

DNATool

Generated by Doxygen 1.9.5



<b>1 eda-lvl5-dnatool README</b>	<b>1</b>
1.1 Respuestas	1
1.1.1 Mejores alineamientos para los mystery genomes:	1
1.2 Bonus points	2
<b>2 Class Index</b>	<b>3</b>
2.1 Class List	3
<b>3 File Index</b>	<b>5</b>
3.1 File List	5
<b>4 Class Documentation</b>	<b>7</b>
4.1 DirCell Struct Reference	7
4.1.1 Detailed Description	7
4.1.2 Member Data Documentation	7
4.1.2.1 d0	7
4.1.2.2 d1	7
4.1.2.3 d2	8
4.1.2.4 d3	8
4.2 DirectionMat Class Reference	8
4.2.1 Constructor & Destructor Documentation	8
4.2.1.1 DirectionMat()	8
4.2.1.2 ~DirectionMat()	9
4.2.2 Member Function Documentation	9
4.2.2.1 at()	9
4.2.2.2 print()	9
4.2.2.3 set()	10
4.2.3 Member Data Documentation	10
4.2.3.1 cols	10
4.2.3.2 rows	10
<b>5 File Documentation</b>	<b>11</b>
5.1 Aligner.cpp File Reference	11
5.1.1 Detailed Description	12
5.1.2 Enumeration Type Documentation	12
5.1.2.1 Direction	12
5.1.3 Function Documentation	12
5.1.3.1 buildAligment()	12
5.1.3.2 fillMat()	13
5.1.3.3 getGlobalAlignment()	13
5.1.3.4 initMat()	14
5.2 Aligner.h File Reference	14
5.2.1 Detailed Description	14
5.2.2 Function Documentation	15

5.2.2.1 buildAlignment()	15
5.2.2.2 fillMat()	15
5.2.2.3 getGlobalAlignment()	15
5.2.2.4 initMat()	15
5.3 Aligner.h	15
5.4 DirectionMat.cpp File Reference	16
5.4.1 Detailed Description	16
5.5 DirectionMat.h File Reference	16
5.5.1 Detailed Description	17
5.6 DirectionMat.h	17
5.7 GBReader.cpp File Reference	18
5.7.1 Detailed Description	18
5.7.2 Function Documentation	18
5.7.2.1 getSequence()	18
5.7.2.2 isNotACGT()	19
5.8 GBReader.h File Reference	19
5.8.1 Detailed Description	19
5.8.2 Function Documentation	20
5.8.2.1 getSequence()	20
5.9 GBReader.h	20
5.10 main.cpp File Reference	20
5.10.1 Detailed Description	21
5.10.2 Function Documentation	21
5.10.2.1 main()	21
5.10.2.2 printElapsedFrom()	21
5.11 main_test.cpp File Reference	21
5.11.1 Detailed Description	22
5.11.2 Function Documentation	22
5.11.2.1 main()	22
5.11.2.2 testAlignment()	22
5.11.2.3 testFillMat()	23
5.11.2.4 testMatInit()	23
5.12 README.md File Reference	23
<b>Index</b>	<b>25</b>

# Chapter 1

## eda-lvl5-dnatool README

- Fecha de entrega: 10/05/2022
- Grupo: 7
  - DALZOTTO, Rafael
  - HEIR, Alejandro Nahuel

22.08 - Algoritmos y Estructuras de Datos - Ingeniería Electrónica - ITBA

### 1.1 Respuestas

#### 1.1.1 Mejores alineamientos para los mystery genomes:

- Mystery genome 1 - OV806939
  - Delta B.1.617.2 - OM202516
  - Puntaje óptimo: 29826
- Mystery genome 2 - MT345875
  - Beta B.1.351 - MZ314996
  - Puntaje óptimo: 29865
- Mystery genome 3 - OU772970
  - Delta B.1.617.2 - OM202516
  - Puntaje óptimo: 29840

Los alineamientos óptimos correspondientes se encuentra en `/resources/test/results/`

## 1.2 Bonus points

- La complejidad computacional del algoritmo Needleman-Wunsch es  $O(mn)$  debido a que el calculo de las direcciones (o scores) de la matriz de alineacion depende de los largos de ambas secuencias.

