Note:

Le nom des fichiers est de couleur rouge.

Le nom des méthodes est de couleur vert.

Les lignes indiquées de couleur bleu correspondent aux lignes du projet initial.

Les lignes indiquées de couleur orange correspondent aux lignes du projet rendu.

Partie I: Fuites mémoires

I) Dans MatchingLibrary

Fichier CommandLine.cpp méthode void CommandLine::prepareCommand(int argc, char \*argv[])

(L.60 à 75,L.60 à 75) La variable std::string\* lastArgument n’est plus un pointeur et n’est plus alloué.

Fichier CommandLineApplication.cpp méthode void CommandLineApplication::run() (L.132, L.132) La méthode readFile retourne un vector<parameters> et pas un parameters. Le premier élément est pris et le vector est libéré.

Fichier CommandLineApplication.cpp méthode void CommandLineApplication::run() (L.214, L.220) Libération des variables (propre à la méthode) executor et parameters.

Fichier CommandLineApplication.cpp méthode void CommandLineApplication::run() (L.199, L.201) Condition

*if (parserName.compare("") != 0) {*

*executor->setParser(parsers[parserName]);*

*}*

changé par

*if (parserName.compare("") != 0) { executor->setParser(parsers[parserName]);*

*}*

*else {*

*executor->setParser(parsers[\_csv]);*

*}*

et dans la méthode std::vector<model::Sequence\*>\* MatchingExecutor::readFile(std::string file) du fichier MatchingExecutor.cpp (L53 à 56) retiré (déplacé) car le parser alloué que si il n’y en avait pas un déjà dans MatchingExecutor n’était pas libéré. Ce changement fait que le destructeur de la classe CommandLineApplication détruit automatiquement le parser.

Fichier CommandLineApplication.cpp (L.225) Destructeur CommandLineApplication::~CommandLineApplication() ajouté pour libérer les maps methods et parsers.

Fichier CSVParser.cpp (L.219,L.219) méthode void\* CSVParser::readFile(void\* document) Libération des variables (propre à la méthode) docr et parser.

Fichier ESCCorrespondence.cpp (L.253 et 260,L.259 et 266) méthode vector<ResultCorrespondence> \*ESCCorrespondence::match(model::Sequence \*s1, model::Sequence \*s2)

La variable int \*\* skipableParent (propre à la méthode) n’est pas libérée donc ajout de delete skipableParent[i] dans la boucle de libération existante et delete skipableParent.

Fichier EXTParser.cpp (L.221,L.221) méthode void\* EXTParser::readFile(void\* document) Libération des variables (propre à la méthode) docr et parser.

Fichier MatchingExecutor.h (L.122,138,186,L.124,142,192) méthodes void setMethod(Correspondence\* cor), void setParameters(Parameters\* par), void setParser(inout::SequenceParser\* p) Libération des attributs (pointeurs) si ils ne sont pas à NULL.

Fichier MatchingExecutor.cpp (L.107,L.129 à 133) méthode std::vector<std::string> MatchingExecutor::match(std::vector<model::Sequence\*>\* target, std::vector<model::Sequence\*>\* ref) Suite aux calculs, libération de la variable et de son contenu vector<tools::ResultCorrespondence>\* result.

Fichier MatchingExecutor.cpp (L.180,L.254 à 261) méthode std::string MatchingExecutor::execute() Libération des variables std::vector<model::Sequence\*> \*referenceSequences et \*targetSequences et de leurs contenus.

Fichier MVMCorrespondence.cpp (L.174,L.213) méthode vector<ResultCorrespondence> \*MVMCorrespondence::match(model::Sequence \*s1, model::Sequence \*s2) Libération de la variable vector<int> \*j\_min

Fichier Sequence.cpp (L.31,L.31) constructeur de copie Sequence::Sequence(const Sequence &copy) Ligne *vSlist->insert(vSlist->begin(), copy.vSlist->at(it))* remplacé par *vSlist->insert(vSlist->begin(), copy.vSlist->at(it)->copy())* car la méthode at() de la classe vector retourne l’élément lui même et non une copie.

Fichier Sequence.cpp (L.61,L.62) méthode void Sequence::removeElement(unsigned int index) Ajout de la ligne *delete vSlist->at(index)* où vSlist est un vector<Element\*> \* car la méthode erase de la classe vector ne libère pas les éléments contenus.

Fichier Sequence.cpp (L.93,L.94) méthode Sequence \*Sequence::copy() Ligne *n->addElement(this->getElement(it)->copy())* remplacé par *n->addElement(this->getElement(it))* car la méthode addElement fait déja une copie.

Fichier XMLParser.cpp (L.85,107,148,L.85,107,149) méthode void \*XMLParser::readFile(void \*document) Libération des variables cha, nu et vec.

