Int. J. Math. And Appl., 9(2)(2021), 195–207

ISSN: 2347-1557

Available Online: http://ijmaa.in/

International Journal of *Mathematics And its Applications*

An Algorithm to Find Eulerian Paths in a Graph and Applications to Bioinformatics

Adishankara Mallik[1,*]

1 415 Cayuga Circle, Mars, Pennslyvania, 16046, United States.

Abstract: We consider a discrete mathematical model to determine the genetic sequence using a list of k-mers from genetic analysis results. We analyze the methods by which to create this program by walking the reader through the process of coding and concepts of the model. Specifically, how a Eulerian path can construct an entire genome from a random list of genetic information broken into chunks.

MSC: 05C90.

Keywords: Applications of graph theory, Genetics, Algorithms, Eulerian Paths and Cycles, Bioinformatics.

1. Introduction

A genome is the whole DNA, and the DNA is a long molecule made up of nucleotides. Each nucleotide is identify with one of four letters: A, C, G and T. Thus, for our purposes, we think of genomes as a long chain of letters or words, where each letter is one of the four letters mentioned [6]. The nucleus of each cell of an organism has a copy of the entire genome. Thus, each organism has trillions of copies of its genome. Being able to read genomes has important applications such us: discovering treatments for diseases [5], screening newborns for diseases [3], detecting food-borne illnesses and contamination [1], detecting the emergence of drug resistant strains of pathogens [4], and various others purposes. Reading genomes is not accomplished solely through lab experiments, there is a significant amount of math and algorithms required as well, a part of which we will go into in this paper. By k-mers, we are referencing the fact that genetic sequences can be in pieces anywhere from 2 to 6, so we use 'k' to designate the ambiguity. For the purposes of this paper, we will mainly be referencing k-mers in bits of three, though it is important to note that the program can be used with k-mers of any length except 1. In graph theory, A Eulerian path is a path that visits every edge once [2]. The use of an algorithm to find the genetic sequence is vital, as without it the amount of time to find a Eulerian path for a genome of billions of nucleotides is unreasonable, even if the experiment is done in segments.

In this article we consider a list of k-mers received from a genetic sequencing device. We assume that there are no errors in the list of k-mers from a sequencing device. The model we study is discrete in time. This paper is organized as follows. In section 2, we briefly explains what DNA is and how we will interpret it. In section 3, we talk about graph theory and reconstruction of a genome by inspection. In section 4 we will discuss Eulerian Cycles and Paths, and the solution to the Genome Reconstruction Problem.

[*] *E-mail: adishankaramallik@gmail.com*

2. Genomic Background

Deoxyribonucleic acid is a molecule we all know. It is composed of 4 nucleotides, Adenine (A), Thymine (T), Cytosine (C), and Guanine (G), Adenine only bonds with Thymine, and Cytosine only bonds with Guanine. In its natural form, it is a double helix with two strands, and this is shown in Figure 1. In order to make it simpler, we show in Figure 2 the untwisted form to emphasize the nucleotides. The green line shows one strand of DNA, the red line is the other strand, and the black lines are meant to show the hydrogen bonds, which is what holds these strands together in real life. The strands are complementary, so in practice, you only need one of strands before running it through a simple loop with four conditionals, one for each nucleotide, to find out the whole genome on both sides. The example shown is if you follow the rules outline above, ATTCGTACTGGCA becomes TAAGCATGACCGT. Therefore, for the purposes of this paper, we only need to focus one strand, like what is shown in Figure 3. So we consider DNA in this paper as nothing but a long but finite string of letters.

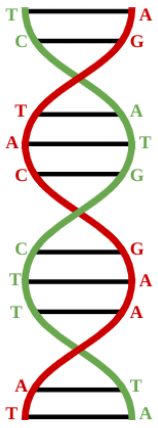


Figure 1. **Simple DNA Image**

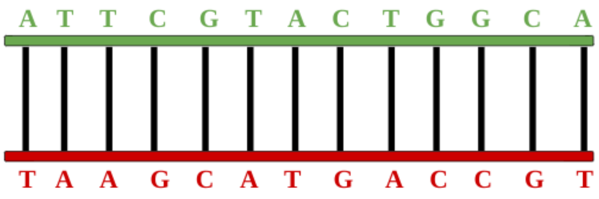


Figure 2. **Straight DNA Sequence**

T A A G C A T G A C C G T

Figure 3. **One Strand DNA Sequence**

2.1. Reading Genomes

The nucleus of each cell has one copy of its genome, and thus when a blood sample is taken from an organism, it has billions of copies of the genome. The human genome has approximately 3 billion nucleotides [7], which is why normal experiments don't use all of data from a single genome, but instead use shorter sequences, and over the course of many experiments, reconstruct the genome. Take for example, the DNA sequence of ATGTCACTAGCAA. As explained above, when we obtain the data, we have several of these genomes. Each of these genomes are broken up into smaller segments. Assume these segments are each about 4 characters long, except a few exceptions due to the length of the genome. We only keep segments, or 4-mers, that are 4 characters long. These are the 4-mers that the program can use. These k-mers are called reads. Because they aren't given in alphabetical order, for the purposes of demonstration, we will be showing them in lexicographic order. Our goal with the program is to reconstruct the genome from these reads.

