LAPORAN TUGAS KECIL 1

IF2211 STRATEGI ALGORITMA

Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force

Disusun oleh:

Naufal Adnan 13522116

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2024

DAFTAR ISI

BAB I
Algoritma Brute Force

1.1 Algoritma Brute Force

Algoritma brute force merupakan sebuah algoritma dengan pendekatan yang lempang (*straightforward*) untuk memecahkan suatu persoalan. Pendekatan yang dilakukan biasanya didasarkan pada pernyataan persoalan (*problem statement*) dan definisi atau konsep yang dilibatkan. Algoritma ini memiliki karakteristik memecahkan persoalan dengan sangat sederhana, langsung, jelas, dan intuitif secara apa adanya (*just solve it*). Hal tersebut membuat pendekatan dengan algoritma brute force mudah dimengerti dan dapat diterapkan untuk memecahkan hampir sebagian besar masalah jika solusi tersebut ada. Namun, di sisi lain algoritma brute force seringkali melibatkan penelusuran semua kemungkinan solusi sebelum akhirnya dievaluasi untuk mengambil solusi terbaik. Oleh karena itu, algoritma brute force jarang menghasilkan algoritma yang mangkus dan dinilai lambat untuk masukan berukuran besar. [1]

1.2 Cyberpunk 2077 Breach Protocol

Cyberpunk 2077 Breach Protocol merupakan sebuah *minigame* yang berisi simulasi peretasan jaringan local dari *ICE* (*Intrusion Countermeasures Electronics*) pada permainan Cyberpunk 2077. Komponen pada permainan ini terdiri dari token berupa dua karakter alfanumerik, matrix dari token-token, sekuens, dan buffer untuk menyusun token. Setiap sekuens memiliki panjang minimal berupa dua token dan memiliki bobot hadiah yang variatif. Permainan dimulai dengan pemain memilih satu token pada posisi baris paling atas dari matriks. Selanjutnya pemain bergerak dengan pola vertikal-horizontal secara bergantian. Sekuens dicocokkan pada token-token yang berada di buffer, di mana satu buffer dapat digunakan pada lebih dari satu sekuens. Gerak pemain dapat diakhiri hingga semua sekuens berhasil dicocokkan atau buffer penuh.

1.3 Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force

Algoritma brute force dapat digunakan untuk memecahkan persoalan pada permainan Cyberpunk 2077 Breach Protocol. Algoritma brute force untuk pencarian diimplementasikan melalui pendekatan konsep *DFS* (*deep-first search*), dengan melakukan *backtracking* untuk menemukan semua kemungkinan gerak di dalam matriks secara rekursif. Langkah-langkah di dalam menyelesaikan persoalan ini adalah sebagai berikut.

1. Ketika mengunjungi salah satu koordinat pada sel matriks, tambahkan isi elemen matriks token ke dalam buffer, tambahkan koordinat sel matriks ke dalam list koordinat sementara, dan tandai isi sel matriks dengan sebuah matriks flag yang bernilai true yang berarti sudah pernah dikunjungi. Sesuai dengan aturan, maka titik permulaan untuk penelusuran yakni elemen matriks pada koordinat (1, 1) atau baris paling atas matriks dimulai dari yang paling kiri.

2. Cek isi buffer dengan sekuens untuk menghitung reward. Lakukan iterasi sebanyak jumlah sekuens untuk melakukan pencocokan apakah token pada buffer mengandung sekuens. Jika terdapat sekuens yang cocok tambahkan *reward* sesuai dengan bobot hadiah sekuens. Jika tidak maka akan berlanjut ke pencocokan sekuens berikutnya hingga seluruh sekuens telah dilakukan proses pencocokan.
3. Setelah mendapat nilai *reward*, lakukan evaluasi dari hasil perolehan *reward*. Jika perolehan *reward* saat ini lebih besar dari pada jumlah *reward* pada penelusuran solusi sebelumnya (lebih besar dari 0 untuk permulaan penelusuran) atau jika perolehan *reward* pada penelusuran saat ini sama dengan *reward* pada penelusuran solusi sebelumnya namun list koordinat sementara sekarang lebih optimal (lebih sedikit langkah) daripada solusi sebelumnya, maka tukar nilai *reward* solusi ke nilai *reward* saat ini dan tukar koordinat solusi dengan koordinat sementara saat ini.
4. Langkah berikutnya adalah cek apakah buffer sudah penuh atau belum. Jika buffer belum penuh, maka akan dicek apakah arah gerak berikutnya adalah vertikal atau horizontal. Jika gerak berikutnya adalah vertikal, maka akan dilakukan traversal pada kolom di mana sel matriks dikunjungi saat ini. Sebaliknya, jika gerak berikutnya adalah horizontal, maka traversal dilakukan pada baris di mana sel matriks dikunjungi saat ini. Pada setiap traversal dilakukan pengecekan apakah sel yang akan dikunjungi selanjutnya sudah pernah dikunjungi sebelumnya. Jika belum, maka secara rekursif akan dilakukan penelusuran kembali mulai dari seperti langkah nomor satu. Penelusuran secara rekursif akan berhenti hingga mencapai basis yaitu ketika buffer telah penuh.
5. Sementara itu, jika buffer sudah penuh, maka akan dilakukan pengosongan pada list koordinat sementara dan set kembali matriks flag ke kondisi false yang berarti belum pernah dikunjungi seluruhnya.
6. Proses penelusuran akan diulangi kembali, namun dengan titik permulaan bergeser ke koordinat (1, 2) atau berlanjut dengan traversal ke kolom berikutnya (sebelah kanannya) di baris paling atas hingga selesai, yaitu seluruh kemungkinan pola yang mungkin dapat terbentuk telah seluruhnya dievaluasi.