II) Dans MatchingToolBoxTest

Fichier CSVParserTest.cpp (L.30,45,60,L.30 à 33, 48 à 51, 66 à 69) méthodes TEST(CSVParserTest, reading\_numeric\_csv\_file), TEST(CSVParserTest, reading\_vector\_csv\_file), TEST(CSVParserTest, reading\_character\_csv\_file) Libération de la variable std::vector<model::Sequence \*> \* sequences et de son contenus. *free* remplacé par *delete*

Fichier XMLParserTest.cpp (L.21,42,68,L.22 à 25, 47 à 50, 77 à 80) méthodes TEST(XMLParserTest, reading\_character\_xml\_files)), TEST(XMLParserTest, reading\_numeric\_xml\_files), TEST(XMLParserTest, reading\_vector\_xml\_files) Libération de la variable std::vector<model::Sequence \*> \* sequences et de son contenus. *free* remplacé par *delete*

Partie II: Projet Tests fonctionnels

Les 3 premiers tests de chaque algorithme utilisent un fichier paramètre. Ce fichier contient les valeurs par défaut, donc c’est comme si on lançait l’algo sans fichier paramètre.

Format: TARGET | REFERENCE

**LEVENSHTEIN**

**TEST\_F(Levenshtein\_SystemTest, levenshtein\_character\_csv)**

c,h,i,e,n,s | n,i,c,h,e

=>distance = 5 OK

**TEST\_F(Levenshtein\_SystemTest, levenshtein\_character\_ext)**

celaestlepetittest | ceciestunpetittest

=>distance = 4 OK

TEST\_F(Levenshtein\_SystemTest, levenshtein\_character\_xml)

relevant | elephant

=>distance = 3 OK

**TEST\_F(Levenshtein\_SystemTest, levenshtein\_numeric\_csv)**

1,2,9,3,3,5,9 | 1,2,8,6,8

=>distance = 5 OK

**TEST\_F(Levenshtein\_SystemTest, levenshtein\_numeric\_ext)**

53181123981435041715112643846 | 5318183499143584715112643846

=>distance = 6 OK

**TEST\_F(Levenshtein\_SystemTest, levenshtein\_numeric\_xml)**

1234567890123456789 | 34567012345670123

=>distance = 8 OK

**TEST\_F(Levenshtein\_SystemTest, levenshtein\_vector\_csv)**

abcde | faghijkclmneo //Les vecteurs sont symbolisés par des lettres, donc

=>distance = 10 OK //abcde correspond à une séquence de 5 vecteurs différents

**TEST\_F(Levenshtein\_SystemTest, levenshtein\_vector\_ext)**

abc | daefbgcb

=>distance = 5 OK

**TEST\_F(Levenshtein\_SystemTest, levenshtein\_vector\_xml)**

abcde | abcde

=>distance = 0 OK

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**DTW**

Une lettre de la séquence target par ligne et une lettre de la séquence référence par colonne

* Pour les characters, lorsque les éléments sont différents, la distance entre les deux éléments est de 1 par défaut.

Exemple: distance(a, p) = 1 distance(g,g) = 0

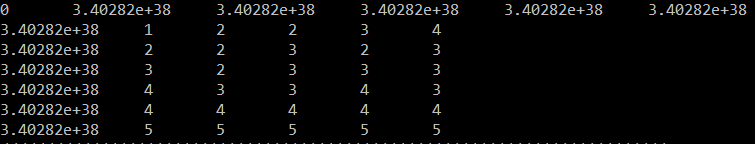
* Pour les vecteurs, lorsque les vecteurs sont différents, la distance entre les deux éléments est égal à sqrt(∑(pow(elementvec1 - elementvec2, 2)))

Exemple: distance(Vec[1,2,3,4,5], Vec[0,0,0,0,0]) = sqrt((1-0)² + (2-0)² + (3-0)² +(4-0)² +(5-0)²) = sqrt(55)

Les matrices distances affichées sont à la puissance 2.