2.2. Basic Genomic Reconstruction by Inspection

The prefix of a segment is everything without the last letter, and the suffix is everything except the first letter.
Using the list of k-mers, ATGT GTCA CACT ACTA CTAG TCAC TAGC GCAA AGCA TGTC, search for similarities between them, specifically, search for k-mer pairs whose suffix and prefix line up, and then from there piece them together. For example, check out the first k-mer in the sequence, ATGT. Its suffix is TGT, so look for a k-mer with the prefix of TGT, which is TGTC, so what we have then is ATGTC, and so even if the k-mers are not in order, you can still piece it together using prefixes and suffixes. This is shown in Figure 4. Another thing is process demonstrates though, is that doing entire genomes by hand is extremely laborious and not advised, particularly because the human genome is three billion nucleotides, even if experiments are only in segments of thousands. The solution to the Genome Reconstruction Program will be to find a way to have an algorithm perform this function.

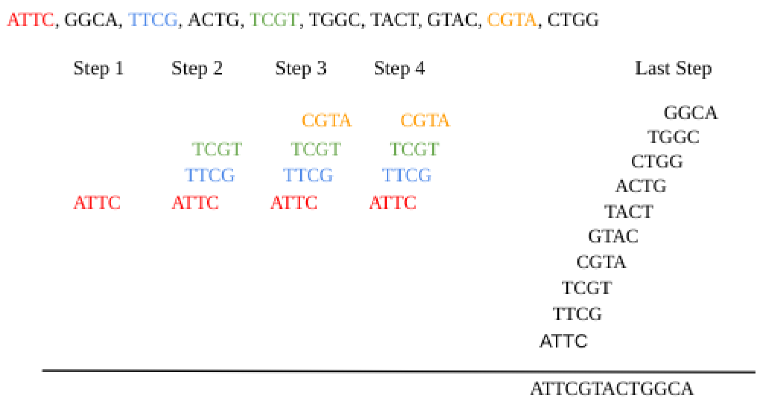


Figure 4. **Reconstruction by Inspection**

3. Computational Aspect

As mentioned in the last section, our goal is to reconstruct the genome from the reads given. In order to further this, we will provide a few definitions.

Definition 3.1 (Genome). *A finite sequence of letters, where each letter can be either C,G,T,A.*

For example, ATCGATCAGTCCG is a genome.

Definition 3.2 (K-mer). *A sequence of k letters (can be C,G,T, or A). A k-mer of a genome is a sub-sequence of the genome of k letters long.*

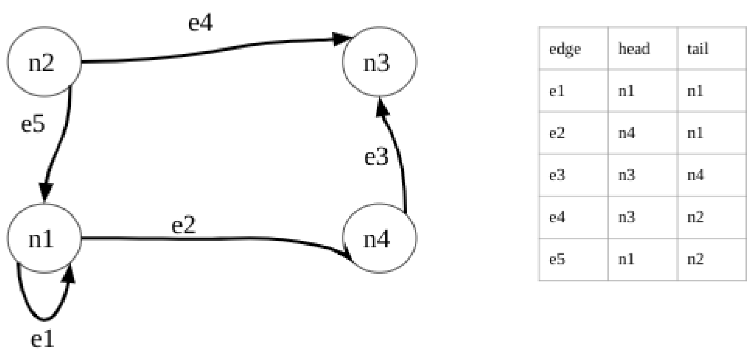
As an example, ATCG is a 4-mer of the genome ATCGATCAGTCCG. In this paper, when we refer to k-mers, we are talking about the list of k-mers from a genome. For example, if the genome is TACGCGACGC, the 3-mers are TAC,ACG,CGC,GCG,CGA,GAC,ACG, and CGC. Notice how ACG is repeated. This is because that sequence appears twice, and k-mers can be repeated any number of times, as long as they appear that many times.

Definition 3.3 (Genome Reconstruction Problem)**.** *The problem we seek to solve is that when given a list of k-mers, we can output the complete sequence of one strand, as explained previously, it is easy to construct the other strand from one. The order of k-mers is irrelevant, and this problem will be considered solved if the genome provided has the same k-mers as the list given.*

3.1. Graph Theory

The algorithm we will describe in this paper will utilize elements of graph theory, and we will describe them in the section below.