Secara garis besar penyelesaian Cyberpunk 2077 Breach Protocol dengan algoritma brute force dapat dilakukan dengan melakukan penelusuran setiap kemungkinan solusi dengan cara yang sistematis, kemudian evaluasi setiap kemungkinan solusi dengan cara membandingkannya dengan solusi sebelumnya, dan bila pencarian berakhir maka solusi optimal telah diperoleh.

BAB II
Implementasi

2.1 Bahasa dan Library

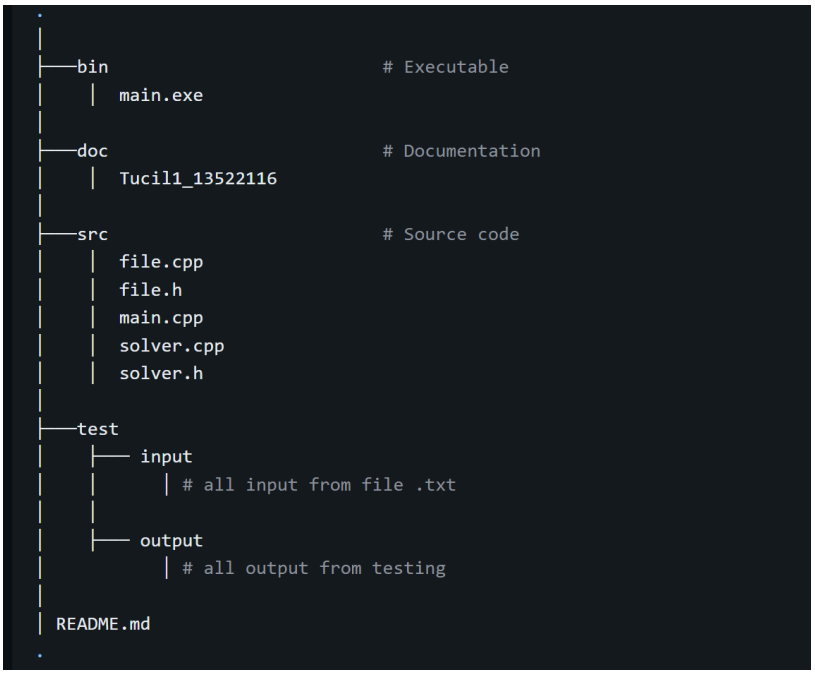
Program diimplementasikan dalam bahasa C++ dengan menggunakan library sebagai berikut.

1. iostream
2. string
3. fstream
4. vector
5. tuple
6. algorithm
7. ctime
8. cstdlib
9. chrono

Digunakan juga *using namespace std;* untuk mempermudah penulisan implementasi program.

2.2 Struktur Program

Struktur program dibuat sedemikian rupa terdiri dari 4 folder. Folder bin berisi *executable file*, folder doc berisi laporan, folder src berisi *source code*, folder test berisi folder input untuk masukan dari file dan folder output untuk menyimpan hasil solusi, serta README.md. Pada folder src terdapat dua file header yaitu file.h untuk file.cpp yang menangani proses input dari file serta penyimpanan dalam bentuk file txt, dan solver.h untuk solver.cpp sebagai pembantu dalam proses pencarian dan mengevaluasi solusi optimal. Lalu program utama terdapat pada file main.cpp.



2.3 Source Code

2.3.1 File file.hpp

```
#ifndef __FILE_H
#define __FILE_H

#include <iostream>
#include <string>
#include <fstream>
#include "solver.hpp"
using namespace std;

class File {
    public:
        ifstream in;
        ofstream out;
        string fileName;

        // konstruktor
        File(string fileName);

        // load & save
        string readFileName();
        Solver fromFile();
        void saveToFile(Solver solve, int duration);
};

#endif
```

