

**LAPORAN TUGAS KECIL 1**  
**IF2211 STRATEGI ALGORITMA**  
**CYBERPUNK 2077 BREACH PROTOCOL SOLVER**



Disusun oleh:  
Abdullah Mubarak 13522101

**Program Studi Teknik Informatika**  
**Sekolah Teknik Elektro dan Informatika**  
**Institut Teknologi Bandung**  
**2022**

## Bab 1. Algoritma Brute Force

Algoritma brute force adalah algoritma yang menyelesaikan suatu masalah dengan pendekatan yang *straitforward*.

Pada penyelesaian Cyberpunk 2077 Breach Protocol dengan algoritma *brute force*, saya menggunakan langkah-langkah algoritma sebagai berikut :

1. Dilakukan perulangan sebanyak jumlah kolom untuk langkah-langkah setelah ini. Hal ini dilakukan untuk mengecek semua kemungkinan titik mulai *sequence*.
2. Dilakukan perulangan sebanyak x dengan x adalah selisih maksimal panjang *buffer* dan panjang *sequence* berhadiah terkecil untuk langkah-langkah setelah ini. Hal ini dilakukan untuk mengecek semua kemungkinan panjang *sequence*.
3. Ditentukan semua langkah *sequence* yang mungkin dengan algoritma *back tracking* dan dilakukan algoritma mencari total nilai *reward* untuk semua *sequence* yang terbentuk.

Algoritma *back tracking* (algoritma 1) yang digunakan untuk menentuk semua langkah *sequence* dengan panjang tertentu :

1. Simpan posisi awal *code* (hanya kolom nya saja) dan tokenya ke *vector*, lalu cek semua kemungkinan posisi selanjutnya (1 sampai jumlah baris, kecuali posisi sekarang) dimulai dari posisi yang paling kecil.
2. Simpan setiap posisi dan token yang telah dikunjungi ke *vector*.
3. Ketika panjang *code* sudah mencapai panjang *buffer* sekarang, cek *reward* yang dapat didapatkan dengan algoritma 2.
4. Cari *code* dengan reward paling besar sebagai *vector code* dan Riwayat posisi *final*.

Algoritma mencari total nilai *reward* dari sebuah *sequence* (algoritma 2):

1. Dicocokkan *sequence* berhadiah pertama terhadap *sequence code* terurut dari elemen pertamanya, jika ditemukan kecocokan *sequence*, maka pencocokan berhenti dan nilai *reward* ditambahkan.
2. Langkah 1 diulangi untuk semua *sequence* berhadiah.

## Bab 2. Source Program

Saya menggunakan Bahasa C++ untuk mengerjakan tugas ini dengan menggunakan library:

1. iostream.
2. fstream.
3. string.
4. vector.
5. chrono.
6. random.

Berikut source codenya:

1. Fungsi getTotalValue

```
int getTotalValue(int SequenceAmount,
                 vector<vector<string>>& matrixSequence,
                 vector<string>& arrayCode,
                 vector<int>& arrayRewards,
                 vector<int>& listSequenceLength,
                 int buff) {
    int totalReward = 0;
    for (int i = 0; i < SequenceAmount; i++){
        vector<int> vectorRow;
        int count = 0;
        int indexCode = 0;
        bool found = false;
        while (indexCode < buff && !found) {
            if (matrixSequence[i][count] == arrayCode[indexCode]) {
                count++;
            } else {
                count = 0;
            }
            if (count == listSequenceLength[i]) {
                totalReward += arrayRewards[i];
                found = true;
            }
            indexCode++;
        }
    }
    return totalReward;
}
```

