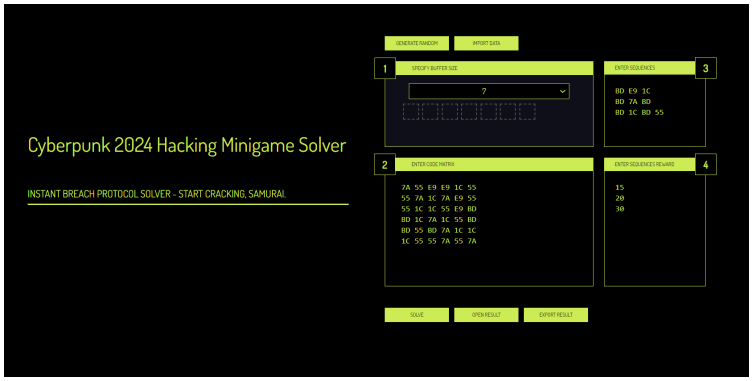
Laporan Tugas Kecil 1 IF2211 Strategi Algoritma

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

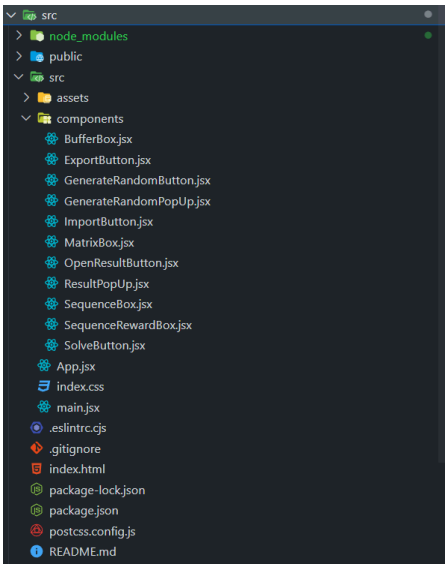
Program ini menggunakan fungsi random() pada library Math bawaan JavaScript untuk menggenerasi data random yang diperlukan berdasarkan data yang diinput oleh user.

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.

```jsx
import React from "react";

const SolveButton = ({ gameData, setOutput, setShowResult, setInputError }) => {
  const handleClick = () => {
    // console.log(gameData);

    const removeEmptyStrings = (arr) => arr.filter((item) => item !== "");

    var buffer_size = gameData.buffer_size;
    var matrix_width = gameData.matrix_width;
    var matrix_height = gameData.matrix_height;

    var matrix = gameData.matrix.map((element) => removeEmptyStrings(element));
    var sequences = gameData.sequences.map((element) =>
      removeEmptyStrings(element)
    );
    var number_of_sequences = gameData.number_of_sequences;
    var sequences_reward = [];
    for (let i = 0; i < gameData.sequences_reward.length; i++) {
      sequences_reward.push(parseInt(gameData.sequences_reward[i]));
    }

    let max_score = gameData.max_score;
    var max_score_found = 0;

    var sequences_list = [];
    var sequences_idx_list = [];
    var score_list = [];

    var output = [];
    var start_time;
```

Gambar 2
Deklarasi variabel

```
  /* Check if matrix is valid */
  for (let i = 0; i < matrix_height; i++) {
    if (matrix[i].length !== matrix_width) {
      console.log("Matrix error!");
      setInputError(true);
      return;
    }
  }

  // console.log(sequences.length);
  // console.log(sequences_reward.length);

  /* Check if sequences and sequences reward match */
  if (sequences.length !== sequences_reward.length) {
    console.log("Sequences and sequences reward does not match!");
    setInputError(true);
    return;
  }

  setInputError(false);
```

Gambar 3
Validasi data pada variabel

```
/* Check if list1 is sublist of list2 function */
function IS_SUB_LIST(l1, l2) {
  const len_l1 = l1.length;
  const len_l2 = l2.length;

  if (len_l1 > len_l2) {
    return false;
  }

  for (let i = 0; i < len_l2 - len_l1 + 1; i++) {
    if (l2.slice(i, i + len_l1).join("") === l1.join("")) {
      return true;
    }
  }
  return false;
}
```

