# Applications of CS 249B in Biological Sequence Alignment

Managing Genomic Information
Using Named Descriptions

Ali Yahya,
Winter-Spring 2009

# Acknowledgements

This project would not have been possible without the support and inspiration of Professor David R. Cheriton, and Teaching Assistant Nicholas A. Dovidio.

# Abstract

As the field of bioinformatics evolves, and as the amount of available biological data increases, the need will arise for computational methods that address the following two limitations of today's techniques to analyze genomic sequences: First, the inability to efficiently represent, manipulate, and analyze statistically significant samples of biological data within reasonable hardware limitations. And second, the inability to accurately determine the computational resources that are required to perform any experiment given its initial parameters and the magnitude of the input size.

To that end, named sub-descriptions (as described in CS 249B) can be used to breakdown character sequences and consolidate their redundancies. The proposed approach resorts to a combination of sequence alignment algorithms to detect redundancies between sequences, and then converts redundant regions into unique sub-descriptions that can be referred to from multiple locations. In the end, the use of named descriptions provides between 25% and 90% of realtime compression of DNA (and other) sequences in computer memory.

# Contents

# Section 1 ~ Introduction

## 1.1 ~ Background ~ Sequencing Technologies

Nowadays, comprehensive human DNA sequencing is starting to become a process that is both fast and low-cost. In the past decade, the amount of time and capital required to fully sequence the three billion base-pairs of an individual's DNA using Sanger Sequencing[1] (the current industry standard) has remained at six months and one million dollars. However, breakthroughs have been made, and those numbers are poised to radically fall by 2010. An example of said breakthroughs can be observed in the technology developed by Pacific Biosciences, a startup biotechnology company that is venture funded by Kleiner Perkins Caufield & Byers and Mohr Davidow Ventures, among others. Pacific Biosciences is developing what they call "a transformative Single Molecule Real Time (SMRT)" DNA Sequencing platform, and already has a prototype of a DNA sequencer that is projected to reduce the aforementioned numbers from six months to less than hour, and from one million dollars to less than $50.[2]

In 2003, the Human Genome Project was completed, and has thereafter become the foundation for much of the work done in bioinformatics. Subsequent projects, including the International HapMap Project, have revealed that even slight variations in genetic code between individuals can have important implications about their biological characteristics, as well as for the development of healthcare and treatment of disease. As a result, the already two billion dollar market for genomic sequencing technologies can be projected to expand as technology capabilities increase, and costs decrease. [3]

## 1.2 ~ Opportunity in Computing

The genetic code of every organism can be represented as a sequence of characters. Each of which can represent one of four base-pairs in DNA & RNA, or one of the 20 amino acids in a protein. The field of bioinformatics today subscribes to the use of computation to represent, analyze, and manipulate said sequences with the goal of learning about and documenting the role of genetics in an organism's biological characteristics. With the advent of faster and more cost-effective sequencing technologies, the analysis of disease and of biological traits in organisms becomes a computer science problem, for as the amount of data becomes virtually unbounded, and the biological samples with which

---

[1] Sanger, Frederick. *Determination of Nucleotide Sequences in DNA ~ Nobel lecture*. Dec. 8, 1980 (accessed April 1, 2009) http://nobelprize.org/nobel_prizes/chemistry/laureates/1980/sanger-lecture.pdf

[2] Martin, Hugh. *13 Mistakes and 13 Brilliant Strokes*. Stanford's Entrepreneurial Thought Leaders Seminar. 01/14/2009. (accessed 03/23/09). http://ecorner.stanford.edu/authorMaterialInfo.html?mid=2106

[3] Pacific Biosciences, Company Backgrounder. (accessed 04/15/09) http://www.pacificbiosciences.com/assets/files/pacbio_backgrounder.pdf

computer scientists and bioinformaticians work becomes statistically significant, the new challenge becomes to find a way to effectively process and find meaningful patterns in the emerging universe of genomic information.

# Section 2 ~ Problem

As the field of bioinformatics evolves, and as biological data becomes more readily available, the need will arise for computational methods that address the following two limitations of today's techniques to analyze genomic sequences:

(i) the inability to efficiently represent, manipulate, and analyze statistically significant samples of biological data within reasonable hardware limitations,

(ii) the inability to accurately determine the computational resources that are required to perform any experiment given its initial parameters and the magnitude of the input size.

To address the first point, let us consider the sheer magnitude of even a single sequence of human DNA. A complete representation of an individual human's genome consists of a string that is approximately three billion characters long, each of which represents a base-pair of single-stranded DNA that can be any of Adenine (A), cytosine (C), guanine (G), or thymine (T). Because there exist four possibilities for each character, the representation of each base-pair in computer memory requires a very minimum of two bits. When multiplied by the number of base-pairs in the human genome, we get the following result:

$$2_{\text{ bits}} * 3e10^{9}{}_{\text{ base pairs}} \approx 6e10^{9}{}_{\text{ bits}} \approx 750_{\text{ megabytes}}$$

And, while being able to represent the very essence of a human being with less than a gigabyte of computer memory sounds reasonable at first, it takes one only a couple searches on Google to realize that, even with today's prohibitively expensive sequencing technologies, over 100 million sequences containing over 99 billion base pairs are stored by only one of the major online databases of genomic information, GenBank.[4] And, as the cost of fast sequencing technologies reaches its trough in the coming decades, those numbers will inevitably explode. Therefore, it becomes clear that, even with the latest developments in volatile and non-volatile computer memory, there is a dire need for efficient computational methods for the representation and manipulation of genomic information. In particular, if the intent is to perform statistical analyses on a large number of genomic sequences via computation, it is desirable to be able to represent said sequences within the limits of a system's available DRAM while sacrificing neither the integrity of the data, nor its efficient accessibility.

---

[4] GenBank Statistics (1982 - 2008). National Center for Biotechnology Information. (accessed 04/16/09) http://www.ncbi.nlm.nih.gov/Genbank/genbankstats.html

Now that we've discussed the primary limitation of today's computational methods in bioinformatics, let's consider the second point. Namely, that it is also desirable to be able to accurately determine the computational resources that are required to perform an arbitrary experiment given its parameters and input size. Otherwise, one will necessarily incur the costs of, either having to overcompensate with hardware for the uncertainty of resource requirements, or having to risk the failure or delay of an experiment. For the most part, today's computational methods in bioinformatics are indiscriminate about the way the data on which they operate is arranged in memory and do little to keep track of the system's worst-case execution patterns. Consequently, as instances of complex value-types are created and copied by a program, the memory demands on the system become increasingly indeterminate. Therefore, there is a need for a mechanism that not only optimizes the allocation of biological descriptions in memory, but that is also more predictable than today's solutions.

# Section 3 ~ Steps to a Solution

## 3.1 ~ Relevant Insights about Biological Sequences

"Nature is a tinkerer and not an inventor." [5]

Evolution is inherently incremental. Molecular structures such as DNA are adapted—through mutation—from preexisting structures, rather than invented from scratch.[6] As a result, the molecular structure and function of an organism's genetic code is inextricably correlated to that of its ancestors and, by transitivity, to that of organisms who share any of those ancestors. And, because the molecular structures of DNA, RNA and proteins can be reduced to sequences of characters, those correlations are visible at a computational level.

On the grand scheme of things, one defining characteristic of biological sequences is their tendency to be extremely similar to related sequences—a tendency that is especially pronounced when the sequences in question originate from the same species, or from organisms with a close common ancestor.[7] In the case of human beings, it is estimated that only ~0.4% of the genetic code of unrelated people differ. When considered at the nucleotide level, genetic variation among individual humans ranges between 0.05% and 0.1%—only one mutation per 1000 base pairs.[8] Hence, while it may take as much as 750

---

[5] Jacob, 1977

[6] Durbin, R., Eddy, S., Krogh A., Mitchison G. *Biological Sequence Analysis*. Cambridge University Press, 1998.

[7] Hunter, Lawrence. *Artificial Intelligence & Molecular Biology*, Molecular Biology for Computer Scientists, Chapter 1. p. 6 - 10, 24 - 32. (accessed 04/2/09). http://www-helix.stanford.edu/bmi214/hunter.pdf

[8] Tang, Hua. Genetic Structure, Self-Identified Race/Ethnicity, and Confounding in Case-Control Association Studies. February 2005. http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=1196372.

megabytes of memory to represent a single human genome, inter- and intrasequence redundancies can be consolidated such that the amount of memory required to represent the genomes of *N* people is reduced from, what would otherwise be *N* * 750 $_{\text{megabytes}}$, to something that is several order of magnitude smaller [9].

## 3.2 ~ Relevant CS 249 Software Engineering Principles

One of the fundamental tenets of CS 249 is that, when modeling complex, real-world entities and attempting to simulate and predict their behavior, it becomes essential to abstract the complexity of the real-world referent system by limiting the granularity and scope to which said entities are described. Therefore, a description of an entity is a focused subset of the important details about the entity, rather than its comprehensive definition.[10]

The concept of a *named description* arises because, in certain cases, the inherent magnitude and complexity of a value-type makes its representation in memory and nominal operations (e.g. assignment and equality), prohibitively expensive. One solution to this complex value-type problem uses named descriptions to implement a type as a composition of pointers to shared, hidden sub-descriptions.[11]

As an illustration of this approach: A sequence of characters that represents a genetic molecular structure can be implemented as a list of pointers. Each pointer refers to a sub-description containing a string of characters that occur somewhere in the sequence. Each sub-description can be used multiple times—either within a single sequence or between multiple sequences—to represent the occurrence of those characters.

This design approach offers two significant benefits that are relevant to this project:

(i) First, the use of named descriptions allows different instances of large, complex value-types to share components that they have in common by referring (via the pointers they encapsulate) to the same sub-descriptions in memory,

(ii) Second, the use of named descriptions simplifies the implementation of the value-type's copy constructor and assignment/comparison operators from what would otherwise be an operation on the several million locations in memory that define the description's value, to an operation on the far smaller set of pointers that refer to the value's underlying sub-descriptions.

---

[9] See test results in the appendix

[10] Cheriton, David. *Object-Oriented Programming from a Modeling and Simulation Perspective,* 455. Winter, 2009. http://cs249b.stanford.edu.

[11] Cheriton, 456

# Section 4 ~ Solution Implemented

## 4.1 ~ Overview

For the purpose of memory optimization, we are concerned with segments within sequences that perfectly correspond to one another. Whenever that is the case—namely, when a base pair in one segment matches that of the other—a redundancy can be eliminated, for only one copy (that can be referred to from both locations) is necessary.

The challenge is that corresponding segments between biological sequences are often separated by non-corresponding segments of variable length. As an example, consider the following two sequences:

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

Note that the first five letters, as well as the last four letters, of the sequences have one-to-one correspondence. However, letters six through eight “ACG” in the first sequence do not exist in the second. This is often referred to as an “insertion mutation” on the first sequence. Ideally, an alignment algorithm processing these two sequences should divide the second sequence so that the corresponding regions are aligned. As a divider, the algorithm can insert gaps, represented by the “–” character, into sequence *B* at each location of the letters “ACG” in sequence *A* to indicate their absence. The following is an illustration of an optimal alignment:

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

Given the above alignment, sequence *B (ATCCGTCGG)* can be represented in memory as the concatenation of two sub-descriptions. The first five letters of sequence *A* can serve as the first sub-description, and the last four letters of sequence *A* can serve as the other. The following section provides a more detailed account of this idea.

## 4.2 ~ Representation of Sequences using Named Descriptions

> Note that this section assumes that we have implemented a sequence alignment algorithm that produces alignments that are optimal for compression. The challenge of implementing said algorithm is addressed in the next section.

In a given alignment of two sequences, it is possible to convert matching (or redundant) regions into unique sub-descriptions that can be referred to from both sequences. This is best illustrated with an example. Consider the following two similar sequences:

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

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

The optimal alignment of the sequences above for memory optimization is the following:

```
AAAGT--TGCA--TTG-GCATTG
A---TACTGCACGTTGCG--TTG
```

Note that the regions of redundancy above are indicated with a light shade of grey. From the above alignment, it is not difficult optimize for memory by storing the following information, instead of the two complete sequences:

The unique sub-descriptions that are referred to (as character arrays) from both sequences are indicated by the bold font.