Definition 3.4 (Directed Graph)**.** *A directed graph is a set of nodes and a set of edges. Each edge connects two nodes. The edges have directions. Given an edge, the two nodes it connects are called its tail and its head. Edges point from their tail to their head. Graphs are visualized with drawings. Each node is drawn as small circle and each edge is drawn as an arrow from the edge's tail to the edge's head. An example is given in Figure 5, where the nodes are n_1, n_2, n_3, and n_4, and the edges are e_1, e_2, e_3, e_4, and e_5.*



edge	head	tail
e1	n1	n1
e2	n4	n1
e3	n3	n4
e4	n3	n2
e5	n1	n2

Figure 5. A Directed Graph

Definition 3.5 (Path)**.** *A path is a sequence of different edges $e_1, e_2, ...e_n$ such that the head of e_i equals the tail of e_{i+1}. An example is given below in Figure 6, where the path is the sequence of edges e_1, e_2, e_3, e_4, e_5.*

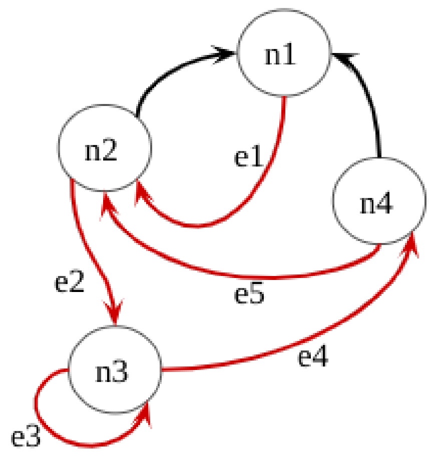


Figure 6. A Path

On the other hand, the sequence of edges e_1, e_2, e_3 in Figure 7 is not a path because the head of e_2 is not the same as the tail of e_3.

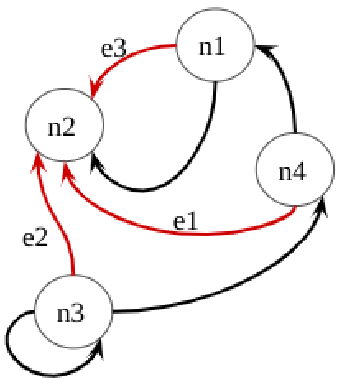


Figure 7. **Not A Path**

3.2. A graph of a list of k-mers

For reasons that will be explained later in this paper, given a list of k-mers we will produce a graph. To that end, here are a few definitions and observations.

Observation 3.6. *A k-mer may appear more than once in a list, but a single k-mer can only appear once in a set. To make this distinction clear, we will use '[ ]' for lists and '{}' for sets.*

Definition 3.7 (Set N(K))**.** *k shall be defined as an integer greater than or equal to 2. Let K be a list of k-mers. A k-mer may appear more than once in the list. N(K) is defined as the set of prefixes and suffixes of the k-mers in K.*

For example, consider this list of 4-mers K=[AATC, ATCG, TCGT, CGTC, GTCG, TCGT]. Notice TCGT appears twice in K. This is allowed because K is a list. From each 4-mer in the list we get its prefix and suffix, which is demonstrated in Figure 8. In black, you see the 3-mers that appear once, and the ones in red have already appeared. The set N(K) is the set of the black 3-mers, i.e. N(K) = {AAT, ATC, TCG, CGT, GTC, CTG}.

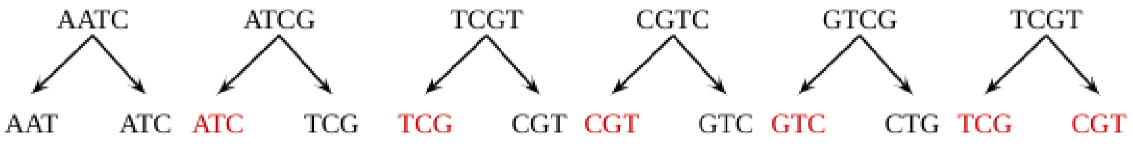


Figure 8. **Getting the Prefixes and Suffixes**

Result 3.8. *Given a k-mer w, we denote its prefix and its suffix by prefix(w) and suffix(w), respectively. For example, the prefix(AGCT)=AGC, and the suffix(AGCT)=GCT.*

Definition 3.9 (The Graph G(K))**.** *Use the definitions of k, K, and N(K) described above. We define the Graph(K) as follows:*

