

LAPORAN TUGAS KECIL 01

IF2211 STRATEGI ALGORITMA

**“Penyelesaian *Minigame Cyberpunk 2077 Breach Protocol* dengan
Algoritma *Brute Force*”**



Disusun oleh:

Muhammad Althariq Fairuz K-01 13522027

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2024

DAFTAR ISI

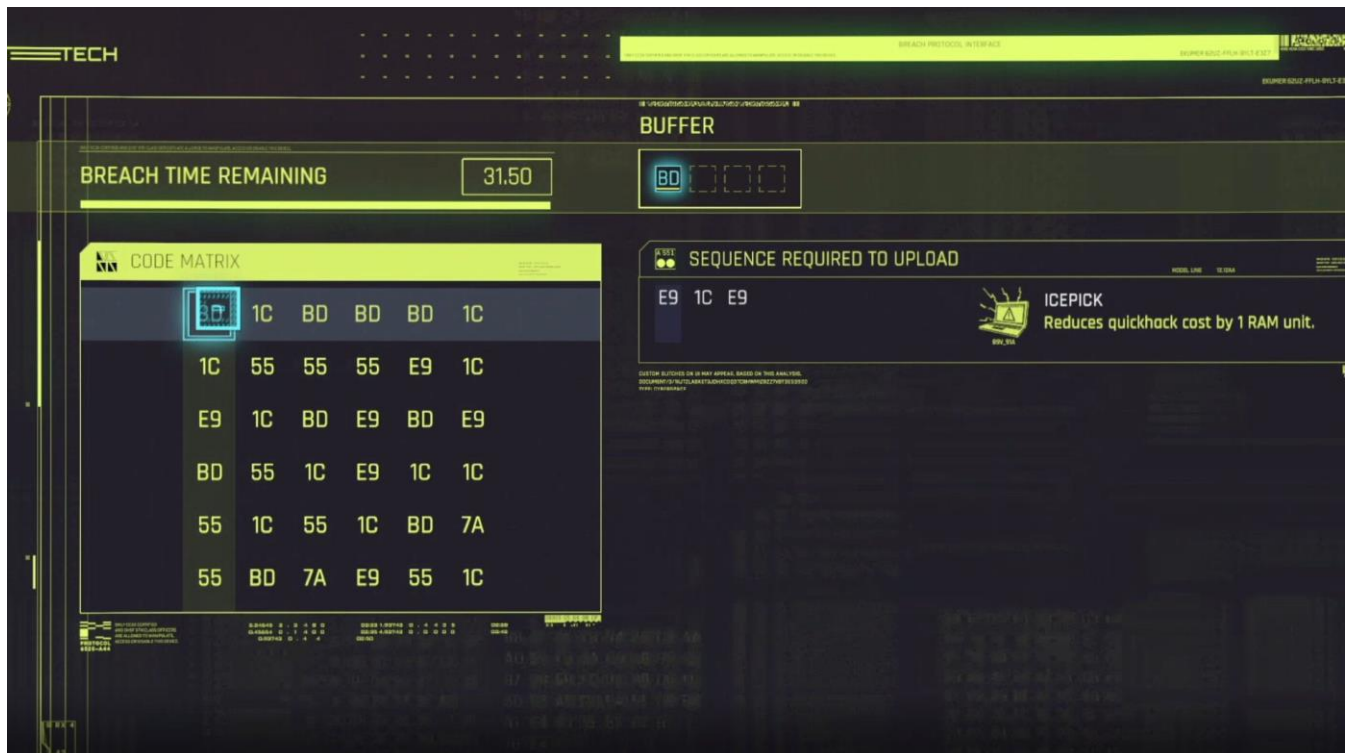
DAFTAR ISI	2
BAB 1	3
BAB 2	4
BAB 3	5
BAB 4	15
BAB 5	16
DAFTAR REFERENSI	25

BAB 1

DESKRIPSI MASALAH

Breach Protocol adalah sebuah *hacking minigame* pada *Videogame Cyberpunk 2077*. Jika pemain bisa menyelesaikan *minigame* ini, pemain akan memperoleh berbagai benefit selama permainan, seperti *in-game currency* atau peningkatan atribut yang bisa digunakan ketika berhadapan dengan musuh.