Gambar 4
Fungsi pembantu IS_SUB_LIST untuk pengecekan sekuens

```
/* Score calculation function */
function CALCULATE_SCORE(sequence) {
  let score = 0;
  for (let i = 0; i < number_of_sequences; i++) {
    if (IS_SUB_LIST(sequences[i], sequence)) {
      score += sequences_reward[i];
    }
  }
  return score;
}
```

Gambar 5
Fungsi pembantu CALCULATE_SCORE untuk menghitung skor sekuens

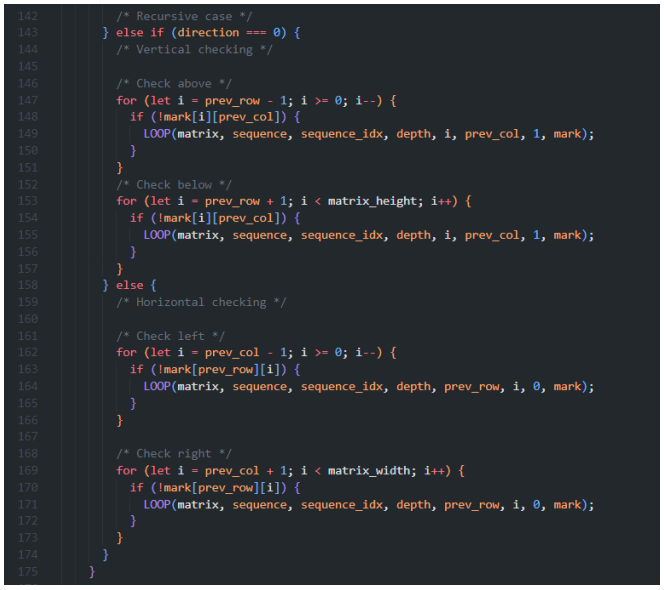
```
/* Recursive LOOP function */
function RECURSION_LOOP(
  sequence,
  sequence_idx,
  depth,
  prev_row,
  prev_col,
  direction,
  mark
) {
  /* Base case */
  if (depth === buffer_size) {
    const score = CALCULATE_SCORE(sequence);

    /* Append data to global arrays */
    if (score >= max_score_found) {
      if (
        !sequences_list.some(
          (arr) => JSON.stringify(arr) === JSON.stringify(sequence)
        )
      ) {
        if (score > max_score_found) {
          max_score_found = score;
          sequences_list.splice(0);
          sequences_idx_list.splice(0);
          score_list.splice(0);
        }

        sequences_list.push(sequence);
        sequences_idx_list.push(sequence_idx);
        score_list.push(score);
      }
    }
```

Gambar 5

Fungsi RECURSION_LOOP (1) basis rekursi sekuens sudah sepanjang buffer



```
    /* Recursive case */
  } else if (direction === 0) {
    /* Vertical checking */

    /* Check above */
    for (let i = prev_row - 1; i >= 0; i--) {
      if (!mark[i][prev_col]) {
        LOOP(matrix, sequence, sequence_idx, depth, i, prev_col, 1, mark);
      }
    }
    /* Check below */
    for (let i = prev_row + 1; i < matrix_height; i++) {
      if (!mark[i][prev_col]) {
        LOOP(matrix, sequence, sequence_idx, depth, i, prev_col, 1, mark);
      }
    }
  } else {
    /* Horizontal checking */

    /* Check left */
    for (let i = prev_col - 1; i >= 0; i--) {
      if (!mark[prev_row][i]) {
        LOOP(matrix, sequence, sequence_idx, depth, prev_row, i, 0, mark);
      }
    }

    /* Check right */
    for (let i = prev_col + 1; i < matrix_width; i++) {
      if (!mark[prev_row][i]) {
        LOOP(matrix, sequence, sequence_idx, depth, prev_row, i, 0, mark);
      }
    }
  }
}
```