**TEST\_F(DTW\_SystemTest, DTW\_character\_csv)**

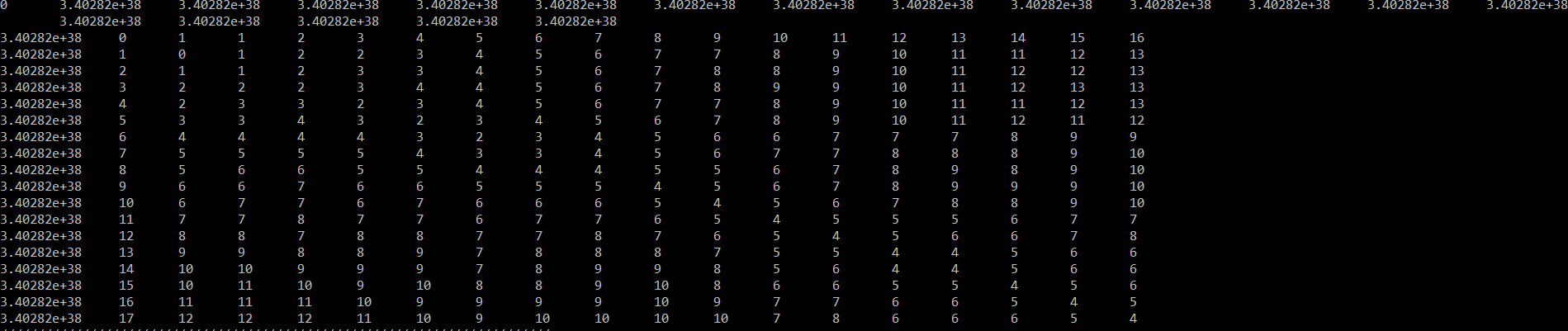
c, h, i, e, n, s | n, i, c, h, e



=>distance = sqrt(5) => OK

**TEST\_F(DTW\_SystemTest, DTW\_character\_ext)**

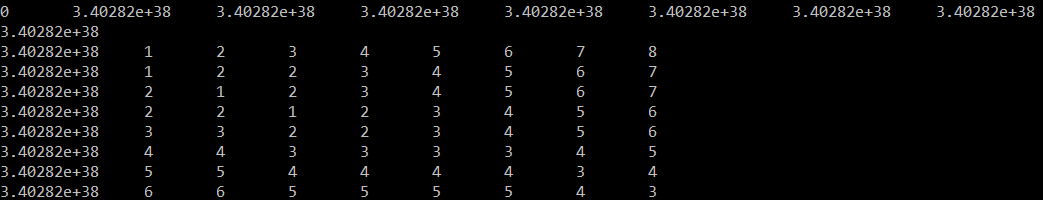
c e l a e s t l e p e t i t t e s t | c e c i e s t u n p e t i t t e s t



=>distance = sqrt(4) => OK

**TEST\_F(DTW\_SystemTest, DTW\_character\_xml)**

r, e, l, e, v, a, n, t | e, l, e, p, h, a, n, t



=>distance = sqrt(3) => OK

**TEST\_F(DTW\_SystemTest, DTW\_numeric\_csv)**

1,2,9,3,3,5,9 | 1,2,8,6,8 (exemple rapport)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | 1 | 2 | 8 | 6 | 8 |
|  | 0 | **∞** | **∞** | **∞** | **∞** | **∞** |
| 1 | **∞** | 0 | 1 | 8 | 13 | 20 |
| 2 | **∞** | 1 | 0 | 6 | 10 | 16 |
| 9 | **∞** | 9 | 7 | 1 | 4 | 5 |
| 3 | **∞** | 11 | 8 | 6 | 4 | 9 |
| 3 | **∞** | 13 | 9 | 11 | 7 | 9 |
| 5 | **∞** | 17 | 12 | 12 | 8 | 10 |
| 9 | **∞** | 25 | 19 | 13 | 11 | 9 |

=>distance = sqrt(9) =3 comme dans le rapport => OK

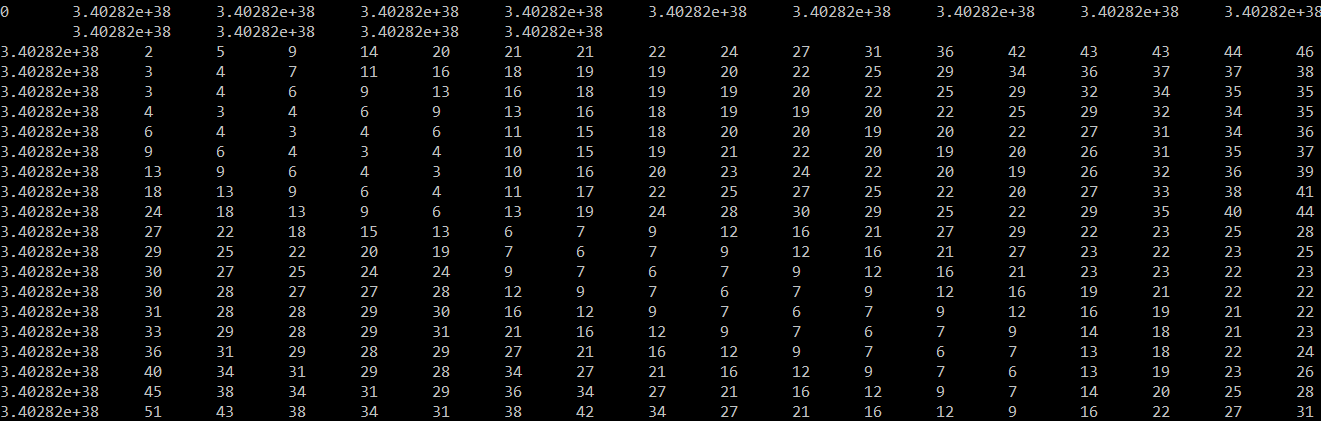
**TEST\_F(DTW\_SystemTest, DTW\_numeric\_ext)**

5 3 1 8 1 1 2 3 9 8 1 4 3 5 0 4 1 7 1 5 1 1 2 6 4 3 8 4 6 | 5 3 1 8 1 8 3 4 9 9 1 4 3 5 8 4 7 1 5 1 1 2 6 4 3 8 4 6

=>distance = sqrt(18) => OK

**TEST\_F(DTW\_SystemTest, DTW\_numeric\_xml)**

1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 | 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3



