

**LAPORAN TUGAS KECIL I**  
**IF2211 STRATEGI ALGORITMA**

Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force



Disusun oleh:

Haikal Assyauqi 13522052

**Program Studi Teknik Informatika**  
**Sekolah Teknik Elektro dan Informatika**  
**Institut Teknologi Bandung**  
**2023**

## Daftar Isi

<b>BAGIAN I ALGORITMA BRUTE FORCE.....</b>	<b>3</b>
<b>BAB II SOURCE PROGRAM.....</b>	<b>4</b>
<b>BAGIAN III SCREENSHOT HASIL TEST .....</b>	<b>15</b>
<b>LINK REPOSITORY .....</b>	<b>21</b>
<b>CHECKLIST.....</b>	<b>21</b>

## BAGIAN I

### ALGORITMA BRUTE FORCE

Algoritma *Brute Force*, adalah algoritma dengan pendekatan yang *straightforward* untuk memecahkan suatu masalah. Algoritma ini biasanya bergantung pada kekuatan komputasi yang tinggi untuk mendapatkan semua solusi yang tepat daripada menggunakan teknik yang canggih. Dalam penyelesaian permainan Cyberpunk 2077 Breach Protocol dengan pendekatan *brute force*, algoritma yang digunakan adalah sebagai berikut:

- Membuat semua kemungkinan solusi berupa gabungan sekuens yang jumlah tokennya lebih kecil atau sama dengan buffer size
- Menghitung poin yang bisa dicapai oleh suatu sekuens
- Melakukan sorting dengan preferensi poin maksimal diikuti langkah terkecil
- Melakukan algoritma *trackback* pada setiap kemungkinan solusi dimulai dari sekuens yang memiliki poin terbesar, hingga selesai

Setelah algoritma diselesaikan semua, maka output yang akan dikeluarkan yakni poin, solusi dengan poin terbanyak dan langkah paling sedikit, koordinat langkah, serta waktu yang digunakan untuk mengeksekusi.

Secara algoritma, kompleksitas waktu sangat tinggi ketika menghitung banyak kemungkinan sekuens yang terbentuk, jika buffer size = infinity, maka kompleksitas waktu mencapai  $O(n^{\text{banyak sekuens}})$ , tetapi karena terdapat ukuran buffer hal ini dapat diperkecil.

## BAB II

### SOURCE PROGRAM

Projek ini ditulis dalam Bahasa C++, menggunakan *library*:

1. `iostream (c++)`
2. `vector (c++)`
3. `string (c++)`
4. `fstream (c++)`
5. `random (c++)`
6. `bits/stdc++.h (c++)`
7. `chrono (c++)`

Terdapat beberapa fungsi buatan sendiri yang digunakan untuk memudahkan pengerjaan

Fungsi meliputi:

- `ceksama`
- `cekduplicatesequence`
- `parsespace`
- `intersection`
- `checkhistory`

Berikut *source code*-nya:

1. Fungsi `ceksama`

Berfungsi untuk cek apakah sequence memiliki sequence penyusun yang sama atau berbeda

```
bool ceksama(vector<int> angka_penyusun_1, vector<int> angka_penyusun_2) {  
    int sama = 0;  
    int ukuranarray1 = angka_penyusun_1.size();  
    int ukuranarray2 = angka_penyusun_2.size();  
    for (int i = 0; i < ukuranarray1; i++) {  
        for (int j = 0; j < ukuranarray2; j++) {  
            if (angka_penyusun_1[i] == angka_penyusun_2[j]) {  
                sama += 1;  
            }  
        }  
    }  
    if (sama > 0) {  
        return false;  
    }  
    else {  
        return true;  
    }  
}
```

## 2. Fungsi cekduplicatesquence

Mengecek apakah sequence yang terbentuk duplikat dari sequence sebelumnya

```
bool cekduplicatesquence(vector<string> sequence1, vector<string> sequence2) {
    int sama = 0;
    for (int i = 0; i < sequence1.size(); i++) {
        if(sequence1[i] == sequence2[i]) {
            sama += 1;
        }
    }
    if (sama == sequence1.size()) {
        return true;
    }
    else {
        return false;
    }
}
```