Saat awal permainan, pemain akan diberikan sebuah matriks dan *sequence* yang berisi token dengan pola tertentu. Pemain diminta untuk mencari pola ini pada matriks yang diberikan dengan hanya bisa bergerak secara horizontal atau vertikal dan setiap elemen pada matriks hanya bisa digunakan satu kali. Permainan selalu dimulai dari baris atas matriks dan pemain diminta untuk memilih salah satu elemen dari sana. Kemudian, pemain akan memilih salah satu elemen lainnya dari kolom yang sama dengan elemen yang dipilih pertama kali. Selanjutnya, pemain akan diminta untuk memilih elemen matriks pada baris yang sama dengan elemen yang dipilih sebelumnya dan seterusnya - selalu bergantian antara kolom dan baris.



Gambar 1.1 Tampilan *Minigame Breach Protocol*

BAB 2

TEORI SINGKAT

2.1 Algoritma Brute Force

Algoritma yang digunakan untuk menyelesaikan permasalahan ini adalah Algoritma *Brute Force*. Program akan mencari semua jenis kombinasi yang setidaknya mengandung salah satu dari semua *sequence*. Kemudian, program akan mencari salah satu kombinasi yang memiliki skor tertinggi.

Langkah-langkah algoritma yang digunakan dalam menyelesaikan permainan tersebut adalah sebagai berikut:

1. Permutasi susunan *buffer*

Program menghasilkan semua kombinasi susunan dari matriks yang diberikan. Kombinasi ini dipastikan memiliki panjang yang sama dengan panjang maksimal *buffer* dan semua kombinasi ini pasti mengandung setidaknya satu *sequence* yang diberikan. Pencarian kombinasi dimulai dari salah satu elemen pada baris pertama matriks. Kemudian, dilanjutkan dengan bergerak secara vertikal dan horizontal secara bergantian tanpa mengunjungi elemen yang sama untuk kedua kalinya. Selain itu, program juga akan mencatat setiap *path* yang telah dilalui suatu kombinasi. Semua kombinasi yang memenuhi akan disimpan dalam *Array of Pair* dengan *Pair* adalah sebuah *class* yang berisi suatu kombinasi yang memenuhi aturan serta *path* yang telah dilaluinya.

2. Solusi

Setelah program menghasilkan seluruh kombinasi yang memenuhi dan menyimpannya dalam *Array of Pair*, program akan melakukan *looping* untuk setiap *Pair* dalam *Array of Pair* dengan semua *sequence* yang diberikan sebelumnya. Skor maksimal akan diinisialisasi dengan 0. Jika suatu *Pair* mengandung lebih dari satu *sequence*, skor untuk *Pair* tersebut akan ditambah sebanyak skor yang dimiliki oleh *sequence* yang bersangkutan. Terakhir, akan dilakukan pengecekan apakah skor untuk *Pair* saat ini lebih besar dari skor maksimal sebelumnya. Jika iya, skor maksimal akan diperbarui dan program akan menyimpan *Pair* saat ini sebagai *Pair* yang memiliki skor tertinggi. Kemudian, program akan mengirimkan skor tertinggi, kombinasi, serta *path* yang dilalui kombinasi tersebut. Namun, jika nilai maksimal tetap 0, program tidak mengembalikan apapun karena tidak ada solusi yang memenuhi.

BAB 3

IMPLEMENTASI PROGRAM

3.1 FOLDER PRIMITIF (src)

a. Main.java

File *Main.java* terdiri dari beberapa *class* :

1. Sequence

Class ini memiliki *member* *tokens* yang berisi kumpulan token dari suatu *sequence* serta skor yang dimilikinya.

A screenshot of a code editor window titled 'Main.java'. The code defines a static class 'Sequence' with two attributes: 'String[] tokens' and 'int score'. It also includes a constructor 'Sequence(String[] tokens, int score)' that initializes these attributes. The code is as follows:

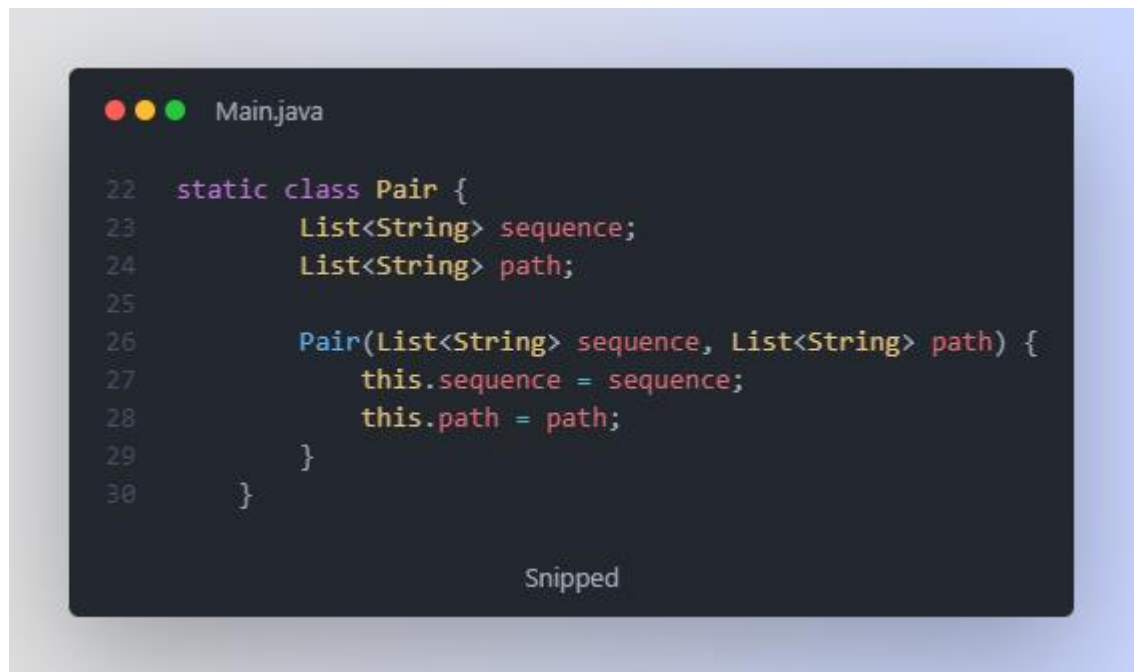
```
12 static class Sequence {
13     String[] tokens;
14     int score;
15
16     Sequence(String[] tokens, int score) {
17         this.tokens = tokens;
18         this.score = score;
19     }
20 }
```

The text 'Snipped' is visible at the bottom of the code block.

Gambar 2.1 *Class Sequence*

2. Pair

Class ini memiliki *member* *sequence* yang berisi kombinasi yang memenuhi aturan serta *path* yang dilaluinya.



Gambar 2.2 *Class Pair*

Selain kedua *class* tersebut, file ini juga memiliki beberapa *method*, diantaranya:

1. Main

Ini adalah fungsi utama, yaitu fungsi yang dijalankan ketika program pertama kali dieksekusi. Fungsi ini berperan dalam mengolah input, baik dari file .txt maupun CLI. Selain itu, fungsi ini juga berperan dalam men-*generate* matriks serta *sequence random* pada input CLI dan juga mencari kombinasi yang memiliki skor tertinggi pada kedua metode input.


```

100 // Brute force all possible combinations
101 long startTime = System.currentTimeMillis(); // Start timer for execution time
102 boolean visited[][] = new boolean[matrix.length][matrix[0].length];
103 List<Pair> result = new ArrayList<Pair>();
104
105 if (bufferLength > matrix[0].length * matrix.length) {
106     bufferLength = matrix[0].length * matrix.length;
107 }
108
109 // For each column in the first row of the matrix, find all possible
110 // combinations
111 for (int i = 0; i < matrix[0].length; i++) {
112     bruteForcePossibleCombination(matrix, bufferLength, sequences, result, new ArrayList<>(), 0, i,
113         true,
114         visited, new ArrayList<>());
115 }
116
117 // Find one combination that has the highest score
118 int maxScore = 0;
119 Pair highestScorePair = null;
120 for (Pair res : result) {
121     int score = 0;
122     for (Sequence sequence : sequences) {
123         if (String.join("", res.sequence).contains(String.join("", sequence.tokens))) {
124             score += sequence.score;
125         }
126     }
127     if (score > maxScore) {
128         maxScore = score;
129         highestScorePair = res;
130     }
131 }
132
133 if (maxScore == 0) {
134     System.out.println("There is no buffer that satisfies the condition");
135 } else {
136     System.out.println("Highest score : " + maxScore);
137     System.out.println("Buffer : ");
138     for (String s : highestScorePair.sequence) {
139         System.out.print(s + " ");
140     }
141     System.out.println("\nPath : ");
142     for (String p : highestScorePair.path) {
143         System.out.println(p);
144     }
145 }
146 long endTime = System.currentTimeMillis();
147 long timeElapsed = endTime - startTime;
148 System.out.println("Time elapsed : " + timeElapsed + " ms\n");
149
150 System.out.println("Do you wish to save the output? (Y/N)");
151 input = new Scanner(System.in);
152 String save = input.nextLine();
153 if (save.toLowerCase().equals("y")) {
154     saveFile(maxScore, highestScorePair, timeElapsed);
155 }
156
157 } catch (FileNotFoundException e) {
158     System.out.println("File doesn't exist : " + e.getMessage());
159 }
160 }