2.3.2 File file.cpp

```cpp
#include "file.hpp"
using namespace std;

File::File(string fileName) {
    File::fileName = fileName;
}

string File::readFileName() {
    string fileName;
    do {
        cout << ">> Masukkan nama file: ";
        cin >> fileName;
        in.open("test/input/" + fileName);
        if (!in.is_open()) {
            cerr << "Error: Tidak ada file dengan nama test/input/" << fileName << "!" << endl;
        }
        cout << "\n";
    } while (!in.is_open());
    in.close();
    return "test/input/" + fileName;
}
```

```cpp
Solver File::fromFile() {
    int buffsize, row, col, countSeq, reward;
    string token;
    in.open(fileName);
    in >> buffsize >> row >> col;
    Solver solve(buffsize, row, col);

    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++) {
            in >> token;
            solve.setElmt(i, j, token);
        }
    }

    in >> countSeq;
    in.ignore();
    for (int i = 0; i < countSeq; i++) {
        getline(in, token);
        in >> reward;
        solve.addSequence(token);
        solve.addReward(reward);
        in.ignore();
    }
    in.close();
    return solve;
}
```

```
void File::saveToFile(Solver solve, int duration) {
    string input;
    cout << "Masukkan nama file: "; cin >> input;
    out.open("test/output/" + input);
    if (!out.is_open()) {
        cout << "Error: Gagal membuat file!" << endl;
        return;
    }

    out << solve.point << endl;
    for (int i = 0; i < solve.coordinate.size(); i++) {
        int x = get<0>(solve.coordinate[i]);
        int y = get<1>(solve.coordinate[i]);
        if (i == solve.coordinate.size() - 1) {
            out << solve.getElmt(x, y);
        }
        else {
            out << solve.getElmt(x, y) << " ";
        }
    }

    out << endl;
    for (int i = 0; i < solve.coordinate.size(); i++) {
        int x = get<0>(solve.coordinate[i]);
        int y = get<1>(solve.coordinate[i]);
        out << y + 1 << ", " << x + 1 << endl;
    }
    out << endl;
    out << duration << " ms";
    out.close();
    cout << "File berhasil disimpan di test/output/" << input << endl;
}
```

2.3.3 File solver.hpp

```
#ifndef __SOLVER_H
#define __SOLVER_H

#include <iostream>
#include <string>
#include <algorithm>
#include <ctime>
#include <cstdlib>
#include <vector>
#include <tuple>
using namespace std;

class Solver {
    public:
        int point;
        int buffer;
        vector<int> reward;
        vector<string> sequence;
        vector<tuple<int, int>> coordinate;
        vector<tuple<int, int>> currCoord;
        vector<vector<string>> matrix;
        vector<vector<bool>> flag;

        // konstruktor
        Solver(int buffsize, int row, int col);

        // setter
        void setElmt(int x, int y, string val);

        // getter
        string getElmt(int x, int y);
        int getRowLength();
        int getColLength();

        // output
        void displayMatrixToken();
        void displayCoordinate();
        void displayBuffer();
        void displayRewardSequence();

        // operasi lainnya
        void addReward(int val);
```

```
        void addSequence(string val);
        int countReward(string val);
        void search(int row, int col, string currToken,
                    bool vertikal, int step);
};

#endif
```

2.3.4 File solver.cpp

```
#include "solver.hpp"
using namespace std;

Solver::Solver(int buffsize, int row, int col) {
    point = 0;
    buffer = buffsize;
    matrix = vector<vector<string>>(row, vector<string>(col));
    flag = vector<vector<bool>>(row, vector<bool>(col, false));
}

void Solver::setElmt(int x, int y, string val) {
    matrix[x][y] = val;
}

string Solver::getElmt(int x, int y) {
    return matrix[x][y];
}

int Solver::getRowLength() {
    return matrix.size();
}

int Solver::getColLength() {
    return matrix[0].size();
}

void Solver::displayMatrixToken() {
    cout << "--Matrix Token--" << endl;
    for (int i = 0; i < getRowLength(); i++) {
        for (int j = 0; j < getColLength(); j++) {
            cout << matrix[i][j] << " ";
        }
        cout << endl;
```

```
    } cout << endl;
}

void Solver::displayCoordinate() {
    cout << "\nKoordinat:\n";
    for (int i = 0; i < coordinate.size(); i++) {
        int x = get<0>(coordinate[i]);
        int y = get<1>(coordinate[i]);
        cout << y + 1 << ", " << x + 1 << endl;
    } cout << endl;
}

void Solver::displayBuffer() {
    cout << "Ukuran buffer: " << buffer << " [ ";
    for (int i = 0; i < buffer; i++) {
        cout << "_ ";
    } cout << "]\n" << endl;
}
```

```
void Solver::displayRewardSequence() {
    cout << "Jumlah sekuens: " << sequence.size() << endl;
    for (int i = 0; i < reward.size(); i++) {
        cout << i + 1 << ". Reward: " << reward[i]<< " - sekuens: " << sequence[i] << endl;
        sequence[i].erase(remove_if(sequence[i].begin(), sequence[i].end(), ::isspace), sequence[i].end());
    }
}

void Solver::addReward(int val) {
    reward.push_back(val);
}

void Solver::addSequence(string val) {
    sequence.push_back(val);
}

int Solver::countReward(string val) {
    int res = 0;
    for (int i = 0; i < sequence.size(); i++) {
        if (val.find(sequence[i]) != string::npos) {
            res += reward[i];
        }
    }
    return res;
}
```

```
void Solver::search(int row, int col, string currToken, bool vertikal, int step) {
    tuple<int, int> p = make_tuple(row, col);
    currCoord.push_back(p);
    flag[row][col] = true;

    int currReward = countReward(currToken);
    if (currReward > point || (currReward == point && currCoord.size() < coordinate.size())) {
        point = currReward;
        coordinate = currCoord;
    }

    if (step < buffer) {
        if (vertikal) {
            for (int i = 0; i < getRowLength(); i++) {
                if (!flag[i][col]) {
                    search(i, col, currToken + getElmt(i, col), !vertikal, step + 1);
                }
            }
        }
        else {
            for (int i = 0; i < getColLength(); i++) {
                if (!flag[row][i]) {
                    search(row, i, currToken + getElmt(row, i), !vertikal, step + 1);
                }
            }
        }
    }

    currCoord.pop_back();
    flag[row][col] = false;
}
```

