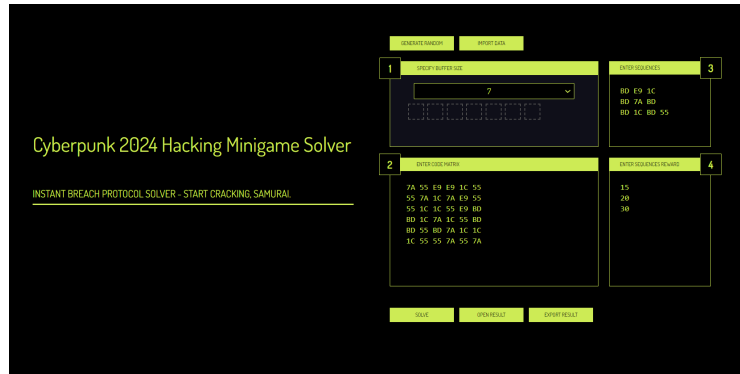


# Laporan Tugas Kecil 1 IF2211 Strategi Algoritma

## Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force



### Dosen Pengampu:

Ir. Rila Mandala, M.Eng., Ph.D.

Monterico Adrian, S.T., M.T.

### Disusun Oleh:

Ahmad Rafi Maliki 1322137

**Teknik Informatika**

**Sekolah Teknik Elektro dan Informatika**

**Institut Teknologi Bandung**

**2024**

## 1. Tentang Program

### 1.1. Deskripsi

Program ini adalah *solver* untuk “CyberPunk 2077 Breach Protocol” yang menggunakan algoritma *brute force* untuk menelusuri seluruh kemungkinan *buffer* dan menemukan *buffer* mana yang paling efisien untuk memperoleh skor tertinggi.

Program merupakan program berbasis web (*non-responsive design*) yang di deploy menggunakan Vercel. Program ini ditulis menggunakan JavaScript, HTML, dan CSS. Program dibuat menggunakan *framework* React dan Tailwind dengan *build tool* Vite.

Segala algoritma yang berkaitan tentang komputasi algoritma yang berkaitan dengan komputasi algoritma ditulis langsung menggunakan JavaScript sebagai bagian dari komponen react.

Desain *User Interface* dari program ini mengadaptasi dari <https://cyberpunk-hacker.com/> dengan menggunakan skema warna yang sama pula.

Program ini buruk dalam menangani ukuran matriks dan buffer yang besar secara sekaligus karena akan berjalan sangat lambat. Namun, jika hanya matriks besar atau buffer besar, kecepatan program ini cukup baik.

### 1.2. Algoritma

#### 1.2.1. Algoritma Searching

Sederhananya, algoritma *brute force* yang digunakan pada program ini akan mencari seluruh kemungkinan *buffer* lalu mencari mana *buffer* yang memiliki skor tertinggi.

Algoritma ini melakukan *looping* secara rekursif dan *backtracking* untuk menelusuri semua kemungkinan *buffer* dari matriks sekuens, pada akar rekursi program akan memilih tiap-tiap elemen dari baris ke-0 pada matrix lalu dilanjutkan dengan pemanggilan rekursi untuk menelusuri sekuens berikutnya.

Pada setiap *looping*, sekuens yang ditelusuri arahnya akan berubah ubah yaitu vertikal dan horizontal (sesuai aturan permainan). Pada *looping* kedua yaitu setelah pemilihan elemen akar, program akan melakukan *looping* lagi untuk menelusuri elemen pada kolom yang sama (penelusuran secara vertikal) yang

dilanjutkan mencari elemen pada baris yang sama (penelusuran secara horizontal) pada *looping* ketiga. Dengan pola tersebut, program akan menelusuri secara horizontal dan vertikal sampai buffer penuh. Elemen yang sudah ditelusuri di tiap sekuens rekursinya ditandai oleh matriks bernilai boolean yang tiap elemennya mewakili tiap elemen pada matriks sekuens.

Setelah *buffer* penuh, sekuens yang baru saja dilewati akan dihitung skor nya. Jika skor melebihi skor terbesar yang sudah ditemukan, sekuens tersebut akan dimasukkan ke variabel berupa array yang menampung semua sekuens yang telah ditemukan dengan mengosongkan variabel tersebut terlebih dulu. Jika skor nya sama dengan skor terbesar yang sudah ditemukan, sekuens tersebut akan langsung dimasukkan ke variabel penampung tersebut. Hal ini dilakukan agar variabel tersebut hanya menampung sekuens dengan skor terbesar.