Sin embargo logramos reducir sustancialmente el uso de memoria, gracias a aprovechar cada bit en los elementos de la matriz de direcciones ([DirectionMat](#)) y debido a que no usamos una matriz de scores, resultando en un uso de memoria de no mas de 250 MB para comparar dos secuencias de unos 30000 pares de base (resultados experimentales). En cambio al usar una matriz de scores de `int32_t`, el uso de memoria se eleva a mas de 3 GB.

- Al cambiar los valores en el sistema de puntaje, cambia el puntaje final de alineamiento óptimo y tambien puede cambiar el alineamiento optimo en si
- Comparamos los genomas de SARS-CoV-2003 y MERS-CoV con Wuhan reference genome
  - Puntaje de MERS-CoV vs Wuhan reference genome: 7240
  - Puntaje de SARS-CoV-2003 vs Wuhan reference genome: 18645
  - Los alineamientos óptimos correspondientes se encuentran en `/resources/test/results/`

## Chapter 2

# Class Index

### 2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">DirCell</a>		
<a href="#">DirCell</a>	contains 4 directions in a single byte . . . . .	<a href="#">7</a>
<a href="#">DirectionMat</a>	. . . . .	<a href="#">8</a>





## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

<a href="#">Aligner.cpp</a>	Genetic sequence aligner, based on Needleman-Wunsch algorithm . . . . .	11
<a href="#">Aligner.h</a>	Genetic sequence aligner, based on Needleman-Wunsch algorithm . . . . .	14
<a href="#">DirectionMat.cpp</a>	Implements 2 bit-size matrix cells . . . . .	16
<a href="#">DirectionMat.h</a>	Implements 2 bit-size matrix cells optimized for memory usage . . . . .	16
<a href="#">GBReader.cpp</a>	Sequence extractor for GenBank-like files . . . . .	18
<a href="#">GBReader.h</a>	Sequence extractor for GenBank-like files . . . . .	19
<a href="#">main.cpp</a>	Runs the genetic sequence aligner . . . . .	20
<a href="#">main_test.cpp</a>	Tests for the genetic sequence aligner . . . . .	21



## Chapter 4

# Class Documentation

### 4.1 DirCell Struct Reference

[DirCell](#) contains 4 directions in a single byte.

```
#include <DirectionMat.h>
```

#### Public Attributes

- `uint8_t d0`: 2
- `uint8_t d1`: 2
- `uint8_t d2`: 2
- `uint8_t d3`: 2

#### 4.1.1 Detailed Description

[DirCell](#) contains 4 directions in a single byte.

#### 4.1.2 Member Data Documentation

##### 4.1.2.1 d0

```
uint8_t DirCell::d0
```

##### 4.1.2.2 d1

```
uint8_t DirCell::d1
```

#### 4.1.2.3 d2

```
uint8_t DirCell::d2
```

#### 4.1.2.4 d3

```
uint8_t DirCell::d3
```

The documentation for this struct was generated from the following file:

- [DirectionMat.h](#)

## 4.2 DirectionMat Class Reference

```
#include <DirectionMat.h>
```

### Public Member Functions

- [DirectionMat](#) (const size\_t rows, const size\_t cols)  
*Constructs a new Direction Mat object.*
- [~DirectionMat](#) ()  
*Destroys the Direction Mat object.*
- uint8\_t [at](#) (size\_t row, size\_t col)  
*Reads a given DirectionMap cell.*
- void [set](#) (uint8\_t val, size\_t row, size\_t col)  
*Writes a given DirecitonMat cell.*
- void [print](#) ()  
*Prints the given [DirectionMat](#), used for debugging.*

### Public Attributes

- const size\_t [rows](#)
- const size\_t [cols](#)

### 4.2.1 Constructor & Destructor Documentation

#### 4.2.1.1 DirectionMat()

```
DirectionMat::DirectionMat (
    const size_t rows,
    const size_t cols )
```

Constructs a new Direction Mat object.

## Parameters

<i>rows</i>	Total rows
<i>cols</i>	Total cols

**4.2.1.2 ~DirectionMat()**

```
DirectionMat::~~DirectionMat ( )
```

Destroys the Direction Mat object.