## 2. Fungsi generateCode

```
void generateCode(int SequenceAmount,
                 vector<string>& arrayCode2,
                 vector<int>& arrayDigit2,
                 vector<int>& arrayDigit,
                 vector<vector<string>>& matrixCode,
                 vector<vector<string>>& matrixSequence,
                 vector<int>& arrayRewards,
                 vector<int>& listSequenceLength,
                 bool isVertical,
                 int x, int buff,
                 int row, int col,
                 int currentRow,
                 int currentCol,
                 vector<string>& arrayCode,
                 int* max) {
    int c;
    if (x == 0) {
        int reward = getTotalValue(SequenceAmount,
                                   matrixSequence,
                                   arrayCode, arrayRewards,
                                   listSequenceLength, buff);
        if (reward > *max) {
            *max = reward;
            arrayCode2.clear();
            arrayDigit2.clear();
            for (int i = 0; i < buff; i++) {
                arrayCode2.push_back(arrayCode[i]);
                arrayDigit2.push_back(arrayDigit[i]);
            }
        }
        return;
    }
    if (isVertical) {
        currentMaxValue = row;
        c = currentRow;
    } else {
        currentMaxValue = col;
        c = currentCol;
    }
}
```

```

for (int digit = 0 ; digit < currentMaxValue; digit++) {
    if (digit != c) {
        arrayCode.push_back(matrixCode[currentRow][currentCol]);
        arrayDigit.push_back(digit);
        if (isVertical) {
            generateCode(sequenceAmount,
                        arrayCode2,
                        arrayDigit2,
                        arrayDigit, matrixCode,
                        matrixSequence, arrayRewards,
                        listSequenceLength,
                        !isVertical,
                        x - 1, buff,
                        row, col,
                        digit, currentCol,
                        arrayCode, max);
        } else {
            generateCode(sequenceAmount,
                        arrayCode2,
                        arrayDigit2,
                        arrayDigit, matrixCode,
                        matrixSequence, arrayRewards,
                        listSequenceLength,
                        !isVertical,
                        x - 1, buff,
                        row, col,
                        currentRow, digit,
                        arrayCode, max);
        }
        arrayCode.pop_back();
        arrayDigit.pop_back();
    }
}
}

```

```

bool isVectorInMatrix (vector<string> v, vector<vector<string>> matrix, int rowMatrix) {
    for (int i = 0; i < rowMatrix; i++) {
        if (v == matrix[i]) {
            return true;
        }
    }
    return false;
}

```

### 3. Fungsi randomize

```
void randomize(vector<vector<string>>& matrixCode,
               int* SequenceAmount,
               int* buff, int* row, int* col,
               vector<vector<string>>& matrixSequence,
               vector<int>& listRewardsAmount,
               vector<int>& listSequenceLength,
               int* minSequenceLength) {
    int amountToken;
    int maxSequenceLength;
    vector<string> listToken;
    string token;
    cout << "Masukkan banyak jenis token: ";
    cin >> amountToken;
    cout << "Masukkan jenis-jenis token: ";
    for (int i = 0; i < amountToken; i++) {
        cin >> token;
        listToken.push_back(token);
    }
    cout << "Masukkan panjang buffer maksimal: ";
    cin >> *buff;
    *minSequenceLength = *buff;

    cout << "Masukkan panjang baris dan kolom dari matriks code: " << endl;
    cout << "Baris: ";
    cin >> *row;
    cout << "Kolom: ";
    cin >> *col;

    cout << "Masukkan banyaknya sequence berhadiah: ";
    cin >> *SequenceAmount;
    cout << "Masukkan panjang maksimal sequence berhadiah: ";
    cin >> maxSequenceLength;

    cout << endl << "Hasil matriks acak: " << endl;

    random_device rd;
    mt19937 eng(rd());

    for (int i = 0; i < *row; i++) {
        vector<string> rowMatrixCode;
        for (int j = 0; j < *col; j++) {
            uniform_int_distribution<int> distr(0, amountToken - 1);
            int index = distr(eng);
            token = listToken[index];
            cout << token << " ";
            rowMatrixCode.push_back(token);
        }
        cout << endl;
        matrixCode.push_back(rowMatrixCode);
    }
}
```

```

cout << endl << "Sequence acak dan reward-nya: " << endl;
for (int i = 0; i < *SequenceAmount; i++) {
    uniform_int_distribution<int> distr(2, maxSequenceLength);
    int sequenceLength = distr(eng);
    if (sequenceLength < *minSequenceLength) {
        *minSequenceLength = sequenceLength;
    }
    bool found = false;

    listSequenceLength.push_back(sequenceLength);

    vector<string> rowMatrixSequence;
    while (found || rowMatrixSequence.empty()) {
        rowMatrixSequence.clear();
        for (int j = 0; j < sequenceLength; j++) {
            uniform_int_distribution<int> distr2(0, amountToken - 1);
            token = listToken[distr2(eng)];
            rowMatrixSequence.push_back(token);
        }
        found = isVectorInMatrix(rowMatrixSequence, matrixSequence, i);
    }
    for (int j = 0; j < sequenceLength; j++) {
        cout << rowMatrixSequence[j] << " ";
    }
    cout << endl;

    uniform_int_distribution<int> distr3(0, 50);
    int rewardAmount = distr3(eng);
    cout << rewardAmount << endl;

    matrixSequence.push_back(rowMatrixSequence);
    listRewardsAmount.push_back(rewardAmount);
}
}