Setelah *looping* rekursif selesai dan semua sekuens dengan skor terbesar ditemukan, program akan mencari sekuens mana yang paling efisien (paling pendek) dengan cara menghilangkan elemen terakhir pada sekuens. Sekuens yang paling efisien adalah sekuens yang paling pendek atau paling banyak dihilangkan elemen terakhirnya namun tetap memiliki skor seperti semula atau maksimal.

Semua sekuens yang tersisa setelah program ini selesai berjalan adalah variasi dari sekuens paling efisien dalam pemecahan permainan ini. Namun, hanya satu sekuens yang akan ditampilkan ke user.

### **1.2.2. Algoritma Generasi Random**

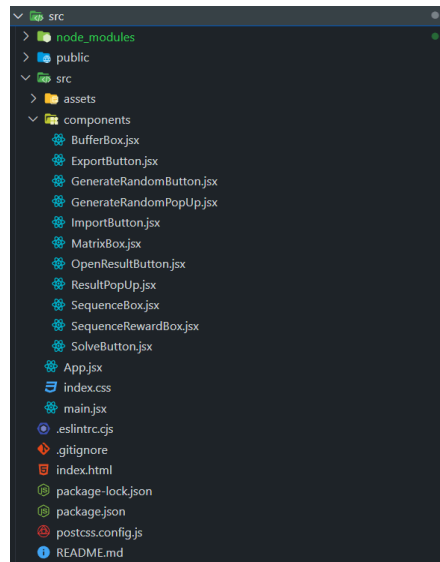
Program ini menggunakan fungsi `random()` pada library `Math` bawaan JavaScript untuk menggenerasi data random yang diperlukan berdasarkan data yang diinput oleh user.

### **1.2.3. Algoritma Import dan Export Data**

Program ini memanfaatkan fitur *native* JavaScript dan HTML untuk mengunggah dan mengunduh data dari *local library user*. Data yang di-*passing* bertipe string dalam file yang berekstensi “.txt”. Program akan melakukan *parsing* data tersebut dari tipe data string ke tipe data objek yang sesuai saat pengunggahan *input* ataupun sebaliknya saat pengunduhan *output*.

### 1.3. Source Code

Berikut merupakan struktur folder React App yang ditulis menggunakan JavaScript menggunakan ekstensi file .jsx. Root program berada pada file index.html.



**Gambar 1**  
Struktur folder

Seluruh gambar yang ditampilkan berikutnya merupakan source code yang berkaitan dengan algoritma *brute force* pada file SolveButton.jsx.

```
1 import React from "react";
2
3 const SolveButton = ({ gameData, setOutput, setShowResult, setInputError }) => {
4   const handleClick = () => {
5     // console.log(gameData);
6
7     const removeEmptyStrings = (arr) => arr.filter((item) => item !== "");
8
9     var buffer_size = gameData.buffer_size;
10    var matrix_width = gameData.matrix_width;
11    var matrix_height = gameData.matrix_height;
12
13    var matrix = gameData.matrix.map((element) => removeEmptyStrings(element));
14    var sequences = gameData.sequences.map((element) =>
15      removeEmptyStrings(element)
16    );
17    var number_of_sequences = gameData.number_of_sequences;
18    var sequences_reward = [];
19    for (let i = 0; i < gameData.sequences_reward.length; i++) {
20      sequences_reward.push(parseInt(gameData.sequences_reward[i]));
21    }
22
23    let max_score = gameData.max_score;
24    var max_score_found = 0;
25
26    var sequences_list = [];
27    var sequences_idx_list = [];
28    var score_list = [];
29
30    var output = [];
31    var start_time;
```

**Gambar 2**  
Deklarasi variabel

```
33  /* Check if matrix is valid */
34  for (let i = 0; i < matrix_height; i++) {
35    if (matrix[i].length !== matrix_width) {
36      console.log("Matrix error!");
37      setInputError(true);
38      return;
39    }
40  }
41
42  // console.log(sequences.length);
43  // console.log(sequences_reward.length);
44
45  /* Check if sequences and sequences reward match */
46  if (sequences.length !== sequences_reward.length) {
47    console.log("Sequences and sequences reward does not match!");
48    setInputError(true);
49    return;
50  }
51
52  setInputError(false);
```