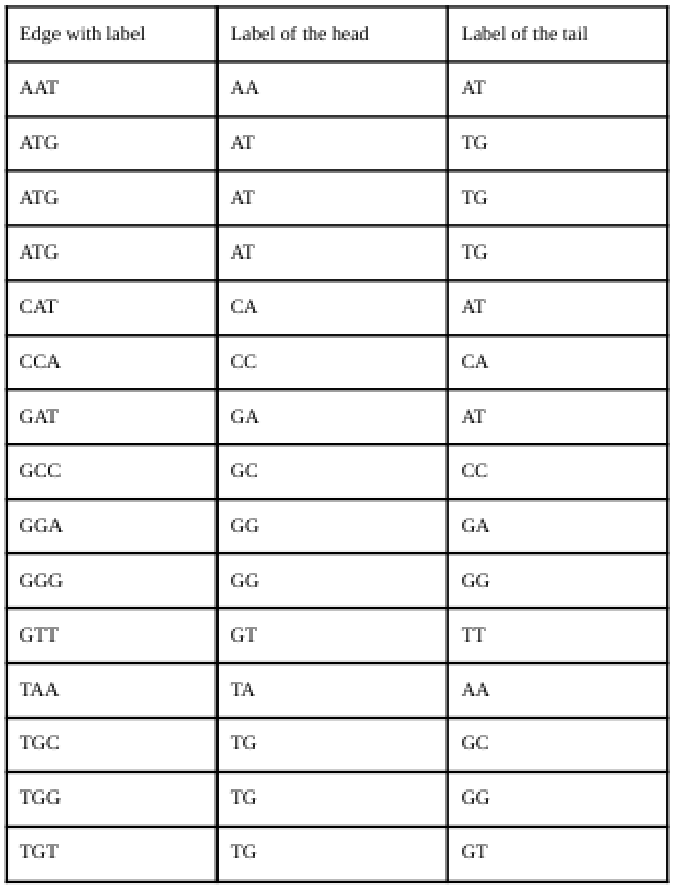
(1). For each k-mer in K, we define an edge that has the same name as the k-mer. Given an edge e, we denote its label as label(e).

(2). For each (k-1)-mer in N(K), we define a node that has as label that (k-1)-mer. Given an edge n, we denote its label as label(n).

(3). Given an edge e its tail is the node t such that prefix(label(e)) = label(t).

(4). Given an edge e its head is the node h such that prefix(label(e)) = label(h).

For example, consider the list of 3-mers: K = [AAT, ATG, ATG, ATG, CAT, CCA, GAT, GCC, GGA, GGG, GTT, TAA, TGC, TGG, TGT]. In this case, N(K) = {AA, AT, TG, CA, CC, GA, GC, GG, GT, TT, TA} is the set for this list of 3-mers. The information for this graph is shown in Figure 9.



Edge with label	Label of the head	Label of the tail
AAT	AA	AT
ATG	AT	TG
ATG	AT	TG
ATG	AT	TG
CAT	CA	AT
CCA	CC	CA
GAT	GA	AT
GCC	GC	CC
GGA	GG	GA
GGG	GG	GG
GTT	GT	TT
TAA	TA	AA
TGC	TG	GC
TGG	TG	GG
TGT	TG	GT

Figure 9. **Graph Information**

The graph from this example above is represented in Figure 10. We label the nodes in black and edges in red. A couple of things you may have noticed, there may be more than one edge from one node to another, and though we labeled the edges, we did not have to, because you can derive the label of the edge from the head and tail.

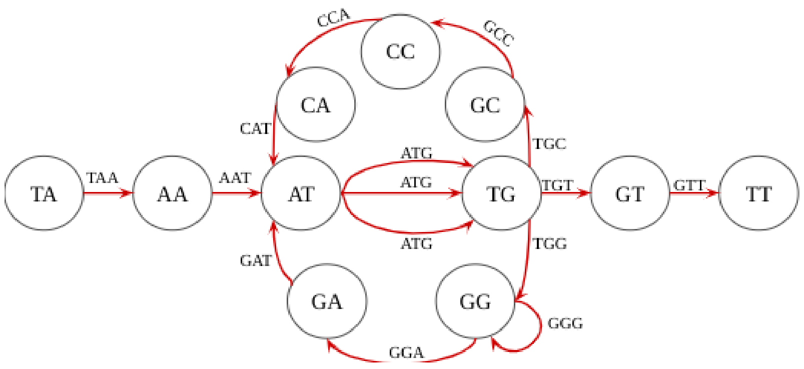


Figure 10. **Graph Example**

3.3. The genome of a path in a graph of a list of k-mers

Definition 3.10. *Given a k-mer k, we define the last letter of k by 'last(k)'. Given two separate strings, x and y, we define their concatenation by '$x + y$'.*

For example: last(ATCAGTA)=A, and ATCAGTA+TGAT= ATCAGTATGAT

Definition 3.11. *Let K be a list of k-mers (as we have defined earlier). Let G(K) be the graph of K. Let $e_1, e_2, e_3, e_4, ...e_n$ be a path in G(K). We define the genome of this path as label(e_1)+last(e_2)+....+last(e_n).*

For example, in Figure 11, we show the same graph as in Figure 10, but we label only the edges in red, that form the path e_1, e_2, e_3, e_4, e_5. In this example, the path is e_1, e_2, e_3, e_4, e_5 Note that label(e_1) = AAT, last(e_2) = ATG, last(e_3) = TCG,last(e_4) = GGG, and last(e_5) = GGA. Thus, the genome of this path is AATGGGA.