=>distance = sqrt(31) => OK

**TEST\_F(DTW\_SystemTest, DTW\_vector\_csv)**

1,2,3,4,5,6,7,8,9

1234,2134,3214,4231,5234,6234,7234,8234,9234

9,8,7,6,5,4,3,2,1

0.1,0.84,5.5,1.127,4.66,12.6,5,2.5,3.77

0.1,0.84,5.5,1.127,4.68,12.6,5,2.5,3.77

|

0,0,0,0,0,0,0,0,0

1,2,3,4,5,6,7,8,9

2,1,3,4,5,6,7,8,9

3,2,1,4,5,6,7,8,9

4,2,3,1,5,6,7,8,9

5,2,3,4,1,6,7,8,9

6,2,3,4,5,1,7,8,9

9,8,7,6,5,4,3,2,1

7,2,3,4,5,6,1,8,9

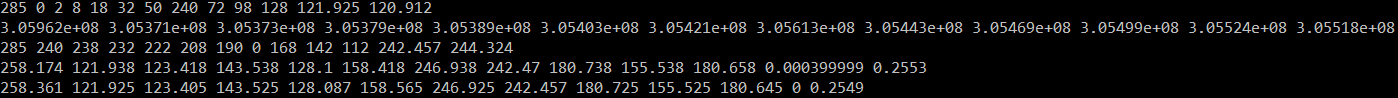
8,2,3,4,5,6,7,1,9

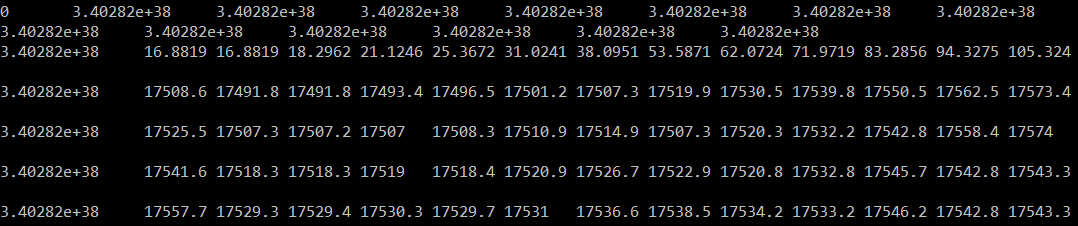
9,2,3,4,5,6,7,8,1

0.1,0.84,5.5,1.127,4.68,12.6,5,2.5,3.77

0.1,0.84,5.5,1.127,4.68,12.6,5.5,2.5,3.7

Matrice des distances vecteurs (premières valeurs vérifiées):





distance=>sqrt(17543.3) => OK

**TEST\_F(DTW\_SystemTest, DTW\_vector\_ext)**

0.15 1.78 2.86 3.30 4.98 5.12 6.18

2598.95 3838.78 2423.01 3838.82 9828.00 65.69 8823.77

9 3 5 1 3 7 2

|

0.15 1.78 2.86 3.31 4.98 5.12 6.18

0.15 1.78 2.86 3.30 4.98 5.12 6.18

0.15 1.78 2.86 3.30 4.99 5.12 6.18

0 0 0 0 0 0 0

2598.95 3838.78 2423.01 3838.82 9828.00 65.69 8823.77

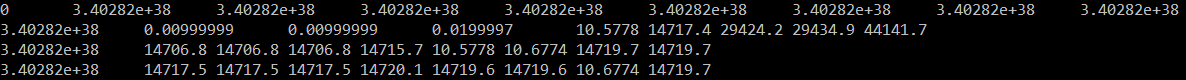
2598.95 3838.78 2423.01 3838.82 9828.10 65.69 8823.77

9 3 5 1 3 7 2

2598.95 3838.78 2423.01 3838.82 9828.00 65.69 8823.77

Matrice des distances vecteurs (non vérifiée car valeurs trop grandes)





distance=>sqrt(1419.7) => OK

**TEST\_F(DTW\_SystemTest, DTW\_vector\_xml)**

1 2 3

4 5 6

7 8 9

1 2 3

4 5 6

|

1 2 3

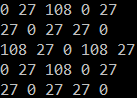
4 5 6

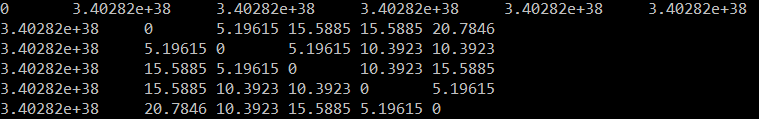
7 8 9

1 2 3

4 5 6

Matrice des distances vecteurs(vérifiée)