**Gambar 3**  
Validasi data pada variabel

```
54  /* Check if list1 is sublist of list2 function */
55  function IS_SUB_LIST(l1, l2) {
56    const len_l1 = l1.length;
57    const len_l2 = l2.length;
58
59    if (len_l1 > len_l2) {
60      return false;
61    }
62
63    for (let i = 0; i < len_l2 - len_l1 + 1; i++) {
64      if (l2.slice(i, i + len_l1).join("") === l1.join("")) {
65        return true;
66      }
67    }
68    return false;
69  }
70
```

**Gambar 4**  
Fungsi pembantu IS\_SUB\_LIST untuk pengecekan sekuens

```
71  /* Score calculation function */
72  function CALCULATE_SCORE(sequence) {
73    let score = 0;
74    for (let i = 0; i < number_of_sequences; i++) {
75      if (IS_SUB_LIST(sequences[i], sequence)) {
76        score += sequences_reward[i];
77      }
78    }
79    return score;
80  }
81
```

**Gambar 5**  
Fungsi pembantu CALCULATE\_SCORE untuk menghitung skor sekuens

```

108      /* Recursive LOOP function */
109      function RECURSION_LOOP(
110          sequence,
111          sequence_idx,
112          depth,
113          prev_row,
114          prev_col,
115          direction,
116          mark
117      ) {
118          /* Base case */
119          if (depth === buffer_size) {
120              const score = CALCULATE_SCORE(sequence);
121
122              /* Append data to global arrays */
123              if (score >= max_score_found) {
124                  if (
125                      !sequences_list.some(
126                          (arr) => JSON.stringify(arr) === JSON.stringify(sequence)
127                      )
128                  ) {
129                      if (score > max_score_found) {
130                          max_score_found = score;
131                          sequences_list.splice(0);
132                          sequences_idx_list.splice(0);
133                          score_list.splice(0);
134                      }
135
136                      sequences_list.push(sequence);
137                      sequences_idx_list.push(sequence_idx);
138                      score_list.push(score);
139                  }
140              }
141          }

```

**Gambar 5**

Fungsi RECURSION\_LOOP (1) basis rekursi sekuens sudah sepanjang buffer

```

142      /* Recursive case */
143      } else if (direction === 0) {
144          /* Vertical checking */
145
146          /* Check above */
147          for (let i = prev_row - 1; i >= 0; i--) {
148              if (!mark[i][prev_col]) {
149                  LOOP(matrix, sequence, sequence_idx, depth, i, prev_col, 1, mark);
150              }
151          }
152          /* Check below */
153          for (let i = prev_row + 1; i < matrix_height; i++) {
154              if (!mark[i][prev_col]) {
155                  LOOP(matrix, sequence, sequence_idx, depth, i, prev_col, 1, mark);
156              }
157          }
158      } else {
159          /* Horizontal checking */
160
161          /* Check left */
162          for (let i = prev_col - 1; i >= 0; i--) {
163              if (!mark[prev_row][i]) {
164                  LOOP(matrix, sequence, sequence_idx, depth, prev_row, i, 0, mark);
165              }
166          }
167          /* Check right */
168          for (let i = prev_col + 1; i < matrix_width; i++) {
169              if (!mark[prev_row][i]) {
170                  LOOP(matrix, sequence, sequence_idx, depth, prev_row, i, 0, mark);
171              }
172          }
173      }
174  }
175  }

```

**Gambar 6**

Fungsi RECURSION\_LOOP (2) kasus rekursi secara vertikal dan horizontal

```

197  /* Master function */
198  function SEARCH_SEQUENCE() {
199    /* Initial LOOP from every element at row 0 */
200    for (let i = 0; i < matrix_width; i++) {
201      /* Initialize marker matrix for keeping track of visited elements */
202      const mark_matrix = Array.from({ length: matrix_height }, () =>
203        Array(matrix_width).fill(false)
204      );
205      mark_matrix[0][i] = true;
206
207      /* Initialize sequence and sequence index matrix*/
208      const sequence = [matrix[0][i]];
209      const sequence_idx = [[0, i]];
210      const prev_row = 0;
211      const prev_col = i;
212      RECURSION_LOOP(
213        sequence.slice(),
214        sequence_idx.slice(),
215        1,
216        prev_row,
217        prev_col,
218        0,
219        mark_matrix.slice()
220      );
221    }
222  }

