|  |
| --- |
|  |
| Smith-Waterman local alignment |
| CSCI-291T |

|  |
| --- |
| Mohamad Mahdi Ziaee 109561770  Spring 2016 |

Contents

[How to run the program? 2](#_Toc443379581)

[/Assignment2/src/com/bio/main/MainProcessor.java 3](#_Toc443379582)

[/Assignment2/src/com/bio/main/PerformanceMonitor.java 4](#_Toc443379583)

[/Assignment2/src/com/bio/main/Process.java 5](#_Toc443379584)

[/Assignment2/src/com/bio/main/io/FileProcessor.java 9](#_Toc443379585)

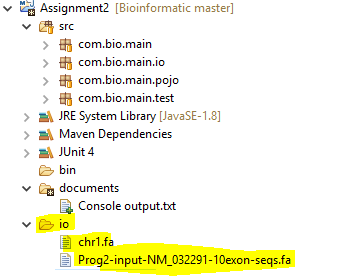
[/Assignment2/src/com/bio/main/pojo/RefSeq.java 12](#_Toc443379586)

[/Assignment2/src/com/bio/main/pojo/SmithWatermanResult.java 13](#_Toc443379587)

[Sample output 14](#_Toc443379588)

# How to run the program?

1. Download JRE from the official website
2. Checkout the code from GitHub account: <https://github.com/momazia/Bioinformatic/tree/master/Assignment2>
3. Create io folder under the main project and put the input files into it:



1. Run the main class:

/Assignment2/src/com/bio/main/MainProcessor.java

# /Assignment2/src/com/bio/main/MainProcessor.java

package com.bio.main;

import com.bio.main.io.FileProcessor;

/\*\*

\* The application runs Smith-Waterman algorithm for given Exon annotations in a

\* chromosome file. Due to the large size of the files, these files are not in

\* repository. Please create the following folder structure for the application

\* to work: <b>/Assignment2/io <b> and include the files needed. The default

\* file names are {@link MainProcessor#EXON\_ANNOT\_FILE\_PATH} and

\* {@link MainProcessor#EXON\_ANNOT\_FILE\_PATH}.

\*

\* @author Mohamad Mahdi Ziaee

\*

\*/

public class MainProcessor {

private static final String EXON\_ANNOT\_FILE\_PATH = FileProcessor.IO\_PATH + "Prog2-input-NM\_032291-10exon-seqs.fa";

private static final String chr\_FILE\_PATH = FileProcessor.IO\_PATH + "chr1.fa";

public static void main(String[] args) {

PerformanceMonitor mainPm = new PerformanceMonitor("Main");

Process.getInstance().run(EXON\_ANNOT\_FILE\_PATH, chr\_FILE\_PATH);

mainPm.end();

}

}

# /Assignment2/src/com/bio/main/PerformanceMonitor.java

package com.bio.main;

/\*\*

\* The class can be used to measure the performance of a process.

\*

\* @author Mohamad Mahdi Ziaee

\*

\*/

public class PerformanceMonitor {

private final long startTimestamp;

private long endTimestamp;

private String processName;

/\*\*

\* Use the default constructor just before the process starts.

\*/

public PerformanceMonitor(String processName) {

this.processName = processName;

System.out.println("Starting [" + processName + "] process...");

startTimestamp = System.currentTimeMillis();

}

/\*\*

\* Invoke this method at the end of the process which also prints the time

\* taken to complete the process.

\*/

public void end() {

endTimestamp = System.currentTimeMillis();

System.out.println(

String.format("Process [%s] ended in %d milliseconds", processName, endTimestamp - startTimestamp));

}

}

# /Assignment2/src/com/bio/main/Process.java

package com.bio.main;

import java.io.IOException;

import java.io.UnsupportedEncodingException;

import java.util.List;

import com.bio.main.io.FileProcessor;

import com.bio.main.pojo.RefSeq;

import com.bio.main.pojo.SmithWatermanResult;

/\*\*

\* The main class in charge of the whole process.

\*

\* @author Mohamad Mahdi Ziaee

\*

\*/

public class Process {

private static Process instance;

// Constants to for scoring purposes.

private static final int SCORE\_INDEL = -1;

private static final int SCORE\_MATCH = 2;

private static final int SCORE\_MISMATCH = -1;

private Process() {

}

public static Process getInstance() {

if (instance == null) {

instance = new Process();

}

return instance;

}

/\*\*

\* The main method to be invoked to print the final result.

\*

\* @param exonAnnotFilePath

\* @param chrFilePath

\*/

public void run(String exonAnnotFilePath, String chrFilePath) {

try {

// Read the db file.

char[] chrArray = extractChromosome(chrFilePath);

// Reading all the exonRefSeqs + strings to look up

List<RefSeq> exonRefSeqs = extractRefSeq(exonAnnotFilePath);

// Looping through each of the Exons and running Smith-Waterman

for (RefSeq refSeq : exonRefSeqs) {

runSmithWaterman(chrArray, refSeq);

}

} catch (IOException e) {

System.out.println("Could not read a file: " + e.getMessage());

}

}

/\*\*

\* Executes SmithWaterman for the given RefSeq on chromosome string and

\* prints the result in console.