### 4.2.1 ~ *Implementation strategy*

One strategy to consolidate redundancies between two aligned sequences when storing them in memory, is to treat sequence *A* as the base sequence and store it in its entirety. Then, it is only necessary to store, from sequence *B,* only what doesn't appear in *A*. This strategy leads to two invariants:

(i) Gap characters in sequence *A* are placeholders for characters in sequence *B* that do not appear in *A*. The occurrence of gaps in sequence *A* are referred to as insertions (of non-gap characters) into sequence *B*. The following example illustrates the insertion of two characters, T and G, into sequence *B*:

```
A:  A G - - C T
B:  A G T G C T
```

(ii) Conversely, gap characters in sequence *B* are placeholders for characters in sequence *A* that do not appear in *B*. The occurrence of gaps in sequence B are referred to as deletions (of non-gap characters) from sequence *B*[12]. The following example illustrates the deletion of two characters, T and G, from sequence *B*:

```
A:  A G T G C T
B:  A G - - C T
```

Now that high-level idea of our compression strategy has been explained, let us the amount of memory (in bytes) that it requires to represent two sequences in memory. For simplicity, we will assume that the amount of memory required to store a single base is one byte. Note

[12] Note that, because sequence A is treated as the base sequence, operations on either sequence are expressed as the steps that must be taken to obtain sequence B from A.

that it must be possible to reproduce the original two sequences from the compressed representation of alignment in memory. There are four factors that directly affect the amount of storage space that is required:

(i) Length of the base sequence—namely, sequence *A*. The base sequence must be stored in its entirety. Memory required:
***Length_SeqA*** *bytes*

(ii) Number of character insertions into sequence *B*. Characters inserted into sequence *B* do not appear in *A*, and must therefore be stored independently. Note that the number of character insertions into sequence *B* is equal to the number of gap insertions into sequence A. Memory required:
***NumGaps_SeqA*** *bytes*

> ***Definition***: a *gap segment* is a substring in a sequence that is composed entirely from gaps, and is delimited on both sides by either the end of the sequence, or a non-gap character.

(iii) Number of gap segments in sequence *A*. Gap segments in sequence *A*, represent character insertions in sequence *B*. In addition to storing the characters themselves (see ii), it is also necessary to track the starting index-location of each insertion. This can be done using an unsigned 8-bit[13] integer (uint8).[14] Memory required:
***NumGapSegments_SeqA*** *bytes*

(iv) Number of gap segments in sequence *B*. Gap segments in sequence *B*, represent character deletions from sequence *B*. It is not necessary to store the value of each character deleted because they are already stored as part of the base sequence. It is, however, necessary to store the start and end index-locations of each deletion. Similarly, each can be represented with an unsigned 8-bit integer. Memory required:
***2 * NumGapSeqments_SeqB*** *bytes*

From the above four factors we can derive the following formula for the amount of memory required to represent two sequences in memory using our compression strategy:

---

[13] 8 bits = 1 byte

[14] this assumes that the original length of each sequence is no greater than 128 characters. Longer sequences can be compressed by running the algorithm repeatedly, on 128 character sub-sequences.

$$MemoryRequired_{\text{bytes}} = Length\_SeqA + NumGaps\_SeqA + NumGapSegments\_SeqA + 2 * NumGapSegments\_SeqB$$

One observation is that no compression is ever done in the storage of sequence *A*. Sequence *A* serves as the data-source with which *B* is consolidated, and as such, must always be stored in its entirety. Therefore, it is useful to frame our compression metric in terms of the amount of memory that is required to store sequence *B*, given that *A* is already in memory. *Percent compression on sequence B given sequence A*, is defined as:

$$1 - \{ (NumGaps\_SeqA + NumGapSegments\_SeqA + 2 * NumGapSegments\_SeqB) / (OriginalLength\_SeqB) \}$$

In other words, the expression above is a metric of comparison between the storage space required to compress and store sequence *B* given sequence *A*, and the storage space required to store sequence *B* uncompressed.

## 4.3 ~ Sequence Alignment Algorithm

The implementation of the sequence alignment algorithm for this project has gone through three major development cycles. The first cycle consisted of the implementation of the simplest form of the Needleman-Wunsch pairwise alignment algorithm. The second cycle consisted of enhancing the algorithm of the first cycle by adding support for variable gap penalties. The third and final cycle consisted of the key milestone to enable the production of true optimal alignments. Namely, bridging the semantic divide (and enforcing the equivalence) between the score that is given to each alignment, and the amount of memory overhead that is required to compress and store that alignment. The following three sections describe the details and implementation of each iteration.

### 4.3.1 ~ *First Generation ~ Needleman Wunsch*

In its simplest form, the Needleman-Wunsch algorithm[15] carries out a global alignment between two sequences. The algorithm was published in 1970 by Saul Needleman and Christian Wunsch, and was the first application of dynamic programming to sequence alignment in bioinformatics.

The fundamental idea behind the Needleman Wunsch algorithm is that a global, best alignment between two sequences can be obtained incrementally. The algorithm is recursive in nature, for the best score of an alignment between two arbitrary sequences of

[15] Needleman, S., Wunsch C. A general method applicable to the search for similarities in the amino acid sequence of two proteins. 1970. http://bit.ly/needleman-wunsch.

length *L* is expressed in terms of the score of the best alignment of subsequences with length *L* – 1.

In order to compute the best alignment between sequence *A* and sequence *B*, the Needleman Wunsch algorithm constructs a matrix *F* with a row for every element in *A*, and a column for every element in *B*. *F* also has one additional row and one additional column such that *F(0,0)* belongs to neither sequence and represents the base case. *F* is indexed by *i* and *j*, such that *F(i, j)* is the score of the best alignment possible between the first *i* elements of *A*, and the first *j* elements of *B.* The score value of *F(i, j)* depends on the scores of the cells *F(i – 1, j – 1)*, *F(i – 1, j)*, and *F(i, j – 1)*.

Before explaining the mathematical definition of that dependence, let's go over the reasons for which it exists. For any position (*i, j)* in matrix *F*, there are three possible scenarios for the best alignment between elements $x_1$, $x_2$, … ,$x_i$ of sequence *A*, and elements $y_1$, $y_2$, … ,$y_j$ of sequence *B*:

(i) in the first scenario, xi and yi are aligned to each other,

(ii) in the second scenario, xi is aligned to a gap, or

(iii) in the third scenario, yj is aligned to a gap.

The following is an illustration of each case:

| | | |
|---|---|---|
| A C T $x_i$ | A C T G $x_i$ | A C $x_i$ – – |
| A C T $y_j$ | A C $y_j$ – – | A C T G $y_j$ |

When computing the score value for *F(i, j)* the algorithm selects, from the three alternatives, the one with the maximum score value. This is illustrated by the following expression:

(Figure 1) [16]

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + s(x_i, y_j), \\ F(i-1,j) - d, \\ F(i,j-1) - d. \end{cases}$$

The first alternative indicates that $x_i$ and $y_i$ are aligned, the second alternative indicates that $x_i$ is aligned to a gap, and the third alternative indicates that $y_i$ is aligned to a gap. $s(x_i, y_i)$ is the match score for $x_i$ *and* $y_i$, and *d* is the gap penalty.

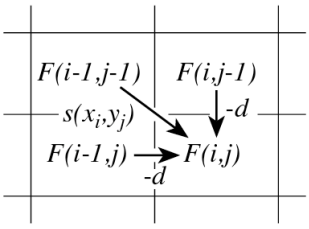
Note that, when used for biological purposes, the Needleman-Wunsch algorithm depends on a score matrix (e.g. BLOSUM62 or PAM) that is given to the program as input so that it can lookup a score value for every possible match of characters. However, for the purpose of memory optimization, we care only about whether or not each pair of characters

[16] Durbin, R., Eddy, S., Krogh A., Mitchison G. *Biological Sequence Analysis*. 20. Cambridge University Press, 1998.

matches, and not about what a match may say about the biological relationship between the sequences. Consequently, we can define three constant values, *kMatchScore*, *kMismatchScore*, and *kGapPenalty*, and refer to their values instead of using a score matrix. In order to favor matches over gaps, *kMatchScore* can be defined as 1, and *kGap* as 0. And, since mismatches should not be tolerated, *kMismatchScore* should be a large negative number (e.g. –1,000,000 or FLT_MIN[17]).

Matrix *F* can be filled recursively from top left to bottom right because the score of each cell (i, j) depends only on the score of the cell right above it (i – 1, j), the cell to its left (i, j – 1), and the cell above it along the diagonal (i – 1, j – 1). This is illustrated in the following diagram from *Biological Sequence Analysis* by Durbin et. al:

(Figure 2) [18]




As matrix *F* is filled, the algorithm keeps track, at every location *F(i, j)*, of the cell from which it was derived—namely, *F(i, j)*, *F(i, j – 1)*, or *F(i – 1, j)*. Using C++, this is easily accomplished with pointers. Let's reconsider the alignment that we used as an example before:

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

The following is the fully populated matrix *F* along with its traceback of pointer dependencies assuming a *kMatchScore* of 1, a *kMismatchScore* of FLT_MIN, and a *kGapPenalty* of 0:

[17] In C++, FLT_MIN is defined under the <float.h> header and is equal to the smallest normalized, finite representable value of type float.

[18] Durbin et. al. 21.

(Figure 3)

| | | A | T | C | C | G | T | C | G | G |
|---|---|---|---|---|---|---|---|---|---|---|
| | **0** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **A** | 0 | **1** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **T** | 0 | 1 | **2** | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| **C** | 0 | 1 | 2 | **3** | 3 | 3 | 3 | 3 | 3 | 3 |
| **C** | 0 | 1 | 2 | 3 | **4** | 4 | 4 | 4 | 4 | 4 |
| **G** | 0 | 1 | 2 | 3 | 4 | **5** | 5 | 5 | 5 | 5 |
| **A** | 0 | 1 | 2 | 3 | 4 | **5** | 5 | 5 | 5 | 5 |
| **C** | 0 | 1 | 2 | 3 | 4 | **5** | 5 | 6 | 6 | 6 |
| **G** | 0 | 1 | 2 | 3 | 4 | **5** | 5 | 6 | 7 | 7 |
| **T** | 0 | 1 | 2 | 3 | 4 | **5** | 6 | 6 | 7 | 7 |
| **C** | 0 | 1 | 2 | 3 | 4 | **5** | 6 | 7 | 7 | 7 |
| **G** | 0 | 1 | 2 | 3 | 4 | **5** | 6 | 6 | 8 | 8 |
| **G** | 0 | 1 | 2 | 3 | 4 | **5** | 6 | 6 | 8 | 9 |

The shaded cells above represent matches between the two sequences. The score in the bottom-right corner of the cell is, by definition, the score of the best possible alignment between the two sequences. The path that is traced by the cells in bold is the *traceback* that produces the alignment in the example:

```
A T C C G A C G T C G G
A T C C G - - - T C G G
```

The traceback is computed after the matrix has been filled, and works by following the path of choices (see *figure 1*) that led to the final score. While executing the traceback, the alignment of the two sequences is built in reverse. We start at the bottom-right corner and follow the pointers that are stored in each cell. Every time the traceback makes a step in the direction of *(i – 1, j – 1)* we add both pairs of symbols, $x_i$ and $y_j$ to the front of the alignment. Whenever the traceback makes a step in the direction of *(i – 1, j)* we add $x_i$ to sequence *A*, and a gap "–" to sequence *B*. Conversely, whenever the traceback makes a step in the direction of *(i, j – 1)* we add a gap to sequence *A*, and $y_j$ *to* sequence *B.*[19]

It is important to note that the alignment above is given a score of 9 by the algorithm, but is not the only alignment with that score. The reason for which more than one alignment exists is that, for certain cells of matrix F, the scores that are produced for the three alternatives by the expression in *figure 1* might not all be different from each other. As a

[19] Durbin, R., Eddy, S., Krogh A., Mitchison G. *Biological Sequence Analysis*. 22. Cambridge University Press, 1998.

result, certain cells of the matrix store multiple pointer dependencies which represent different (but with equal score) potential paths for the traceback. The following is a list the three other alignments for sequences *A* and *B* that have a score of 9:

```
A T C C G A C G T C G G
A T _ C _ _ C G T C G G

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

A T C C G A C G T C G G
A T C _ _ _ C G T C G G
```

## *Summary of Test Results (first generation)*

*Example, Sequence Test Run 1.0* ~ The algorithm described was tested on the first 64 characters of the following two sequences. For this test run we used the following parameter values[20]:

| | | |
|---|---|---|
| *kMatchScore* | = | 1 |
| *kMismatchScore* | = | *FLT_MIN* |
| *kGapPenalty* | = | 0 |

Sequence A

Species: Balaenoptera Musculus (Blue Whale)
Region: First 64 characters, mitochondrial DNA, complete genome
Accession #: NC_001601[21]
Sequence:

GTTAATTACTAATCAGCCCATGATCATAACATAACTGAGGTTTCATACATTTGGTATTTTTTTA

Sequence B

Species: Balaena Mysticetus (Bowhead Whale)
Region: First 64 Characters, mitochondrial DNA, complete genome
Accession #: AP006472[22]
Sequence:

GTTAATGTAGCTTAAATATTTATAAAGCAAAACACTGAAAATGTTTAGATGGGTTTAATTAACC

[20] See Appendix B for more information about any test run

[21] National Center for Biotechnology Information. Record Accession #NC_001601. http://bit.ly/OBkcS.