## 3. Fungsi parsespace

Parsing sequence yang masuk berupa banyak token ketika input file. memisahkan token dengan token lainnya yang dipisahkan oleh spasi

```
vector<string> parsespace(string kode) {
    vector<string> array;
    string temp;
    for (int i = 0; i < kode.length(); i++) {
        if(kode[i] != ' ') {
            temp += kode[i];
        }
        else {
            array.push_back(temp);
            temp = "";
        }
        if (i == kode.length()-1) {
            array.push_back(temp);
        }
    }
    return array;
}
```

#### 4. Fungsi intersection

Membuat sequence baru yang memiliki hubungan dengan sequence lain

Misal: Sequence 1 = AA BB CC DD; Sequence 2 = CC DD EE;

Sequence baru = AA BB CC DD EE

```
vector<vector<string> > intersection(vector<string> sequence1, vector<string>
sequence2) {
    vector<vector<string> > collection;
    for(int i = 0; i < sequence1.size(); i++) {
        vector<string> section;
        if(sequence1[i] == sequence2[0]) {
            int idx1 = i;
            int idx2 = 0;
            bool same = true;
            while(same == true and idx1 < sequence1.size()) {
                if(sequence1[idx1] == sequence2[idx2]) {
                    idx1 += 1;
                    idx2 += 1;
                }
                else {
                    same = false;
                }
            }
            if (same == true) {
                for(int j = 0; j < sequence1.size(); j++) {
                    section.push_back(sequence1[j]);
                }
                for(int k = idx2; k < sequence2.size(); k++) {
                    section.push_back(sequence2[k]);
                }
                collection.push_back(section);
            }
        }
    }
    return collection;
}
```

## 5. Fungsi checkhistory

Berfungsi untuk menguji apakah titik sudah pernah dijelajahi atau belum

```
bool checkhistory(int baris, int kolom, vector<vector<vector<int> > > history, int idx)
{
    int sama = 0;
    for(int i = 0; i < history[idx].size(); i++) {
        if(history[idx][i][0] == baris and history[idx][i][1] == kolom) {
            sama += 1;
        }
    }
    if (sama > 0) {
        return false;
    }
    else {
        return true;
    }
}
```

Code input

```
if(input == 1) {
    string path = "../src/";
    string namafile;
    cout << "Masukkan nama file yang akan diinput: ";
    cin >> namafile;
    path += namafile;
    ifstream file(path);
    while (file >> buffer_size) {
        file >> map_width >> map_height;
        for(int i = 0; i < map_height; i++) {
            line = { };
            for(int j = 0; j < map_width; j++) {
                file >> kode;
                file >> ws;
                line.push_back(kode);
            }
            map.push_back(line);
        }
        string token; // Sequence
        file >> total_sequence;
        file >> ws;
        int ulang = total_sequence;
        for(int i = 0; i < ulang; i++) {
            getline(file, token);
        }
    }
}
```

```

        vector<string> parsetoken = parsespace(token);
        kumpulankode.push_back(parsetoken);
        file >> angka;
        poin.push_back(angka);
        file >> ws;
    }
}

// Input CLI
else if(input == 2){
    int batasbawah = 0;
    int bataspointatas = 100;
    random_device rand_dev;
    mt19937 generator(rand_dev());
    int jumlahtoken;
    cin >> jumlahtoken;
    string token_unik;
    vector<string> kumpulan_token_unik;
    // Save Token
    for(int i = 0; i < jumlahtoken; i++) {
        cin >> token_unik;
        kumpulan_token_unik.push_back(token_unik);
    }
    cin >> buffer_size;
    cin >> map_width >> map_height;
    for(int i = 0; i < map_height; i++) {
        line = { };
        for(int j = 0; j < map_width; j++) {
            uniform_int_distribution<int> map(batasbawah, jumlahtoken-1);
            line.push_back(kumpulan_token_unik[map(generator)]);
        }
        map.push_back(line);
    }
    int panjang_maksimal;
    int panjangsequence;
    cin >> total_sequence;
    cin >> panjang_maksimal;
    for(int i = 0; i < total_sequence; i++) {
        line = { };
        uniform_int_distribution<int> panjangseq(2, panjang_maksimal);
        uniform_int_distribution<int> map(batasbawah, jumlahtoken-1);
        uniform_int_distribution<int> poinrand(batasbawah, bataspointatas);
        panjangsequence = panjangseq(generator);
        for(int i = 0; i < panjangsequence; i++) {

