCUACACACCUCAGCGUUGAUAUA

AAGUUCAAGACUGAAGGAUUAUGUGUUGACAUACCAGGCAUACCAAAGGACA

UGACCUACCGUAGACUCAUCUCUAUGAUGGGUUUCAAAAUGAAUUACCAAGU

CAAUGGUUACCCUAAUAUGUUUAUCACCCGCGAAGAAGCUAUUCGUCACGUU

CGUGCGUGGAUUGGCUUUGAUGUAGAGGGCUGUCAUGCAACUAGAGAUGCUG

UGGGUACUAACCUACCUCUCCAGCUAGGAUUUUCUACAGGUGUUAACUUAGU

AGCUGUACCGACUGGUUAUGUUGACACUGAAAAUAACACAGAAUUCACCAGA

GUUAAUGCAAAACCUCCACCAGGUGACCAGUUUAAACAUCUUAUACCACUCAU

GUAUAAAGGCUUGCCCUGGAAUGUAGUGCGUAUUAAGAUAGUACAAAUGCUC

AGUGAUACACUGAAAGGAUUGUCAGACAGAGUCGUGUUCGUCCUUUGGGCGC

AUGGCUUUGAGCUUACAUCAAUGAAGUACUUUGUCAAGAUUGGACCUGAAAG

AACGUGUUGUCUGUGUGACAAACGUGCAACUUGCUUUUCUACUUCAUCAGAU

ACUUAUGCCUGCUGGAAUCAUUCUGUGGGUUUUGACUAUGUCUAUAACCCAU

UUAUGAUUGAUGUUCAGCAGUGGGGCUUUACGGGUAACCUUCAGAGUAACCA

UGACCAACAUUGCCAGGUACAUGGAAAUGCACAUGUGGCUAGUUGUGAUGCU

AUCAUGACUAGAUGUUUAGCAGUCCAUGAGUGCUUUGUUAAGCGCGUUGAUU

GGUCUGUUGAAUACCCUAUUAUAGGAGAUGAACUGAGGGUUAAUUCUGCUUG

CAGAAAAGUACAACACAUGGUUGUGAAGUCUGCAUUGCUUGCUGAUAAGUUU

CCAGUUCUUCAUGACAUUGGAAAUCCAAAGGCUAUCAAGUGUGUGCCUCAGG

CUGAAGUAGAAUGGAAGUUCUACGAUGCUCAGCCAUGUAGUGACAAAGCUUA

CAAAAUAGAGGAACUCUUCUAUUCUUAUGCUACACAUCACGAUAAAUUCACU

GAUGGUGUUUGUUUGUUUUGGAAUUGUAACGUUGAUCGUUACCCAGCCAAUG

CAAUUGUGUGUAGGUUUGACACAAGAGUCUUGUCAAACUUGAACUUACCAGG

CUGUGAUGGUGGUAGUUUGUAUGUGAAUAAGCAUGCAUUCCACACUCCAGCU

UUCGAUAAAAGUGCAUUUACUAAUUUAAAGCAAUUGCCUUUCUUUUACUAUU

CUGAUAGUCCUUGUGAGUCUCAUGGCAAACAAGUAGUGUCGGAUAUUGAUUA

UGUUCCACUCAAAUCUGCUACGUGUAUUACACGAUGCAAUUUAGGUGGUGCU

GUUUGCAGACACCAUGCAAAUGAGUACCGACAGUACUUGGAUGCAUAUAAUA

UGAUGAUUUCUGCUGGAUUUAGCCUAUGGAUUUACAAACAAUUUGAUACUUA

UAACCUGUGGAAUACAUUUACCAGGUUACAGAGUUUAGAAAAUGUGGCUUAU

AAUGUUGUUAAUAAAGGACACUUUGAUGGACACGCCGGCGAAGCACCUGUUU

CCAUCAUUAAUAAUGCUGUUUACACAAAGGUAGAUGGUAUUGAUGUGGAGAU

CUUUGAAAAUAAGACAACACUUCCUGUUAAUGUUGCAUUUGAGCUUUGGGCU