**4.2.2 Member Function Documentation****4.2.2.1 at()**

```
uint8_t DirectionMat::at (
    size_t row,
    size_t col )
```

Reads a given DirectionMap cell.

## Parameters

<i>row</i>	
<i>col</i>	

## Returns

uint8\_t

**4.2.2.2 print()**

```
void DirectionMat::print ( )
```

Prints the given [DirectionMat](#), used for debugging.

## Parameters

<i>mat</i>	initialized <a href="#">DirectionMat</a>
------------	--

#### 4.2.2.3 set()

```
void DirectionMat::set (
    uint8_t val,
    size_t row,
    size_t col )
```

Writes a given DirecitonMat cell.

##### Parameters

<i>val</i>	Data to store
<i>row</i>	
<i>col</i>	

### 4.2.3 Member Data Documentation

#### 4.2.3.1 cols

```
const size_t DirectionMat::cols
```

#### 4.2.3.2 rows

```
const size_t DirectionMat::rows
```

The documentation for this class was generated from the following files:

- [DirectionMat.h](#)
- [DirectionMat.cpp](#)

## Chapter 5

# File Documentation

### 5.1 Aligner.cpp File Reference

Genetic sequence aligner, based on Needleman-Wunsch algorithm.

```
#include "Aligner.h"
#include <iostream>
#include <cstdlib>
#include <algorithm>
#include <vector>
```

#### Enumerations

- enum [Direction](#) { [VERTICAL](#) , [DIAGONAL](#) , [HORIZONTAL](#) }

#### Functions

- void [initMat](#) ([DirectionMat](#) &mat)  
*Initializes a [DirectionMat](#) for storing Needleman-Wunsch directions.*
- int32\_t [fillMat](#) ([DirectionMat](#) &mat, const string &seq1, const string &seq2)  
*Completes a [DirectionMat](#) based on two given genetic sequences.*
- void [buildAlignment](#) ([DirectionMat](#) &mat, const string &seq1, const string &seq2, array< string, 3 > &output)  
*Builds the optimal alignment for two genetic sequences and an already filled [DirectionMat](#).*
- int32\_t [getGlobalAlignment](#) (const string &seq1, const string &seq2, array< string, 3 > &alignment)  
*Runs aligner for two acgt-like strings.*

### 5.1.1 Detailed Description

Genetic sequence aligner, based on Needleman-Wunsch algorithm.

#### Authors

DALZOTTO, Rafael; HEIR, Alejandro - Grupo 7

#### Version

0.1

#### Date

2022-05-08

#### Copyright

Copyright (c) 2022 - 22.08 EDA - ITBA

### 5.1.2 Enumeration Type Documentation

#### 5.1.2.1 Direction

enum [Direction](#)

##### Enumerator

VERTICAL	
DIAGONAL	
HORIZONTAL	

### 5.1.3 Function Documentation

#### 5.1.3.1 buildAligment()

```
void buildAligment (
    DirectionMat & mat,
    const string & seq1,
    const string & seq2,
    array< string, 3 > & output )
```

Builds the optimal alignment for two genetic sequences and an already filled [DirectionMat](#).



## Parameters

<i>mat</i>	
<i>seq1</i>	String with one acgt-like genetic sequence.
<i>seq2</i>	String with one acgt-like genetic sequence.
<i>output</i>	3-string array to store the optimal alignment.

## 5.1.3.2 fillMat()

```
int32_t fillMat (  
    DirectionMat & mat,  
    const string & seq1,  
    const string & seq2 )
```

Completes a [DirectionMat](#) based on two given genetic sequences.

## Parameters

<i>mat</i>	
<i>seq1</i>	String with one acgt-like genetic sequence.
<i>seq2</i>	String with one acgt-like genetic sequence.

## Returns

int32\_t Optimal score

## 5.1.3.3 getGlobalAlignment()

```
int32_t getGlobalAlignment (  
    const string & seq1,  
    const string & seq2,  
    array< string, 3 > & alignment )
```

Runs aligner for two acgt-like strings.

## Parameters

<i>seq1</i>	String with one acgt-like genetic sequence.
<i>seq2</i>	String with one acgt-like genetic sequence.
<i>alignment</i>	3-string array to store the optimal alignment.