```



```

        line.push_back(kumpulan_token_unik[map(generator)]);
    }
    kumpulankode.push_back(line);
    poin.push_back(poinrand(generator));
}
cout << "Matriks permainan: " << endl;
for(int i = 0; i < map_height; i++) {
    for(int j = 0; j < map_width; j++) {
        cout << map[i][j] << " ";
    }
    cout << endl;
}
cout << "Sequence: " << endl;
for(int i = 0; i < kumpulankode.size(); i++) {
    for(int j = 0; j < kumpulankode[i].size(); j++) {
        cout << kumpulankode[i][j] << " ";
    }
    cout << endl;
    cout << poin[i] << endl;
}
}

```

### Pembentukan sequence

```

for (int i = 0; i < total_sequence; i++) {
    if(kumpulankode[i].size() <= buffer_size) {
        num = {};
        num.push_back(i);
        sequence.push_back(kumpulankode[i]);
        angka_penyusun.push_back(num);
    }
}
// cout << angka_penyusun.size();
// cout << angka_penyusun[0]
// Cek Angka Penyusun

for (int i = 2; i <= total_sequence; i++) { // Membentuk digit sequence
    int kemungkinan_Sequences = sequence.size();
    for (int j = 0; j < kemungkinan_Sequences; j++) {
        for (int k = 0; k < kemungkinan_Sequences; k++) {
            bool sama = ceksama(angka_penyusun[j], angka_penyusun[k]);
            if (sama == true) {
                vector<string> unionsequence = {};
                vector<string> sequence1 = sequence[j];
                vector<string> sequence2 = sequence[k];
            }
        }
    }
}

```

```

vector<int> angka_penyusun1 = angka_penyusun[j];
vector<int> angka_penyusun2 = angka_penyusun[k];
vector<int> unionangka_penyusun = {};
for (int l = 0; l < sequence1.size(); l++) {
    unionsequence.push_back(sequence1[l]);
}
for(int m = 0; m < sequence2.size(); m++) {
    unionsequence.push_back(sequence2[m]);
}
for (int n = 0; n < angka_penyusun1.size(); n++) {
    unionangka_penyusun.push_back(angka_penyusun1[n]);
}
for (int o = 0; o < angka_penyusun2.size(); o++) {
    unionangka_penyusun.push_back(angka_penyusun2[o]);
}
int dupe = 0;
for (int p = 0; p < sequence.size(); p++) {
    if(unionsequence.size() == sequence[p].size()) {
        bool duplicate = cekduplicatesequence(unionsequence,
sequence[p]);

        if (duplicate == true) {
            dupe += 1;
        }
    }
}
if (dupe == 0 and unionsequence.size() <= buffer_size) {
    sequence.push_back(unionsequence);
    angka_penyusun.push_back(unionangka_penyusun);
}

// Kemungkinan terjadi interse
vector<vector<string> > shortener =
intersection(sequence1,sequence2);
for (int a = 0; a < shortener.size(); a++) {
    if(shortener[a].size() <= buffer_size) {
        sequence.push_back(shortener[a]);
        angka_penyusun.push_back(unionangka_penyusun);
    }
}
}
}
}
}
}

```

## Sort Sequence

```
for(int i = 0; i < sequence.size(); i++) {
    for(int j = i+1; j < sequence.size(); j++) {
        vector<string> tempseq;
        int tempint;
        if(kumpulanpoin[j] > kumpulanpoin[i]) {
            tempseq = sequence[i];
            sequence[i] = sequence[j];
            sequence[j] = tempseq;
            tempint = kumpulanpoin[i];
            kumpulanpoin[i] = kumpulanpoin[j];
            kumpulanpoin[j] = tempint;
        }
        else if(kumpulanpoin[i] == kumpulanpoin[j]) {
            if(sequence[i].size() > sequence[j].size()) {
                tempseq = sequence[i];
                sequence[i] = sequence[j];
                sequence[j] = tempseq;
                tempint = kumpulanpoin[i];
                kumpulanpoin[i] = kumpulanpoin[j];
                kumpulanpoin[j] = tempint;
            }
        }
    }
}
```