distance=>sqrt(0)=>OK

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**LCS**

**TEST\_F(LCS\_SystemTest, LCS\_character\_csv)**

g,a,c | a,g,c,a,t

=>distance = 2 comme dans l’exemple du rapport

**TEST\_F(LCS\_SystemTest, LCS\_character\_ext)**

p o r t e | r a p p o r t e r

=>distance = 5 comme dans l’exemple du rapport

**Autres tests avec LCS**

**TEST\_F(LCS\_SystemTest, LCS\_fichier\_vide\_csv):**

On passe un fichier vide à la reférence

et un fichier non vide au niveau de la target

quand on le compile , le programme bug avec une exception qui est levé

de type std::out\_of\_range

Comme le programme n'est pas fini , il ya des fuites mémoires

**TEST\_F(LCS\_SystemTest, LCS\_fichier\_type\_different\_csv):**

comparaison de fichier de nature caractere non identique

type de fichiers différents

target : xml

reférence : csv

le programme compile sans aucune exception n'est indiquée , il applique l'algoritme mais

avec des distances = 0 et sans correspondance

Result

Distance = 0

Correspondance

<

-

- - - - - - -

a b a c c t y

Longest common sequence : {}

**TEST\_F(LCS\_SystemTest, LCS\_fichier\_type\_different\_nature):**

comparaison de fichier de nature non identique

et de type de fichiers différents

target : xml , numéric

reférence : csv , charactere

le programme compile sans aucune exception n'est indiquée , il applique l'algoritme mais

avec des distances = 0 et sans correspondance

**TEST\_F(LCS\_SystemTest, LCS\_fichier\_nature\_different\_Num\_Char\_csv):**

comparaison de fichier de nature non identique

et type de fichiers identique

target : NUM , csv

ref : char , csv

Marche sans aucune exception avec une distance = 0

Result

Distance = 0

Correspondance

9 2 9 3 4 3 5 6 9 1

- - - - - - - - - -

- - - - - - -

a b a c c t y

Longest common sequence : {}

**TEST\_F(LCS\_SystemTest, LCS\_fichier\_nature\_different\_Vector\_Num\_csv):**

comparaison de fichier de nature non identique

et type de fichiers identique

target : csv , vector

reférence : csv , numeric

Avec un fichier de parametre , il y a une exception de type SequenceMatchingException qui est lévé lors de

la lecture du fichier parametre

**TEST\_F(LCS\_SystemTest, LCS\_fichier\_nature\_different\_Vector\_Num\_csv\_2):**

comparaison de fichier de nature non identique

et type de fichiers identique

target : csv , vector

reférence : csv , numeric

Sans un fichier de parametre , alors pas d'erreur , l'algo s'applique et arrive à trouver

des séquences communs et des distances (vu que ce sont des vecteurs numéric)

En sortie deux fichiers contenant le long sequence commun

**TEST\_F(LCS\_SystemTest, LCS\_fichier\_nature\_different\_Char\_Vector\_csv):**

comparaison de fichier de nature non identique

et type de fichiers identique

target : csv , Character

ref : csv , vector

Sans un fichier de parametre , une exception est lévé lors de la lecture du fichier CSV

Erreur ligne 20, type ERREUR\_FILE : CSV Reading error : unable to open the file

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**CDP**

**TEST\_F(CDP\_SystemTest, CDP\_numeric\_csv)**

1,2,9,16,9,25,37 | 1,2,8,16,16 (exemple rapport)

=>distance = 1.6 comme dans l’exemple du rapport

Les 8 autres tests sont posés mais il n’y a pas de résultat attendu défini.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**FSM**

**TEST\_F(FSM\_SystemTest, FSM\_numeric\_csv)**

1,2,9,16,9,25,37 | 1,2,8,16,16 (exemple rapport)

=>distance = 0.56 comme dans l’exemple du rapport

Les 8 autres tests sont posés mais il n’y a pas de résultat attendu défini.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ESC**

**TEST\_F(ESC\_SystemTest, ESC\_numeric\_csv)**

73,1,2,3,49,12,9,123,33 | 1,2,3,49,64,34,9,22 (exemple rapport)

=>distance = 6.8 comme dans l’exemple du rapport

Les 8 autres tests sont posés mais il n’y a pas de résultat attendu défini.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**MVM**

Les 9 tests sont posés mais il n’y a pas de résultat attendu défini.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Format XML**

L’ensemble des tests de cette partie utilise l’algorithme de levenshtein afin de faciliter l’analyse des résultats obtenus.

La conclusion de ces tests est que le nom des balises n’a pas d’importance.

**TEST(XMLFormat\_SystemTest, XMLFormat\_testKey\_idS\_String\_char)** remplace le résultat de la balise idS = string par la valeur x dans les deux fichiers.

Aucune exception n’est levé lors de l’appel de run() pour exécuter l’algorithme et le résultat retourné est correct.

**TEST(XMLFormat\_SystemTest, XMLFormat\_testKey\_idS\_targetOnly\_String\_char**) fait le même test à la différence que seul le fichier target à la valeur x pour la clé idS.

Le bon résultat est retourné.

**TEST(XMLFormat\_SystemTest, XMLFormat\_testKey\_id\_string\_num)** remplace le résultat de la balise id (sous balise de idS) = string par la valeur x dans les deux fichiers.

Le bon résultat est retourné.