[22] National Center for Biotechnology Information. Record Accession #AP006472. http://bitly.com/wBxsD.

Sequence Test Run 1.0 ~ Results

| | |
|---|---|
| Alignment score | 43.00 |
| Number of alignments produced | 22,464 |
| Original length of sequence A | 64 |
| Original length of sequence B | 64 |
| Length of best alignment | 85 |
| Number of gaps inserted into sequence A | 21 |
| Number of gaps inserted into sequence B | 21 |
| Number of gap segments in sequence A | 12 |
| Number of gap segments in sequence B | 9 |
| Percent compression on sequence B given sequence A | **20.31%** |

Best alignment:

```
A: GTTAAT-TA-CT--AATCAGCCCA-TGATCATAA--CATA-A- ...
B: GTTAATGTAGCTTAAAT------ATT--T-ATAAAGCA-AAAC ...

... -CTGAGGTTTCATACAT-TT--GGTAT---TTTT--TT-A--
... ACTGA------A-A-ATGTTTAG--ATGGGTT-TAATTAACC
```

## *Explanation & Analysis of Test Results (first generation)*

Alignment score

The scoring model that the first generation of the algorithm uses is simple. Because it imposes no penalty on the occurrence of gaps or their location, the resulting score depends only on the number of one-to-one base-pair matches between the two sequences of the alignment. A score of 43.00 indicates that 43 bases in sequence A are aligned to corresponding bases in sequence B. Note that the alignment score is equal to the original length of one sequence minus the number of gaps in the other:

$$Score = Original\_Length\_SeqA - Gaps\_SeqB, \text{ or}$$
$$Score = Original\_Length\_SeqB - Gaps\_SeqA$$

Alignments produced

In its present state, the algorithm is indifferent to the manner in which gaps are introduced into the alignment. Consequently, there are many ways to insert gaps into each sequence and maintain the same score. For this pair of sequences, the algorithm produced 22,464 unique alignments, each with 43 one-to-one base-pair correspondences (and an alignment score of 43.00).

Length of Best Alignment

The length of the best alignment produced by the algorithm is inversely correlated to the amount of compression in memory that can be achieved by consolidating redundancies between the two sequences. In the worst of cases—namely, an alignment with zero base-pair matches—the alignment length would be the sum of the lengths of the original sequences. In this example, the worst-case length would be 128, for both sequences have

an original length of 64. The best alignment produced by the algorithm has a length of 85, which is equal to the worst-case length (128) minus the number of base-pair matches in the alignment (43).

Number of Gap Segments

The number of gap segments is also inversely correlated to the amount of compression that can be achieved on two sequences from a given alignment. A high number of gap segments implies a highly fragmented alignment, which translates to fragmentation in memory which makes the use of space less efficient. Moreover, any strategy to consolidate redundancies between the two sequences requires keeping track of the start and end indexes of each gap segment. Therefore, a high number of gap segments translates to a large amount of storage overhead. In this example, the total number of gap segments is 21 —12 gap segments in sequence A, and 9 gap segments in sequence B.

Percent Compression on Sequence B, given Sequence A

This value is calculated from the expression derived in section 4.2.1

$$MemoryRequired_{\text{bytes}} = Length\_SeqA + NumGaps\_SeqA + NumGapSegments\_SeqA + 2 * NumGapSegments\_SeqB$$

In our test run example, the above expression evaluates to the following:

$$1 - \{ (21 + 12 + 2 * 9) / (64) \} = \textbf{20.31\% compression}$$

Best Alignment

In order to identify the best alignment from the 22,464 that were produced, we simply sort them in descending order, in terms of percent compression. As a result, the first item on the list of alignments is the best possible alignment for the given sequences.

### 4.3.2 ~ *Second Generation ~ Affine Gap Penalties*

The first generation of our sequence alignment algorithm relies on a simplistic gap scoring model—namely, one in which all gaps are regarded equal regardless of where they are located in the sequence. This model is not ideal for the following two reasons:

(i) First, because of its indiscriminate approach to gap insertion, the first generation of the algorithm produces an exorbitantly large number of alignments for many sequence pairs. This is a problem because it is expensive to store a large number of alignments, albeit temporarily, during the alignment selection process.

(ii) Second, because of its indifference to the number of gap segments in each sequence, the resulting alignments are suboptimal. As seen in the previous section, the amount of memory required to represent an alignment in memory depends directly on the number of gap segments in each sequence. A more effective approach would minimize the number of gap segments introduced into each alignment by clustering the occurrence of gaps.

It would be ideal to have more fine grained control over the penalty that is incurred every time a gap is introduced. In order to preserve $O(n^2)$ execution time, the standard approach is to assume an affine cost structure[23]; namely,

$$gap\_penalty = -d - (g - 1)e$$

Where *d* is the gap open penalty, *g* is the number of gaps in a gap segment, and *e* is a gap extension penalty. More specifically, an implementation of the above affine cost structure should score the insertion of gaps in terms of:

(i) kGapOpenPenalty_SeqA ~ penalty incurred per gap segment in sequence A

(ii) kGapOpenPenalty_SeqB ~ penalty incurred per gap segment in sequence B

(iii) kGapPenalty_SeqA ~ penalty incurred for a standard gap in sequence A

(iv) kGapPenalty_SeqB ~ penalty incurred for a standard gap in sequence B

In order to accommodate the above requirements, the dynamically programmed model of the first generation of the algorithm must be updated to: first, differentiate between the first gap in a gap segment, and the gaps that follow it; and second, to keep track of the number of consecutive gaps in any given gap segment. This can be accomplished by introducing two new matrices—one dedicated to keeping track of gaps in sequence *A*, and the other to keeping track of gaps in sequence *B*. As a result, the second generation of the algorithm resorts to three matrices to compute an alignment that is more optimized for compression:

---

[23] Durbin, R., Eddy, S., Krogh A., Mitchison G. *Biological Sequence Analysis*. 29. Cambridge University Press, 1998.

```
A C T xi      A C T G xi      A C xi - -
A C T yj      A C yj - -      A C T G yj
```

(i) Matrix *M* ~ let *M(i, j)* be the best score up to (i, j) given that $x_i$ is aligned to $y_i$ (left)

(ii) Matrix *Ix* ~ let *Ix(i, j)* be the best score given that $x_i$ is aligned to a gap (central)

(iii) Matrix *Iy* ~ let *Iy(i, j)* be the best score given that $y_i$ is aligned to a gap (right)

The recurrence relations for the three matrices now become:

$$M(i,j) = \max \begin{cases} M(i-1,j-1)+s(x_i,y_j), \\ I_x(i-1,j-1)+s(x_i,y_j), \\ I_y(i-1,j-1)+s(x_i,y_j); \end{cases}$$

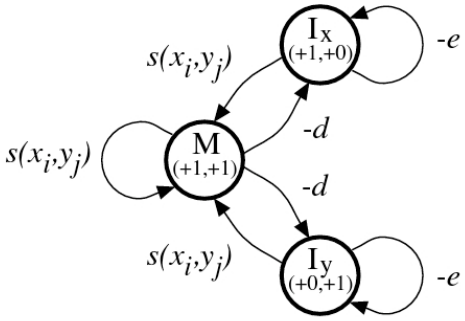
(Figure 3) [24]

$$I_x(i,j) = \max \begin{cases} M(i-1,j)-d, \\ I_x(i-1,j)-e; \end{cases}$$

$$I_y(i,j) = \max \begin{cases} M(i,j-1)-d, \\ I_y(i,j-1)-e. \end{cases}$$

The following diagram illustrates the new gap scoring model as a finite state machine. Each matrix represents a state in which the alignment can be. Note that transitions from matrix *M* to matrix *Ix* or *Iy* (the opening of a gap) result in a penalty of *d*, whereas transitions from *Ix* or *Iy* to themselves (continuing gaps) result in a penalty of *e*.

(Figure 4) [25]




As we had done previously, we can extract the alignment itself from the populated matrices using a traceback procedure.

[24] Durbin, R., Eddy, S., Krogh A., Mitchison G. *Biological Sequence Analysis*. 30. Cambridge University Press, 1998.

[25] ibid

## *Summary of Test Results (second generation)*

*Example, Sequence Test Run 2.0* ~ The algorithm described was tested on the same two sequences tested with the first generation of the algorithm. For this test run we used the following parameter values:

| | | |
|---|---|---|
| *kGapOpenPenalty_SeqA* | = | 0.9 |
| *kGapOpenPenalty_SeqB* | = | 0.9 |
| *kGapPenalty_SeqA* | = | 0.2 |
| *kGapPenalty_SeqB* | = | 0.0 |

Note that the value of *kGapPenalty_SeqA* is greater than that of *kGapPenalty_SeqB* because, gap insertions into sequence *A* are more expensive than gap insertions into sequence *B* (see section 4.2.1).

Sequence A

Species: Balaenoptera Musculus (Blue Whale)
Region: First 64 characters, mitochondrial DNA, complete genome
Accession #: NC_001601[26]

```
GTTAATTACTAATCAGCCCATGATCATAACATAACTGAGGTTTCATACATTTGGTATTTTTTTA
```

Sequence B

Species: Balaena Mysticetus (Bowhead Whale)
Region: First 64 Characters, mitochondrial DNA, complete genome
Accession #: AP006472[27]

```
GTTAATGTAGCTTAAATATTTATAAAGCAAAACACTGAAAATGTTTAGATGGGTTTAATTAACC
```

Sequence Test Run 2.0 ~ Results

| | | Values of 1.0 | % change |
|---|---|---|---|
| Alignment score | 23.40 | 43.00 | |
| Number of alignments produced | 36 | 22,464 | -99.84% |
| Original length of sequence A | 64 | 64 | |
| Original length of sequence B | 64 | 64 | |
| Length of best alignment | 86 | 85 | 1.18% |
| Number of gaps inserted into sequence A | 22 | 21 | 4.76% |
| Number of gaps inserted into sequence B | 22 | 21 | 4.76% |
| Number of gap segments in sequence A | 9 | 12 | -25.00% |
| Number of gap segments in sequence B | 8 | 9 | -11.11% |
| Percent compression on sequence B given sequence A | **26.56%** | **20.31%** | **30.77%** |

[26] National Center for Biotechnology Information. Record Accession #NC_001601. http://bit.ly/OBkcS.