```

#### 4. Fungsi utama (*main function*)

```
int main() {

    // deklarasi variabel
    string line;
    int i = 1;
    int buff;
    int row;
    int col;
    vector<vector<string>> matrixCode;
    int SequenceAmount;
    vector<vector<string>> matrixSequence;
    vector<int> listRewardsAmount;
    vector<int> listSequenceLength;
    int minSequenceLength;
    int input;

    cout << "Pilih input yang ingin digunakan (1 atau 2):" << endl << "1. File" << endl << "2. Randomize" << endl;
    cin >> input;

    if (input == 1) {
        string filename;

        cout << "Masukkan nama file : ";
        cin >> filename;
        ifstream inputfile (filename);

        while (!inputfile.is_open()) {
            cout << "Nama file tidak valid!" << endl;
            cout << "Masukkan Nama file : ";
            cin >> filename;
            inputfile.open(filename);
        }
    }
```

```
    while (getline(inputfile, line)) {
        if (i == 1) {
            buff = stoi(line);
            minSequenceLength = buff;
        } else if (i == 2) {
            row = line[0] - '0';
            col = line[2] - '0';
        } else if (i < row + 3) {
            vector<string> rowMatrix;
            int c = 0;
            int indexRow = 0;

            while (indexRow < col) {
                rowMatrix.push_back(line.substr(c,2));
                c += 3;
                indexRow++;
            }

            matrixCode.push_back(rowMatrix);
        } else if (i == row + 3) {
            SequenceAmount = stoi(line);
        } else {
            vector<string> rowMatrixtRewards;
            int count = 0;
            if ((i - (row + 3)) % 2 == 1) {
                int c = 0;
                int length = line.length();
                if (length < minSequenceLength) {
                    minSequenceLength = length;
                }
                while(c < length){
                    rowMatrixtRewards.push_back(line.substr(c,2));
                    c += 3;
                }
                matrixSequence.push_back(rowMatrixtRewards);
                listSequenceLength.push_back((length + 1) / 3);
            }
        }
        i++;
    }
```



```

    } else {
        listRewardsAmount.push_back(stoi(line));
    }
}
i++;
}
inputfile.close();
} else {
    randomize(matrixCode,
              &SequenceAmount,
              &buff, &row, &col,
              matrixSequence,
              listRewardsAmount,
              listSequenceLength,
              &minSequenceLength);
}

cout << endl << "Hasil: " << endl;

auto start = chrono :: high_resolution_clock :: now();
vector<string> arrayCode;
vector<string> arrayCode2;
vector<string> arrayCodeFinal;
vector<int> arrayDigit;
vector<int> arrayDigit2;
vector<int> arrayDigitFinal;
int indexStartCol;
int maxbuff;
int max = 0;
int value;

for (int i = 0; i < col; i++) {
    for (int currentCodeLength = minSequenceLength; currentCodeLength <= buff; currentCodeLength++) {
        value = 0;
        arrayDigit.clear();
        arrayCode.clear();
    }
}

```

```

        generateCode(sequenceAmount,
                    arrayCode2,
                    arrayDigit2,
                    arrayDigit, matrixCode,
                    matrixSequence,
                    listRewardsAmount,
                    listSequenceLength, true,
                    currentCodeLength, currentCodeLength,
                    row, col, 0, i, arrayCode, &value);
    if (value > max) {
        max = value;
        indexStartCol = i;
        maxbuff = currentCodeLength;
        arrayDigitFinal.clear();
        arrayCodeFinal.clear();
        for (int j = 0; j < currentCodeLength; j++) {
            arrayDigitFinal.push_back(arrayDigit2[j]);
            arrayCodeFinal.push_back(arrayCode2[j]);
        }
    }
}

}

auto end = chrono :: high_resolution_clock :: now();
auto duration = chrono :: duration_cast <chrono :: milliseconds> (end - start);
int x = 1;
int y = indexStartCol + 1;
cout << "Hadiah maksimal: " << max << endl;

cout << "Code paling optimal: ";
if (arrayDigitFinal.size() != 0) {
    for (int i = 0; i < maxbuff; i++) {
        cout << arrayCodeFinal[i] << " ";
    }
}
cout << endl;
cout << "Jejak posisi token pada code (row, col): " << endl;
cout << x << ", " << y << endl;