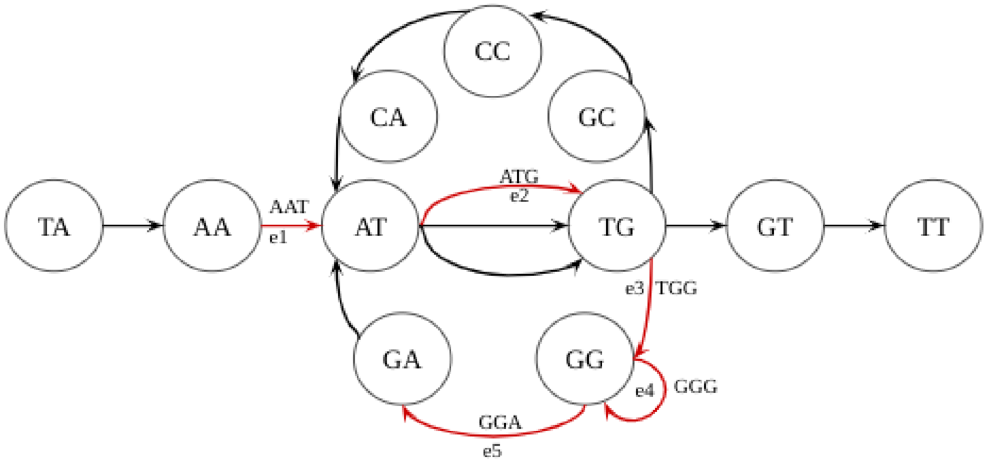


Figure 11. **Path Example**

Observation 3.12. *Let K be a list of k-mers. Let G(K) be the graph of K. Let $e_1, e_2, e_3, e_4, ...e_n$ be a path in G(K). The k-mers of the genome of this path is the list of the labels of the edges in the path.*

In the example from Figure 11, the genome of the path e_1, e_2, e_3, e_4, e_5 is AATGGGA. The list of 3-mers is K=[AAT, ATG, TCG, GGG, GGA]. On the other hand, the list of labels of the edges in this path is [label(e_1), label(e_2), label(e_3), label(e_4), label(e_5)]=[AAT, ATG, TCG, GGG, GGA], which is the same as the genome of the path.

Observation 3.13. *Let K be a list of k-mers. Let G(K) be the graph of K. Let $e_1, e_2, e_3, e_4, ...e_n$ be a path in G(K) that contains all the edges of G. Then the list of k-mers of the path is the genome of K.*

Note that this genome is the answer to our genome reconstruction problem. So, we can simplify the genome reconstruction problem to finding a path in G(K) that contains all of the edges.

For example, consider the graph and path in Figure 12. The graph is G(K), where K=[TAA, TGG, GGG, AAT, ATG, GGA, GAT, ATG, TGC, GCC, CCA, CAT, ATG, TGT, GTT]. The edges are not labeled, but their labels can easily be derived from the nodes. In this example, the path $e_1, e_2, e_3, e_4, ...e_15$ contains all of edges of the graph. Its genome is TAATGCCATGGGATGTT. The of 3-mers is [TAA, AAT, ATG, TGG, GGG, GGA, GAT, ATG, TGC, GCC, CCA, CAT, ATG, TGT, GTT], which is the same list as K, but in a different order.

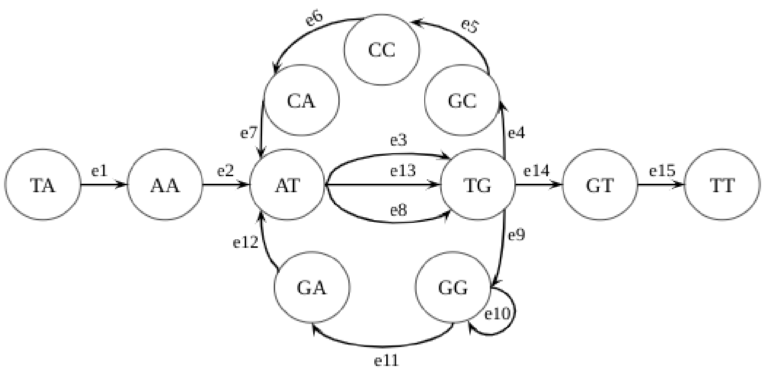


Figure 12. **Genomic Path**

Observation 3.14. *There could be more than one path that contains all the edges, and these different paths can lead to different genomes. For example, as shown in Figure 13, the same graph as above has a different path containing all edges.*

In this example, the path $e_1, e_2, e_3, e_4, ...e_15$ contains all of edges of the graph, and its genome is TAATGGGATGCCATGTT, which is different than the genome we obtained above,TAATGCCATGGGATGTT, with the same graph but different path.