```

Snipped

Gambar 2.4 Main Function (bagian 2)


```

160 else if (choice == 2) {
161
162     System.out.println("Input number of unique tokens : ");
163     Integer numberOfUniqueToken = Integer.parseInt(input.nextLine());
164     while (numberOfUniqueToken < 2) {
165         System.out.println("Number of unique tokens must be at least 2. Please input again :");
166         numberOfUniqueToken = Integer.parseInt(input.nextLine());
167     }
168     System.out.println();
169
170     System.out.println("Input token, each of them must be separated with a space : ");
171     String tokens = input.nextLine();
172     String[] token = tokens.split(" ");
173     while (!isUnique(token) || token.length != numberOfUniqueToken) {
174         System.out.println(
175             "Number of tokens that has been input doesn't equal with the number of unique tokens or the tokens aren't unique. Please input again : ");
176         tokens = input.nextLine();
177         token = tokens.split(" ");
178     }
179     System.out.println();
180
181     System.out.println("Input the length of buffer : ");
182     Integer bufferLength = Integer.parseInt(input.nextLine());
183     while (bufferLength < 2) {
184         System.out.println("Length of buffer must be at least 2");
185         bufferLength = Integer.parseInt(input.nextLine());
186     }
187     System.out.println();
188
189     System.out.println(
190         "Input length of column and row of a matrix, each of them must be separated with a space : ");
191     String matrixSize = input.nextLine();
192
193     while (matrixSize.split(" ").length != 2 || Integer.parseInt(matrixSize.split(" ")[0]) < 1
194         || Integer.parseInt(matrixSize.split(" ")[1]) < 1) {
195         if (matrixSize.split(" ").length != 2) {
196             System.out.println("Input must be two numbers separated by a space");
197         } else {
198             System.out.println("Input must be higher than 0");
199         }
200         matrixSize = input.nextLine();
201     }
202
203     String[] matrixSizeArr = matrixSize.split(" ");
204     Integer matrixCol = Integer.parseInt(matrixSizeArr[0]);
205     Integer matrixRow = Integer.parseInt(matrixSizeArr[1]);
206     System.out.println();
207
208     System.out.println("Input maximum length of a sequence : ");
209     Integer maxSequenceLength = Integer.parseInt(input.nextLine());
210     while (maxSequenceLength < 2) {
211         System.out.println("Maximum length of a sequence must be at least 2");
212         maxSequenceLength = Integer.parseInt(input.nextLine());
213     }
214
215     System.out.println();
216
217     System.out.println("Input number of sequence : ");
218     Integer numberOfSequence = Integer.parseInt(input.nextLine());
219     while (numberOfSequence < 1) {
220         System.out.println("Number of sequence must be at least 1. Please input again :");
221         numberOfSequence = Integer.parseInt(input.nextLine());
222     }
223     System.out.println();
224
225     // Generate matrix
226     String[][] matrix = new String[matrixRow][matrixCol];
227     for (int i = 0; i < matrixRow; i++) {
228         for (int j = 0; j < matrixCol; j++) {
229             matrix[i][j] = token[(int) (Math.random() * numberOfUniqueToken)];
230         }
231     }

```

Snipped

Gambar 2.5 Main Function (bagian 3)

```

233     System.out.println("Here are the generated matrix : ");
234     for (String[] row : matrix) {
235         for (String elm : row) {
236             System.out.print(elm + " ");
237         }
238         System.out.println();
239     }
240     System.out.println();
241
242     // Generate sequences
243     Set<String> uniqueSequences = new HashSet<String>();
244     List<Sequence> sequences = new ArrayList<Sequence>();
245
246     for (int i = 0; i < numberOfSequence; i++) {
247         int sequenceLength = (int) (Math.random() * maxSequenceLength + 1);
248         while (sequenceLength < 2) {
249             sequenceLength = (int) (Math.random() * maxSequenceLength + 1);
250         }
251         String[] sequence = new String[sequenceLength];
252         for (int j = 0; j < sequence.length; j++) {
253             sequence[j] = token[(int) (Math.random() * numberOfUniqueToken)];
254         }
255
256         int score = (int) (Math.random() * 100);
257
258         // Convert sequence to string
259         String sequenceStr = Arrays.toString(sequence);
260
261         // Cek whether sequence is already in set
262         while (uniqueSequences.contains(sequenceStr)) {
263             sequence = new String[sequenceLength];
264             for (int j = 0; j < sequence.length; j++) {
265                 sequence[j] = token[(int) (Math.random() * numberOfUniqueToken)];
266             }
267             // Update sequence string, convert sequence to string ([a, b, c] -> "[a,b,c]")
268             sequenceStr = Arrays.toString(sequence);
269         }
270
271         // Add sequence to set unique so that there is no duplicate sequence
272         uniqueSequences.add(sequenceStr);
273
274         // Add sequence to list
275         sequences.add(new Sequence(sequence, score));
276     }
277
278     System.out.println(
279         "Here are the generated sequences and their score : ");
280     for (Sequence sequence : sequences) {
281         String[] sequenceToken = sequence.tokens;
282         for (int i = 0; i < sequenceToken.length; i++) {
283             System.out.print(sequenceToken[i] + " ");
284         }
285         System.out.println();
286         System.out.println(sequence.score + "\n\n");
287     }
288