```

```

    for (int i = 0; i < maxbuff - 1; i++) {
        if (i % 2 == 0) {
            x = arrayDigitFinal[i] + 1;
        } else {
            y = arrayDigitFinal[i] + 1;
        }
        cout << x << ", " << y << endl;
    }
}
cout << endl;
cout << duration.count() << "ms" << endl;
cout << "Apakah ingin menyimpan solusi? (y/n)" << endl;

char isSave;
cin >> isSave;
if (isSave == 'y') {
    string filename;
    cout << "Masukkan nama file: ";
    cin >> filename;
    ofstream outputFile(filename);

    while (!outputFile.is_open()) {
        cout << "Nama atau path file tidak valid!" << endl;
        cout << "Masukkan Nama file : ";
        cin >> filename;
        outputFile.open(filename);
    }
    outputFile << max;
    outputFile << endl;
    for (int o = 0; o < maxbuff; o++) {
        outputFile << arrayCodeFinal[o] << " ";
    }
    outputFile << endl;

    if (arrayDigitFinal.size() != 0) {
        outputFile << 1 << ", " << y << endl;
        for (int o = 0; o < maxbuff - 1; o++) {

```

```

            if (o % 2 == 0) {
                x = arrayDigitFinal[o] + 1;
            } else {
                y = arrayDigitFinal[o] + 1;
            }
            outputFile << x << ", " << y << endl;
        }
    }
    outputFile << endl;
    outputFile << duration.count() << "ms";
    outputFile.close();
    cout << "Data berhasil tertulis ke file." << endl;
}
}

```

# 1. Masukan dari file 1

```
Pilih input yang ingin digunakan (1 atau 2):
1. File
2. Randomize
1
Masukkan nama file : input1.txt

Hasil:
Hadiah maksimal: 102
Code paling optimal: 55 1C 55 BD 7A E9
Jejak posisi token pada code (row, col):
1, 1
2, 1
2, 3
4, 3
4, 4
3, 4

285ms
Apakah ingin menyimpan solusi? (y/n)
y
Masukkan nama file: ../test/test1.txt
Data berhasil tertulis ke file.
```

File *input*:

```
1 7
2 6 6
3 55 55 1C 55 E9 1C
4 1C BD 55 55 7A E9
5 7A 1C 7A E9 BD 55
6 BD 7A BD 7A 55 1C
7 7A 7A 1C 7A E9 55
8 55 BD 55 55 1C 1C
9 3
10 BD 7A
11 32
12 1C 55
13 29
14 7A E9
15 41
```

File *output* :

```
1 102
2 55 1C 55 BD 7A E9
3 1, 4
4 2, 4
5 2, 3
6 4, 3
7 4, 4
8 3, 4
9
10 285ms
```

## 2. Masukan dari file 2

```
Pilih input yang ingin digunakan (1 atau 2):
1. File
2. Randomize
1
Masukkan nama file : input2.txt

Hasil:
Hadiah maksimal: 106
Code paling optimal: 1C 55 BD 7A 1C 7A E9
Jejak posisi token pada code (row, col):
1, 1
5, 1
5, 2
6, 2
6, 3
4, 3
4, 4

47196ms
Apakah ingin menyimpan solusi? (y/n)
y
Masukkan nama file: ../test/test2.txt
Data berhasil tertulis ke file.
```

File *input*

```
1 10
2 6 6
3 1C BD 55 55 7A E9
4 BD 7A BD 7A 55 1C
5 55 55 1C 55 7A 1C
6 7A 1C 7A E9 BD 55
7 55 BD 55 BD 1C 1C
8 7A 7A 1C 7A E9 55
9 3
10 BD 7A 1C
11 50
12 1C 55
13 33
14 7A E9
15 23
```

File *output*

```
1 106
2 1C 55 BD 7A 1C 7A E9
3 1, 4
4 5, 4
5 5, 2
6 6, 2
7 6, 3
8 4, 3
9 4, 4
10
11 47196ms
```

## 3. Masukan dari file 3

Pilih input yang ingin digunakan (1 atau 2):

1. File
2. Randomize

1

Masukkan nama file : input3.txt

Hasil:

Hadiah maksimal: 50

Code paling optimal: 7A BD 7A BD 1C BD 55

Jejak posisi token pada code (row, col):

1, 1

4, 1

4, 3

5, 3

5, 6

3, 6

3, 1

259ms

Apakah ingin menyimpan solusi? (y/n)

y

Masukkan nama file: ../test/test3.txt

Data berhasil tertulis ke file.