```

**Gambar 7**

Fungsi SEARCH\_SEQUENCE (1) untuk memulai pemanggilan fungsi RECURSION\_LOOP

```

203  if (max_score_found > 0) {
204    // console.log("Score: ${max_score_found}");
205    // console.log("Sequence:", sequences_list[max_score_idx].join(" "));
206
207    var making_efficient = true;
208
209    var new_sequences_list = JSON.parse(JSON.stringify(sequences_list));
210    var new_sequences_idx_list = JSON.parse(
211      JSON.stringify(sequences_idx_list)
212    );
213
214    var new_score = JSON.parse(JSON.stringify(score_list));
215    var temp_score = JSON.parse(JSON.stringify(score_list));
216
217    /* While loop for making sequence shorter */
218    while (making_efficient) {
219      console.log("Making efficient...");
220      temp_score = JSON.parse(JSON.stringify(new_score));
221      new_score = [];
222
223      let temp_sequences_list = JSON.parse(
224        JSON.stringify(new_sequences_list)
225      );
226      let temp_sequences_idx_list = JSON.parse(
227        JSON.stringify(new_sequences_idx_list)
228      );
229      for (let i = 0; i < new_sequences_list.length; i++) {
230        new_sequences_list[i].pop();
231        new_sequences_idx_list[i].pop();
232        // console.log(new_sequences_list[i]);
233        // console.log(CALCULATE_SCORE(new_sequences_list[i]));
234        new_score.push(CALCULATE_SCORE(new_sequences_list[i]));
235      }
236      const max_score = Math.max(...new_score);
237      if (max_score < max_score_found) {
238        making_efficient = false;
239        sequences_list = temp_sequences_list;
240        sequences_idx_list = temp_sequences_idx_list;
241        score_list = temp_score;
242      }
243    }
244  }

```

**Gambar 8**

Fungsi SEARCH\_SEQUENCE (2) mencari sekuens yang paling efisien

```

const max_score_idx = score_list.indexOf(max_score_found);
var new_idx_list = sequences_idx_list[max_score_idx].map((idx) => [
  idx[1] + 1,
  idx[0] + 1,
]);

// console.log(sequences_list);

/* Output Array Handler */
output.push(`${max_score_found}`);
output.push(sequences_list[max_score_idx].join(" "));
new_idx_list = new_idx_list.map((idx) => {
  output.push(idx.join(", "));
});
let end_time = Date.now() - start_time;
output.push(`${end_time} ms`);
setOutput(output);

const scrollTop = document.documentElement.scrollTop;
document.body.dataset.scrollY = scrollTop;
document.body.style.position = "fixed";
document.body.style.top = `-${scrollTop}px`;
setShowResult(true);
} else {
  /* Output Array Handler */
  let end_time = Date.now() - start_time;
  output.push(`${end_time} ms`);
  output.push("No solution");
  setOutput(output);
  const scrollTop = document.documentElement.scrollTop;
  document.body.dataset.scrollY = scrollTop;
  document.body.style.position = "fixed";
  document.body.style.top = `-${scrollTop}px`;
  setShowResult(true);
  // console.log("No sequence found");
}
}

```

**Gambar 9**

Fungsi SEARCH\_SEQUENCE (3) memasukkan sekuens paling efisien ke variabel output

#### 1.4. Repository

Keseluruhan program beserta dokumentasi dapat diakses pada [https://github.com/rafimaliki/Tucil1\\_13522137](https://github.com/rafimaliki/Tucil1_13522137)



## 2. Cara Menjalankan Program

Program dapat dijalankan dengan mengakses <https://cryberpunk2024.vercel.app/> atau dengan meng-*clone* repository [https://github.com/rafimaliki/Tucil1\\_13522137](https://github.com/rafimaliki/Tucil1_13522137) untuk menjalankan program di localhost.

Jika ingin menjalankan program pada localhost, pastikan [node.js](#) sudah terinstal pada device anda. Jika sudah, langkah selanjutnya sebagai berikut:

- 1) Clone dari repository dengan menggunakan command pada terminal:  
***git clone https://github.com/rafimaliki/Tucil1\_13522137***
- 2) Masuk ke folder src dengan menggunakan command pada terminal:  
***cd Tucil1\_13522137\src***
- 3) Unduh modul node.js dengan menggunakan command pada terminal:  
***npm i***
- 4) Jalankan *development server* dengan menggunakan command pada terminal:  
***npm run dev***
- 5) Buka laman <http://localhost:5174/> pada *web browser*

### 3. Cara Menggunakan Program

**Cyberpunk 2024 Hacking Minigame Solver**

INSTANT BREACH PROTOCOL SOLVER - START CRACKING, SAMURAI.