AAGCGUAACAUUAAACCAGUGCCAGAGAUUAAGAUACUCAAUAAUUUGGGUG

UUGAUAUCGCUGCUAAUACUGUAAUCUGGGACUACAAAAGAGAAGCCCCAGC

ACAUGUAUCUACAAUAGGUGUCUGCACAAUGACUGACAUUGCCAAGAAACCU

ACUGAGAGUGCUUGUUCUUCACUUACUGUCUUGUUUGAUGGUAGAGUGGAAG

GACAGGUAGACCUUUUUAGAAACGCCCGUAAUGGUGUUUUAAUAACAGAAGG

UUCAGUCAAAGGUCUAACACCUUCAAAGGGACCAGCACAAGCUAGCGUCAAU

GGAGUCACAUUAAUUGGAGAAUCAGUAAAAACACAGUUUAACUACUUUAAGA

AAGUAGACGGCAUUAUUCAACAGUUGCCUGAAACCUACUUUACUCAGAGCAG

AGACUUAGAGGAUUUUAAGCCCAGAUCACAAAUGGAAACUGACUUUCUCGAG

CUCGCUAUGGAUGAAUUCAUACAGCGAUAUAAGCUCGAGGGCUAUGCCUUCG

AACACAUCGUUUAUGGAGAUUUCAGUCAUGGACAACUUGGCGGUCUUCAUUU

AAUGAUAGGCUUAGCCAAGCGCUCACAAGAUUCACCACUUAAAUUAGAGGAU

UUUAUCCCUAUGGACAGCACAGUGAAAAAUUACUUCAUAACAGAUGCGCAAA

CAGGUUCAUCAAAAUGUGUGUGUUCUGUGAUUGAUCUUUUACUUGAUGACUU

UGUCGAGAUAAUAAAGUCACAAGAUUUGUCAGUGAUUUCAAAAGUGGUCAAG

GUUACAAUUGACUAUGCUGAAAUUUCAUUCAUGCUUUGGUGUAAGGAUGGAC

AUGUUGAAACCUUCUACCCAAAACUACAAGCAAGUCAAGCGUGGCAACCAGG

UGUUGCGAUGCCUAACUUGUACAAGAUGCAAAGAAUGCUUCUUGAAAAGUGU

GACCUUCAGAAUUAUGGUGAAAAUGCUGUUAUACCAAAAGGAAUAAUGAUGA

AUGUCGCAAAGUAUACUCAACUGUGUCAAUACUUAAAUACACUUACUUUAGC

UGUACCCUACAACAUGAGAGUUAUUCACUUUGGUGCUGGCUCUGAUAAAGGA

GUUGCACCAGGUACAGCUGUGCUCAGACAAUGGUUGCCAACUGGCACACUACU

UGUCGAUUCAGAUCUUAAUGACUUCGUCUCCGACGCAGAUUCUACUUUAAUU

GGAGACUGUGCAACAGUACAUACGGCUAAUAAAUGGGACCUUAUUAUUAGCG

AUAUGUAUGACCCUAGGACCAAACAUGUGACAAAAGAGAAUGACUCUAAAGA

AGGGUUUUUCACUUAUCUGUGUGGAUUUAUAAAGCAAAAACUAGCCCUGGGU

GGUUCUAUAGCUGUAAAGAUAACAGAGCAUUCUUGGAAUGCUGACCUUUACA

AGCUUAUGGGCCAUUUCUCAUGGUGGACAGCUUUUGUUACAAAUGUAAAUGC

AUCAUCAUCGGAAGCAUUUUUAAUUGGGGCUAACUAUCUUGGCAAGCCGAAG

GAACAAAUUGAUGGCUAUACCAUGCAUGCUAACUACAUUUUCUGGAGGAACA

CAAAUCCUAUCCAGUUGUCUUCCUAUUCACUCUUUGACAUGAGCAAAUUUCCU

CUUAAAUUAAGAGGAACUGCUGUAAUGUCUCUUAAGGAGAAUCAAAUCAAUG

AUAUGAUUUAUUCUCUUCUGGAAAAAGGUAGGCUUAUCAUUAGAGAAAACAA