## Returns

int32\_t Optimal alignment score

### 5.1.3.4 initMat()

```
void initMat (
    DirectionMat & mat )
```

Initializes a [DirectionMat](#) for storing Needleman-Wunsch directions.

#### Parameters

<i>rows</i>	
<i>cols</i>	

## 5.2 Aligner.h File Reference

Genetic sequence aligner, based on Needleman-Wunsch algorithm.

```
#include <string>
#include <array>
#include "DirectionMat.h"
```

### Functions

- void [initMat](#) ([DirectionMat](#) &mat)  
*Initializes a [DirectionMat](#) for storing Needleman-Wunsch directions.*
- int32\_t [fillMat](#) ([DirectionMat](#) &mat, const std::string &seq1, const std::string &seq2)
- void [buildAlignment](#) ([DirectionMat](#) &mat, const std::string &seq1, const std::string &seq2, std::array< std::string, 3 > &output)
- int32\_t [getGlobalAlignment](#) (const std::string &seq1, const std::string &seq2, std::array< std::string, 3 > &alignment)

### 5.2.1 Detailed Description

Genetic sequence aligner, based on Needleman-Wunsch algorithm.

#### Authors

DALZOTTO, Rafael; HEIR, Alejandro - Grupo 7

#### Version

0.1

#### Date

2022-05-08

#### Copyright

Copyright (c) 2022 - 22.08 EDA - ITBA

## 5.2.2 Function Documentation

### 5.2.2.1 buildAligment()

```
void buildAligment (
    DirectionMat & mat,
    const std::string & seq1,
    const std::string & seq2,
    std::array< std::string, 3 > & output )
```

### 5.2.2.2 fillMat()

```
int32_t fillMat (
    DirectionMat & mat,
    const std::string & seq1,
    const std::string & seq2 )
```

### 5.2.2.3 getGlobalAlignment()

```
int32_t getGlobalAlignment (
    const std::string & seq1,
    const std::string & seq2,
    std::array< std::string, 3 > & alignment )
```

### 5.2.2.4 initMat()

```
void initMat (
    DirectionMat & mat )
```

Initializes a [DirectionMat](#) for storing Needleman-Wunsch directions.

#### Parameters

<i>rows</i>	
<i>cols</i>	

## 5.3 Aligner.h

[Go to the documentation of this file.](#)

```
1
11 #ifndef ALIGNER_H
12 #define ALIGNER_H
13
14 #include <string>
15 #include <array>
16
17 #include "DirectionMat.h"
18
19 void initMat(DirectionMat &mat);
20
21 int32_t fillMat(DirectionMat &mat, const std::string &seq1, const std::string &seq2);
22
23 void buildAlignent(DirectionMat &mat, const std::string &seq1, const std::string &seq2,
24                  std::array<std::string, 3> &output);
25
26 int32_t getGlobalAlignment(const std::string &seq1, const std::string &seq2,
27                          std::array<std::string, 3> &alignment);
28
29 #endif
```

## 5.4 DirectionMat.cpp File Reference

Implements 2 bit-size matrix cells.

```
#include "DirectionMat.h"
```

### 5.4.1 Detailed Description

Implements 2 bit-size matrix cells.

#### Authors

DALZOTTO, Rafael; HEIR, Alejandro - Grupo 7

#### Version

0.1

#### Date

2022-05-08

#### Copyright

Copyright (c) 2022 - 22.08 EDA - ITBA

## 5.5 DirectionMat.h File Reference

Implements 2 bit-size matrix cells optimized for memory usage.

```
#include <iostream>
```

## Classes

- struct [DirCell](#)  
*DirCell contains 4 directions in a single byte.*
- class [DirectionMat](#)

### 5.5.1 Detailed Description

Implements 2 bit-size matrix cells optimized for memory usage.