GENERATE RANDOM IMPORT DATA

**1** SPECIFY BUFFER SIZE

4

**2** ENTER CODE MATRIX

7A 55 E9 E9 1C 55

**3** ENTER SEQUENCES

E9 1C 55

**4** ENTER SEQUENCES REWARD

15

SOLVE

This page is created based on <https://cyberpunk-hacker.com/> to fulfill Tugas Kecil 1 IF220 Strategi Algoritma using brute force algorithm

GitHub Reference Website 13/2/2027

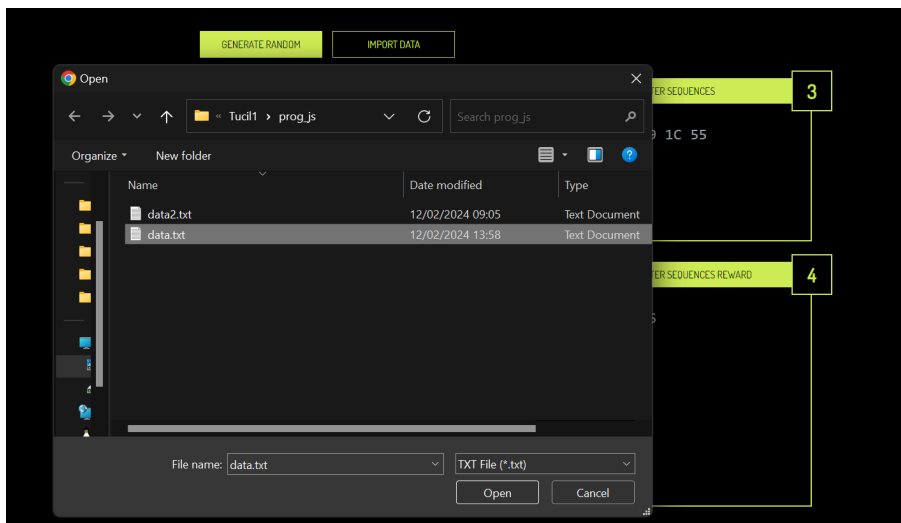
**Gambar 10**  
Tampilan awal program

Sebelum dijalankan, kita harus menginput data dengan salah satu dari tiga cara yaitu, *manual data input*, *data import* dari file .txt, atau *generate random data*. Setelah data terinput secara valid, jalankan algoritma dengan menekan tombol “SOLVE”



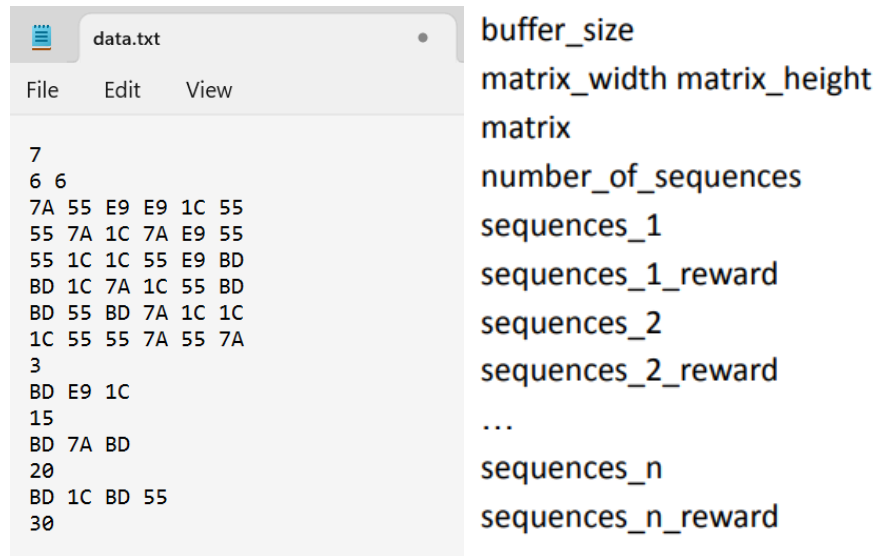
**Gambar 11**  
Input data manual

Untuk menginput data secara manual, dapat langsung diketik pada kolom yang sudah tersedia atau menggunakan *drop down* khusus untuk memilih ukuran buffer.