## Pencarian langkah optimal

```
bool found = false;
int row = 0;
vector<int> step;
vector<int> buntu;
vector<vector<int>> > jejaklaki;
int i = 0;
// Ex : sequence ada 4
while (found == false and i < sequence.size()) {
    int row = 0;
    bool change = false;
    while (change == false and row < map_width) {
        vector<vector<vector<int>> > > history; //Menyimpan langkah
        vector<vector<int>> > jejak;
        vector<int> posisi = {0,0};
        for(int j = 0; j < sequence[i].size(); j++) {
            history.push_back({{-1,-1}});
```

```

    }
    posisi[1] = row;
    int idx = 0; // Banyaknya indeks yang telah diolah
    bool optimal = true; // Cek jalan buntu
    int ver = 0; // Mengetahui vertikal atau horizontal
    if(sequence[i][idx] == map[0][posisi[1]]) { // Error ketika row == elemen
        idx += 1;
        jejak.push_back({0,row});
    }
    while (optimal == true and idx < sequence[i].size()) {
        if(ver % 2 == 0) {
            for(int start = 0; start < map.size(); start++) {
                if(sequence[i][idx] == map[start][posisi[1]] and
checkhistory(start,posisi[1],history,idx) and start != posisi[0]) {
                    idx += 1;
                    ver += 1;
                    posisi[0] = start;
                    jejak.push_back(posisi);
                    break;
                }
                else if(ver == 0 and (sequence[i][idx] !=
map[start][posisi[1]] or checkhistory(start,posisi[1],history,idx) == false) and start
== map.size()-1) {
                    optimal = false;
                }
                else if((sequence[i][idx] != map[start][posisi[1]] or
checkhistory(start,posisi[1],history,idx) == false)) and start == map.size()-1) {
//Error
                    if(idx == 0) {
                        optimal = false;
                        break;
                    }
                    // cout << "SALAH" << endl;
                    history[idx-1].push_back(posisi);
                    idx -= 1;
                    ver -= 1;
                    jejak.pop_back();
                    posisi = jejak[idx];
                    break;
                }
            }
        }
        else if(ver % 2 == 1) {
            for(int start = 0; start < map[0].size(); start++) {

```

```

        if(sequence[i][idx] == map[posisi[0]][start] and
checkhistory(posisi[0],start,history,idx) and start != posisi[1]) {
            idx += 1;
            ver += 1;
            posisi[1] = start;
            jejak.push_back(posisi);
            break;
        }
        else if(ver == 0 and (sequence[i][idx] !=
map[posisi[0]][start] or checkhistory(posisi[0],start,history,idx) == false) and start
== map[0].size()-1) {
            optimal = false;
        }
        else if((sequence[i][idx] != map[posisi[0]][start] or
checkhistory(posisi[0],start,history,idx) == false) and start == map[0].size()-1) {
            // cout << "SALAH" << endl;
            if(idx == 0) {
                optimal = false;
                break;
            }
            else {
                history[idx-1].push_back(posisi);
                idx -= 1;
                ver -= 1;
                jejak.pop_back();
                posisi = jejak[idx];
                break;
            }
        }
        // else if(start == map[0].size()-1) {
        //     cout << posisi[0] << " " << start << endl;
        //     cout << (checkhistory(posisi[0],start,history,idx) ==
false) << endl;
        // }
    }
}
if(idx == sequence[i].size() and kumpulanpoin[i] > max) {
    found = true;
    change = true;
    max = kumpulanpoin[i];
    cout << "Poin maksimal " << kumpulanpoin[i] << endl;
    finalkode = sequence[i];
    jejakkaki = jejak;
}
}

```

```

        row += 1;
    }
    i += 1;
}

```

## Output

```

auto end = chrono::high_resolution_clock::now();
double elapsed_time_ms = chrono::duration<double, milli>(end - start).count();

if(max == 0) {
    cout << "Poin maksimal " << max << endl;
    cout << endl;
}
for(int i = 0; i < finalkode.size(); i++) {
    cout << finalkode[i] << " ";
}
cout << endl;
for(int i = 0; i < jejaklaki.size(); i++) {
    cout << jejaklaki[i][1]+1 << ", " << jejaklaki[i][0]+1 << endl;
}
cout << time << " ms" << endl;
char yorn;
cout << "Apakah ingin disimpan di file txt? (y/n) ";
cin >> yorn;
if(yorn == 'y') {
    string pathinput = "../test/";
    string fileinput;
    cout << "Masukkan nama file: ";
    cin >> fileinput;
    pathinput += fileinput;
    ofstream infile;
    infile.open(pathinput);
    infile << max << "\n";
    for(int i = 0; i < finalkode.size(); i++) {
        infile << finalkode[i] << " ";
    }
    infile << "\n";
    for (int i = 0; i < jejaklaki.size(); i++) {
        infile << jejaklaki[i][1] + 1 << ", " << jejaklaki[i][0] + 1;
        infile << "\n";
    }
    infile << time << " ms";
}