```

Snipped

Gambar 2.6 Main Function (bagian 4)

```

Main.java

289         // Brute force all combinatio
290         long startTime = System.currentTimeMillis();
291         boolean visited[][] = new boolean[matrix.length][matrix[0].length];
292         List<Pair> result = new ArrayList<Pair>();
293
294         if (bufferLength > matrix[0].length * matrix.length) {
295             bufferLength = matrix[0].length * matrix.length;
296         }
297
298         for (int i = 0; i < matrix[0].length; i++) {
299             bruteForcePossibleCombination(matrix, bufferLength, sequences, result, new ArrayList<String>(), 0,
300                 i,
301                 true,
302                 visited, new ArrayList<String>());
303         }
304
305         // Find one combination that has the highest score
306         int maxScore = 0;
307         Pair highestScorePair = null;
308         for (Pair res : result) {
309             int score = 0;
310             for (Sequence sequence : sequences) {
311                 if (String.join("", res.sequence).contains(String.join("", sequence.tokens))) {
312                     score += sequence.score;
313                 }
314             }
315             if (score > maxScore) {
316                 maxScore = score;
317                 highestScorePair = res;
318             }
319         }
320
321         if (maxScore == 0) {
322             System.out.println("There is no buffer that satisfies the condition");
323         } else {
324             System.out.println("Highest score : " + maxScore);
325             System.out.println("Buffer : ");
326             for (String s : highestScorePair.sequence) {
327                 System.out.print(s + " ");
328             }
329             System.out.println("\nPath : ");
330             for (String p : highestScorePair.path) {
331                 System.out.println(p);
332             }
333         }
334         long endTime = System.currentTimeMillis();
335         long timeElapsed = endTime - startTime;
336         System.out.println("Time elapsed : " + timeElapsed + " ms\n");
337
338         System.out.println("Do you wish to save the output? (Y/N)");
339         input = new Scanner(System.in);
340         String save = input.nextLine();
341         if (save.toLowerCase().equals("y")) {
342             saveFile(maxScore, highestScorePair, timeElapsed);
343         }
344     }
345     System.out.println("\nPlease choose one input method : ");
346     System.out.println("1. Via .txt");
347     System.out.println("2. Via CLI");
348     System.out.println("3. Exit");
349     choice = Integer.parseInt(input.nextLine());
350     while (choice != 1 && choice != 2 && choice != 3) {
351         System.out.println("Please choose only 1 or 2 or 3 : ");
352         choice = Integer.parseInt(input.nextLine());
353     }
354 }
355 input.close();
356 }

```

Snipped

Gambar 2.7 Main Function (bagian 5)