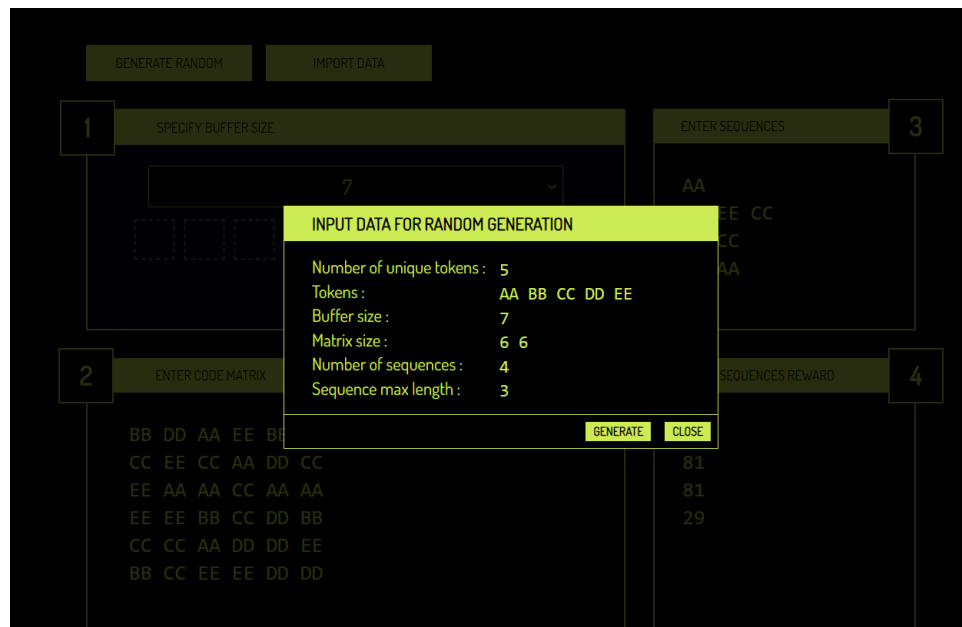


**Gambar 12**  
Import data

Untuk menginput data menggunakan file .txt dapat dengan menekan tombol “IMPORT DATA” lalu memilih file .txt yang diinginkan

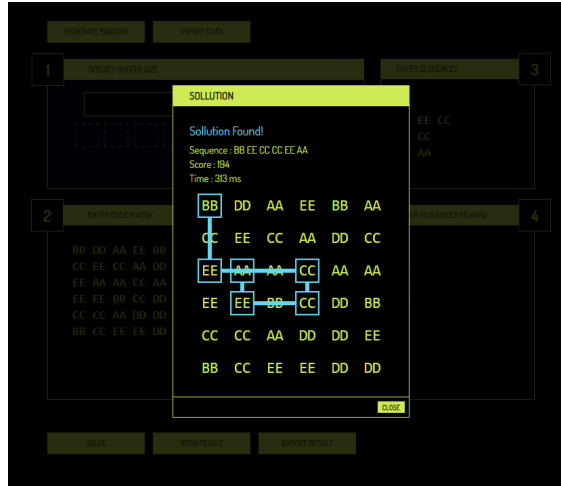


**Gambar 13**  
Template data file .txt

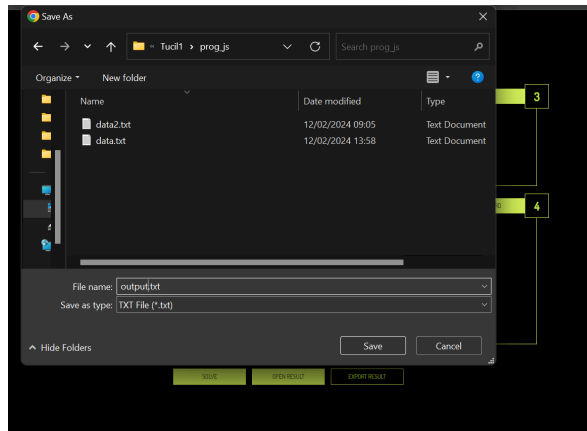


**Gambar 14**  
Generator data random

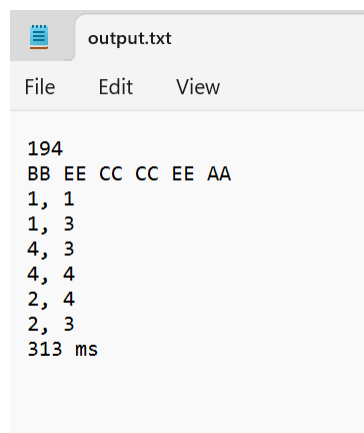
Untuk menggenerasi data secara random, dapat menekan tombol “GENERATE RANDOM”, kemudian mengisi data yang diinginkan dan dilanjutkan dengan menekan tombol “GENERATE”