CAGAGUUGUGGUUUCAAGUGAUAUUCUUGUUAACAACUAAACGAACAUGUUU

AUUUUCUUAUUAUUUCUUACUCUCACUAGUGGUAGUGACCUUGACCGGUGCA

CCACUUUUGAUGAUGUUCAAGCUCCUAAUUACACUCAACAUACUUCAUCUAU

GAGGGGGGUUUACUAUCCUGAUGAAAUUUUUAGAUCAGACACUCUUUAUUUA

ACUCAGGAUUUAUUUCUUCCAUUUUAUUCUAAUGUUACAGGGUUUCAUACUA

UUAAUCAUACGUUUGGCAACCCUGUCAUACCUUUUAAGGAUGGUAUUUAUUU

UGCUGCCACAGAGAAAUCAAAUGUUGUCCGUGGUUGGGUUUUUGGUUCUACC

AUGAACAACAAGUCACAGUCGGUGAUUAUUAUUAACAAUUCUACUAAUGUUG

UUAUACGAGCAUGUAACUUUGAAUUGUGUGACAACCCUUUCUUUGCUGUUUC

UAAACCCAUGGGUACACAGACACAUACUAUGAUAUUCGAUAAUGCAUUUAAU

UGCACUUUCGAGUACAUAUCUGAUGCCUUUUCGCUUGAUGUUUCAGAAAAGU

CAGGUAAUUUUAAACACUUACGAGAGUUUGUGUUUAAAAAUAAAGAUGGGUU

UCUCUAUGUUUAUAAGGGCUAUCAACCUAUAGAUGUAGUUCGUGAUCUACCU

UCUGGUUUUAACACUUUGAAACCUAUUUUUAAGUUGCCUCUUGGUAUUAACA

UUACAAAUUUUAGAGCCAUUCUUACAGCCUUUUCACCUGCUCAAGACAUUUG

GGGCACGUCAGCUGCAGCCUAUUUUGUUGGCUAUUUAAAGCCAACUACAUUU

AUGCUCAAGUAUGAUGAAAAUGGUACAAUCACAGAUGCUGUUGAUUGUUCUC

AAAAUCCACUUGCUGAACUCAAAUGCUCUGUUAAGAGCUUUGAGAUUGACAA

AGGAAUUUACCAGACCUCUAAUUUCAGGGUUGUUCCCUCAGGAGAUGUUGUG

AGAUUCCCUAAUAUUACAAACUUGUGUCCUUUUGGAGAGGUUUUUAAUGCUA

CUAAAUUCCCUUCUGUCUAUGCAUGGGAGAGAAAAAAAAUUUCUAAUUGUGU

UGCUGAUUACUCUGUGCUCUACAACUCAACAUUUUUUUCAACCUUUAAGUGC

UAUGGCGUUUCUGCCACUAAGUUGAAUGAUCUUUGCUUCUCCAAUGUCUAUG

CAGAUUCUUUUGUAGUCAAGGGAGAUGAUGUAAGACAAAUAGCGCCAGGACA

AACUGGUGUUAUUGCUGAUUAUAAUUAUAAAUUGCCAGAUGAUUUCAUGGGU

UGUGUCCUUGCUUGGAAUACUAGGAACAUUGAUGCUACUUCAACUGGUAAUU

AUAAUUAUAAAUAUAGGUAUCUUAGACAUGGCAAGCUUAGGCCCUUUGAGAG

AGACAUAUCUAAUGUGCCUUUCUCCCCUGAUGGCAAACCUUGCACCCCACCUG

CUCUUAAUUGUUAUUGGCCAUUAAAUGAUUAUGGUUUUUACACCACUACUGG

CAUUGGCUACCAACCUUACAGAGUUGUAGUACUUUCUUUUGAACUUUUAAAU

GCACCGGCCACGGUUUGUGGACCAAAAUUAUCCACUGACCUUAUUAAGAACCA

GUGUGUCAAUUUUAAUUUUAAUGGACUCACUGGUACUGGUGUGUUAACUCCU