2. bruteForcePossibleCombination

Ini adalah *method* yang digunakan untuk men-*generate* semua kombinasi yang memenuhi aturan dari permainan, yaitu semua kombinasi yang bisa dicapai hanya dengan bergerak secara vertikal tau horizontal, tidak mengunjungi elemen matriks lebih dari satu kali, dan mengandung setidaknya satu *sequence*.

```
358 static void bruteForcePossibleCombination(String matrix[][], int bufferLength, List<Sequence> sequences,
359 List<Pair> result, List<String> temp, int row, int col, boolean isHorizontal, boolean visited[][],
360 List<String> tempPath) {
361
362     // Find all valid coordinates (Not out of bound and haven't been visited yet)
363     if (row < 0 || row >= matrix.length || col < 0 || col >= matrix[0].length || visited[row][col]) {
364         return;
365     }
366
367     // If valid, add token to temporary sequence (temp) and mark the coordinate as
368     // visited and add the path to temporary path (tempPath)
369     temp.add(matrix[row][col]);
370     visited[row][col] = true;
371     tempPath.add(Integer.toString(col + 1) + " " + Integer.toString(row + 1));
372
373     // If the length of sequence (temp) is less than the buffer length, find all
374     // possible combinations recursively based on the direction until the length of
375     // sequence (temp) is equal to the buffer length
376     if (temp.size() < bufferLength) {
377         isHorizontal = !isHorizontal;
378         if (isHorizontal) {
379             for (int j = 0; j < matrix[0].length; j++) {
380                 if (j != col && !visited[row][j]) {
381                     bruteForcePossibleCombination(matrix, bufferLength, sequences, result, temp, row, j,
382 isHorizontal,
383 visited, tempPath);
384                 }
385             }
386         } else {
387             for (int i = 0; i < matrix.length; i++) {
388                 if (i != row && !visited[i][col]) {
389                     bruteForcePossibleCombination(matrix, bufferLength, sequences, result, temp, i, col,
390 isHorizontal,
391 visited, tempPath);
392                 }
393             }
394         }
395     }
```

Snipped

Gambar 2.8 bruteForcePossibleCombination (bagian 1)

Main.java

```
396 // If the length of sequence (temp) is equal to the buffer length, check whether
397 // the sequence contains any of the sequences
398 else if (temp.size() == bufferLength) {
399     String tempString = String.join("", temp);
400     for (Sequence sequence : sequences) {
401         if (tempString.contains(String.join("", sequence.tokens))) {
402             // If it contains, add the sequence and the path to the result
403             result.add(new Pair(new ArrayList<String>(temp), new ArrayList<String>(tempPath)));
404             break;
405         }
406     }
407 }
408
409 // After the sequence has been checked, remove the last token from the sequence
410 // and mark the coordinate as unvisited and remove the last path from the path
411 visited[row][col] = false;
412 temp.remove(temp.size() - 1);
413 tempPath.remove(tempPath.size() - 1);
414 }
```

Snipped

Gambar 2.9 bruteForcePossibleCombination (bagian 2)

3. saveFile

Ini adalah *method* yang digunakan untuk menyimpan hasil *output* dalam format .txt.

Main.java

```
416 static void saveFile(int maxScore, Pair highestScore, long timeElapsed) {
417     System.out.println("Input file name: ");
418     String saveLocation = input.nextLine();
419     File saveFile = new File("../test/" + saveLocation + ".txt");
420     while (saveFile.exists()) {
421         System.out.println("File already exist, please use another name : ");
422         saveLocation = input.nextLine();
423         saveFile = new File("../test/" + saveLocation + ".txt");
424     }
```

Snipped

Gambar 2.10 saveFile (bagian 1)

```

Main.java

425 try {
426     saveFile.createNewFile();
427     FileWriter writer = new FileWriter(saveFile);
428     writer.write(maxScore + "\n");
429     if (highestScore == null) {
430         writer.write("There is no buffer that satisfies the condition\n");
431     } else {
432         for (String s : highestScore.sequence) {
433             writer.write(s + " ");
434         }
435         for (String p : highestScore.path) {
436             writer.write(p + "\n");
437         }
438     }
439     writer.write("\n" + timeElapsed + " ms");
440     writer.close();
441     System.out.println("File has been saved : " + saveFile.getName());
442
443 } catch (Exception e) {
444     System.out.println("An error occurred : " + e.getMessage());
445     e.printStackTrace();
446 }
447 }

```

Snipped

Gambar 2.11 saveFile (bagian 2)

4. Unique

Ini adalah *method* yang digunakan untuk mengecek apakah token yang di input pada CLI sudah unik atau belum.

```

Main.java

449     static boolean isUnique(String[] token) {
450         // Check whether a token is unique or not
451         Set<String> uniqueTokens = new HashSet<String>(Arrays.asList(token));
452         return uniqueTokens.size() == token.length;
453     }