**Gambar 15**  
Tampilan solusi

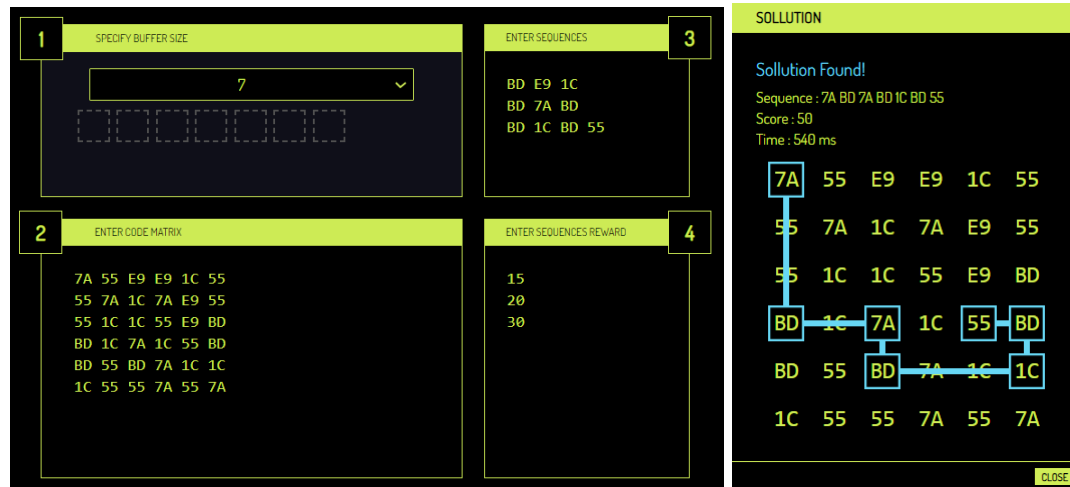


**Gambar 15**  
Tampilan “EXPORT RESULT”

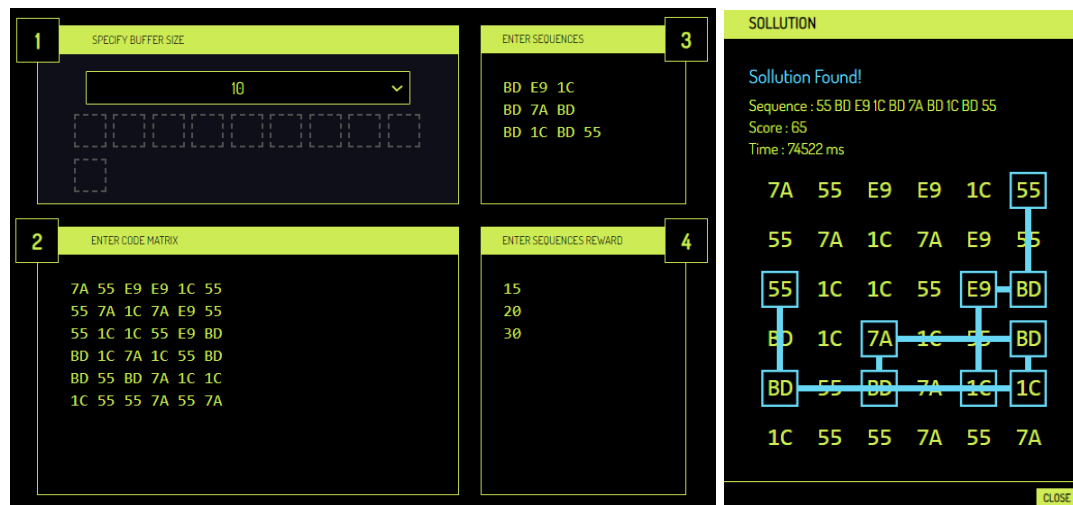


**Gambar 16**  
Isi output.txt

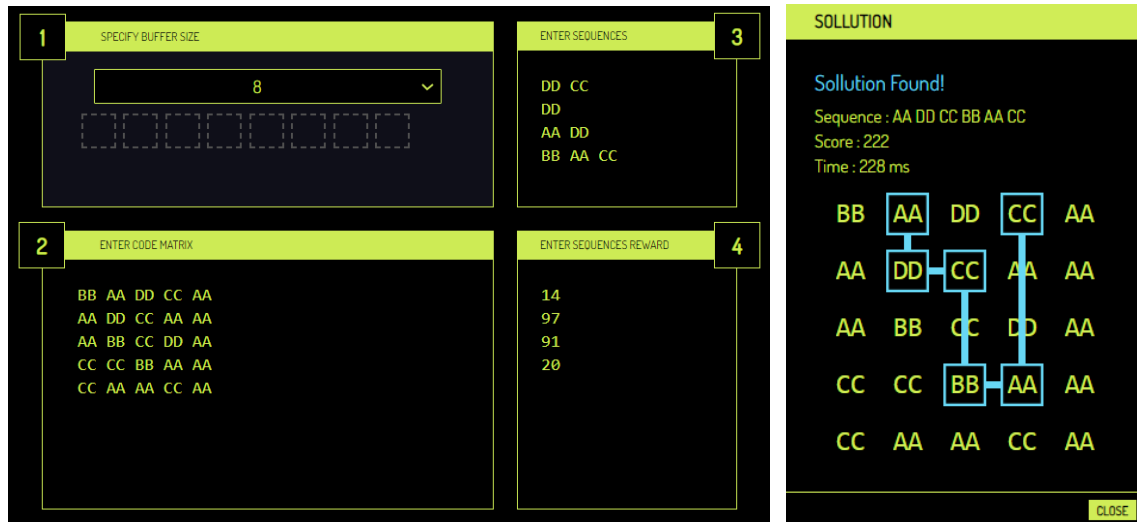
#### 4. Testing



**Gambar 17**  
Test case 1 (data dari spek)



**Gambar 18**  
Test case 2 (full solution dari Test case 1 ditemukan dengan buffer ukuran 10)



**Gambar 19**

Test case 3 (contoh buffer bisa tidak penuh)



**Gambar 20**

Test case 4 (contoh tidak ada solusi)

1

SPECIFY BUFFER SIZE

5

2

ENTER CODE MATRIX

CC CC DD BB AA AA CC AA  
BB DD BB BB AA DD BB AA  
DD DD BB BB BB CC DD BB  
AA CC CC CC CC BB DD AA  
BB BB BB AA BB BB BB BB  
CC CC AA BB DD BB AA AA  
AA AA DD AA AA BB CC AA  
BB BB DD BB BB AA BB AA

3

ENTER SEQUENCES

AA  
CC AA AA  
BB  
BB BB

4

ENTER SEQUENCES REWARD

28  
59  
76  
96

SOLLUTION

Sollution Found!  
Sequence : CC AA AA BB BB  