Gambar 6

Fungsi RECURSION_LOOP (2) kasus rekursi secara vertikal dan horizontal

```
/* Master function */
function SEARCH_SEQUENCE() {
  /* Initial LOOP from every element at row 0 */
  for (let i = 0; i < matrix_width; i++) {
    /* Initialize marker matrix for keeping track of visited elements */
    const mark_matrix = Array.from({ length: matrix_height }, () =>
      Array(matrix_width).fill(false)
    );
    mark_matrix[0][i] = true;

    /* Initialize sequence and sequence index matrix*/
    const sequence = [matrix[0][i]];
    const sequence_idx = [[0, i]];
    const prev_row = 0;
    const prev_col = i;
    RECURSION_LOOP(
      sequence.slice(),
      sequence_idx.slice(),
      1,
      prev_row,
      prev_col,
      0,
      mark_matrix.slice()
    );
  }
```

Gambar 7

Fungsi SEARCH_SEQUENCE (1) untuk memulai pemanggilan fungsi RECURSION_LOOP

```
  if (max_score_found > 0) {
    // console.log(`Score: ${max_score_found}`);
    // console.log("Sequence:", sequences_list[max_score_idx].join(" "));

    var making_efficient = true;

    var new_sequences_list = JSON.parse(JSON.stringify(sequences_list));
    var new_sequences_idx_list = JSON.parse(
      JSON.stringify(sequences_idx_list)
    );

    var new_score = JSON.parse(JSON.stringify(score_list));
    var temp_score = JSON.parse(JSON.stringify(score_list));

    /* While loop for making sequence shorter */
    while (making_efficient) {
      console.log("Making efficient...");
      temp_score = JSON.parse(JSON.stringify(new_score));
      new_score = [];

      let temp_sequences_list = JSON.parse(
        JSON.stringify(new_sequences_list)
      );
      let temp_sequences_idx_list = JSON.parse(
        JSON.stringify(new_sequences_idx_list)
      );
      for (let i = 0; i < new_sequences_list.length; i++) {
        new_sequences_list[i].pop();
        new_sequences_idx_list[i].pop();
        // console.log(new_sequences_list[i]);
        // console.log(CALCULATE_SCORE(new_sequences_list[i]));
        new_score.push(CALCULATE_SCORE(new_sequences_list[i]));
      }
      const max_score = Math.max(...new_score);
      if (max_score < max_score_found) {
        making_efficient = false;
        sequences_list = temp_sequences_list;
        sequences_idx_list = temp_sequences_idx_list;
        score_list = temp_score;
      }
    }
```

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

2. Cara Menjalankan Program

Program dapat dijalankan dengan mengakses https://cyberpunk2024.vercel.app/ atau dengan meng-*clone* repository 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*

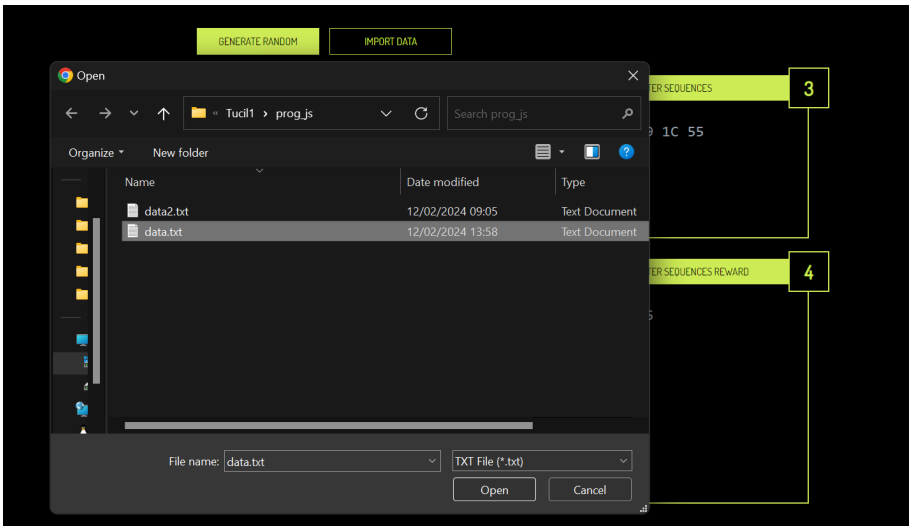
3. Cara Menggunakan Program

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”