[27] National Center for Biotechnology Information. Record Accession #AP006472. http://bitly.com/wBxsD.

Best alignment:

```
A: GTTAAT-TA-CT-AATCAGCCCATGA--TCATAA--CATA--- ...
B: GTTAATGTAGCTTAA-------AT-ATTT-ATAAAGCA-AAAC ...

... ACTGAGGTTTCATACA--TTT-----GGTATTTTT--TTA---
... ACTGA------A-A-ATGTTTAGATGGG----TTTAATTAACC
```

## *Analysis of Test Results (second generation)*

Alignment score

The scoring model that the second generation of the algorithm uses distinguishes between opening and continuing gaps. It also distinguishes between gaps in sequence *A* and gaps in sequence *B*. The score of the resulting alignment is defined by the following expression:

$$
\begin{aligned}
\text{Score} = \quad & \text{NumGapSegments\_SeqA} * \text{kGapOpenPenalty\_SeqA} + \\
& \text{NumGapSegments\_SeqB} * \text{kGapOpenPenalty\_SeqB} + \\
& \text{kGapPenalty\_SeqA} * (\text{NumGaps\_SeqA} - 1) + \\
& \text{kGapPenalty\_SeqB} * (\text{NumGaps\_SeqB} - 1)
\end{aligned}
$$

Alignments produced

The fine grained scoring model of the second generation algorithm enables it to evaluate each alignment it produces with more precision. As a result, it is able to narrow down its output to fewer and better alignments than its predecessor. Whereas the previous generation produced over 20,000 “equivalent” alignments for the sequence-pair, this generation produces only 36.

Number of Gap Segments [28]

The greatest improvement brought upon by the second generation of the algorithm is in the number of gap segments that are opened in the two sequences. Because the alignment score incurs a high penalty (0.9) every time it opens a gap segment, the algorithm is forced to minimize their occurrence. The result is an alignment with 17 gap segments instead of 21 from the previous generation.

[28] See section 4.2.1 for a definition of “*gap segment*”

### 4.3.3 ~ *Third Generation* ~ *Alignment Score ⇔ Memory Overhead*

The second generation of our sequence alignment algorithm relied on four parameters—*kGapOpenPenalty* and *kGapPenalty* for each sequence—to produce alignments that are near optimal for compression. Nonetheless, the values assigned to the four parameters are somewhat arbitrary and have little to no connection to the memory requirements that each parameter translates into during compression.

The third generation of the sequence alignment algorithm bridges the semantic divide (and enforces the equivalence) between the score that is given to each alignment and amount of memory overhead that would be required to compress and store that alignment. As a result, the alignments produced by the algorithm are perfectly tailored for our compression strategy, and are therefore truly optimal.

The implementation of this algorithm generation is characterized by two major differences:

(i) First, the algorithm seeks to minimize, not maximize, the alignment score. In this scoring model, the score of an alignment represents memory overhead in bytes; hence, alignments with lower scores are more compressible than those with higher scores.

(ii) Second, to make the score of each alignment equal to the memory overhead that is associated with it, the four parameters of the algorithm's scoring model become:

$$
\begin{aligned}
kGapOpenOverhead\_SeqA &= 1 \\
kGapOpenOverhead\_SeqB &= 2 \\
kGapOverhead\_SeqA &= 1 \\
kGapOverhead\_SeqB &= 0
\end{aligned}
$$

While the above modifications may seem trivial, the challenge is to have well defined base cases along the edges of all three matrices such that the traceback takes into account every gap and every gap segment and penalizes accordingly. The reason for the challenge is that, in this model, end gaps[29] are essential, whereas before they were not. For a more elaborate account about the intricacies of end gaps in the *Needleman Wunsch with Affine Gap Penalties* algorithm, refer to *Bioinformatics, Sequence and Genome Analysis*[30] by David W. Mount.

---

[29] gaps at the beginning or end of a sequence

[30] Mount, David W. *Bioinformatics, Sequence and Genome Analysis*, chapter 3. Cold Spring Harbor Laboratory Press, 2004.

## *Summary of Test Results (third generation)*

*Example, Sequence Test Run 3.0* ~ The algorithm described was tested on the same two sequences tested with the first and second generations of the algorithm. For this test run we used the parameter values given above.

Sequence A
Species: Balaenoptera Musculus (Blue Whale)
Region: First 64 characters, mitochondrial DNA, complete genome
Accession #: NC_001601[31]
Sequence:

```
GTTAATTACTAATCAGCCCATGATCATAACATAACTGAGGTTTCATACATTTGGTATTTTTTTA
```

Sequence B
Species: Balaena Mysticetus (Bowhead Whale)
Region: First 64 Characters, mitochondrial DNA, complete genome
Accession #: AP006472[32]
Sequence:

```
GTTAATGTAGCTTAAATATTTATAAAGCAAAACACTGAAAATGTTTAGATGGGTTTAATTAACC
```

Sequence Test Run 3.0 ~ Results

| | | | |
|---|---|---|---|
| Alignment score | 42.00 | 23.40 | |
| Number of alignments produced | 72 | 36 | 100.00% |
| Original length of sequence A | 64 | 64 | |
| Original length of sequence B | 64 | 64 | |
| Length of optimal alignment | 88 | 86 | 2.33% |
| Number of gaps inserted into sequence A | 24 | 22 | 9.09% |
| Number of gaps inserted into sequence B | 24 | 22 | 9.09% |
| Number of gap segments in sequence A | 8 | 9 | -11.11% |
| Number of gap segments in sequence B | 5 | 8 | -37.50% |
| Percent compression on sequence B given sequence A | 34.38% | 26.56% | **29.41%** |

## *Analysis of Test Results (third generation)*

Alignment score

The alignment score is equal to the amount of memory (in bytes) required to store sequence *B* given that sequence *A* is already in memory:

score = *NumGaps_SeqA +*
*NumGapSegments_SeqA +*
*2 * NumGapSegments_SeqB*

[31] National Center for Biotechnology Information. Record Accession #NC_001601. http://bit.ly/OBkcS.

[32] National Center for Biotechnology Information. Record Accession #AP006472. http://bitly.com/wBxsD.

# Section 5 ~ Conclusion

In the end, the use of named descriptions to optimize the amount of memory required to represent character sequences is highly effective. See the Appendix A for the results of an extensive compendium of tests that were performed on all three iterations of the algorithm. The amount of compression that can be achieved on any given pair of sequences ranges from 25% to over 90%, depending on the structural and semantic similarity of the sequences in question.

# Appendix A ~ Test Run Summary & Results

## Sequence Alignment Algorithm ~ First Generation

| **Sequence Test Run 1.0** | | *Organism* | *Region* | *Span* |
|---|---|---|---|---|
| Sequence A | Accession NC_001601 http://bit.ly/OBkcS | Balaenoptera Musculus (Blue Whale) | Mitochondrial DNA | Bases 0 - 64 |
| | GTTAATTACTAATCAGCCCATGATCATAACATAACTGAGGTTTCATACATTTGGTATTTTTTTA | | | |
| Sequence B | GTTAATGTAGCTTAAATATTTATAAAGCAAAACACTGAAAATGTTTAGATGGGTTTAATTAACC | | | |
| | Accession AP006472 http://bit.ly/wBxsD | Balaena Mysticetus (Bowhead Whale) | Mitochondrial DNA | Bases 0 - 64 |
| | | | | |
| **Algorithm Parameters** | | | | |
| Match Score | | 1 | | |
| Mismatch Score | | FLT_MIN | | |
| Gap Penalty | | 0 | | |
| | | | | |
| | | | | |
| **Test Results** | | | | |
| Alignment score | | 43.00 | | |
| Number of alignments produced | | 22,464 | | |
| Original length of sequence A | | 64 | | |
| Original length of sequence B | | 64 | | |
| Length of optimal alignment | | 85 | | |
| Number of gaps inserted into sequence A | | 21 | | |
| Number of gaps inserted into sequence B | | 21 | | |
| Number of gap segments in sequence A | | 12 | | |
| Number of gap segments in sequence B | | 9 | | |
| Percent compression on sequence B given sequence A | | **20.31%** | | |
| **Alignment** | | | | |

```
A: GTTAAT-TA-CT--AATCAGCCCA-TGATCATAA--CATA-A ...
B: GTTAATGTAGCTTAAAT------ATT--T-ATAAAGCA-AAA ...

... --CTGAGGTTTCATACAT-TT--GGTAT---TTTT--TT-A--
... CACTGA------A-A-ATGTTTAG--ATGGGTT-TAATTAACC
```

| **Sequence Test Run 1.1** | | *Organism* | *Region* | *Span* |
|---|---|---|---|---|
| Sequence A | Accession AF304073 http://bit.ly/10deJe | Physeter catodon (Sperm Whale) | cytochrome b, mitochondrial protein | Bases 0 - 100 |
| | ATGACCAACATCCGAAAATCACACCCATTAATAAAAATCATTAACAATGCATTCATCGACCTCCCTACCCCATCAA ACATTTCCTCATGATGAAACTTCG | | | |
| Sequence B | ATGACCAACATCCGAAAAACACACCCATTGATAAAAATCGTCAACAACGCATTCATCGACCTCCCTACTCCATCAA ACATCTCCTCATGATGAAATTTCG | | | |
| | Accession U72040 http://bit.ly/iube4 | Kogia breviceps (Pygmy Sperm Whale) | cytochrome b, mitochondrial protein | Bases 0 - 100 |
| | | | | |
| **Algorithm Parameters** | | | | |
| Match Score | | 1 | | |
| Mismatch Score | | FLT_MIN | | |
| Gap Penalty | | 0 | | |
| | | | | |
| | | | | |
| **Test Results** | | | | |
| Alignment score | | 90.00 | | |
| Number of alignments produced | | 960 | | |
| Original length of sequence A | | 100 | | |
| Original length of sequence B | | 100 | | |
| Length of optimal alignment | | 110 | | |
| Number of gaps inserted into sequence A | | 10 | | |
| Number of gaps inserted into sequence B | | 10 | | |
| Number of gap segments in sequence A | | 7 | | |
| Number of gap segments in sequence B | | 7 | | |
| Percent compression on sequence B given sequence A | | **69.00%** | | |
| **Alignment** | | | | |

```
A: ATGACCAACATCCG-AAAATCACACCCATT-AATAAAAATC---ATTAACAATGC ...
B: ATGACCAACATCCGAAAAA-CACACCCATTGA-TAAAAATCGTCA---ACAA--C ...

... --ATTCATCGACCTCCCTAC-CCCATCAAACAT-TTCCTCATGATGAAACT-TCG
... GCATTCATCGACCTCCCTACTCC-ATCAAACATCT-CCTCATGATGAAA-TTTCG
```

| **Sequence Test Run 1.2** | | *Organism* | *Region* | *Span* |
|---|---|---|---|---|
| Sequence A | Accession EF093041 http://bit.ly/NmVDT | Lagenorhynchus obliquidens (Pacific white-sided dolphin) | cytochrome b, mitochondrial protein | Bases 0 - 100 |
| | ATGACCAACATCCGAAAAACACACCCACTAATAAAAATCCTCAATGACGCATTCATCGACCTACCCACTCCATCTAACATCTCCTCATGATGAAACTTTG | | | |
| Sequence B | ATGACCAACATCCGAAAAACACACCCACTAATAAAAATTATCAATAACGCATTCATTGACCTACCCACTCCATCTAACATCTCCTCATGATGAAACTTTG | | | |
| | Accession AF084057 http://bit.ly/bmrCE | Pseudorca Crassidens (False Killer Whale) | cytochrome b, mitochondrial protein | Bases 0 - 100 |
| | | | | |
| **Algorithm Parameters** | | | | |
| Match Score | | 1 | | |
| Mismatch Score | | FLT_MIN | | |
| Gap Penalty | | 0 | | |
| | | | | |
| | | | | |
| **Test Results** | | | | |
| Alignment score | | 96.00 | | |
| Number of alignments produced | | 1 | | |
| Original length of sequence A | | 100 | | |
| Original length of sequence B | | 100 | | |
| Length of optimal alignment | | 104 | | |
| Number of gaps inserted into sequence A | | 4 | | |
| Number of gaps inserted into sequence B | | 4 | | |
| Number of gap segments in sequence A | | 3 | | |
| Number of gap segments in sequence B | | 3 | | |
| Percent compression on sequence B given sequence A | | **87.00%** | | |
| **Alignment** | | | | |

```
A: ATGACCAACATCCGAAAAACACACCCACTAATAAAAATCCT--CAATGA-CG ...
B: ATGACCAACATCCGAAAAACACACCCACTAATAAAAAT--TATCAAT-AACG ...

... CATTCA-TCGACCTACCCACTCCATCTAACATCTCCTCATGATGAAACTTTG
... CATTCATT-GACCTACCCACTCCATCTAACATCTCCTCATGATGAAACTTTG
```