Score : 259  
Time : 52 ms

CC CC DD BB AA AA CC AA  
BB DD BB BB AA DD BB AA  
DD DD BB BB BB CC DD BB  
AA CC CC CC CC BB DD AA  
BB BB BB AA BB BB BB BB  
CC CC AA BB DD BB AA AA  
AA AA DD AA AA BB CC AA  
BB BB DD BB BB AA BB AA

CLOSE

**Gambar 21**  
Test case 5 (matriks 8x8)

1

SPECIFY BUFFER SIZE

16

2

ENTER CODE MATRIX

CC DD AA AA  
AA DD CC BB  
AA DD CC DD  
AA AA CC BB

3

ENTER SEQUENCES

CC CC AA  
CC BB DD  
CC BB  
BB DD DD

4

ENTER SEQUENCES REWARD

10  
74  
64  
84

SOLLUTION

Sollution Found!  
Sequence : AA CC BB DD DD AA CC CC AA  
Score : 232  
Time : 298 ms

CC DD AA AA  
AA DD CC BB  
AA DD CC DD  
AA AA CC BB

CLOSE

**Gambar 22**  
Test case 6 (buffer 16)



## 5. Lampiran

### Checklist Program

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa kesalahan	✓	
2. Program berhasil dijalankan	✓	
3. Program dapat membaca masukan 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	✓	