2.3.5 File main.cpp

```
#include <chrono>
#include "file.hpp"
#include "solver.hpp"
using namespace std;

void clearScreen() {
    #ifdef _WIN32
    system("cls");
    #else
    system("clear");
    #endif
};

void displayHeader() {
    cout << R"(
   ___      _  _    _                          _ __                          _
  / __|    | || | | |__      ___         _ _   |  _ \   _  _      _ _       | |__
 | (__      \_, | |  _ \    / -_)       |  _|  | .__/ | +| |    |   \     | / /
  \___|    _|__/  |_.__/    \___|      _|_|_   |_|__    \_,_|   |_||_|    |_\_\;
_|"""""|_|"""""|_|"""""|_|"""""|_|"""""|_|"""""|_|"""""|_|"""""|_|"""""|_|"""""|
---0-0---`-0-0---`-0-0---`-0-0---`-0-0---`-0-0---`-0-0---`-0-0---`-0-0---`-0-0--

# ----------WELCOME TO AWKWK-SYSTEM----------- #
# -------Cyberpunk 2077 Breach Protocol------- #
# Temukan solusi dan dapatkan reward maksimal! #
-----------------------------------------------)" << "\n\n";
};

void displayMenu() {
    cout << "Pilih metode masukan:\n1. File\n2. Keyboard & random\n3. Keluar" << endl;
};

int inputChoice(int start, int end) {
    int choice;
    do {
        cout << ">> Masukkan pilihan: "; cin >> choice;
        if (choice < start || choice > end) {
            cout << "Error: Input tidak valid!\n";
        } cout << "\n";
    } while (choice < start || choice > end);
    return choice;
}

bool isStringExists(const vector<string>& list, const string& str) {
    return find(list.begin(), list.end(), str) != list.end();
}

bool isAlphanumeric(const string& str) {
    int i = 0;
    bool flag = true;
    while (i < str.length() && flag) {
        if (!isalnum(str[i])) {
            flag = false;
        }
        i++;
    }
    return flag;
}
```

```
void runGame(int choice, Solver solve) {
    File file("");
    solve.displayMatrixToken();
    solve.displayBuffer();
    solve.displayRewardSequence();

    cout << "\nApakah ingin melihat solusi?\n1. Ya\n2. Tidak" << endl;
    choice = inputChoice(1, 2);
    if (choice == 1) {
        // brute force
        auto start = std::chrono::steady_clock::now();
        for (int i = 0; i < solve.getColLength(); i++) {
            solve.search(0, i, solve.getElmt(0, i), true, 1);
        }
        auto end = std::chrono::steady_clock::now();
        auto duration = std::chrono::duration_cast<std::chrono::milliseconds>(end - start);
        // output solusi
        cout << "---Solusi---\nReward: " << solve.point << "\nBuffer: ";
        for (int i = 0; i < solve.coordinate.size(); i++) {
            int x = get<0>(solve.coordinate[i]);
            int y = get<1>(solve.coordinate[i]);
            cout << solve.getElmt(x, y) << " ";
        }
        solve.displayCoordinate();
        cout << duration.count() << " ms" << endl;
        // save
        cout << "\nApakah ingin menyimpan solusi?\n1. Ya\n2. Tidak" << endl;
        choice = inputChoice(1, 2);
        if (choice == 1) {
            file.saveToFile(solve, duration.count());
        }
    }
    cout << "\nPress enter to continue...";
    cin.ignore();
    cin.get();
}

int main() {
    bool run = true;
    int choice, countToken, countSeq, sizeSeq, randnum, randreward, buffer, row, col;
    string fileName, input;
    vector<string> token;
    File file("");
    Solver solve(0, 0, 0);
    while (run)
    {
        clearScreen();
        displayHeader();
        displayMenu();
        choice = inputChoice(1, 3);
```

```
switch (choice)
{
case 1:
    clearScreen();
    displayHeader();
    fileName = file.readFileName();
    file = File(fileName);
    solve = file.fromFile();
    runGame(choice, solve);
    break;

case 2:
    clearScreen();
    displayHeader();

    do {
        cout << "Masukkan jumlah token unik: "; cin >> countToken;
        if (countToken < 1) {
            cout << "Error: Jumlah token tidak boleh kosong!"<< endl;
        }
    } while (countToken < 1);

    cout << endl << "Masukkan token unik" << endl;
    token.resize(countToken);
    for (int i = 0; i < countToken; i++) {
        do {
            cout << "Token unik ke-" << i + 1 << ": "; cin >> input;
            if (input.length() != 2 || !isAlphanumeric(input)) {
                cout << "Error: Token harus terdiri dari 2 karakter alfanumerik!\nSilakan masukkan kembali." << endl;
            }
            else if (isStringExists(token, input)) {
                cout << "Error: Token sudah ada, pastikan token unik!\nSilakan masukkan kembali." << endl;
            }
        } while (input.length() != 2 || isStringExists(token, input) || !isAlphanumeric(input));
        token[i] = input;
    }

    cout << "\nMasukkan ukuran buffer: "; cin >> buffer;
    cout << "Masukkan baris matriks: "; cin >> row;
    cout << "Masukkan kolom matriks: "; cin >> col;
    solve = Solver(buffer, row, col);
    srand(time(nullptr));
    for (int i = 0; i < solve.getRowLength(); i++) {
        for (int j = 0; j < solve.getColLength(); j++) {
            int randomIndex = rand() % token.size();
            solve.setElmt(i, j, token[randomIndex]);
        }
    }

    do {
        cout << "Masukkan jumlah sekuens: "; cin >> countSeq;
        if (countSeq < 1) {
            cout << "Error: Jumlah sekuens tidak boleh kosong!"<< endl;
        }
    } while (countSeq < 1);
```