| **Sequence Test Run 1.3** | | *Organism* | *Region* | *Span* |
|---|---|---|---|---|
| Sequence A | Accession EF093025 http://bit.ly/iK1lw | Lissodelphis Borealis (Northern Right Whale Dolphin) | cytochrome b, mitochondrial | Bases 0 - 100 |
| | ATGACCAACATCCGAAAAACACACCCACTAATAAAAATCCTCAATGACGCATTCATCGACCTACCCACTCCATCTAACATCTCCTCATGATGAAACTTTG | | | |
| Sequence B | ATGACCAACATCCGAAAAACACACCCACTAATAAAAATCCTCAATAACACATTCATCGACCTACCCACTCCATCTAACATCTCCTCATGATGAAACTTTG | | | |
| | Accession AY257155 http://bit.ly/fjF7g | Lagenorhynchus Obscurus (Dusky Dolphin) | cytochrome b, mitochondrial | Bases 0 - 100 |
| | | | | |
| **Algorithm Parameters** | | | | |
| Match Score | | 1 | | |
| Mismatch Score | | FLT_MIN | | |
| Gap Penalty | | 0 | | |
| | | | | |
| | | | | |
| **Test Results** | | | | |
| Alignment score | | 97.00 | | |
| Number of alignments produced | | 2 | | |
| Original length of sequence A | | 100 | | |
| Original length of sequence B | | 100 | | |
| Length of optimal alignment | | 103 | | |
| Number of gaps inserted into sequence A | | 3 | | |
| Number of gaps inserted into sequence B | | 3 | | |
| Number of gap segments in sequence A | | 1 | | |
| Number of gap segments in sequence B | | 2 | | |
| Percent compression on sequence B given sequence A | | **92.00%** | | |
| **Alignment** | | | | |

```
A: ATGACCAACATCCGAAAAACACACCCACTAATAAAAATCCTCAATGA---CG ...
B: ATGACCAACATCCGAAAAACACACCCACTAATAAAAATCCTCAAT-AACAC- ...

... CATTCATCGACCTACCCACTCCATCTAACATCTCCTCATGATGAAACTTTG
... -ATTCATCGACCTACCCACTCCATCTAACATCTCCTCATGATGAAACTTTG
```

| **Sequence Test Run 1.4** | | *Organism* | *Region* | *Span* |
|---|---|---|---|---|
| Sequence A | Accession FJ919248 http://bit.ly/7GDXy | Human rotavirus A | outer capsid protein (VP4) | Bases 0 - 100 |
| | ATTTATAGACAACTTCTCACTAATTACTATTCGGTAGACTTGCATGACGAAATAGAACAGATTGGATCGGAGAAAA CTCAAAATGTGACGGTAAATCCAG | | | |
| Sequence B | GGCTATAAAATGGCTTCGCTCATTTATAGACAGCTTCTCACTAATTCATATTCAGTAGATTTATATGATGAAATAG AGCAAATTGGATCAGAAAAAACTC | | | |
| | Accession EU839962 http://bit.ly/M5ibV | Human rotavirus G9P[8] | outer capsid protein (VP4) | Bases 0 - 100 |
| | | | | |
| **Algorithm Parameters** | | | | |
| Match Score | | 1 | | |
| Mismatch Score | | FLT_MIN | | |
| Gap Penalty | | 0.1 | | |
| Text | | | | |
| | | | | |
| **Test Results** | | | | |
| Alignment score | | 64.90 | | |
| Number of alignments produced | | 50,400 | | |
| Original length of sequence A | | 100 | | |
| Original length of sequence B | | 100 | | |
| Length of optimal alignment | | 131 | | |
| Number of gaps inserted into sequence A | | 31 | | |
| Number of gaps inserted into sequence B | | 31 | | |
| Number of gap segments in sequence A | | 6 | | |
| Number of gap segments in sequence B | | 13 | | |
| Percent compression on sequence B given sequence A | | **37.00%** | | |

**Alignment**

```
A: --------------------ATTTATAGACAA-CTTCTCACTAATT-ACTATTC-GGTAGACT-- ...
B: GGCTATAAAATGGCTTCGCTCATTTATAGAC-AGCTTCTCACTAATTCA-TATTCAG-TAGA-TTT ...

  ... ---TGCATGACGAAATAGA--ACAGATTGGATCGGAGAAAACTCAAAATGTGACGGTAAATCCAG
  ... ATATG-AT---GAAATAGAGCA-A-ATTGGATC--AGAAAA-----AA-----C------T-C--
```

| **Sequence Test Run 1.5** | | *Organism* | *Region* | *Span* |
|---|---|---|---|---|
| Sequence A | Accession DQ022290 http://bit.ly/15pwqT | Panthera leo (lion) | cytochrome b, mitochondrial | Bases 0 - 64 |
| | ATGACCAACATTCGAAAATCACACCCCCTTGTCAAAATTATTAATCACTCATTCATTGATCTTC | | | |
| Sequence B | TGGATTATCCGATATCTACATGCCAACGGAGCCTCCATATTCTTTATCTGTCTATACATGCACG | | | |
| | Accession FJ895266 http://bit.ly/GnXGR | Panthera tigris altaica (Amur tiger) | cytochrome b, mitochondrial | Bases 0 - 64 |
| | | | | |
| **Algorithm Parameters** | | | | |
| Match Score | | 1 | | |
| Mismatch Score | | FLT_MIN | | |
| Gap Penalty | | 0.2 | | |
| | | | | |
| | | | | |
| **Test Results** | | | | |
| Alignment score | | 31.00 | Values on the left have been scaled so that they reflect the alignment of sequence A and B if their length were 100 characters instead of 64. | |
| Number of alignments produced | | 37,800 | | |
| Original length of sequence A | | 100 | | |
| Original length of sequence B | | 100 | | |
| Length of optimal alignment | | 138 | | |
| Number of gaps inserted into sequence A | | 38 | | |
| Number of gaps inserted into sequence B | | 38 | | |
| Number of gap segments in sequence A | | 22 | | |
| Number of gap segments in sequence B | | 17 | | |
| Percent compression on sequence B given sequence A | | **6.00%** | | |
| **Alignment** | | | | |

```
A: ---A-TGA-CC-A-ACAT-TCGAAA-AT--CACAC----CC-CC ...
B: TGGATT-ATCCGATA--TCT----ACATGCCA-ACGGAGCCTCC...

... -T-TGTCAAAATTATTAATCACT--C-ATT-CATTG-ATCTTC-
... ATAT-TC----T--TT-ATC--TGTCTA-TACAT-GCA----CG
```

| **Sequence Test Run 1.6** | | *Organism* | *Region* | *Span* |
|---|---|---|---|---|
| Sequence A | Accession NM_001828 http://bit.ly/rvhic | Homo sapiens (human) | Charcot-Leyden crystal protein | Bases 0 - 64 |
| | ATTTAAATTCTGCAGCTCAGAGATTCACACAGAAGTCTGGACACAATTCAGAAGAGCCACCCA | | | |
| Sequence B | GACAGAAGACTGGACACAATTCCGAAGAGTCGCCCAGAAGGAGAGAACAATGTCATCACTACC | | | |
| | Accession FJ613346 http://bit.ly/uXtDu | Papio anubis (olive baboon) | LGALS14 | Bases 0 - 64 |
| | | | | |
| **Algorithm Parameters** | | | | |
| Match Score | | 1 | | |
| Mismatch Score | | FLT_MIN | | |
| Gap Penalty | | 0.2 | | |
| | | | | |
| | | | | |
| **Test Results** | | | | |
| Alignment score | | 31.80 | Values on the left have been scaled so that they reflect the alignment of sequence A and B if their length were 100 characters instead of 64. | |
| Number of alignments produced | | 2,250 | | |
| Original length of sequence A | | 100 | | |
| Original length of sequence B | | 100 | | |
| Length of optimal alignment | | 134 | | |
| Number of gaps inserted into sequence A | | 34 | | |
| Number of gaps inserted into sequence B | | 34 | | |
| Number of gap segments in sequence A | | 17 | | |
| Number of gap segments in sequence B | | 17 | | |
| Percent compression on sequence B given sequence A | | **15.00%** | | |
| **Alignment** | | | | |

```
A: -----ATTTAA-ATTCTG--CAGC---T-C--AGAGATT--CAC ...
B: GACAG----AAGA--CTGGACA-CAATTCCGAAGAG--TCGC-C...

... ACAGAAGTCTG---GACACAAT-TCAGA--AGAGCC-ACCCA
... -CAGAAG---GAGAGA-ACAATGTC--ATCA----CTAC-C-
```

# Sequence Alignment Algorithm ~ Second Generation

| **Sequence Test Run 2.0** | | *Organism* | *Region* | *Span* |
|---|---|---|---|---|
| Sequence A | Accession NC_001601 http://bit.ly/OBkcS | Balaenoptera Musculus (Blue Whale) | Mitochondrial DNA | Bases 0 - 64 |
| | GTTAATTACTAATCAGCCCATGATCATAACATAACTGAGGTTTCATACATTTGGTATTTTTTTA | | | |
| Sequence B | GTTAATGTAGCTTAAATATTTATAAAGCAAAACACTGAAAATGTTTAGATGGGTTTAATTAACC | | | |
| | Accession AP006472 http://bit.ly/wBxsD | Balaena Mysticetus (Bowhead Whale) | Mitochondrial DNA | Bases 0 - 64 |
| **Algorithm Parameters** | | | | |
| Match Score | | 1 | | |
| Mismatch Score | | FLT_MIN | | |
| Sequence A ~ Gap Open Penalty | | 0.9 | | |
| Sequence B ~ Gap Open Penalty | | 0.9 | | |
| Sequence A ~ Gap Extension Penalty | | 0.2 | | |
| Sequence B ~ Gap Extension Penalty | | 0.0 | | |
| **Test Results** | | | **Values of 1.0** | **% change** |
| Alignment score | | 23.40 | 43.00 | |
| Number of alignments produced | | 36 | 22,464 | -99.84% |
| Original length of sequence A | | 64 | 64 | |
| Original length of sequence B | | 64 | 64 | |
| Length of optimal alignment | | 86 | 85 | 1.18% |
| Number of gaps inserted into sequence A | | 22 | 21 | 4.76% |
| Number of gaps inserted into sequence B | | 22 | 21 | 4.76% |
| Number of gap segments in sequence A | | 9 | 12 | -25.00% |
| Number of gap segments in sequence B | | 8 | 9 | -11.11% |
| Percent compression on sequence B given sequence A | | **26.56%** | 20.31% | **30.77%** |
| **Alignment** | | | | |

```
A: GTTAAT-TA-CT--AATCAGCCCA-TGATCATAA--CATA-- ...
B: GTTAATGTAGCTTAAAT------ATT--T-ATAAAGCA-AAA ...

... -ACTGAGGTTTCATACAT-TT------GGTATTTTT--TTA
... CACTGA------A-A-ATGTTTAGATGGG----TTTAATTA
```