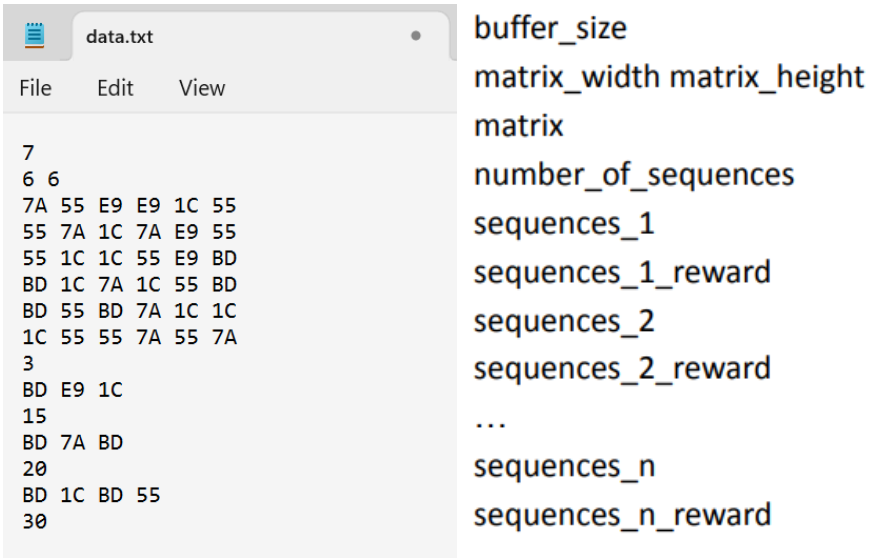
Gambar 11
Input data manual

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

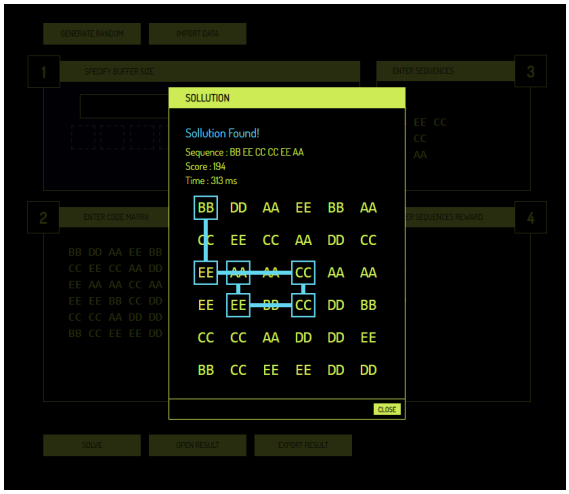
Untuk menginput data menggunakan file .txt dapat dengan menekan tombol “IMPORT DATA” lalu memilih file .txt yang diinginkan



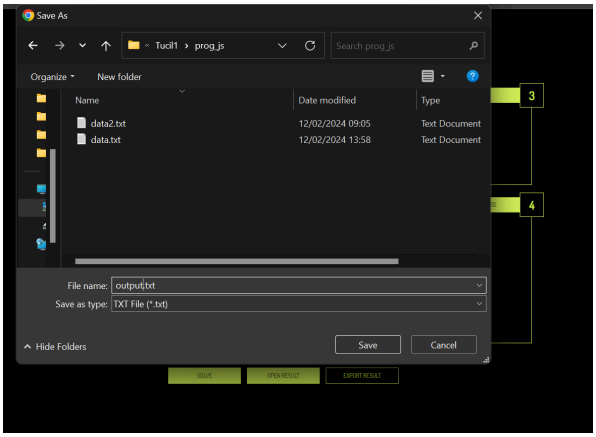
Gambar 13
Template data file .txt

Gambar 14
Generator data random

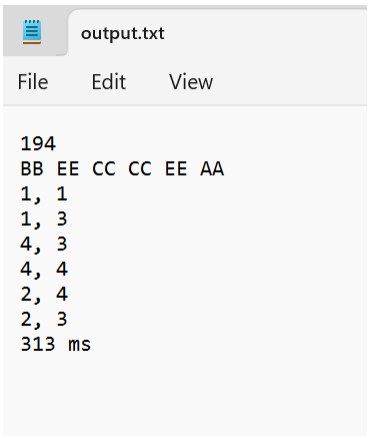
Untuk menggenerasi data secara random, dapat menekan tombol “GENERATE RANDOM”, kemudian mengisi data yang diinginkan dan dilanjutkan dengan menekan tombol “GENERATE”