\*

\* @param chrArray

\* @param refSeq

\*/

private void runSmithWaterman(char[] chrArray, RefSeq refSeq) {

System.out.println(refSeq.getHeader());

SmithWatermanResult optimumResult = smithWaterman(refSeq.getStr().toCharArray(), chrArray);

System.out.println(optimumResult);

}

/\*\*

\* Returns a list of ReqSeqs by reading the Exon annotation file path given.

\*

\* @param exonAnnotFilePath

\* @return

\* @throws IOException

\*/

private List<RefSeq> extractRefSeq(String exonAnnotFilePath) throws IOException {

return FileProcessor.getInstance().readAnnorationFile(exonAnnotFilePath);

}

/\*\*

\* Extracts the Chromosome file path given and returns its array of

\* characters.

\*

\* @param chrFilePath

\* @return

\* @throws UnsupportedEncodingException

\* @throws IOException

\*/

private char[] extractChromosome(String chrFilePath) throws UnsupportedEncodingException, IOException {

return FileProcessor.getInstance().readChromoseFile(chrFilePath).toCharArray();

}

/\*\*

\* The method runs Smith Waterman algorithm on chr string using the RefSeq's

\* string.

\*

\* @param strArray

\* @param chrArray

\* @return returns the optimum score together with the position of it in

\* both strings in terms of i and j indexes.

\*/

private SmithWatermanResult smithWaterman(char[] strArray, char[] chrArray) {

SmithWatermanResult optimumSmithWatermanResult = new SmithWatermanResult();

int patternSize = strArray.length;

// Creating a vertical Table for the calculation.

int[][] vTable = new int[patternSize][2];

for (int jIndex = 1; jIndex < chrArray.length; jIndex++) {

for (int iIndex = 0; iIndex < patternSize; iIndex++) {

if (iIndex == 0) {

vTable[iIndex][1] = 0; // Initial value for [0][1]

} else {

vTable[iIndex][1] = findMax(vTable, iIndex, chrArray[jIndex], strArray[iIndex]);

}

// Keeping the biggest score if found one.

if (vTable[iIndex][1] > optimumSmithWatermanResult.getScore()) {

optimumSmithWatermanResult.setScore(vTable[iIndex][1]);

optimumSmithWatermanResult.setiIndex(iIndex);

optimumSmithWatermanResult.setjIndex(jIndex);

}

}

// Shifting the result to left column prior to repeating for the

// next charater in chr.

for (int iIndex = 0; iIndex < patternSize; iIndex++) {

vTable[iIndex][0] = vTable[iIndex][1];

}

}

return optimumSmithWatermanResult;

}

/\*\*

\* For a given vTable, it applies Smith-Waterman calculation and finds the

\* maximum score among left, top and diagonal items in the table. The result

\* cannot be lower than zero.

\*

\* @param vTable

\* @param iIndex

\* @param charAtChr

\* @param charAtPattern

\* @return

\*/

public int findMax(int[][] vTable, int iIndex, char charAtChr, char charAtPattern) {

int left = vTable[iIndex][0] + SCORE\_INDEL;

int top = vTable[iIndex - 1][1] + SCORE\_INDEL;

int diagonal = vTable[iIndex - 1][0] + (charAtChr == charAtPattern ? SCORE\_MATCH : SCORE\_MISMATCH);

return Math.max(Math.max(left, top), Math.max(diagonal, 0));

}

}

# /Assignment2/src/com/bio/main/io/FileProcessor.java

package com.bio.main.io;

import java.io.BufferedReader;

import java.io.FileInputStream;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.UnsupportedEncodingException;

import java.nio.file.Files;

import java.nio.file.Paths;

import java.util.ArrayList;

import java.util.Iterator;

import java.util.List;

import org.apache.commons.io.input.BoundedInputStream;

import com.bio.main.pojo.RefSeq;

/\*\*

\* The main class in charge of reading from and writing to a file.

\*

\* @author Mohamad Mahdi Ziaee

\*

\*/

public class FileProcessor {

public static final String IO\_PATH = "../Assignment2/io/";

private static FileProcessor instance;

private static final int chr\_FA\_STARTING\_BYTES = 6; // Includes '\n' at the

// end of the first

// line.

private FileProcessor() {

}

public static FileProcessor getInstance() {

if (instance == null) {

instance = new FileProcessor();

}

return instance;

}

/\*\*

\* Reads the file given and constructs a list of {@link RefSeq} which

\* contains a header and string.