```

Snipped

Gambar 2.12 isUnique

BAB 4

EKSPERIMEN

Initialize Program

```

/$$$$$/ $$/ /$$$$$/ $$$$$$/ $$$$$$/ $$$$$$/ $$/ /$$$$$/ $$$$$$/ $$$$$$/ $$$$$$/
/$_ $| $$ /$$ $_ $| $$ _____ $$_ $| $$$ $| $| $$$ /$$/
$$ \ $ $$$ /$$ \ $| $$$ $$$ \ $ $$$ \ $| $$$ $| $$$$| $| $$$ /$$/
| $$$ \ $$$/ $$$$$$| $$$$$$| $$$$$$| $$$$$$| $$$ $| $$$ $$$ $| $$$$$/
| $$$ \ $$$/ $$$ $| $$$/ $$$ $| $$$/ $$$ $| $$$ $$$ $$$ $$$ /$$/
| $$$ $$$ | $$$ $$$ \ $| $$$ $$$ \ $| $$$ $$$ | $$$ \ $$$ \ $$$ /$$/
| $$$$$$/ | $$$ $$$$$$| $$$$$$| $$$ $| $$$ | $$$$$| $$$ \ $| $$$ \ $$$ /$$/
\___/ \___/ \___/ \___/ \___/ \___/ \___/ \___/ \___/ \___/ \___/

/$$$$$/ $$$$$$/ $$$$$$/ $$$$$$/ $$$$$$/ $$/ /$$$$$/ $$$$$$/ $$$$$$/ $$$$$$/
| $$$ $| $$$ $_ $| $$$ /$_ $$$/$_ $| $$$ | $$$ | $$$ \ $| $$$ \ $| $$$ \ $$$
| $$$ \ $| $$$ \ $| $$$ $$$ \ $| $$$ \ $$$ | $$$ | $$$ | $$$ \ $| $$$ \ $$$
| $$$$$$| $$$$$$| $$$$$$| $$$$$$| $$$ | $$$ | $$$ $$$ | $$$ | $$$ | $$$
| $$$ $| $$$ $| $$$/ $$$ $| $$$ | $$$ | $$$ | $$$ | $$$ | $$$ | $$$
| $$$ \ $$$ \ $| $$$ $$$ | $$$ | $$$ $| $$$ | $$$ | $$$ | $$$ | $$$
| $$$$$$| $$$ | $| $$$$$$| $$$ | $| $$$$$$| $$$ | $$$ | $$$ | $$$
\___/ \___/ \___/ \___/ \___/ \___/ \___/ \___/ \___/ \___/ \___/

Please choose one input method :
1. Via .txt
2. Via CLI
3. Exit
█

```

Gambar 3.1 Tampilan awal program

Input dari File .txt (Opsi 1)

```
Please choose one input method :
1. Via .txt
2. Via CLI
3. Exit
1
Input filename :
soal1
Highest score : 50
Sequence :
7A BD 7A BD 1C BD 55
Path :
1 1
1 4
3 4
3 5
6 5
6 3
1 3
Time elapsed : 83 ms

Do you wish to save the output? (Y/N)
N
```

Gambar 3.2 Output dari Soal1.txt


```
Please choose one input method :
1. Via .txt
2. Via CLI
3. Exit
1
Input filename :
soal2
Highest score : 10
Sequence :
1C E9 55 BD 7A E9 7A BD 55 55 7A BD E9 E9 1C 1C 7A 55 1C
Path :
2 1
2 10
1 10
1 2
2 2
2 9
1 9
1 3
2 3
2 8
1 8
1 4
2 4
2 7
1 7
1 5
2 5
2 6
1 6
Time elapsed : 553 ms

Do you wish to save the output? (Y/N)
n
```

Gambar 3.3 Output dari Soal2.txt

```
Please choose one input method :
1. Via .txt
2. Via CLI
3. Exit
1
Input filename :
soal3
Highest score : 10
Sequence :
E9 BD E9 55 7A 55 7A 1C BD E9 55 E9 55 7A 1C 7A 1C BD E9 BD
Path :
10 1
10 2
9 2
9 1
8 1
8 2
7 2
7 1
1 1
1 2
5 2
5 1
4 1
4 2
3 2
3 1
2 1
2 2
6 2
6 1
Time elapsed : 2119 ms

Do you wish to save the output? (Y/N)
y
Input file name:
Jawaban3
File has been saved : Jawaban3.txt
```

Gambar 3.4 Output dari Soal3.txt

```
Please choose one input method :
1. Via .txt
2. Via CLI
3. Exit
1
Input filename :
soal4
Highest score : 92
Sequence :
1G FD BC 5A FD 8E 8E
Path :
2 1
2 4
6 4
6 1
3 1
3 2
1 2
Time elapsed : 98 ms

Do you wish to save the output? (Y/N)
y
Input file name:
Jawaban4
File has been saved : Jawaban4.txt
```