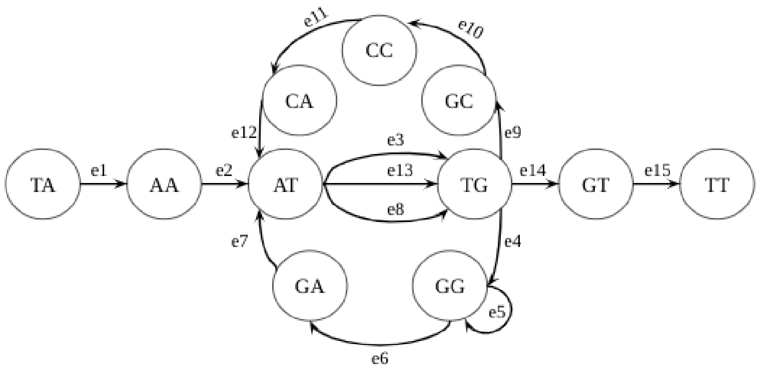


Figure 13. **Genomic Path 2**

4. Eulerian Path and Cycles

4.1. Eulerian paths and conditions for their existence

Let K be a list of k-mers. In the last section, we simplified the genome reconstruction problem to finding a path in G(K) that contains all of the edges.

Definition 4.1 (Eulerian Path)**.** *A path that contains all of the edges in the graph.*

So, if K is list of k-mers, to solve the genome reconstruction problem, we have to find an Eulerian Path of G(K). Remember we are not worried about the fact that there could be multiple genomes, we are fine if we produce one genome.

Definition 4.2 (Indegrees and Outdegrees)**.** *Let n be a node in a directed graph. We define the indegree of n, indeg(n), as the number of edges that have n as a head. We define the outdegree of n, outdeg(n), as the number of edges that have n as its tail.*

Theorem 4.3. *Let G be a directed graph with the nodes N and edges E. Assume G is connected. G has an Eulerian path that is not a cycle only if there is exists a node x that the outdeg(x)-indeg(x)=1, and a node y such that the indeg(y)-*

outdeg(y)=1, and for every other node n, indeg(n)=outdeg(n). Furthermore, any Eulerian Path starts with edge having x as its tail will ends with an edge having y as its head.

An example of this is shown in Figure 14.

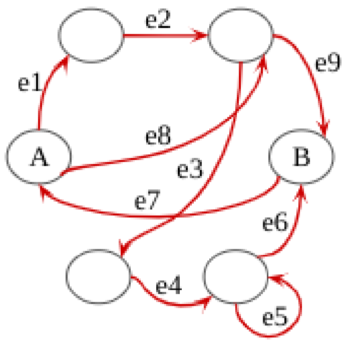


Figure 14. **Eulerian Path**

4.2. An algorithm to find an Eulerian cycle

In this subsection, we will describe an algorithm to find an Eulerian cycle. The process will be shown visually in the next few pictures, starting at Figure 15. Consider the graph of Figure 15. Pick any edge, and label it e_1, see Figure 16. Create a path until it is no longer possible to continue, see Figure 17.

If we had visited all the edges by the time we had to stop we would be finished. But often, as shown with the cycle e_1, e_2, e_3, e_4, the cycle does not contain all of the edges. But, because this graph is connected, there should be one node that is the tail of a black edge. The next step is to go around the cycle and stop at one of these nodes. The red node in Figure 18 is the first node visited in the cycle that has a black edge coming out of it. We always stop at the same node we start when tracing a cycle, so we stopped at the red node. But the red node was chosen because it has an edge coming out that was not in the cycle. This means that we do not need to stop, we can continue and obtain a cycle that contains all the edges of the red cycle and some new edges. This is illustrated in Figure 19. In our example, we ended up with an Eulerian cycle and we are done. This is not always the case, but we can just continue repeating this strategy until we eventually end with an Eulerian cycle.

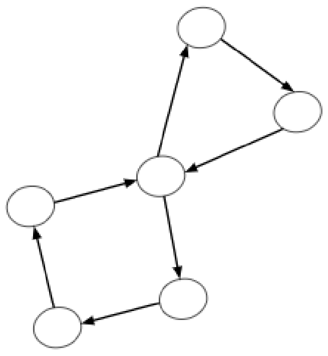


Figure 15. **Graph we use as example.**

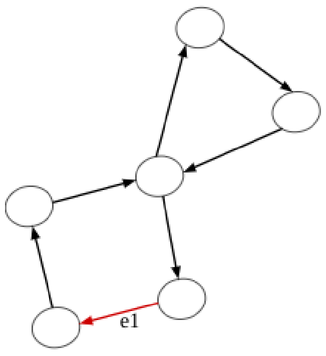


Figure 16. **Picking the first edge**

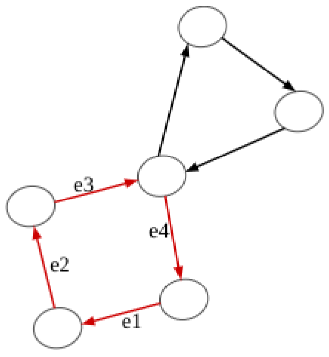


Figure 17. **We can not extend the path e_1, e_2, e_3, e_4.**

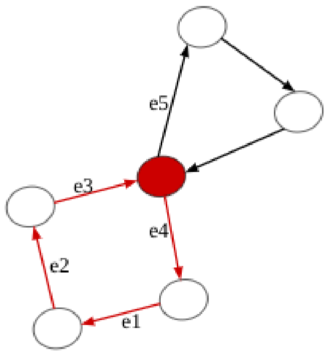


Figure 18. **Going through the same cycle as in Figure 17 but starting from the red node.**