UCUUCAAAGAGAUUUCAACCAUUUCAACAAUUUGGCCGUGAUGUUUCUGAUU

UCACUGAUUCCGUUCGAGAUCCUAAAACAUCUGAAAUAUUAGACAUUUCACC

UUGCGCUUUUGGGGGUGUAAGUGUAAUUACACCUGGAACAAAUGCUUCAUCU

GAAGUUGCUGUUCUAUAUCAAGAUGUUAACUGCACUGAUGUUUCUACAGCAA

UUCAUGCAGAUCAACUCACACCAGCUUGGCGCAUAUAUUCUACUGGAAACAA

UGUAUUCCAGACUCAAGCAGGCUGUCUUAUAGGAGCUGAGCAUGUCGACACU

UCUUAUGAGUGCGACAUUCCUAUUGGAGCUGGCAUUUGUGCUAGUUACCAUA

CAGUUUCUUUAUUACGUAGUACUAGCCAAAAAUCUAUUGUGGCUUAUACUAU

GUCUUUAGGUGCUGAUAGUUCAAUUGCUUACUCUAAUAACACCAUUGCUAUA

CCUACUAACUUUUCAAUUAGCAUUACUACAGAAGUAAUGCCUGUUUCUAUGG

CUAAAACCUCCGUAGAUUGUAAUAUGUACAUCUGCGGAGAUUCUACUGAAUG

UGCUAAUUUGCUUCUCCAAUAUGGUAGCUUUUGCACACAACUAAAUCGUGCA

CUCUCAGGUAUUGCUGCUGAACAGGAUCGCAACACACGUGAAGUGUUCGCUC

AAGUCAAACAAAUGUACAAAACCCCAACUUUGAAAUAUUUUGGUGGUUUUAA

UUUUUCACAAAUAUUACCUGACCCUCUAAAGCCAACUAAGAGGUCUUUUAUU

GAGGACUUGCUCUUUAAUAAGGUGACACUCGCUGAUGCUGGCUUCAUGAAGC

AAUAUGGCGAAUGCCUAGGUGAUAUUAAUGCUAGAGAUCUCAUUUGUGCGCA

GAAGUUCAAUGGACUUACAGUGUUGCCACCUCUGCUCACUGAUGAUAUGAUU

GCUGCCUACACUGCUGCUCUAGUUAGUGGUACUGCCACUGCUGGAUGGACAU

UUGGUGCUGGCGCUGCUCUUCAAAUACCUUUUGCUAUGCAAAUGGCAUAUAG

GUUCAAUGGCAUUGGAGUUACCCAAAAUGUUCUCUAUGAGAACCAAAAACAA

AUCGCCAACCAAUUUAACAAGGCGAUUAGUCAAAUUCAAGAAUCACUUACAA

CAACAUCAACUGCAUUGGGCAAGCUGCAAGACGUUGUUAACCAGAAUGCUCA

AGCAUUAAACACACUUGUUAAACAACUUAGCUCUAAUUUUGGUGCAAUUUCA

AGUGUGCUAAAUGAUAUCCUUUCGCGACUUGAUAAAGUCGAGGCGGAGGUAC

AAAUUGACAGGUUAAUUACAGGCAGACUUCAAAGCCUUCAAACCUAUGUAAC

ACAACAACUAAUCAGGGCUGCUGAAAUCAGGGCUUCUGCUAAUCUUGCUGCU

ACUAAAAUGUCUGAGUGUGUUCUUGGACAAUCAAAAAGAGUUGACUUUUGUG

GAAAGGGCUACCACCUUAUGUCCUUCCCACAAGCAGCCCCGCAUGGUGUUGUC

UUCCUACAUGUCACGUAUGUGCCAUCCCAGGAGAGGAACUUCACCACAGCGCC

AGCAAUUUGUCAUGAAGGCAAAGCAUACUUCCCUCGUGAAGGUGUUUUUGUG

UUUAAUGGCACUUCUUGGUUUAUUACACAGAGGAACUUCUUUUCUCCACAAA

UAAUUACUACAGACAAUACAUUUGUCUCAGGAAAUUGUGAUGUCGUUAUUGG