| **Sequence Test Run 2.1** | | *Organism* | *Region* | *Span* |
|---|---|---|---|---|
| Sequence A | Accession AF304073 http://bit.ly/10deJe | Physeter catodon (Sperm Whale) | cytochrome b, mitochondrial protein | Bases 0 - 100 |
| | ATGACCAACATCCGAAAATCACACCCATTAATAAAAATCATTAACAATGCATTCATCGACCTCCCTACCCCATCAA ACATTTCCTCATGATGAAACTTCG | | | |
| Sequence B | ATGACCAACATCCGAAAAACACACCCATTGATAAAAATCGTCAACAACGCATTCATCGACCTCCCTACTCCATCAA ACATCTCCTCATGATGAAATTTCG | | | |
| | Accession U72040 http://bit.ly/iube4 | Kogia breviceps (Pygmy Sperm Whale) | cytochrome b, mitochondrial protein | Bases 0 - 100 |
| **Algorithm Parameters** | | | | |
| Match Score | | 1 | | |
| Mismatch Score | | FLT_MIN | | |
| Sequence A ~ Gap Open Penalty | | 0.9 | | |
| Sequence B ~ Gap Open Penalty | | 0.9 | | |
| Sequence A ~ Gap Extension Penalty | | 0.2 | | |
| Sequence B ~ Gap Extension Penalty | | 0.0 | | |
| **Test Results** | | | **Values of 1.1** | **% change** |
| Alignment score | | 76.80 | 90.00 | |
| Number of alignments produced | | 384 | 960 | -60.00% |
| Original length of sequence A | | 100 | 100 | |
| Original length of sequence B | | 100 | 100 | |
| Length of optimal alignment | | 110 | 110 | 0.00% |
| Number of gaps inserted into sequence A | | 10 | 10 | 0.00% |
| Number of gaps inserted into sequence B | | 10 | 10 | 0.00% |
| Number of gap segments in sequence A | | 7 | 7 | 0.00% |
| Number of gap segments in sequence B | | 7 | 7 | 0.00% |
| Percent compression on sequence B given sequence A | | **69.00%** | 69.00% | **0.00%** |
| **Alignment** | | | | |

```
A: ATGACCAACATCCGA-AAATCACACCCATT-AATAAAAATC---ATTAACAATGC ...
B: ATGACCAACATCCGAAAAA-CACACCCATTGA-TAAAAATCGTCA---ACAA--C ...

... --ATTCATCGACCTCCCTAC-CCCATCAAACAT-TTCCTCATGATGAAACT-TCG
... GCATTCATCGACCTCCCTACTCC-ATCAAACATCT-CCTCATGATGAAA-TTTCG
```

| **Sequence Test Run 2.2** | | *Organism* | *Region* | *Span* |
|---|---|---|---|---|
| Sequence A | Accession EF093041 http://bit.ly/NmVDT | Lagenorhynchus obliquidens (Pacific white-sided dolphin) | cytochrome b, mitochondrial protein | Bases 0 - 100 |
| | ATGACCAACATCCGAAAAACACACCCACTAATAAAAATCCTCAATGACGCATTCATCGACCTACCCACTCCATCTAACATCTCCTCATGATGAAACTTTG | | | |
| Sequence B | ATGACCAACATCCGAAAAACACACCCACTAATAAAAATTATCAATAACGCATTCATTGACCTACCCACTCCATCTAACATCTCCTCATGATGAAACTTTG | | | |
| | Accession AF084057 http://bit.ly/bmrCE | Pseudorca Crassidens (False Killer Whale) | cytochrome b, mitochondrial protein | Bases 0 - 100 |
| **Algorithm Parameters** | | | | |
| Match Score | | 1 | | |
| Mismatch Score | | FLT_MIN | | |
| Sequence A ~ Gap Open Penalty | | 0.9 | | |
| Sequence B ~ Gap Open Penalty | | 0.9 | | |
| Sequence A ~ Gap Extension Penalty | | 0.2 | | |
| Sequence B ~ Gap Extension Penalty | | 0.0 | | |
| **Test Results** | | | **Values of 1.2** | **% change** |
| Alignment score | | 90.40 | 96.00 | |
| Number of alignments produced | | 1 | 1 | 0.00% |
| Original length of sequence A | | 100 | 100 | |
| Original length of sequence B | | 100 | 100 | |
| Length of optimal alignment | | 104 | 104 | 0.00% |
| Number of gaps inserted into sequence A | | 4 | 4 | 0.00% |
| Number of gaps inserted into sequence B | | 4 | 4 | 0.00% |
| Number of gap segments in sequence A | | 3 | 3 | 0.00% |
| Number of gap segments in sequence B | | 3 | 3 | 0.00% |
| Percent compression on sequence B given sequence A | | **87.00%** | 87.00% | **0.00%** |
| **Alignment** | | | | |

```
A: ATGACCAACATCCGAAAAACACACCCACTAATAAAAATCCT--CAATGA-CG ...
B: ATGACCAACATCCGAAAAACACACCCACTAATAAAAAT--TATCAAT-AACG ...

... CATTCA-TCGACCTACCCACTCCATCTAACATCTCCTCATGATGAAACTTTG
... CATTCATT-GACCTACCCACTCCATCTAACATCTCCTCATGATGAAACTTTG
```

| **Sequence Test Run 2.3** | | *Organism* | *Region* | *Span* |
|---|---|---|---|---|
| Sequence A | Accession EF093025 http://bit.ly/iK1lw | Lissodelphis Borealis (Northern Right Whale Dolphin) | cytochrome b, mitochondrial | Bases 0 - 100 |
| | ATGACCAACATCCGAAAAACACACCCACTAATAAAAATCCTCAATGACGCATTCATCGACCTACCCACTCCATCTAACATCTCCTCATGATGAAACTTTG | | | |
| Sequence B | ATGACCAACATCCGAAAAACACACCCACTAATAAAAATCCTCAATAACACATTCATCGACCTACCCACTCCATCTAACATCTCCTCATGATGAAACTTTG | | | |
| | Accession AY257155 http://bit.ly/fjF7g | Lagenorhynchus Obscurus (Dusky Dolphin) | cytochrome b, mitochondrial | Bases 0 - 100 |
| **Algorithm Parameters** | | | | |
| Match Score | | 1 | | |
| Mismatch Score | | FLT_MIN | | |
| Sequence A ~ Gap Open Penalty | | 0.9 | | |
| Sequence B ~ Gap Open Penalty | | 0.9 | | |
| Sequence A ~ Gap Extension Penalty | | 0.2 | | |
| Sequence B ~ Gap Extension Penalty | | 0.0 | | |
| **Test Results** | | | **Values of 1.3** | **% change** |
| Alignment score | | 93.90 | 97.00 | |
| Number of alignments produced | | 1 | 2 | -50.00% |
| Original length of sequence A | | 100 | 100 | |
| Original length of sequence B | | 100 | 100 | |
| Length of optimal alignment | | 103 | 103 | 0.00% |
| Number of gaps inserted into sequence A | | 3 | 3 | 0.00% |
| Number of gaps inserted into sequence B | | 3 | 3 | 0.00% |
| Number of gap segments in sequence A | | 1 | 1 | 0.00% |
| Number of gap segments in sequence B | | 2 | 2 | 0.00% |
| Percent compression on sequence B given sequence A | | **92.00%** | 92.00% | **0.00%** |
| **Alignment** | | | | |

```
A: ATGACCAACATCCGAAAAACACACCCACTAATAAAAATCCTCAATGA---CG ...
B: ATGACCAACATCCGAAAAACACACCCACTAATAAAAATCCTCAAT-AACAC- ...

... CATTCATCGACCTACCCACTCCATCTAACATCTCCTCATGATGAAACTTTG
... -ATTCATCGACCTACCCACTCCATCTAACATCTCCTCATGATGAAACTTTG
```

| **Sequence Test Run 2.4** | | *Organism* | *Region* | *Span* |
|---|---|---|---|---|
| Sequence A | Accession FJ919248 http://bit.ly/7GDXy | Human rotavirus A | outer capsid protein (VP4) | Bases 0 - 100 |
| | ATTTATAGACAACTTCTCACTAATTACTATTCGGTAGACTTGCATGACGAAATAGAACAGATTGGATCGGAGAAAA CTCAAAATGTGACGGTAAATCCAG | | | |
| Sequence B | GGCTATAAAATGGCTTCGCTCATTTATAGACAGCTTCTCACTAATTCATATTCAGTAGATTTATATGATGAAATAG AGCAAATTGGATCAGAAAAAACTC | | | |
| | Accession EU839962 http://bit.ly/M5ibV | Human rotavirus G9P[8] | outer capsid protein (VP4) | Bases 0 - 100 |
| **Algorithm Parameters** | | | | |
| Match Score | | 1 | | |
| Mismatch Score | | FLT_MIN | | |
| Sequence A ~ Gap Open Penalty | | 0.9 | | |
| Sequence B ~ Gap Open Penalty | | 0.9 | | |
| Sequence A ~ Gap Extension Penalty | | 0.2 | | |
| Sequence B ~ Gap Extension Penalty | | 0.0 | | |
| **Test Results** | | | **Values of 1.4** | **% change** |
| Alignment score | | 51.80 | 64.90 | |
| Number of alignments produced | | 192 | 50,400 | -99.62% |
| Original length of sequence A | | 100 | 100 | |
| Original length of sequence B | | 100 | 100 | |
| Length of optimal alignment | | 131 | 131 | 0.00% |
| Number of gaps inserted into sequence A | | 31 | 31 | 0.00% |
| Number of gaps inserted into sequence B | | 31 | 31 | 0.00% |
| Number of gap segments in sequence A | | 6 | 6 | 0.00% |
| Number of gap segments in sequence B | | 12 | 13 | -7.69% |
| Percent compression on sequence B given sequence A | | **39.00%** | 37.00% | **5.41%** |

**Alignment**

```
A: ---------------------ATTTATAGACAA-CTTCTCACTAATTAC-TATTC-GGTAGACT- ...
B: GGCTATAAAATGGCTTCGCTCATTTATAGAC-AGCTTCTCACTAATT-CATATTCAG-TAGA-TT ...

... ----TGCATGACGAAATAGA--ACAGATTGGATCGGAGAAAACTCAAAATGTGACGGTAAATCCAG
... TATATG-ATGA---AATAGAGCA-A-ATTGGATC--AGAAA------AA----AC------TC---
```

| **Sequence Test Run 2.5** | | *Organism* | *Region* | *Span* |
|---|---|---|---|---|
| Sequence A | Accession DQ022290 http://bit.ly/15pwqT | Panthera leo (lion) | cytochrome b, mitochondrial | Bases 0 - 100 |
| | ATGACCAACATTCGAAAATCACACCCCCTTGTCAAAATTATTAATCACTCATTCATTGATCTTCCCACTCCACCCA ATATCTCAGCATGATGAAACTTTG | | | |
| Sequence B | TGGATTATCCGATATCTACATGCCAACGGAGCCTCCATATTCTTTATCTGTCTATACATGCACGTAGGACGAGGAA TATACTACGGCTCCTACACCTTCT | | | |
| | Accession FJ895266 http://bit.ly/GnXGR | Panthera tigris altaica (Amur tiger) | cytochrome b, mitochondrial | Bases 0 - 100 |
| **Algorithm Parameters** | | | | |
| Match Score | | 1 | | |
| Mismatch Score | | FLT_MIN | | |
| Sequence A ~ Gap Open Penalty | | 0.9 | | |
| Sequence B ~ Gap Open Penalty | | 0.9 | | |
| Sequence A ~ Gap Extension Penalty | | 0.2 | | |
| Sequence B ~ Gap Extension Penalty | | 0.0 | | |
| **Test Results** | | | **Values of 1.5** | **% change** |
| Alignment score | | 27.30 | 31.00 | |
| Number of alignments produced | | 72 | 37,800 | -99.81% |
| Original length of sequence A | | 100 | 100 | |
| Original length of sequence B | | 100 | 100 | |
| Length of optimal alignment | | 139 | 138 | 0.72% |
| Number of gaps inserted into sequence A | | 39 | 38 | 2.63% |
| Number of gaps inserted into sequence B | | 39 | 38 | 2.63% |
| Number of gap segments in sequence A | | 19 | 22 | -13.64% |
| Number of gap segments in sequence B | | 14 | 17 | -17.65% |
| Percent compression on sequence B given sequence A | | **14.00%** | 6.00% | **133.33%** |
| **Alignment** | | | | |

```
A: AT-GACCAACATT---CGA-AAATC-ACA--CC--C----CCTTGTC-AAAATTATTAATCACTCATTCA ...
B: -TGG------ATTATCCGATA--TCTACATGCCAACGGAGCC---TCCA----TATT-----CT--TT-A ...

... T-TGATCT-TCCCAC-T-C-C-----ACCC----AATAT-------CTCAGCATGATGAA-AC-TT-TG
... TCTG-TCTAT---ACATGCACGTAGGA--CGAGGAATATACTACGGCTC--C----T--ACACCTTCT-
```