#### Authors

DALZOTTO, Rafael; HEIR, Alejandro - Grupo 7

#### Version

0.1

#### Date

2022-05-08

#### Copyright

Copyright (c) 2022 - 22.08 EDA - ITBA

## 5.6 DirectionMat.h

[Go to the documentation of this file.](#)

```
1
11 #ifndef DIRECTIONMAT_H
12 #define DIRECTIONMAT_H
13
14 #include <iostream>
15
16 typedef struct
17 {
18     uint8_t d0 : 2;
19     uint8_t d1 : 2;
20     uint8_t d2 : 2;
21     uint8_t d3 : 2;
22 } DirCell;
23
24 class DirectionMat
25 {
26 public:
27     DirectionMat(const size_t rows, const size_t cols);
28     ~DirectionMat();
29
30     uint8_t at(size_t row, size_t col);
31     void set(uint8_t val, size_t row, size_t col);
32     void print();
33
34     const size_t rows;
35     const size_t cols;
36
37 private:
38     DirCell *data;
39     size_t compactCols;
40 };
41
42 #endif
```

## 5.7 GBReader.cpp File Reference

Sequence extractor for GenBank-like files.

```
#include "GBReader.h"  
#include <algorithm>
```

### Functions

- bool [isNotACGT](#) (char c)  
*Checks if a given character is not 'a', 'c', 'g' or 't'.*
- bool [getSequence](#) (string &output, istream &file)  
*Extracts an acgt-like string from a GenBank file.*

### 5.7.1 Detailed Description

Sequence extractor for GenBank-like files.

#### Authors

DALZOTTO, Rafael; HEIR, Alejandro - Grupo 7

#### Version

0.1

#### Date

2022-05-08

#### Copyright

Copyright (c) 2022 - 22.08 EDA - ITBA

### 5.7.2 Function Documentation

#### 5.7.2.1 [getSequence\(\)](#)

```
bool getSequence (  
    string & output,  
    istream & file )
```

Extracts an acgt-like string from a GenBank file.

## Parameters

<i>output</i>	String to store the acgt sequence
<i>file</i>	istream file handler.

## Returns

true Success

false Failure

### 5.7.2.2 isNotACGT()

```
bool isNotACGT (
    char c )
```

Checks if a given character is not 'a', 'c', 'g' or 't'.

## Parameters

<i>c</i>	
----------	--

## Returns

true It's not

false It is

## 5.8 GBReader.h File Reference

Sequence extractor for GenBank-like files.

```
#include <iostream>
```

### Functions

- bool [getSequence](#) (std::string &output, std::istream &file)

### 5.8.1 Detailed Description

Sequence extractor for GenBank-like files.

## Authors

DALZOTTO, Rafael; HEIR, Alejandro - Grupo 7

**Version**

0.1

**Date**

2022-05-08

**Copyright**

Copyright (c) 2022 - 22.08 EDA - ITBA

## 5.8.2 Function Documentation

### 5.8.2.1 getSequence()

```
bool getSequence (
    std::string & output,
    std::istream & file )
```

## 5.9 GBReader.h

[Go to the documentation of this file.](#)

```
1
12 #ifndef GBREADER_H
13 #define GBREADER_H
14
15 #include <iostream>
16
17 bool getSequence(std::string &output, std::istream &file);
18
19 #endif
```

## 5.10 main.cpp File Reference

Runs the genetic sequence aligner.

```
#include <iostream>
#include <fstream>
#include <chrono>
#include "GBReader.h"
#include "Aligner.h"
```

### Functions

- void [printElapsedFrom](#) (time\_point< steady\_clock > &timestamp)
- int [main](#) (int argc, char \*\*argv)



### 5.10.1 Detailed Description

Runs the genetic sequence aligner.

#### Authors

DALZOTTO, Rafael; HEIR, Alejandro - Grupo 7

#### Version

0.1

#### Date

2022-05-08

#### Copyright

Copyright (c) 2022 - 22.08 EDA - ITBA

### 5.10.2 Function Documentation

#### 5.10.2.1 main()

```
int main (
    int argc,
    char ** argv )
```

#### 5.10.2.2 printElapsedFrom()

```
void printElapsedFrom (
    time_point< steady_clock > & timestamp )
```

## 5.11 main\_test.cpp File Reference

Tests for the genetic sequence aligner.

```
#include <iostream>
#include <string>
#include <array>
#include <fstream>
#include <sstream>
#include "GBReader.h"
#include "Aligner.h"
```

## Functions

- bool `testMatInit` (`DirectionMat` &mat, `DirectionMat` &expectedMat)
- bool `testFillMat` (`DirectionMat` &mat, const string &s1, const string &s2, int32\_t expectedScore, `DirectionMat` &expectedMat)
- bool `testAlignment` (`DirectionMat` &mat, const string &s1, const string &s2, array< string, 3 > &expectedAlignment)
- int `main` ()

### 5.11.1 Detailed Description

Tests for the genetic sequence aligner.