Gambar 15
Tampilan solusi

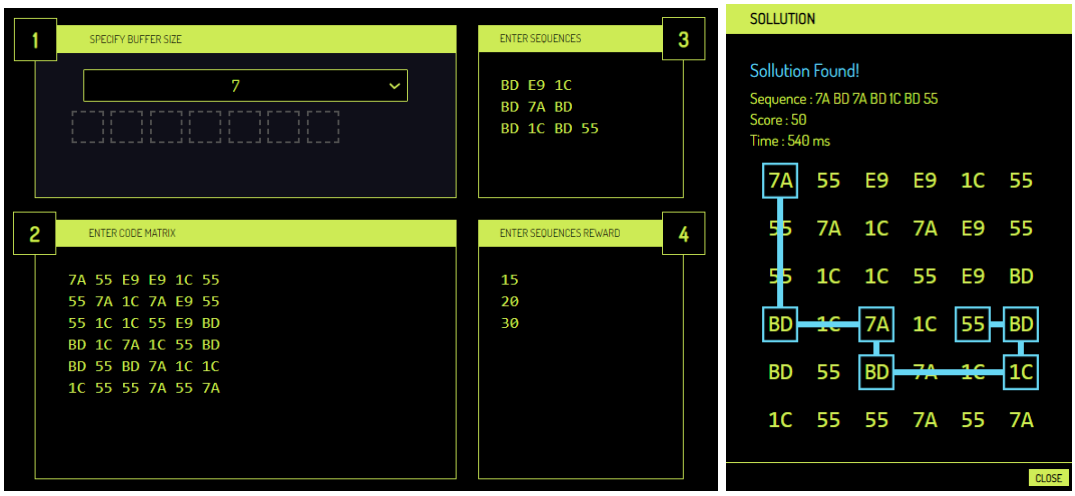


Gambar 15
Tampilan “EXPORT RESULT”



Gambar 16
Isi output.txt

4. Testing

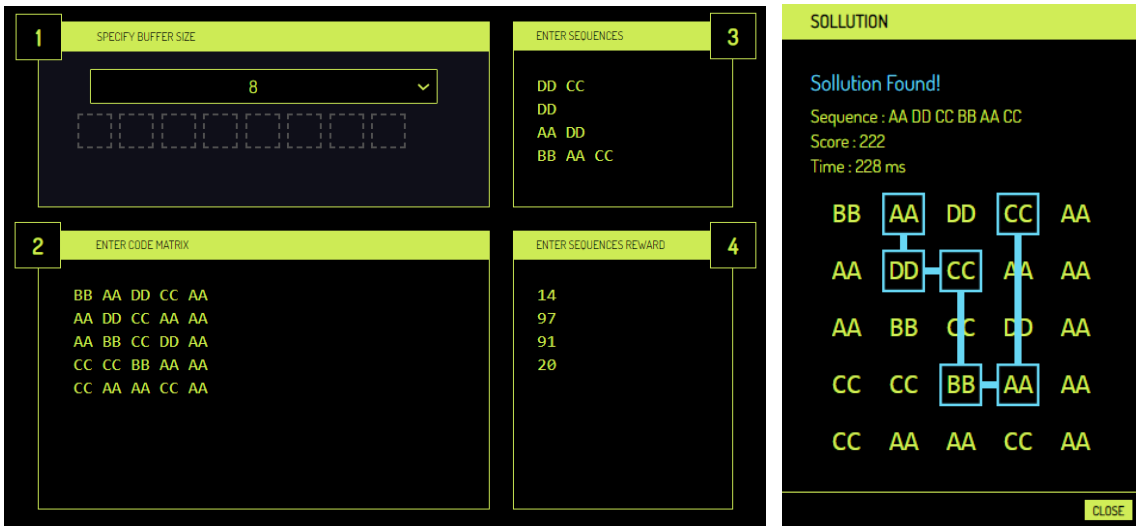


Gambar 17

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)

Gambar 21

Test case 5 (matriks 8x8)

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	✓	