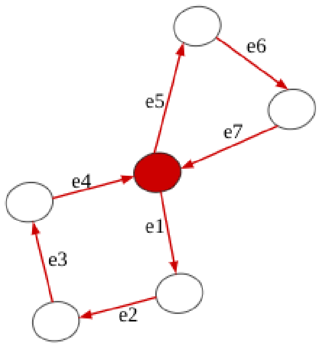


Figure 19. **Completed Eulerian Cycle Algorithm**

4.3. An Algorithm to Find an Eulerian Path

Our original goal, was to find an Eulerian Path, not a cycle. Step 1 is to verify such a path exists. So we perform the calculations described previously, for every node n, the $indeg(n) - outdeg(n) = 0$, and that for one node x, the $indeg(x) - outdeg(x) = 1$, and for one node y, the $indeg(y) - outdeg(y) = -1$. We then add an edge with x as its head and with y as its tail, which in effect creates a Eulerian cycle, as now the $indeg(n) - outdeg(n) = 0$ for every node. Next, we find an Eulerian cycle in this new graph. Then, retrace the cycle starting with the edge that comes after the added edge and stop just before the turn of the added edge. This is the Eulerian path. This is shown in Figures 20-23.

Figure 20 shows the example we work with and the extra edge added. Figure 22 shows the Eulerian cycle we find following the algorithm described in the last subsection. Figure 23 shows the Eulerian path.

5. Conclusions

In this article we described the mathematics behind the reconstruction of genomes. This theory has been fundamental to the field of bio-informatics and reading genomes would not be possible without it. We restricted ourselves to the ideal scenario where the information given to us is complete and without errors. In practice, leading with the imperfections of real life data in not trivial and, in fact, has lead to the field of bio-informatics that has many applications beyond genome reconstructions.