\*

\* @param filePath

\* @param geneId

\* @return

\* @throws IOException

\*/

public List<RefSeq> readAnnorationFile(String filePath) throws IOException {

List<RefSeq> result = new ArrayList<>();

// Reading the whole file line by line

List<String> lines = Files.readAllLines(Paths.get(filePath));

Iterator<String> iterator = lines.iterator();

while (iterator.hasNext()) {

result.add(new RefSeq(iterator.next(), iterator.next()));

}

return result;

}

/\*\*

\* Reads the chromosome file given and puts it into a string. It skips the

\* first 6 bytes containing text ">chr" and replaces it with an empty space.

\*

\* @param chrFilePath

\* @return

\* @throws IOException

\* @throws UnsupportedEncodingException

\*/

public String readChromoseFile(String chrFilePath) throws UnsupportedEncodingException, IOException {

FileInputStream file = new FileInputStream(chrFilePath);

// Since the file given is big, that is why we want to skip the

// unnecessary part of the file instead of reading the whole file.

file.skip(chr\_FA\_STARTING\_BYTES);

try (BufferedReader br = new BufferedReader(new InputStreamReader(new BoundedInputStream(file)))) {

int fileChar;

StringBuilder response = new StringBuilder();

// Adding an empty character to the beginning of the string.

response.append(' ');

while ((fileChar = br.read()) != -1) {

// Making sure to only put the letters in the string. This is to

// exclude '\n' (New lines)

if (Character.isLetter(fileChar)) {

response.append((char) fileChar);

}

}

return response.toString();

}

}

}

# /Assignment2/src/com/bio/main/pojo/RefSeq.java

package com.bio.main.pojo;

/\*\*

\* A POJO to keep RefSeq structure.

\*

\* @author Mohamad Mahdi Ziaee

\*

\*/

public class RefSeq {

/\*\*

\* Header holds the RefSeq annotation.

\*/

private String header;

/\*\*

\* The string to look for.

\*/

private String str;

/\*\*

\* The only constructor of the POJO.

\*

\* @param header

\* @param str

\*/

public RefSeq(String header, String str) {

this.header = header;

this.str = str;

}

public String getHeader() {

return header;

}

/\*\*

\* Returns the string attached to this RefSeq by prefixing it with an empty

\* space first.

\*

\* @return

\*/

public String getStr() {

return ' ' + str;

}

}

# /Assignment2/src/com/bio/main/pojo/SmithWatermanResult.java

package com.bio.main.pojo;

/\*\*

\* The class holds values of the result of running SmithWaterman algorithm. It

\* contains a maximum score and locations of the found score.

\*

\* @author Mohamad Mahdi Ziaee

\*

\*/

public class SmithWatermanResult {

private int iIndex;

private int jIndex;

private int score;

@Override

public String toString() {

return "Score: [" + this.getScore() + "] iIndex: [" + this.getiIndex() + "] jIndex: [" + this.getjIndex() + "]";

}

public int getiIndex() {

return iIndex;

}

public void setiIndex(int iIndex) {

this.iIndex = iIndex;

}

public int getjIndex() {

return jIndex;

}

public void setjIndex(int jIndex) {

this.jIndex = jIndex;

}

public int getScore() {

return score;

}

public void setScore(int score) {

this.score = score;

}

}

# Sample output

Starting [Main] process...

>chr1.66999824.67000051.NM\_032291\_exon\_0\_0\_chr1\_66999825\_f.+

Score: [454] iIndex: [227] jIndex: [67000051]

>chr1.67091529.67091593.NM\_032291\_exon\_1\_0\_chr1\_67091530\_f.+

Score: [128] iIndex: [64] jIndex: [67091593]

>chr1.67098752.67098777.NM\_032291\_exon\_2\_0\_chr1\_67098753\_f.+

Score: [50] iIndex: [25] jIndex: [67098777]

>chr1.67101626.67101698.NM\_032291\_exon\_3\_0\_chr1\_67101627\_f.+

Score: [144] iIndex: [72] jIndex: [67101698]

>chr1.67105459.67105516.NM\_032291\_exon\_4\_0\_chr1\_67105460\_f.+

Score: [114] iIndex: [57] jIndex: [67105516]

>chr1.67108492.67108547.NM\_032291\_exon\_5\_0\_chr1\_67108493\_f.+

Score: [110] iIndex: [55] jIndex: [67108547]

>chr1.67109226.67109402.NM\_032291\_exon\_6\_0\_chr1\_67109227\_f.+

Score: [352] iIndex: [176] jIndex: [67109402]

>chr1.67126195.67126207.NM\_032291\_exon\_7\_0\_chr1\_67126196\_f.+

Score: [24] iIndex: [12] jIndex: [33358004]

>chr1.67133212.67133224.NM\_032291\_exon\_8\_0\_chr1\_67133213\_f.+

Score: [24] iIndex: [12] jIndex: [13840000]

>chr1.67136677.67136702.NM\_032291\_exon\_9\_0\_chr1\_67136678\_f.+

Score: [50] iIndex: [25] jIndex: [67136702]

Process [Main] ended in 915275 milliseconds