| **Sequence Test Run 2.6** | | *Organism* | *Region* | *Span* |
|---|---|---|---|---|
| Sequence A | Accession NM_001828 http://bit.ly/rvhic | Homo sapiens (human) | Charcot-Leyden crystal protein | Bases 0 - 100 |
| | ATTTAAATTCTGCAGCTCAGAGATTCACACAGAAGTCTGGACACAATTCAGAAGAGCCACCCAGAAGGAGACAACA ATGTCCCTGCTACCCGTGCCATAC | | | |
| Sequence B | GACAGAAGACTGGACACAATTCCGAAGAGTCGCCCAGAAGGAGAGAACAATGTCATCACTACCCGTACCATACACA CTGCCTGTTTCCTTGTCTGTTGGT | | | |
| | Accession FJ613346 http://bit.ly/uXtDu | Papio anubis (olive baboon) | LGALS14 | Bases 0 - 100 |
| **Algorithm Parameters** | | | | |
| Match Score | | 1 | | |
| Mismatch Score | | FLT_MIN | | |
| Sequence A ~ Gap Open Penalty | | 0.9 | | |
| Sequence B ~ Gap Open Penalty | | 0.9 | | |
| Sequence A ~ Gap Extension Penalty | | 0.2 | | |
| Sequence B ~ Gap Extension Penalty | | 0.0 | | |
| **Test Results** | | | **Values of 1.6** | **% change** |
| Alignment score | | 41.90 | 31.80 | |
| Number of alignments produced | | 16 | 2,250 | -99.29% |
| Original length of sequence A | | 100 | 100 | |
| Original length of sequence B | | 100 | 100 | |
| Length of optimal alignment | | 137 | 134 | 2.24% |
| Number of gaps inserted into sequence A | | 37 | 34 | 8.82% |
| Number of gaps inserted into sequence B | | 37 | 34 | 8.82% |
| Number of gap segments in sequence A | | 7 | 17 | -58.82% |
| Number of gap segments in sequence B | | 10 | 17 | -41.18% |
| Percent compression on sequence B given sequence A | | **36.00%** | 15.00% | **140.00%** |

**Alignment**

```
      A: ATTTAAATTCTGCAGCTCAGAGATTCACACAGAAGTCTGGACACAATT-CAGAAGAG-C-CACC ...
      B: ----------G-A---CAGA---------AGA---CTGGACACAATTCC-GAAGAGTCGC--C ...

... CAGAAG--GAGACAACAATGTC--C-CTGCTACCCGTG--CCATAC--------------------------
... CAGAAGGAGAG--AACAATGTCATCAC---TACCC--GTACCATACACACTGCCTGTTTCCTTGTCTGTTGGT
```

# Sequence Alignment Algorithm ~ Third Generation

| **Sequence Test Run 3.0** | | *Organism* | *Region* | *Span* |
|---|---|---|---|---|
| Sequence A | Accession NC_001601 http://bit.ly/OBkcS | Balaenoptera Musculus (Blue Whale) | Mitochondrial DNA | Bases 0 - 64 |
| | GTTAATTACTAATCAGCCCATGATCATAACATAACTGAGGTTTCATACATTTGGTATTTTTTTA | | | |
| Sequence B | GTTAATGTAGCTTAAATATTTATAAAGCAAAACACTGAAAATGTTTAGATGGGTTTAATTAACC | | | |
| | Accession AP006472 http://bit.ly/wBxsD | Balaena Mysticetus (Bowhead Whale) | Mitochondrial DNA | Bases 0 - 64 |
| **Algorithm Parameters** | | | | |
| Mismatch Score | | FLT_MIN | | |
| Sequence A ~ Gap Open Overhead | | 1 | | |
| Sequence B ~ Gap Open Overhead | | 2 | | |
| Sequence A ~ Gap Extension Overhead | | 1 | | |
| Sequence B ~ Gap Extension Overhead | | 0 | | |
| | | | | |
| **Test Results** | | | **Values of 2.0** | **% change** |
| Alignment score | | 45.00 | 23.40 | |
| Number of alignments produced | | 72 | 36 | 100.00% |
| Original length of sequence A | | 64 | 64 | |
| Original length of sequence B | | 64 | 64 | |
| Length of optimal alignment | | 93 | 86 | 8.14% |
| Number of gaps inserted into sequence A | | 29 | 22 | 31.82% |
| Number of gaps inserted into sequence B | | 29 | 22 | 31.82% |
| Number of gap segments in sequence A | | 8 | 9 | -11.11% |
| Number of gap segments in sequence B | | 4 | 8 | -50.00% |
| Percent compression on sequence B given sequence A | | **29.69%** | 26.56% | **11.76%** |
| **Alignment** | | | | |

```
A: GTTAAT-TA-CT--AATCAGCCCATGATCAT-----------AACA ...
B: GTTAATGTAGCTTAAAT------------ATTTATAAAGCAAAACA ...

... TAACTGAGGTTTCATACA----TTT-----GGTATTTTT--TTA---
... ---CTGA----------AAATGTTTAGATGGG----TTTAATTAACC
```

| **Sequence Test Run 3.1** | | *Organism* | *Region* | *Span* |
|---|---|---|---|---|
| Sequence A | Accession AF304073 http://bit.ly/10deJe | Physeter catodon (Sperm Whale) | cytochrome b, mitochondrial protein | Bases 0 - 100 |
| | ATGACCAACATCCGAAAATCACACCCATTAATAAAAATCATTAACAATGCATTCATCGACCTCCCTACCCCATCAA ACATTTCCTCATGATGAAACTTCG | | | |
| Sequence B | ATGACCAACATCCGAAAAACACACCCATTGATAAAAATCGTCAACAACGCATTCATCGACCTCCCTACTCCATCAA ACATCTCCTCATGATGAAATTTCG | | | |
| | Accession U72040 http://bit.ly/iube4 | Kogia breviceps (Pygmy Sperm Whale) | cytochrome b, mitochondrial protein | Bases 0 - 100 |
| **Algorithm Parameters** | | | | |
| Mismatch Score | | FLT_MIN | | |
| Sequence A ~ Gap Open Overhead | | 1 | | |
| Sequence B ~ Gap Open Overhead | | 2 | | |
| Sequence A ~ Gap Extension Overhead | | 1 | | |
| Sequence B ~ Gap Extension Overhead | | 0 | | |
| | | | | |
| **Test Results** | | | **Values of 2.1** | **% change** |
| Alignment score | | 31.00 | 76.80 | |
| Number of alignments produced | | 528 | 384 | 37.50% |
| Original length of sequence A | | 100 | 100 | |
| Original length of sequence B | | 100 | 100 | |
| Length of optimal alignment | | 111 | 110 | 0.91% |
| Number of gaps inserted into sequence A | | 11 | 10 | 10.00% |
| Number of gaps inserted into sequence B | | 11 | 10 | 10.00% |
| Number of gap segments in sequence A | | 6 | 7 | -14.29% |
| Number of gap segments in sequence B | | 7 | 7 | 0.00% |
| Percent compression on sequence B given sequence A | | **69.00%** | 69.00% | **0.00%** |
| **Alignment** | | | | |

```
A: ATGACCAACATCCGAAA-ATCACACCCATT-AATAAAAAT------CATTAACAAT ...
B: ATGACCAACATCCGAAAAA-CACACCCATTGA-TAAAAATCGTCAAC---AAC--- ...

... GCATTCATCGACCTCCCTAC-CCCATCAAACATT-TCCTCATGATGAAACTT-CG
... GCATTCATCGACCTCCCTACTC-CATCAAACA-TCTCCTCATGATGAAA-TTTCG
```

| **Sequence Test Run 3.2** | | *Organism* | *Region* | *Span* |
|---|---|---|---|---|
| Sequence A | Accession EF093041 http://bit.ly/NmVDT | Lagenorhynchus obliquidens (Pacific white-sided dolphin) | cytochrome b, mitochondrial protein | Bases 0 - 100 |
| | ATGACCAACATCCGAAAAACACACCCACTAATAAAAATCCTCAATGACGCATTCATCGACCTACCCACTCCATCTAACATCTCCTCATGATGAAACTTTG | | | |
| Sequence B | ATGACCAACATCCGAAAAACACACCCACTAATAAAAATTATCAATAACGCATTCATTGACCTACCCACTCCATCTAACATCTCCTCATGATGAAACTTTG | | | |
| | Accession AF084057 http://bit.ly/bmrCE | Pseudorca Crassidens (False Killer Whale) | cytochrome b, mitochondrial protein | Bases 0 - 100 |
| **Algorithm Parameters** | | | | |
| Mismatch Score | | FLT_MIN | | |
| Sequence A ~ Gap Open Overhead | | 1 | | |
| Sequence B ~ Gap Open Overhead | | 2 | | |
| Sequence A ~ Gap Extension Overhead | | 1 | | |
| Sequence B ~ Gap Extension Overhead | | 0 | | |
| | | | | |
| **Test Results** | | | **Values of 2.2** | **% change** |
| Alignment score | | 13.00 | 90.40 | |
| Number of alignments produced | | 1 | 1 | 0.00% |
| Original length of sequence A | | 100 | 100 | |
| Original length of sequence B | | 100 | 100 | |
| Length of optimal alignment | | 104 | 104 | 0.00% |
| Number of gaps inserted into sequence A | | 4 | 4 | 0.00% |
| Number of gaps inserted into sequence B | | 4 | 4 | 0.00% |
| Number of gap segments in sequence A | | 3 | 3 | 0.00% |
| Number of gap segments in sequence B | | 3 | 3 | 0.00% |
| Percent compression on sequence B given sequence A | | **87.00%** | 87.00% | **0.00%** |