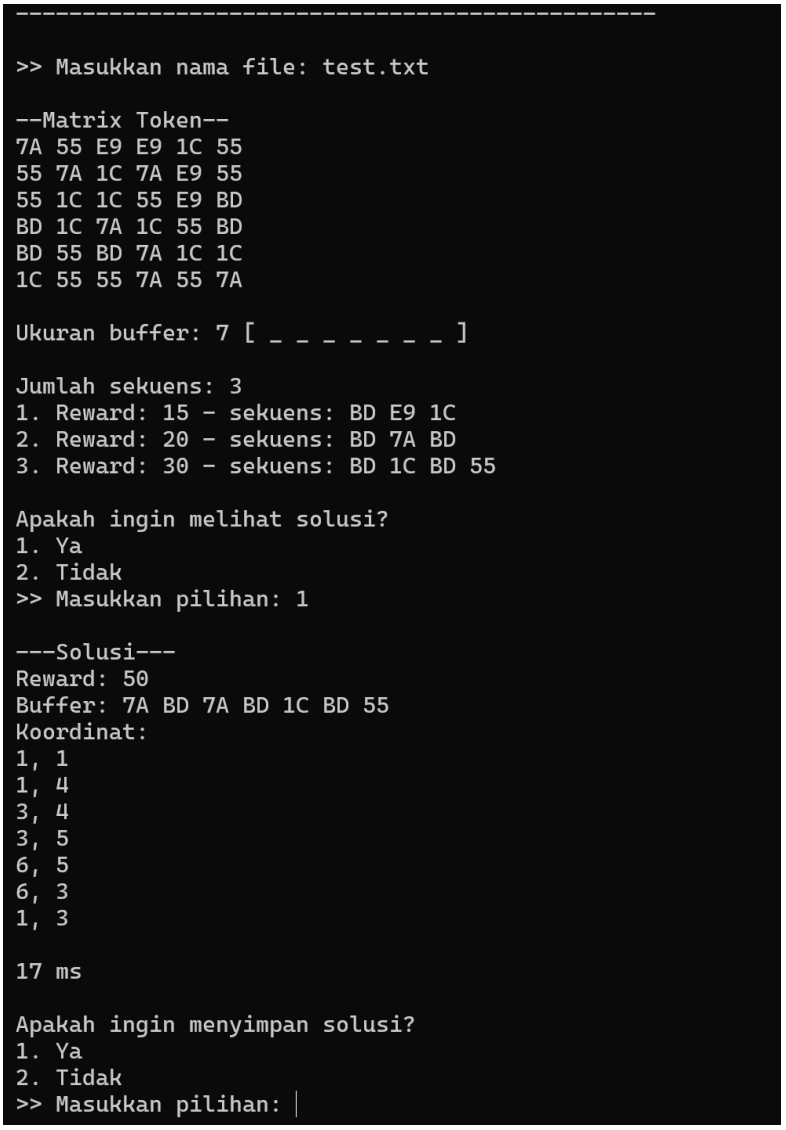
```
            do {
                cout << "Masukkan ukuran maksimal sekuens: "; cin >> sizeSeq; cout << endl;
                if (sizeSeq < 2) {
                    cout << "Error: Ukuran minimal sekuens adalah 2!"<< endl;
                }
            } while (sizeSeq < 2);

            for (int i = 0; i < countSeq; i++) {
                do {
                    randnum = rand() % (sizeSeq - 1) + 2;
                    input = "";
                    for (int j = 0; j < randnum; j++) {
                        int randomIndex = rand() % token.size();
                        input = input + " " + token[randomIndex];
                    }
                } while (isStringExists(solve.sequence, input));
                solve.addSequence(input);
                randreward = rand() % 41 + 10;
                solve.addReward(randreward);
            }
            runGame(choice, solve);
            break;

        case 3:
            run = false;

        default:
            break;;
        }
    }
}
```