#### Authors

DALZOTTO, Rafael; HEIR, Alejandro - Grupo 7

#### Version

0.1

#### Date

2022-05-08

#### Copyright

Copyright (c) 2022 - 22.08 EDA - ITBA

### 5.11.2 Function Documentation

#### 5.11.2.1 `main()`

```
int main ( )
```

#### 5.11.2.2 `testAlignment()`

```
bool testAlignment (
    DirectionMat & mat,
    const string & s1,
    const string & s2,
    array< string, 3 > & expectedAlignment )
```

### 5.11.2.3 testFillMat()

```
bool testFillMat (
    DirectionMat & mat,
    const string & s1,
    const string & s2,
    int32_t expectedScore,
    DirectionMat & expectedMat )
```

### 5.11.2.4 testMatInit()

```
bool testMatInit (
    DirectionMat & mat,
    DirectionMat & expectedMat )
```

## 5.12 README.md File Reference



# Index

- ~DirectionMat
  - DirectionMat, [9](#)
- Aligner.cpp, [11](#)
  - buildAligment, [12](#)
  - DIAGONAL, [12](#)
  - Direction, [12](#)
  - fillMat, [13](#)
  - getGlobalAlignment, [13](#)
  - HORIZONTAL, [12](#)
  - initMat, [14](#)
  - VERTICAL, [12](#)
- Aligner.h, [14](#)
  - buildAligment, [15](#)
  - fillMat, [15](#)
  - getGlobalAlignment, [15](#)
  - initMat, [15](#)
- at
  - DirectionMat, [9](#)
- buildAligment
  - Aligner.cpp, [12](#)
  - Aligner.h, [15](#)
- cols
  - DirectionMat, [10](#)
- d0
  - DirCell, [7](#)
- d1
  - DirCell, [7](#)
- d2
  - DirCell, [7](#)
- d3
  - DirCell, [8](#)
- DIAGONAL
  - Aligner.cpp, [12](#)
- DirCell, [7](#)
  - d0, [7](#)
  - d1, [7](#)
  - d2, [7](#)
  - d3, [8](#)
- Direction
  - Aligner.cpp, [12](#)
- DirectionMat, [8](#)
  - ~DirectionMat, [9](#)
  - at, [9](#)
  - cols, [10](#)
  - DirectionMat, [8](#)
  - print, [9](#)
  - rows, [10](#)
  - set, [10](#)
- DirectionMat.cpp, [16](#)
- DirectionMat.h, [16](#)
- fillMat
  - Aligner.cpp, [13](#)
  - Aligner.h, [15](#)
- GBReader.cpp, [18](#)
  - getSequence, [18](#)
  - isNotACGT, [19](#)
- GBReader.h, [19](#)
  - getSequence, [20](#)
- getGlobalAlignment
  - Aligner.cpp, [13](#)
  - Aligner.h, [15](#)
- getSequence
  - GBReader.cpp, [18](#)
  - GBReader.h, [20](#)
- HORIZONTAL
  - Aligner.cpp, [12](#)
- initMat
  - Aligner.cpp, [14](#)
  - Aligner.h, [15](#)
- isNotACGT
  - GBReader.cpp, [19](#)
- main
  - main.cpp, [21](#)
  - main\_test.cpp, [22](#)
- main.cpp, [20](#)
  - main, [21](#)
  - printElapsedFrom, [21](#)
- main\_test.cpp, [21](#)
  - main, [22](#)
  - testAlignment, [22](#)
  - testFillMat, [22](#)
  - testMatInit, [23](#)
- print
  - DirectionMat, [9](#)
- printElapsedFrom
  - main.cpp, [21](#)
- README.md, [23](#)
- rows
  - DirectionMat, [10](#)

set

DirectionMat, [10](#)

testAlignment

main\_test.cpp, [22](#)

testFillMat

main\_test.cpp, [22](#)

testMatInit

main\_test.cpp, [23](#)

VERTICAL

Aligner.cpp, [12](#)