**Alignment**

```
A: ATGACCAACATCCGAAAAACACACCCACTAATAAAAATCCT--CAATGA-CG ...
B: ATGACCAACATCCGAAAAACACACCCACTAATAAAAAT--TATCAAT-AACG ...

... CATTCA-TCGACCTACCCACTCCATCTAACATCTCCTCATGATGAAACTTTG
... CATTCATT-GACCTACCCACTCCATCTAACATCTCCTCATGATGAAACTTTG
```

| **Sequence Test Run 3.3** | | *Organism* | *Region* | *Span* |
|---|---|---|---|---|
| Sequence A | Accession EF093025 http://bit.ly/16E6VM | Lissodelphis Borealis (Northern Right Whale Dolphin) | cytochrome b, mitochondrial | Bases 0 - 100 |
| | ATGACCAACATCCGAAAAACACACCCACTAATAAAAATCCTCAATGACGCATTCATCGACCTACCCACTCCATCTA ACATCTCCTCATGATGAAACTTTG | | | |
| Sequence B | ATGACCAACATCCGAAAAACACACCCACTAATAAAAATCCTCAATAACACATTCATCGACCTACCCACTCCATCTA ACATCTCCTCATGATGAAACTTTG | | | |
| | Accession AY257155 http://bit.ly/fjF7g | Lagenorhynchus Obscurus (Dusky Dolphin) | cytochrome b, mitochondrial | Bases 0 - 100 |
| **Algorithm Parameters** | | | | |
| Mismatch Score | | FLT_MIN | | |
| Sequence A ~ Gap Open Overhead | | 1 | | |
| Sequence B ~ Gap Open Overhead | | 2 | | |
| Sequence A ~ Gap Extension Overhead | | 1 | | |
| Sequence B ~ Gap Extension Overhead | | 0 | | |
| | | | | |
| **Test Results** | | | **Values of 2.3** | **% change** |
| Alignment score | | 8.00 | 93.90 | |
| Number of alignments produced | | 3 | 1 | 200.00% |
| Original length of sequence A | | 100 | 100 | |
| Original length of sequence B | | 100 | 100 | |
| Length of optimal alignment | | 105 | 103 | 1.94% |
| Number of gaps inserted into sequence A | | 5 | 3 | 66.67% |
| Number of gaps inserted into sequence B | | 5 | 3 | 66.67% |
| Number of gap segments in sequence A | | 1 | 1 | 0.00% |
| Number of gap segments in sequence B | | 1 | 2 | -50.00% |
| Percent compression on sequence B given sequence A | | **92.00%** | 92.00% | **0.00%** |
| **Alignment** | | | | |

```
A: ATGACCAACATCCGAAAAACACACCCACTAATAAAAATCCTCA-----ATGAC ...
B: ATGACCAACATCCGAAAAACACACCCACTAATAAAAATCCTCAATAACA---- ...

... GCATTCATCGACCTACCCACTCCATCTAACATCTCCTCATGATGAAACTTTG
... -CATTCATCGACCTACCCACTCCATCTAACATCTCCTCATGATGAAACTTTG
```

| **Sequence Test Run 3.4** | | *Organism* | *Region* | *Span* |
|---|---|---|---|---|
| Sequence A | Accession FJ919248 http://bit.ly/7GDXy | Human rotavirus A | outer capsid protein (VP4) | Bases 0 - 100 |
| | ATTTATAGACAACTTCTCACTAATTACTATTCGGTAGACTTGCATGACGAAATAGAACAGATTGGATCGGAGAAAACTCAAAATGTGACGGTAAATCCAG | | | |
| Sequence B | GGCTATAAAATGGCTTCGCTCATTTATAGACAGCTTCTCACTAATTCATATTCAGTAGATTTATATGATGAAATAGAGCAAATTGGATCAGAAAAAACTC | | | |
| | Accession EU839962 http://bit.ly/M5ibV | Human rotavirus G9P[8] | outer capsid protein (VP4) | Bases 0 - 100 |
| **Algorithm Parameters** | | | | |
| Mismatch Score | | FLT_MIN | | |
| Sequence A ~ Gap Open Overhead | | 1 | | |
| Sequence B ~ Gap Open Overhead | | 2 | | |
| Sequence A ~ Gap Extension Overhead | | 1 | | |
| Sequence B ~ Gap Extension Overhead | | 0 | | |
| | | | | |
| **Test Results** | | | **Values of 2.4** | **% change** |
| Alignment score | | 59.00 | 51.80 | |
| Number of alignments produced | | 240 | 192 | 25.00% |
| Original length of sequence A | | 100 | 100 | |
| Original length of sequence B | | 100 | 100 | |
| Length of optimal alignment | | 134 | 131 | 2.29% |
| Number of gaps inserted into sequence A | | 34 | 31 | 9.68% |
| Number of gaps inserted into sequence B | | 34 | 31 | 9.68% |
| Number of gap segments in sequence A | | 7 | 6 | 16.67% |
| Number of gap segments in sequence B | | 9 | 12 | -25.00% |
| Percent compression on sequence B given sequence A | | **41.00%** | 39.00% | **5.13%** |
| **Alignment** | | | | |

```
A: ----------------------ATTTATAGACAA-CTTCTCACTAATTAC-TATTC-GGTAGACTT-- ...
B: GGCTATAAAATGGCTTCGCTCATTTATAGAC-AGCTTCTCACTAATT-CATATTCAG-TAGA-TTTA ...

... ---GCATGACGAAATAGAACAG---ATTGGATCGGAGA--AAACTCAAAATGTGACGGTAAATCCAG
... TATG-ATGA---AATAGA---GCAAATTGGATC--AGAAAAAACTC--------------------
```

| **Sequence Test Run 3.5** | | *Organism* | *Region* | *Span* |
|---|---|---|---|---|
| Sequence A | Accession DQ022290 http://bit.ly/15pwqT | Panthera leo (lion) | cytochrome b, mitochondrial | Bases 0 - 100 |
| | ATGACCAACATTCGAAAATCACACCCCCTTGTCAAAATTATTAATCACTCATTCATTGATCTTCCCACTCCACCCA ATATCTCAGCATGATGAAACTTTG | | | |
| Sequence B | TGGATTATCCGATATCTACATGCCAACGGAGCCTCCATATTCTTTATCTGTCTATACATGCACGTAGGACGAGGAA TATACTACGGCTCCTACACCTTCT | | | |
| | Accession FJ895266 http://bit.ly/GnXGR | Panthera tigris altaica (Amur tiger) | cytochrome b, mitochondrial | Bases 0 - 100 |
| **Algorithm Parameters** | | | | |
| Mismatch Score | | FLT_MIN | | |
| Sequence A ~ Gap Open Overhead | | 1 | | |
| Sequence B ~ Gap Open Overhead | | 2 | | |
| Sequence A ~ Gap Extension Overhead | | 1 | | |
| Sequence B ~ Gap Extension Overhead | | 0 | | |
| | | | | |
| **Test Results** | | | **Values of 2.5** | **% change** |
| Alignment score | | 80.00 | 27.30 | |
| Number of alignments produced | | 16 | 72 | -77.78% |
| Original length of sequence A | | 100 | 100 | |
| Original length of sequence B | | 100 | 100 | |
| Length of optimal alignment | | 155 | 139 | 11.51% |
| Number of gaps inserted into sequence A | | 55 | 39 | 41.03% |
| Number of gaps inserted into sequence B | | 55 | 39 | 41.03% |
| Number of gap segments in sequence A | | 14 | 19 | -26.32% |
| Number of gap segments in sequence B | | 6 | 14 | -57.14% |
| Percent compression on sequence B given sequence A | | **19.00%** | 14.00% | **35.71%** |
| **Alignment** | | | | |

```
A: ---ATGACCAACATT---CGAAAATCACACCCCCTTGTCAAAATTAT-TA-AT--CA-------CTC---ATTC---A...
B: TGG---------ATTATCCGA----------------------TATCTACATGCCAACGGAGCCTCCATATTCTTTA...

  ... T-TGATCTTCCCACTCCACCCAATATCTCAGCATG-A--T--------GAA---ACT-----------------TTG
  ... TCTG-TCT--------------ATA------CATGCACGTAGGACGAGGAATATACTACGGCTCCTACACCTTCT--
```

| **Sequence Test Run 3.6** | | *Organism* | *Region* | *Span* |
|---|---|---|---|---|
| Sequence A | Accession NM_001828 http://bit.ly/rvhic | Homo sapiens (human) | Charcot-Leyden crystal protein | Bases 0 - 100 |
| | ATTTAAATTCTGCAGCTCAGAGATTCACACAGAAGTCTGGACACAATTCAGAAGAGCCACCCAGAAGGAGACAACA ATGTCCCTGCTACCCGTGCCATAC | | | |
| Sequence B | GACAGAAGACTGGACACAATTCCGAAGAGTCGCCCAGAAGGAGAGAACAATGTCATCACTACCCGTACCATACACA CTGCCTGTTTCCTTGTCTGTTGGT | | | |
| | Accession FJ613346 http://bit.ly/uXtDu | Papio anubis (olive baboon) | LGALS14 | Bases 0 - 100 |
| **Algorithm Parameters** | | | | |
| Mismatch Score | | FLT_MIN | | |
| Sequence A ~ Gap Open Overhead | | 1 | | |
| Sequence B ~ Gap Open Overhead | | 2 | | |
| Sequence A ~ Gap Extension Overhead | | 1 | | |
| Sequence B ~ Gap Extension Overhead | | 0 | | |
| | | | | |
| **Test Results** | | | **Values of 2.6** | **% change** |
| Alignment score | | 63.00 | 41.90 | |
| Number of alignments produced | | 1,152 | 16 | 7,100.00% |
| Original length of sequence A | | 100 | 100 | |
| Original length of sequence B | | 100 | 100 | |
| Length of optimal alignment | | 140 | 137 | 2.19% |
| Number of gaps inserted into sequence A | | 40 | 37 | 8.11% |
| Number of gaps inserted into sequence B | | 40 | 37 | 8.11% |
| Number of gap segments in sequence A | | 7 | 7 | 0.00% |
| Number of gap segments in sequence B | | 8 | 10 | -20.00% |
| Percent compression on sequence B given sequence A | | **37.00%** | 36.00% | **2.78%** |

**Alignment**

```
A: -ATTTAAATTCTGCAGCTCAGAGATTCACACAGAAGTCTGGACACAATT-CAGAAGA---GCCACCCAGA ...
B: GA----------------CAGA---------AGA---CTGGACACAATTCC-GAAGAGTCG---CCCAGA ...

... AG--GAGACAACAATGTC----CCTGCTACCCGTG--CCATAC--------------------------
... AGGAGAGA--ACAATGTCATCAC----TACCC--GTACCATACACACTGCCTGTTTCCTTGTCTGTTGGT
```

| **Sequence Test Run 3.7** | | *Organism* | *Region* | *Span* |
|---|---|---|---|---|
| Sequence A | Accession EF093041 http://bit.ly/NmVDT | Lagenorhynchus obliquidens (Pacific white-sided dolphin) | cytochrome b, mitochondrial protein | Amino Acids 0 - 100 |
| | MTNIRKTHPLMKILNDAFIDLPTPSNISSWWNFGSLLGLCLIMQILTGLFLAMHYTPDTSTAFSSVAHICRDVNYG WFIRYLHANGASMFFICLYAHIGR | | | |
| Sequence B | MTNIRKTHPLMKIINNAFIDLPTPSNISSWWNFGSLLGLCLIMQILTGLFLAMHYTPDTSTAFSSVAHICRDVNYG WFIRYLHANGASMFFICLYAHIGR | | | |
| | Accession AF084057 http://bit.ly/bmrCE | Pseudorca Crassidens (False Killer Whale) | cytochrome b, mitochondrial protein | Amino Acids 0 - 100 |
| **Algorithm Parameters** | | | | |
| Mismatch Score | | FLT_MIN | | |
| Sequence A ~ Gap Open Overhead | | 1 | | |
| Sequence B ~ Gap Open Overhead | | 2 | | |
| Sequence A ~ Gap Extension Overhead | | 1 | | |
| Sequence B ~ Gap Extension Overhead | | 0 | | |
| | | | | |
| **Test Results** | | | | |
| Alignment score | | 7.00 | | |
| Number of alignments produced | | 1 | | |
| Original length of sequence A | | 100 | | |
| Original length of sequence B | | 100 | | |
| Length of optimal alignment | | 105 | | |
| Number of gaps inserted into sequence A | | 5 | | |
| Number of gaps inserted into sequence B | | 5 | | |
| Number of gap segments in sequence A | | 1 | | |
| Number of gap segments in sequence B | | 1 | | |
| Percent compression on sequence B given sequence A | | **92.00%** | | |
| **Alignment** | | | | |

```
A: MTNIRKTHPLMKILNDAFI-----DLPTPSNISSWWNFGSLLGLCLIMQILTG ...
B: MTNIRKTHPLMKI-----INNAFIDLPTPSNISSWWNFGSLLGLCLIMQILTG ...

  ... LFLAMHYTPDTSTAFSSVAHICRDVNYGWFIRYLHANGASMFFICLYAHIGR
  ... LFLAMHYTPDTSTAFSSVAHICRDVNYGWFIRYLHANGASMFFICLYAHIGR
```