CAUCAUUAACAACACAGUUUAUGAUCCUCUGCAACCUGAGCUUGACUCAUUCA

AAGAAGAGCUGGACAAGUACUUCAAAAAUCAUACAUCACCAGAUGUUGAUCU

UGGCGACAUUUCAGGCAUUAACGCUUCUGUCGUCAACAUUCAAAAAGAAAUU

GACCGCCUCAAUGAGGUCGCUAAAAAUUUAAAUGAAUCACUCAUUGACCUUC

AAGAAUUGGGAAAAUAUGAGCAAUAUAUUAAAUGGCCUUGGUAUGUUUGGCU

CGGCUUCAUUGCUGGACUAAUUGCCAUCGUCAUGGUUACAAUCUUGCUUUGU

UGCAUGACUAGUUGUUGCAGUUGCCUCAAGGGUGCAUGCUCUUGUGGUUCUU

GCUGCAAGUUUGAUGAGGAUGACUCUGAGCCAGUUCUCAAGGGUGUCAAAUU

ACAUUACACAUAAACGAACUUAUGGAUUUGUUUAUGAGAUUUUUUACUCUUA

GAUCAAUUACUGCACAGCCAGUAAAAAUUGACAAUGCUUCUCCUGCAAGUAC

UGUUCAUGCUACAGCAACGAUACCGCUACAAGCCUCACUCCCUUUCGGAUGGC

UUGUUAUUGGCGUUGCAUUUCUUGCUGUUUUUCAGAGCGCUACCAAAAUAAU

UGCGCUCAAUAAAAGAUGGCAGCUAGCCCUUUAUAAGGGCUUCCAGUUCAUU

UGCAAUUUACUGCUGCUAUUUGUUACCAUCUAUUCACAUCUUUUGCUUGUCG

CUGCAGGUAUGGAGGCGCAAUUUUUGUACCUCUAUGCCUUGAUAUAUUUUCU

ACAAUGCAUCAACGCAUGUAGAAUUAUUAUGAGAUGUUGGCUUUGUUGGAAG

UGCAAAUCCAAGAACCCAUUACUUUAUGAUGCCAACUACUUUGUUUGCUGGC

ACACACAUAACUAUGACUACUGUAUACCAUAUAACAGUGUCACAGAUACAAU

UGUCGUUACUGAAGGUGACGGCAUUUCAACACCAAAACUCAAAGAAGACUAC

CAAAUUGGUGGUUAUUCUGAGGAUAGGCACUCAGGUGUUAAAGACUAUGUCG

UUGUACAUGGCUAUUUCACCGAAGUUUACUACCAGCUUGAGUCUACACAAAU

UACUACAGACACUGGUAUUGAAAAUGCUACAUUCUUCAUCUUUAACAAGCUU

GUUAAAGACCCACCGAAUGUGCAAAUACACACAAUCGACGGCUCUUCAGGAG

UUGCUAAUCCAGCAAUGGAUCCAAUUUAUGAUGAGCCGACGACGACUACUAG

CGUGCCUUUGUAAGCACAAGAAAGUGAGUACGAACUUAUGUACUCAUUCGUU

UCGGAAGAAACAGGUACGUUAAUAGUUAAUAGCGUACUUCUUUUUCUUGCUU

UCGUGGUAUUCUUGCUAGUCACACUAGCCAUCCUUACUGCGCUUCGAUUGUG

UGCGUACUGCUGCAAUAUUGUUAACGUGAGUUUAGUAAAACCAACGGUUUAC

GUCUACUCGCGUGUUAAAAAUCUGAACUCUUCUGAAGGAGUUCCUGAUCUUC

UGGUCUAAACGAACUAACUAUUAUUAUUAUUCUGUUUGGAACUUUAACAUUG

CUUAUCAUGGCAGACAACGGUACUAUUACCGUUGAGGAGCUUAAACAACUCC

UGGAACAAUGGAACCUAGUAAUAGGUUUCCUAUUCCUAGCCUGGAUUAUGUU