BAB III

Eksperimen

3.1 Input dari File

3.1.1 Test



3.1.2 Test 1

```
------------------------------------------------

>> Masukkan nama file: test1.txt

--Matrix Token--
88 X1 F4 X1 88
6C X1 6C F4 NY
X1 NY NY NY 88
0K 0K 6C NY 0K

Ukuran buffer: 7 [ _ _ _ _ _ _ _ ]

Jumlah sekuens: 7
1. Reward: 45 - sekuens: 0K NY 88
2. Reward: 18 - sekuens: 0K NY
3. Reward: 45 - sekuens: 88 88 X1
4. Reward: 10 - sekuens: 0K 88
5. Reward: 29 - sekuens: 6C NY NY
6. Reward: 27 - sekuens: 0K F4
7. Reward: 36 - sekuens: 88 6C

Apakah ingin melihat solusi?
1. Ya
2. Tidak
>> Masukkan pilihan: 1

---Solusi---
Reward: 108
Buffer: 88 0K 0K NY 88 88 X1
Koordinat:
1, 1
1, 4
2, 4
2, 3
5, 3
5, 1
2, 1

2 ms

Apakah ingin menyimpan solusi?
1. Ya
2. Tidak
>> Masukkan pilihan:
```

3.1.3 Test 2

```
------------------------------------------

>> Masukkan nama file: test2.txt

--Matrix Token--
FF N0 N0 N0 DD
7A N0 N0 N0 1C
FF 1C 7A 7A N0
7A DD 7A 7A 7A
1C N0 1C 1C DD

Ukuran buffer: 10 [ _ _ _ _ _ _ _ _ _ _ ]

Jumlah sekuens: 5
1. Reward: 50 - sekuens: N0 FF 7A FF
2. Reward: 45 - sekuens: FF DD
3. Reward: 21 - sekuens: 1C 1C 1C
4. Reward: 30 - sekuens: N0 1C
5. Reward: 16 - sekuens: N0 1C FF FF

Apakah ingin melihat solusi?
1. Ya
2. Tidak
>> Masukkan pilihan: 1

---Solusi---
Reward: 91
Buffer: N0 1C FF FF DD
Koordinat:
2, 1
2, 3
1, 3
1, 1
5, 1

220 ms

Apakah ingin menyimpan solusi?
1. Ya
2. Tidak
>> Masukkan pilihan:
```

3.1.3 Test 3

```
----------------------------------------------

>> Masukkan nama file: test3.txt

--Matrix Token--
N4 A1 NC N4 N4
A1 2D N4 N4 2D
2D 2D N4 4O N4
A1 2D 4O NC 2D
2D 2D 2D 2D NC
N4 2D 2D 40 NC

Ukuran buffer: 8 [ _ _ _ _ _ _ _ _ ]

Jumlah sekuens: 7
1. Reward: 10 - sekuens: NC NC
2. Reward: 16 - sekuens: 4O 2D NC
3. Reward: 49 - sekuens: A1 4O NC
4. Reward: 21 - sekuens: 2D 2D
5. Reward: 28 - sekuens: A1 A1
6. Reward: 15 - sekuens: 2D NC 2D
7. Reward: 32 - sekuens: NC A1

Apakah ingin melihat solusi?
1. Ya
2. Tidak
>> Masukkan pilihan: 1

---Solusi---
Reward: 109
Buffer: N4 N4 A1 A1 4O NC A1
Koordinat:
4, 1
4, 2
1, 2
1, 4
3, 4
3, 1
2, 1

62 ms

Apakah ingin menyimpan solusi?
1. Ya
2. Tidak
>> Masukkan pilihan:
```

3.1.4 Test 4

```
------------------------------------------------

>> Masukkan nama file: test4.txt

--Matrix Token--
1A CT 87 F3 87 1A BD BD 87 1A
CT 1A 1A 1A 1A BD 87 BD F3 CT
1A CT CT F3 BD CT 1A F3 F3 1A
F3 87 F3 1A CT 87 F3 1A 1A 87
CT F3 87 CT F3 87 F3 CT CT 1A
CT BD 1A 1A CT 1A 1A F3 BD 87
87 F3 BD 87 F3 CT BD BD 87 F3
87 BD 1A 1A CT 87 1A BD CT 1A
CT F3 F3 F3 CT BD CT BD CT 1A
F3 CT CT F3 CT F3 CT CT CT CT

Ukuran buffer: 5 [ _ _ _ _ _ ]

Jumlah sekuens: 7
1. Reward: 33 - sekuens: BD F3
2. Reward: 35 - sekuens: CT CT
3. Reward: 29 - sekuens: F3 87
4. Reward: 20 - sekuens: 87 87
5. Reward: 11 - sekuens: CT F3
6. Reward: 27 - sekuens: CT BD
7. Reward: 28 - sekuens: BD BD CT

Apakah ingin melihat solusi?
1. Ya
2. Tidak
>> Masukkan pilihan: 1

---Solusi---
Reward: 124
Buffer: CT CT BD F3 87
Koordinat:
2, 1
2, 3
5, 3
5, 5
3, 5

19 ms

Apakah ingin menyimpan solusi?
1. Ya
2. Tidak
>> Masukkan pilihan:
```