Gambar 3.5 Output dari Soal4.txt

Input dari CLI (Opsi 2)

```
Please choose one input method :
1. Via .txt
2. Via CLI
3. Exit
2
Input number of unique tokens :
5

Input token, each of them must be separated with a space :
A1 B2 C3 D4 E5

Input the length of buffer :
6

Input length of column and row of a matrix, each of them must be separated with a space :
5 8

Input maximum length of a sequence :
5

Input number of sequence :
5

Here are the generated matrix :
E5 C3 B2 A1 E5
B2 E5 E5 D4 C3
B2 B2 D4 E5 E5
B2 D4 E5 B2 A1
C3 B2 B2 E5 B2
C3 D4 C3 C3 C3
C3 E5 A1 D4 C3
D4 E5 E5 D4 E5

Here are the generated sequences and their score :
C3 B2 A1
22

E5 B2
18
```

Gambar 3.6 Input dari input melalui CLI (bagian 1)

```
C3 C3  
53
```

```
D4 C3 D4  
68
```

```
B2 A1 E5  
9
```

```
Highest score : 121
```

```
Buffer :
```

```
E5 C3 C3 D4 C3 D4
```

```
Path :
```

```
1 1
```

```
1 6
```

```
4 6
```

```
4 7
```

```
1 7
```

```
1 8
```

```
Time elapsed : 80 ms
```

```
Do you wish to save the output? (Y/N)
```

```
Y
```

```
Input file name:
```

```
Jawaban5
```

```
File has been saved : Jawaban5.txt
```

Gambar 3.7 Output dari input melalui CLI (bagian 2)
Output disimpan pada Jawaban5.txt

```
Please choose one input method :
1. Via .txt
2. Via CLI
3. Exit
2
Input number of unique tokens :
3

Input token, each of them must be separated with a space :
AA BB CC

Input the length of buffer :
7

Input length of column and row of a matrix, each of them must be separated with a space :
8 9

Input maximum length of a sequence :
8

Input number of sequence :
12

Here are the generated matrix :
BB BB CC BB BB CC BB BB
AA CC BB AA CC BB BB CC
AA CC BB CC AA CC CC AA
BB CC AA BB CC CC AA CC
AA CC AA AA AA CC BB CC
CC CC BB BB BB CC BB BB
BB CC AA BB BB BB CC AA
CC AA CC CC CC CC AA AA
CC AA AA AA CC CC BB CC
```

Gambar 3.8 Input dari CLI (bagian 1)

Here are the generated sequences and their score :

CC AA CC BB
73

CC AA BB CC CC CC AA
69

CC AA
40

CC BB AA CC
1

AA CC CC CC AA CC
90

AA CC AA BB CC AA
57

BB BB BB CC AA BB BB AA
80

BB AA AA BB
66

AA CC
99

AA BB BB CC
25

Gambar 3.9 Input dari CLI (bagian 2)

```
BB CC CC CC AA CC BB CC
1

BB BB BB AA CC BB BB BB
51

Highest score : 278
Buffer :
CC AA CC BB AA AA BB
Path :
3 1
3 4
5 4
5 7
3 7
3 5
7 5
Time elapsed : 3589 ms

Do you wish to save the output? (Y/N)
Y
Input file name:
Jawaban6
File has been saved : Jawaban6.txt
```

Gambar 3.10 Output dari input CLI (bagian 3)
Output disimpan pada Jawaban6.txt

BAB 5

PENUTUP

5.1 Kesimpulan

Melalui tugas besar ini, saya menjadi belajar banyak hal terkait library dan bahasa pemrograman Java. Algoritma *brute force* dapat menyelesaikan hampir segala macam persoalan algoritma, namun tidak efisien dan memakan banyak memori.

5.2 Link Repository

Link repository untuk tugas kecil 1 mata kuliah IF2211 Strategi Algoritma adalah sebagai berikut

Link : [AlthariqFairuz/Tucil_13522027 \(github.com\)](https://github.com/AlthariqFairuz/Tucil_13522027)

5.3 Tabel Checkpoint Program

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa kesalahan	✓	
2. Program berhasil dijalankan	✓	
3. Program dapat membacamasukan berkas .txt	✓	
4. Program dapat menghasilkan masukan secara acak	✓	
5. Solusi yang diberikan program optimal	✓	
6. Program dapat menyimpan solusi dalam berkas .txt	✓	
7. Program memiliki GUI		✓

DAFTAR REFERENSI

[Java ArrayList \(w3schools.com\)](https://www.w3schools.com/java/)

[Set in Java - GeeksforGeeks](https://www.geeksforgeeks.org/set-in-java/)

[Algoritma-Brute-Force-\(2022\)-Bag1.pdf \(itb.ac.id\)](https://www.itb.ac.id/~bag1/Algoritma-Brute-Force-(2022)-Bag1.pdf)