ACUACAAUUUGCCUAUUCUAAUCGGAACAGGUUUUUGUACAUAAUAAAGCUU

GUUUUCCUCUGGCUCUUGUGGCCAGUAACACUUGCUUGUUUUGUGCUUGCUG

CUGUCUACAGAAUUAAUUGGGUGACUGGCGGGAUUGCGAUUGCAAUGGCUUG

UAUUGUAGGCUUGAUGUGGCUUAGCUACUUCGUUGCUUCCUUCAGGCUGUUU

GCUCGUACCCGCUCAAUGUGGUCAUUCAACCCAGAAACAAACAUUCUUCUCAA

UGUGCCUCUCCGGGGGACAAUUGUGACCAGACCGCUCAUGGAAAGUGAACUU

GUCAUUGGUGCUGUGAUCAUUCGUGGUCACUUGCGAAUGGCCGGACACUCCC

UAGGGCGCUGUGACAUUAAGGACCUGCCAAAAGAGAUCACUGUGGCUACAUC

ACGAACGCUUUCUUAUUACAAAUUAGGAGCGUCGCAGCGUGUAGGCACUGAU

UCAGGUUUUGCUGCAUACAACCGCUACCGUAUUGGAAACUAUAAAUUAAAUA

CAGACCACGCCGGUAGCAACGACAAUAUUGCUUUGCUAGUACAGUAAGUGAC

AACAGAUGUUUCAUCUUGUUGACUUCCAGGUUACAAUAGCAGAGAUAUUGAU

UAUCAUUAUGAGGACUUUCAGGAUUGCUAUUUGGAAUCUUGACGUUAUAAUA

AGUUCAAUAGUGAGACAAUUAUUUAAGCCUCUAACUAAGAAGAAUUAUUCGG

AGUUAGAUGAUGAAGAACCUAUGGAGUUAGAUUAUCCAUAAAACGAACAUGA

AAAUUAUUCUCUUCCUGACAUUGAUUGUAUUUACAUCUUGCGAGCUAUAUCA

CUAUCAGGAGUGUGUUAGAGGUACGACUGUACUACUAAAAGAACCUUGCCCA

UCAGGAACAUACGAGGGCAAUUCACCAUUUCACCCUCUUGCUGACAAUAAAU

UUGCACUAACUUGCACUAGCACACACUUUGCUUUUGCUUGUGCUGACGGUAC

UCGACAUACCUAUCAGCUGCGUGCAAGAUCAGUUUCACCAAAACUUUUCAUCA

GACAAGAGGAGGUUCAACAAGAGCUCUACUCGCCACUUUUUCUCAUUGUUGC

UGCUCUAGUAUUUUUAAUACUUUGCUUCACCAUUAAGAGAAAGACAGAAUGA

AUGAGCUCACUUUAAUUGACUUCUAUUUGUGCUUUUUAGCCUUUCUGCUAUU

CCUUGUUUUAAUAAUGCUUAUUAUAUUUUGGUUUUCACUCGAAAUCCAGGAU

CUAGAAGAACCUUGUACCAAAGUCUAAACGAACAUGAAACUUCUCAUUGUUU

UGACUUGUAUUUCUCUAUGCAGUUGCAUAUGCACUGUAGUACAGCGCUGUGC

AUCUAAUAAACCUCAUGUGCUUGAAGAUCCUUGUAAGGUACAACACUAGGGG

UAAUACUUAUAGCACUGCUUGGCUUUGUGCUCUAGGAAAGGUUUUACCUUUU

CAUAGAUGGCACACUAUGGUUCAAACAUGCACACCUAAUGUUACUAUCAACU

GUCAAGAUCCAGCUGGUGGUGCGCUUAUAGCUAGGUGUUGGUACCUUCAUGA

AGGUCACCAAACUGCUGCAUUUAGAGACGUACUUGUUGUUUUAAAUAAACGA

ACAAAUUAAAAUGUCUGAUAAUGGACCCCAAUCAAACCAACGUAGUGCCCCCC

GCAUUACAUUUGGUGGACCCACAGAUUCAACUGACAAUAACCAGAAUGGAGG