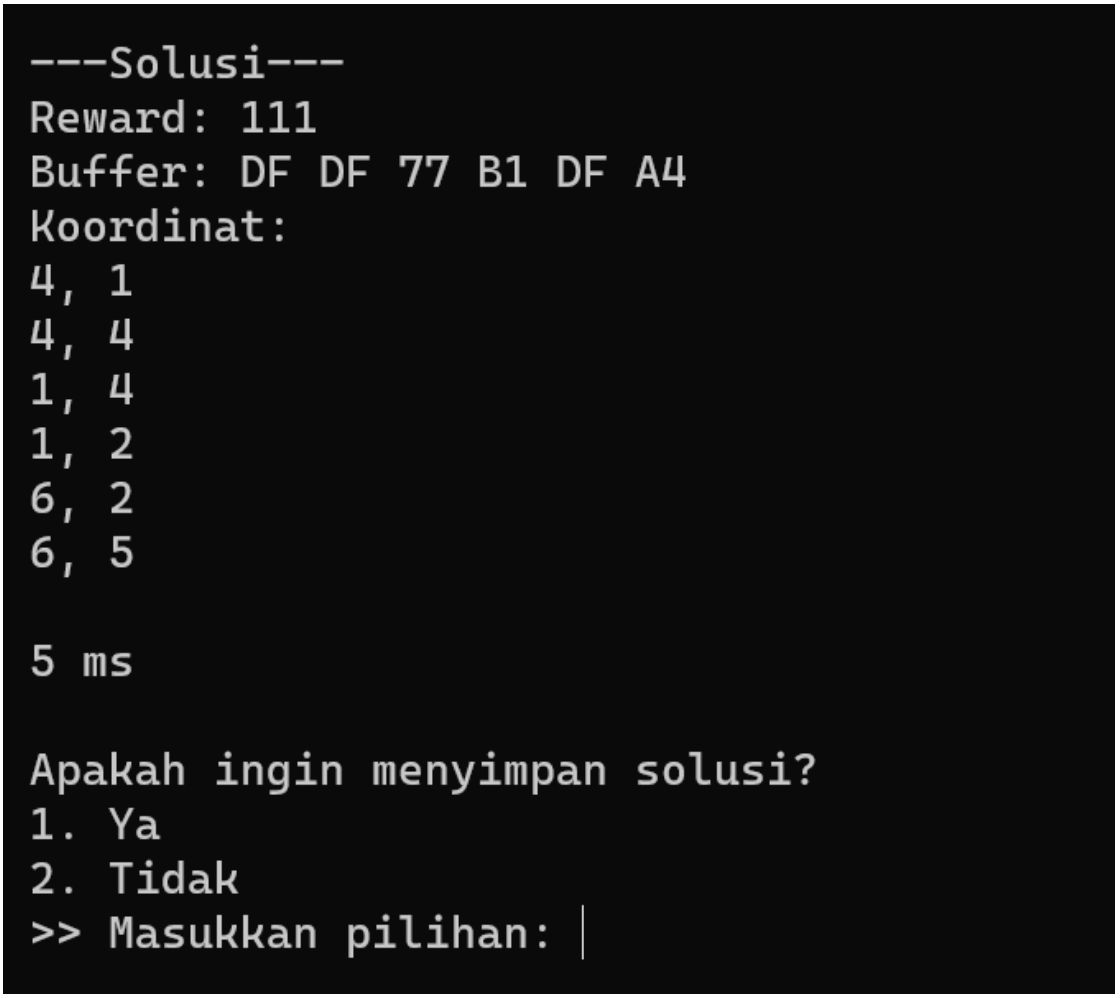
3.2 Generate Random

3.1.5 Test 5

- Input

```
-----------------------------------------------

Masukkan jumlah token unik: 5

Masukkan token unik
Token unik ke-1: d[
Error: Token harus terdiri dari 2 karakter alfanumerik!
Silakan masukkan kembali.
Token unik ke-1: DF
Token unik ke-2: A4
Token unik ke-3: 9J
Token unik ke-4: B1
Token unik ke-5: 77

Masukkan ukuran buffer: 6
Masukkan baris matriks: 6
Masukkan kolom matriks: 6
Masukkan jumlah sekuens: 4
Masukkan ukuran maksimal sekuens: 4

--Matrix Token--
77 9J 9J DF DF 77
B1 B1 A4 A4 DF DF
9J 9J 77 B1 77 77
77 77 B1 DF 9J DF
DF 77 9J A4 9J A4
77 9J 9J A4 77 A4

Ukuran buffer: 6 [ _ _ _ _ _ _ ]

Jumlah sekuens: 4
1. Reward: 48 - sekuens:  B1 DF A4
2. Reward: 28 - sekuens:  77 B1
3. Reward: 21 - sekuens:  B1 B1 B1 9J
4. Reward: 35 - sekuens:  DF DF

Apakah ingin melihat solusi?
1. Ya
2. Tidak
>> Masukkan pilihan:
```