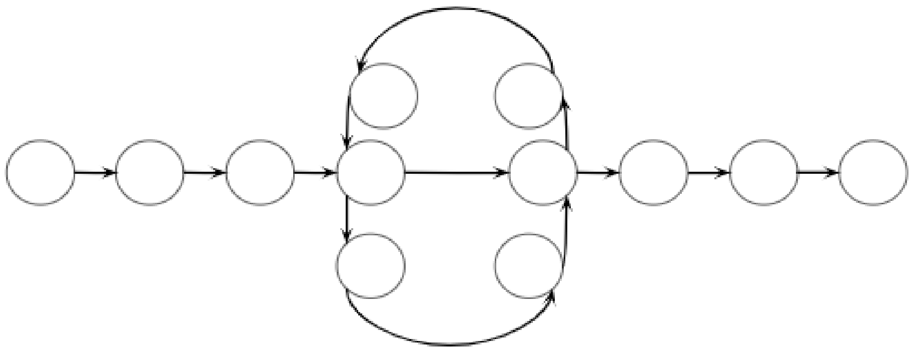


Figure 20. **Example of a graph**

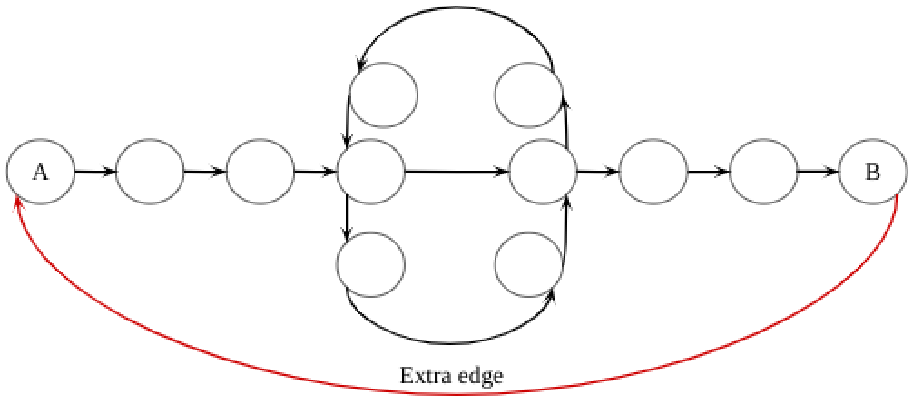


Figure 21. **Adding the extra edge.**

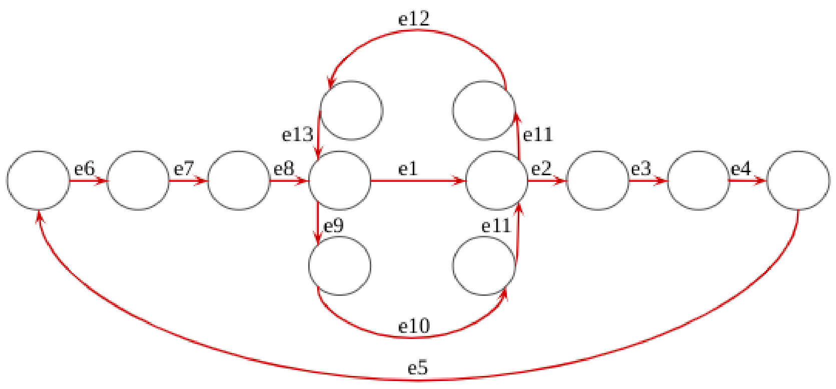


Figure 22. **Finding the Eulerian cycle.**

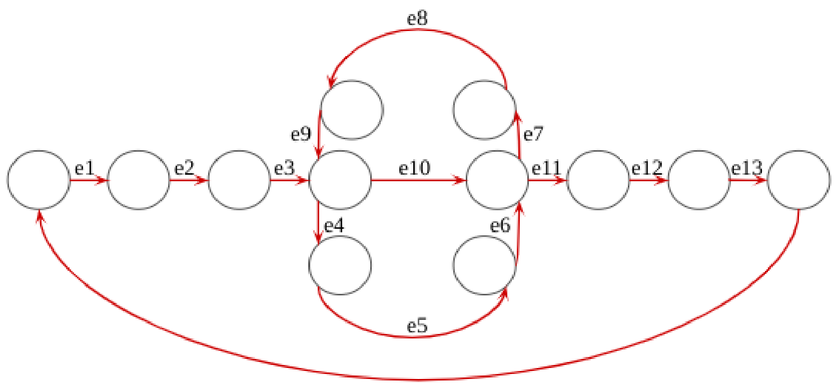


Figure 23. **Getting the Eulerian path from the cycle.**

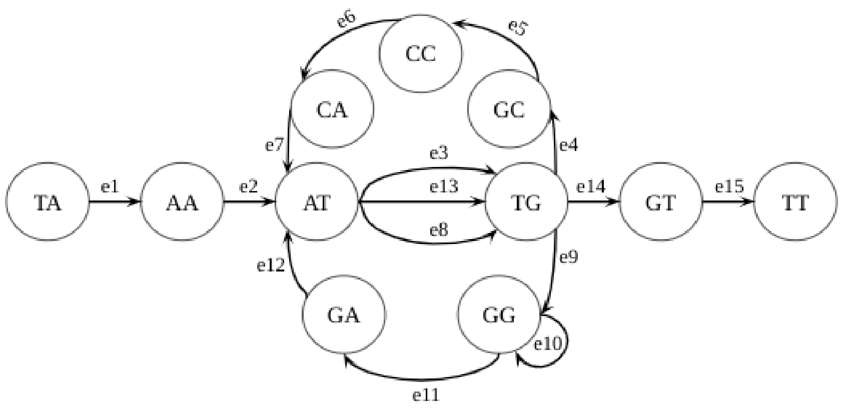


Figure 24. **The Eulerian Path of the Genome we were considering**

References

[1] M. W. Allard, E. Strain, H. Rand, D. Melka, W. A. Correll, L. Hintz, E. Stevens, R. Timme, S. Lomonaco, Y. Chen, S. M. Musser, and E. W. Brown, *Whole genome sequencing uses for foodborne contamination and compliance: Discovery of an emerging contamination event in an ice cream facility using whole genome sequencing*, Infection, Genetics and Evolution, 73(2019), 214–220.

[2] R. Balakrishnan and K. Ranganathan, *A textbook of graph theory (Second edition)*, Springer, (2012).

[3] J. S. Berg and C. M. Powell, *Potential uses and inherent challenges of using genome-scale sequencing to augment current newborn screening*, Cold Spring Harbor Perspectives in Medicine, 5(12)(2015), a023150. https://doi.org/10.1101/cshperspect.a023150

[4] C. J. Houldcroft, M. A. Beale and J. Breuer, *Clinical and biological insights from viral genome sequencing*, Nature Reviews Microbiology, 15(3)(2017), 183–192.

[5] J. Peto, *Breast cancer susceptibility—A new look at an old model*, Cancer Cell, 1(5)(2002), 411–412.

[6] L. A. Urry, M. L. Cain, S. A. Wasserman, P. V. Minorsky, J. B. Reece and N. A. Campbell, *Campbell biology (Eleventh edition)*, Pearson Education Inc., (2017).

[7] J. C. Venter, M. D. Adams, E. W. Myers, P. W. Li, R. J. Mural, G. G. Sutton, H. O. Smith, M. Yandell, C. A. Evans, R. A. Holt, J. D. Gocayne, P. Amanatides, R. M. Ballew, D. H. Huson, J. R. Wortman, Q. Zhang, C. D. Kodira, X. H. Zheng, L. Chen and X. Zhu, *The sequence of the human genome*, Science, 291(5507)(2001), 1304–1351.