```

### BAGIAN III SCREENSHOT HASIL TEST

Input: inputtxt1.txt (Teks) (Small)

<pre> E:\Semester 4\Strategi Algoritma\Tucil1_13522052\bin&gt;main.exe Pilih cara input: 1. File 2. CLI Masukkan cara input: 1 Masukkan nama file yang akan diinput: inputtext1.txt Poin maksimal 92 1G FD BC 5A FD 8E 2, 1 2, 4 6, 4 6, 1 3, 1 3, 2 0 ms Apakah ingin disimpan di file txt? (y/n) y Masukkan nama file: tc1.txt         </pre>	<pre> 92 1G FD BC 5A FD 8E 2, 1 2, 4 6, 4 6, 1 3, 1 3, 2 0 ms You, 16         </pre>
---	--

Input: inputtxt2.txt (Teks) (Medium)

<pre> PS E:\Semester 4\Strategi Algoritma\Tucil1_13522052&gt; ./run.bat  E:\Semester 4\Strategi Algoritma\Tucil1_13522052&gt;g++ src/tucil.cpp -o bin/main  E:\Semester 4\Strategi Algoritma\Tucil1_13522052&gt;cd bin  E:\Semester 4\Strategi Algoritma\Tucil1_13522052\bin&gt;main.exe Pilih cara input: 1. File 2. CLI Masukkan cara input: 1 Masukkan nama file yang akan diinput: inputtext2.txt Poin maksimal 184 GR PS PS MM AB PS AB 1, 1 1, 4 4, 4 4, 1 5, 1 5, 10 9, 10 86 ms         </pre>	<pre> 184 GR PS PS MM AB PS AB 1, 1 1, 4 4, 4 4, 1 5, 1 5, 10 9, 10 82 ms You, 10 hour         </pre>
--	---

Input: inputtxt3.txt (Teks) (Large)

```
PS E:\Semester 4\Strategi Algoritma\Tucil1_13522052> ./run.bat

E:\Semester 4\Strategi Algoritma\Tucil1_13522052>g++ src/tucil.cpp -o
bin/main

E:\Semester 4\Strategi Algoritma\Tucil1_13522052>cd bin

E:\Semester 4\Strategi Algoritma\Tucil1_13522052\bin>main.exe
Pilih cara input:
1. File
2. CLI
Masukkan cara input: 1
Masukkan nama file yang akan diinput: inputtext3
Poin maksimal 255
DD DD AA DD BB BB CC
2, 1
2, 2
1, 2
1, 7
2, 7
2, 4
1, 4
8788 ms
Apakah ingin disimpan di file txt? (y/n) y
Masukkan nama file: tc3
```

```
255 You, 22 hour
DD DD AA DD BB BB CC
2, 1
2, 2
1, 2
1, 7
2, 7
2, 4
1, 4
8788 ms
```



## Input: CLI Small

```
PS E:\Semester 4\Strategi Algoritma\Tucil1_13522052> ./run.bat

E:\Semester 4\Strategi Algoritma\Tucil1_13522052>g++ src/tucil.cpp -o bin/main

E:\Semester 4\Strategi Algoritma\Tucil1_13522052>cd bin

E:\Semester 4\Strategi Algoritma\Tucil1_13522052\bin>main.exe
Pilih cara input:
1. File
2. CLI
Masukkan cara input: 2
5
JK T4 8T EA MK
8
5 3
3
3
Matriks permainan:
8T EA T4 MK JK
8T MK 8T MK MK
JK EA T4 8T EA
Sequence:
T4 JK 8T
33
8T T4
3
MK MK
1
Poin maksimal 37
MK MK 8T T4 JK 8T
4, 1
4, 2
3, 2
3, 3
1, 3
1, 1
9 ms
Apakah ingin disimpan di file txt? (y/n) y
Masukkan nama file: tc4.txt
```

```
37      You, 15 m
MK MK 8T T4 JK 8T
4, 1
4, 2
3, 2
3, 3
1, 3
1, 1
9 ms
```