File *input*

```
1 7
2 6 6
3 7A 55 E9 E9 1C 55
4 55 7A 1C 7A E9 55
5 55 1C 1C 55 E9 BD
6 BD 1C 7A 1C 55 BD
7 BD 55 BD 7A 1C 1C
8 1C 55 55 7A 55 7A
9 3
10 BD E9 1C
11 15
12 BD 7A BD
13 20
14 BD 1C BD 55
15 30
```

File *output*

```
1 50
2 7A BD 7A BD 1C BD 55
3 1, 1
4 4, 1
5 4, 3
6 5, 3
7 5, 6
8 3, 6
9 3, 1
10
11 259ms
```

4. Masukan dari *keyboard* 1

```

Pilih input yang ingin digunakan (1 atau 2):
1. File
2. Randomize
2
Masukkan banyak jenis token: 3
Masukkan jenis-jenis token: AA BB 8U
Masukkan panjang buffer maksimal: 6
Masukkan panjang baris dan kolom dari matriks code:
Baris: 5
Kolom: 6
Masukkan banyaknya sequence berhadiah: 3
Masukkan panjang maksimal sequence berhadiah: 4

Hasil matriks acak:
8U BB 8U AA BB BB
8U AA BB 8U BB 8U
AA AA BB 8U AA 8U
AA BB BB 8U 8U AA
8U 8U 8U BB AA AA

Sequence acak dan reward-nya:
AA AA
32
AA BB BB
44
BB BB 8U
10

Hasil:
Hadiah maksimal: 76
Code paling optimal: 8U AA AA AA BB BB
Jejak posisi token pada code (row, col):
1, 1
3, 1
3, 2
2, 2
2, 3
3, 3

28ms
Apakah ingin menyimpan solusi? (y/n)
y
Masukkan nama file: ../test/test4.txt
Data berhasil tertulis ke file.

```

```

1 76
2 8U AA AA AA BB BB
3 1, 3
4 3, 3
5 3, 2
6 2, 2
7 2, 3
8 3, 3
9
10 28ms

```

5. Masukan dari *keyboard* 2

```
Pilih input yang ingin digunakan (1 atau 2):
1. File
2. Randomize
2
Masukkan banyak jenis token: 5
Masukkan jenis-jenis token: AA 88 I0 I0 JJ
Masukkan panjang buffer maksimal: 7
Masukkan panjang baris dan kolom dari matriks code:
Baris: 10
Kolom: 9
Masukkan banyaknya sequence berhadiah: 4
Masukkan panjang maksimal sequence berhadiah: 5
```

```
Hasil matriks acak:
I0 I0 I0 I0 JJ JJ 88 I0 JJ
AA 88 I0 I0 AA JJ 88 I0 88
I0 AA I0 I0 AA 88 I0 JJ 88
I0 AA JJ AA I0 I0 AA JJ I0
I0 I0 88 AA I0 I0 88 JJ AA
I0 I0 I0 I0 AA I0 I0 I0 I0
I0 I0 88 88 I0 I0 I0 I0 AA
JJ I0 88 JJ 88 I0 JJ 88 88
I0 I0 I0 JJ I0 I0 88 JJ JJ
I0 JJ I0 I0 I0 JJ I0 88 AA
```

```
Sequence acak dan reward-nya:
```

```
JJ 88 AA
3
I0 AA
48
AA AA JJ I0
4
I0 JJ AA
47
```

```
Hasil:
Hadiah maksimal: 95
Code paling optimal: I0 I0 AA I0 JJ AA
Jejak posisi token pada code (row, col):
1, 1
3, 1
3, 2
8, 2
8, 1
2, 1
```

```
23222ms
```

```
Apakah ingin menyimpan solusi? (y/n)
y
Masukkan nama file: ../test/test5.txt
Data berhasil tertulis ke file.
```



```

1 95
2 I0 IO AA IO JJ AA
3 1, 1
4 3, 1
5 3, 2
6 8, 2
7 8, 1
8 2, 1
9
10 23222ms

```

6. Masukan dari *keyboard* 3

```

Pilih input yang ingin digunakan (1 atau 2):
1. File
2. Randomize
2
Masukkan banyak jenis token: 2
Masukkan jenis-jenis token: AA BB
Masukkan panjang buffer maksimal: 2
Masukkan panjang baris dan kolom dari matriks code:
Baris: 1
Kolom: 2
Masukkan banyaknya sequence berhadiah: 1
Masukkan panjang maksimal sequence berhadiah: 2

Hasil matriks acak:
AA BB

Sequence acak dan reward-nya:
AA AA
18

Hasil:
Hadiah maksimal: 0
Code paling optimal:
0ms
Apakah ingin menyimpan solusi? (y/n)
y
Masukkan nama file: ../test/test6.txt
Data berhasil tertulis ke file.

```

```

1 0
2
3
4 0ms

```

## Bab 4

Link repo github: <https://github.com/b33rk/STIMA>