**TEST(XMLFormat\_SystemTest, XMLFormat\_testKey\_id\_referenceOnly\_string\_num)** remplace le résultat de la balise id uniquement dans le fichier reference par la valeur x.

Le bon résultat est retourné.

Les tests suivants sont les mêmes que ci dessus mais avec le type vector. Le résultat retourné est correct:

**TEST(XMLFormat\_SystemTest, XMLFormat\_testKey\_idS\_string\_vec)**

**TEST(XMLFormat\_SystemTest, XMLFormat\_testKey\_idS\_targetOnly\_string\_vec)**

**TEST(XMLFormat\_SystemTest, XMLFormat\_testKey\_id\_string\_vec)**

**TEST(XMLFormat\_SystemTest, XMLFormat\_testKey\_id\_targetOnly\_string\_vec)**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Différents Types puis différents Formats**

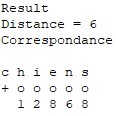
L’ensemble des tests de cette partie utilise l’algorithme de levenshtein afin de faciliter l’analyse des résultats obtenus.

Le premier type ou format indiqué dans le test est celui indiqué dans la commande.

exemple:

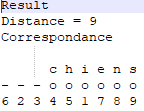
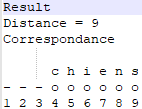
Dans le test **TEST(DifferentTypeFormat\_SystemTest, DifferentType\_char\_num)**, le type indiqué est character.

**TEST(DifferentTypeFormat\_SystemTest, DifferentType\_char\_num)** Test entre le type character et numérique. Ce test ne lève pas d’exception mais le résultat retourné est correct:

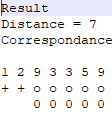


**TEST(DifferentTypeFormat\_SystemTest, DifferentType\_char\_vec)** Test entre le type character et vecteur. Ce test ne lève pas d’exception.

A la suite de son exécution, plusieurs fichiers sont créé. Leurs résultats par rapport aux deux séquences présentées dans le résultat est correct mais ces résultats n’ont pas de sens par rapport aux deux fichiers target et référence donnés en paramètres:



**TEST(DifferentTypeFormat\_SystemTest, DifferentType\_num\_char)** Test entre le type numéric et character. Le test ne lève pas d’exception. Le résultat n’est pas correct car les lettres deviennent des 0.



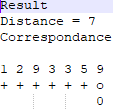
**TEST(DifferentTypeFormat\_SystemTest, DifferentType\_num\_vec)** Test entre le type numéric et vecteur.

Même chose que le test TEST(DifferentType\_SystemTest, DifferentType\_char\_vec)

**TEST(DifferentTypeFormat\_SystemTest, DifferentType\_vec\_char)** Test entre le type vecteur et character. Ce test fonctionne. Une exception est levée.

**TEST(DifferentTypeFormat\_SystemTest, DifferentType\_vec\_num)** Test entre le type vecteur et numéric. Ce test fonctionne. Une exception est levée.

**TEST(DifferentTypeFormat\_SystemTest, DifferentFormat\_csv\_xml)** Test entre le format csv et xml. Le test n’est pas validé. Aucune exception est levée et le second fichier est mal interprété ce qui génère plusieurs fichiers résultats incorrect:



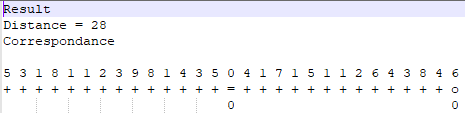
**TEST(DifferentTypeFormat\_SystemTest, DifferentFormat\_csv\_ext)** Test entre le format csv et ext. Le test n’est pas validé. Une exception est levée mais non gérée par la librairie.

**TEST(DifferentTypeFormat\_SystemTest, DifferentFormat\_xml\_csv)** Test entre le format xml et csv. Une exception est levée mais non gérée par la librairie.

**TEST(DifferentTypeFormat\_SystemTest, DifferentFormat\_xml\_ext)** Test entre le format xml et ext. Une exception est levée mais non gérée par la librairie.

**TEST(DifferentTypeFormat\_SystemTest, DifferentFormat\_ext\_csv)** Test entre le format ext et csv. Le test fonctionne, une exception est levée.

**TEST(DifferentTypeFormat\_SystemTest, DifferentFormat\_ext\_xml)** Test entre le format ext et xml. Le test ne passe pas. Aucune exception est levée. Plusieurs résultats qui n’ont pas de sens sont retournés:



**Séquences corrompues**

**Fichiers en param vides:**

TEST(CorruptFile\_SystemTest, CorruptFile\_csv\_charTAR\_vide):

* Méthode qui teste l’exécution de l’algo si le fichier target\_character.csv est vide
* Le test ne passe pas si le fichier est vide, levée d’une exception
* Ce test ne passe pas car une exception externe à la librairie est levée.

TEST(CorruptFile\_SystemTest, CorruptFile\_csv\_vecTAR\_vide):

* Méthode qui teste l’exécution de l’algo si le fichier target\_vector.csv est vide
* Le test ne passe pas si le fichier est vide, levée d’une exception
* Ce test ne passe pas car une exception externe à la librairie est levée.