Input: CLI Medium

```
Pilih cara input:
1. File
2. CLI
Masukkan cara input: 2
5
AA BB CC DD EE
10
8 8
5
4
Matriks permainan:
CC DD BB EE AA CC EE CC
EE EE AA DD BB CC DD DD
BB AA CC BB BB CC BB AA
AA EE EE AA DD DD CC BB
AA BB AA CC EE EE BB DD
EE CC EE AA CC BB EE DD
AA DD BB CC BB BB EE EE
CC DD EE EE DD BB DD DD
Sequence:
DD CC
6
DD DD EE
85
CC AA
30
BB AA CC
69
DD DD DD
58
Poin maksimal 248
DD CC AA DD DD DD EE BB AA CC
2, 1
2, 6
4, 6
4, 2
7, 2
7, 8
3, 8
3, 1
5, 1
5, 6
788 ms
Apakah ingin disimpan di file txt? (y/n) y
Masukkan nama file: tc6
```

```
248
DD CC AA DD DD DD EE BB AA CC
2, 1
2, 6
4, 6
4, 2
7, 2
7, 8
3, 8
3, 1
5, 1
5, 6
788 ms
```

## Input CLI Large

```
PS E:\Semester 4\Strategi Algoritma> ./run.bat  
E:\Semester 4\Strategi Algoritma>g++ src/tucil.cpp -o bin/main  
E:\Semester 4\Strategi Algoritma>cd bin  
E:\Semester 4\Strategi Algoritma\bin>main.exe  
Pilih cara input:  
1. File  
2. CLI  
Masukkan cara input: 2  
5  
JK T4 8T KI II  
10  
8 8  
7  
5
```

Matriks permainan:

```
8T KI T4 II JK 8T II 8T  
II II JK KI T4 8T KI KI  
T4 JK 8T T4 T4 8T T4 JK  
JK II II JK KI KI 8T T4  
JK T4 JK 8T II II T4 KI  
II 8T II JK 8T T4 II KI  
JK KI T4 8T T4 T4 II II  
8T KI II II KI T4 KI KI
```

Sequence:

KI 8T

6

KI KI II II

14

JK T4 8T T4

9

KI 8T KI

70

8T 8T T4 KI II

75

II T4 JK 8T

45

II KI

58

Poin maksimal 209

8T 8T T4 KI II KI 8T KI

1, 1

1, 8

6, 8

6, 4

2, 4

2, 1

6, 1

6, 4

42671 ms

Apakah ingin disimpan di file txt? (y/n) y

Masukkan nama file: tc5.txt

## Input CLI Random

```
E:\Semester 4\Strategi Algoritma\Tucil1_13522052\bin>main.exe
Pilih cara input:
1. File
2. CLI
Masukkan cara input: 2
6
AB MI PS GR GP MM
10
8 8
5
7
Matriks permainan:
GR GP PS MM AB GR MM GR
MM MM MI GR MI GR GP GP
MI AB GR MI PS GR MI AB
MI MM MM AB GR GP PS PS
AB PS AB PS MM MM MI GR
MM GR MM AB GR MI MM GP
AB GP MI PS PS MI MM MM
PS GP MM MM GP PS GP GP
Sequence:
GR PS AB
65
GR GP MM AB GR AB
30
MI AB PS GP
56
GP GP GR PS MI
78
GP GR MM PS MI PS PS
14
Poin maksimal 143
AB GP GP GR PS MI GR PS AB
5, 1
5, 8
8, 8
8, 1
3, 1
3, 2
4, 2
4, 5
1, 5
11 ms
Apakah ingin disimpan di file txt? (y/n) y
Masukkan nama file: tc7
```

	AB	GP	GP	GR	PS	MI	GR	PS	AB
143									
5, 1									
5, 8									
8, 8									
8, 1									
3, 1									
3, 2									
4, 2									
4, 5									
1, 5									

11 ms You, 37 seconds

## LINK REPOSITORY

[https://github.com/Haikalin/Tucil1\\_13522052](https://github.com/Haikalin/Tucil1_13522052)

## CHECKLIST

Poin	Ya	Tidak
Program dapat dikompilasi tanpa kesalahan	✓	
Program berhasil dijalankan	✓	
Program dapat membaca membaca masukan berkas.txt	✓	
Program dapat menghasilkan masukan secara acak	✓	
Solusi yang diberikan program optimal	✓	
Program dapat menyimpan solusi dalam berkas .txt	✓	
Menampilkan GUI		✓