ACGCAAUGGGGCAAGGCCAAAACAGCGCCGACCCCAAGGUUUACCCAAUAAUA

CUGCGUCUUGGUUCACAGCUCUCACUCAGCAUGGCAAGGAGGAACUUAGAUU

CCCUCGAGGCCAGGGCGUUCCAAUCAACACCAAUAGUGGUCCAGAUGACCAAA

UUGGCUACUACCGAAGAGCUACCCGACGAGUUCGUGGUGGUGACGGCAAAAU

GAAAGAGCUCAGCCCCAGAUGGUACUUCUAUUACCUAGGAACUGGCCCAGAA

GCUUCACUUCCCUACGGCGCUAACAAAGAAGGCAUCGUAUGGGUUGCAACUG

AGGGAGCCUUGAAUACACCCAAAGACCACAUUGGCACCCGCAAUCCUAAUAAC

AAUGCUGCCACCGUGCUACAACUUCCUCAAGGAACAACAUUGCCAAAAGGCUU

CUACGCAGAGGGAAGCAGAGGCGGCAGUCAAGCCUCUUCUCGCUCCUCAUCAC

GUAGUCGCGGUAAUUCAAGAAAUUCAACUCCUGGCAGCAGUAGGGGAAAUUC

UCCUGCUCGAAUGGCUAGCGGAGGUGGUGAAACUGCCCUCGCGCUAUUGCUGC

UAGACAGAUUGAACCAGCUUGAGAGCAAAGUUUCUGGUAAAGGCCAACAACA

ACAAGGCCAAACUGUCACUAAGAAAUCUGCUGCUGAGGCAUCUAAAAAGCCU

CGCCAAAAACGUACUGCCACAAAACAGUACAACGUCACUCAAGCAUUUGGGAG

ACGUGGUCCAGAACAAACCCAAGGAAAUUUCGGGGACCAAGACCUAAUCAGA

CAAGGAACUGAUUACAAACAUUGGCCGCAAAUUGCACAAUUUGCUCCAAGUG

CCUCUGCAUUCUUUGGAAUGUCACGCAUUGGCAUGGAAGUCACACCUUCGGG

AACAUGGCUGACUUAUCAUGGAGCCAUUAAAUUGGAUGACAAAGAUCCACAA

UUCAAAGACAACGUCAUACUGCUGAACAAGCACAUUGACGCAUACAAAACAU

UCCCACCAACAGAGCCUAAAAAGGACAAAAAGAAAAAGACUGAUGAAGCUCA

GCCUUUGCCGCAGAGACAAAAGAAGCAGCCCACUGUGACUCUUCUUCCUGCGG

CUGACAUGGAUGAUUUCUCCAGACAACUUCAAAAUUCCAUGAGUGGAGCUUC

UGCUGAUUCAACUCAGGCAUAAACACUCAUGAUGACCACACAAGGCAGAUGG

GCUAUGUAAACGUUUUCGCAAUUCCGUUUACGAUACAUAGUCUACUCUUGUG

CAGAAUGAAUUCUCGUAACUAAACAGCACAAGUAGGUUUAGUUAACUUUAAU

CUCACAUAGCAAUCUUUAAUCAAUGUGUAACAUUAGGGAGGACUUGAAAGAG

CCACCACAUUUUCAUCGAGGCCACGCGGAGUACGAUCGAGGGUACAGUGAAU

AAUGCUAGGGAGAGCUGCCUAUAUGGAAGAGCCCUAAUGUGUAAAAUUAAUU

UUAGUAGUGCUAUCCCCAUGUGAUUUUAAUAGCUUCUUAGGAGAAUGACAAA

AAAAAAAAAAAAAAAA

Curriculum Vitae

Candidate’s full name: Nabil Fannoush

Universities attended (with dates and degrees obtained): University of Garyounis (2001, BSc in Computer Science)