TEST(CorruptFile\_SystemTest, CorruptFile\_xml\_numTAR\_vide):

* Méthode qui teste l’exécution de l’algo si le fichier target\_numeric.xml est vide
* Le test ne passe pas si le fichier est vide, levée d’une exception
* Ce test passe, une exception de type SequenceMatchingException est levée.

TEST(CorruptFile\_SystemTest, CorruptFile\_xml\_vecTAR\_vide):

* Méthode qui teste l’exécution de l’algo si le fichier target\_vector.xml est vide
* Le test ne passe pas si le fichier est vide, levée d’une exception
* Ce test passe, une exception de type SequenceMatchingException est levée.

TEST(CorruptFile\_SystemTest, CorruptFile\_ext\_charTAR\_vide):

* Méthode qui teste l’exécution de l’algo si le fichier target\_character.extl est vide
* Le test ne passe pas si le fichier est vide, levée d’une exception
* Ce test ne passe pas car une exception externe à la librairie est levée.

TEST(CorruptFile\_SystemTest, CorruptFile\_ext\_vecTAR\_vide):

* Méthode qui teste l’exécution de l’algo si le fichier target\_vector.extl est vide
* Le test ne passe pas si le fichier est vide, levée d’une exception
* Ce test ne passe pas car une exception externe à la librairie est levée.

**Suppression séparateurs et ajout contenu erroné:**

TEST(CorruptFile\_SystemTest, CorruptFile\_csv\_charTAR\_Supp\_Separateurs):

* Méthode qui teste l’ajout de contenu erroné après suppression des séparateurs dans le fichier target\_character.csv
* La méthode est censé lever une exception si le format du fichier est incorrect
* 1er contenu testé: **chiens** (virgules servant de séparateurs retirés)
  + Le test ne passe pas car pas d’exception levée
* 2ème contenu testé: **chiens256871é'**
  + Le test ne passe pas car pas d’exception levée
* 3ème contenu testé: 1ère ligne (**chiens256871é')** et 2nde ligne (**chiens256871é'**)
  + Le test ne passe pas car pas d’exception levée
* 4ème contenu testé: **c,h,i,e,n,s,2;5,4%,1**
  + Le test ne passe pas car pas d’exception levée

TEST(CorruptFile\_SystemTest, CorruptFile\_csv\_vecTAR\_Supp\_Separateurs):

* Méthode qui teste l’ajout de contenu erroné après suppression des séparateurs dans le fichier target\_vector.csv
* 1er contenu testé: **1,2,3,4,5,6,l,k,i**
  + Le test ne passe pas car pas d’exception levée
* 2eme contenu testé: **123456789** (sans les virgules qui servent de séparateurs)
  + Le test ne passe pas car pas d’exception levée de type SequenceMatchingException

TEST(CorruptFile\_SystemTest, CorruptFile\_csv\_numTAR\_Supp\_Separateurs):

* Méthode qui teste l’ajout de contenu erroné après suppression des séparateurs dans le fichier target\_numeric.csv
* 1er contenu testé: **1,2,9,3,3,5,9,m,p,i,k**
  + Le test ne passe pas car pas d’exception levée

*On peut supposer que le système accepte les caractères séparés comme les numériques. Toutefois, le résultat n’est pas forcément correct car non vérifié*

* 2eme contenu testé: **1,2,9,3,3,5,9,mstgsegr,gwdrgp**
  + Le test ne passe pas car pas d’exception levée => *il serait possible de regrouper les caractères, cependant, le résultat n’est pas vérifié.*
* 3eme contenu testé: **123456789**
  + Le test ne passe pas car pas d’exception levée => *Il est possible de supprimer les séparateurs (virgule), elles seraient alors dispensables. Le résultat n’est pas vérifié.*
* 4eme contenu testé: **123456789,qzrzAZEdsfq,2hjsqd**
  + Le test ne passe pas car pas d’exception levée => *Il est possible de mélanger à la fois des caractères avec des numériques, le résultat n’est pas vérifié.*
* 5ème contenu testé: 1ere ligne (**123456789,qzrzAZEdsfq,2hjsqd**) et 2nde ligne (**123456789,qzrzAZEdsfq,2hjsqd**)
  + Le test ne passe pas car pas d’exception levée

Pour conclure, que ce soit caractères ou numériques mélangés, aucune exception n’a été levée malgré le changement de type et la suppression de séparateurs.

TEST(CorruptFile\_SystemTest, CorruptFile\_xml\_numTAR\_Supp\_Separateurs):

* Méthode qui teste l’ajout de contenu erroné après suppression des séparateurs dans le fichier target\_numeric.xml
* 1er contenu testé: suppression d’une balise de fermeture
  + Le test passe, une exception de type SequenceMatchingException est levée
* 2eme contenu testé : rajouter du contenu dans le champ value tel que **value="2reqcrq"** le tout séparé de balises
  + Le test ne passe pas car pas d’exception levée
* 3eme contenu testé: changer le type de la séquence: string -> integer
  + Le test ne passe pas car pas d’exception levée
* 4eme contenu testé: ajouter des caractères avec la séquence d’entiers tel que **id="integer" value="1fdqserqze"**
  + Le test ne passe pas car pas d’exception levée

Les résultats de chaque cas où les tests ne passent sans lever d’exception ne sont pas vérifiés.

Finalement, il semble que l’on puisse mélanger n’importe quels types de variables ensemble même si le type précisé dans idS et id de chaque ligne est différent. Il n’y a aucune analyse du type des valeurs données. La seule contrainte est qu’il faut respecter la mise en forme xml pour cela: mise en place des balises,...

TEST(CorruptFile\_SystemTest, CorruptFile\_xml\_charTAR\_Supp\_Separateurs):

* Méthode qui teste l’ajout de contenu erroné après suppression des séparateurs dans le fichier target\_numeric.xml

Comme vu précédemment et sachant que cela ne fonctionne pas car vérifié, les tests de suppression de balises ne seront pas faits pour ce test.

* 1er test effectué: suppression d’un champ xml **value = “<valeur>”**
* Levée d’une exception de type “Refus d’accès à la mémoire” car aucune valeur n’a été renseignée dans le champ en raison de la suppression

Une exception non gérée par la librairie est levée.

Il est donc impossible de supprimer la structure du xml sans provoquer de levée d’exception. Cependant, les valeurs prises dans le champ value et id de chaque ligne à analyser sont totalement libres: on peut mettre ce que l’on veut pour les valeurs ainsi que pour le type

* 2eme test effectué: modification du champ id tel que **<Element id="bouh" value="r865456413" />**
  + Le test est passé sans lever d’exception, ce qui confirme ce qui est supposé au-dessus.

Une dernière méthode de test a été effectué mais le résultat de chaque modification regroupe déjà tout ce qui a été fait précédemment après avoir effectué les mêmes tests sur la méthode:

TEST(CorruptFile\_SystemTest, CorruptFile\_xml\_vecTAR\_Supp\_Separateurs).

Partie III: Modification du code de la bibliothèque (MatchingLibrary)

Dans TOUT le projet: \_\_min() et \_\_max() remplacé par std::min et std::max car \_\_min et \_\_max sont propre à VisualStudio.

Fichier CDPCorrespondence.cpp méthode float CDPCorrespondence::min3(float f1, float f2, float f3) (L.11, L.11) supprimée. La fonction standard std::min() la remplace.

Fichier ParamParser.cpp méthode void \*ParamParser::readFile(void \*document) (L.201, L.201) Erreur trouvé qui ne mettait pas la variable v1\_node à jour :

*v1\_node->next\_sibling()* remplacé par *v1\_node = v1\_node->next\_sibling()*

Fichier MatchingExecutor.cpp méthode std::vector<model::Sequence\*>\* MatchingExecutor::readFile(std::string file) (L.58 à 70) Lignes supprimées car incohérent. Le fichier parameters ne détermine pas le type des séquences

Fichier MatchingExecutor.cpp méthode std::vector<model::Sequence\*>\* MatchingExecutor::readFile(std::string file) (L.73, L.77 à 85) L’appel à la méthode readFile pour parser les données ne fonctionnait pas pour le type xml. Les fichiers sont d’abord parsé une première fois puis au format rapid\_xml.

Fichier MathcingExecutor.cpp méthode void MatchingExecutor::writeOutput(std::string& directoryPath, std::vector<std::string> res) (L.137 à 141, L.209 à 214) Les dossiers contenant les fichiers de sorties sont nommés ainsi pour savoir directement quelle méthode, format et type de données ont été utilisés:

resultat\_target\_reference\_method\_type\_parser

Fichier Parameters.cpp méthode void Parameters::swapSequences() Nouvelle méthode créée. Elle a pour but d’échanger les valeurs entre celles de la séquence target et référence. Cette méthode est appelée dans les méthodes match(model::Sequence \*s1, model::Sequence \*s2) de ESCCorrespondence(L.26), FSMCorrespondance(L.21) et MVMCorrespondence(L.24)

Dans le rapport de Mr MEUNIER il est écrit pour les algorithmes ESC et FSM que la longueur de la séquence target doit être plus longue que la séquence référence. Dans le rapport de Mr ADEN HASSAN il est écrit la même chose pour l’algorithme MVM.

Sans l’appel de cette méthode, les algorithmes lisent au delà de la limite des conteneurs de parameters ce qui provoque un crash. Avec cette méthode, les algorithmes s’arrêtent pile à la dernière valeur des conteneurs de parameters.

Fichier ESCCorrespondence.cpp (L.25 à 30,L.30 à 35) méthode vector<ResultCorrespondence> \*ESCCorrespondence::match(model::Sequence \*s1, model::Sequence \*s2) Les variables s1 et s2 ont été inversés dû à l’appel de *swapSequences().* De plus, sans cela, une exception est levé (ne devrait pas) (L.58, L.63)

sur cette ligne: *parametrerVectors(cs2, cs1);*

Même chose avec FSMCorrespondance et MVMCorrespondence.