- Output


---Solusi---
Reward: 111
Buffer: DF DF 77 B1 DF A4
Koordinat:
4, 1
4, 4
1, 4
1, 2
6, 2
6, 5
5 ms
Apakah ingin menyimpan solusi?
1. Ya
2. Tidak
>> Masukkan pilihan:

3.1.6 Test 6

- Input

```
Masukkan jumlah token unik: 5

Masukkan token unik
Token unik ke-1: 1A
Token unik ke-2: BD
Token unik ke-3: 55
Token unik ke-4: 7A
Token unik ke-5: E9

Masukkan ukuran buffer: 7
Masukkan baris matriks: 7
Masukkan kolom matriks: 7
Masukkan jumlah sekuens: 3
Masukkan ukuran maksimal sekuens: 4

--Matrix Token--
1A BD BD 55 BD 1A BD
E9 55 BD 55 7A 7A BD
E9 7A E9 55 BD 1A BD
BD BD 7A BD BD 55 7A
E9 BD 55 7A 7A 55 E9
BD 55 E9 1A BD 1A 7A
1A E9 BD 55 BD E9 7A

Ukuran buffer: 7 [ _ _ _ _ _ _ _ ]

Jumlah sekuens: 3
1. Reward: 49 - sekuens:  E9 7A 1A E9
2. Reward: 32 - sekuens:  BD 1A E9
3. Reward: 42 - sekuens:  1A BD BD

Apakah ingin melihat solusi?
1. Ya
2. Tidak
>> Masukkan pilihan:
```

- Output

```
---Solusi---
Reward: 81
Buffer: BD BD 1A E9 7A 1A E9
Koordinat:
3, 1
3, 7
1, 7
1, 2
6, 2
6, 3
1, 3

62 ms

Apakah ingin menyimpan solusi?
1. Ya
2. Tidak
>> Masukkan pilihan:
```

3.1.7 Test 7

- Input

```
------------------------------------------------

Masukkan jumlah token unik: 6

Masukkan token unik
Token unik ke-1: WW
Token unik ke-2: 3R
Token unik ke-3: B5
Token unik ke-4: DF
Token unik ke-5: 99
Token unik ke-6: 1J

Masukkan ukuran buffer: 13
Masukkan baris matriks: 4
Masukkan kolom matriks: 5
Masukkan jumlah sekuens: 7
Masukkan ukuran maksimal sekuens: 3

--Matrix Token--
DF DF 3R 3R 1J
1J 3R 1J B5 99
3R DF WW 99 DF
1J 1J DF 99 B5

Ukuran buffer: 13 [ _ _ _ _ _ _ _ _ _ _ _ _ _ ]

Jumlah sekuens: 7
1. Reward: 48 - sekuens:  WW B5 99
2. Reward: 47 - sekuens:  WW DF 1J
3. Reward: 30 - sekuens:  DF 99 1J
4. Reward: 46 - sekuens:  99 99
5. Reward: 40 - sekuens:  B5 WW DF
6. Reward: 23 - sekuens:  DF 99
7. Reward: 18 - sekuens:  WW 99

Apakah ingin melihat solusi?
1. Ya
2. Tidak
>> Masukkan pilihan: 1
```

- Output

```
---Solusi---
Reward: 146
Buffer: 3R WW DF 1J 99 99 DF 99 1J
Koordinat:
3, 1
3, 3
2, 3
2, 4
4, 4
4, 3
5, 3
5, 2
1, 2

400 ms

Apakah ingin menyimpan solusi?
1. Ya
2. Tidak
>> Masukkan pilihan:
```

3.1.8 Test 8

- Input

```
Masukkan jumlah token unik: 4

Masukkan token unik
Token unik ke-1: HH
Token unik ke-2: 34
Token unik ke-3: D9
Token unik ke-4: 7C

Masukkan ukuran buffer: 10
Masukkan baris matriks: 7
Masukkan kolom matriks: 7
Masukkan jumlah sekuens: 4
Masukkan ukuran maksimal sekuens: 5

--Matrix Token--
34 D9 34 D9 HH HH D9
34 7C 7C D9 7C 34 HH
D9 D9 HH 7C 34 34 D9
34 7C D9 D9 7C D9 7C
34 HH HH 34 HH D9 7C
7C 7C 7C D9 D9 7C 7C
34 HH HH HH D9 D9 D9

Ukuran buffer: 10 [ _ _ _ _ _ _ _ _ _ _ ]

Jumlah sekuens: 4
1. Reward: 30 - sekuens:  HH 34 34 HH
2. Reward: 44 - sekuens:  D9 7C
3. Reward: 14 - sekuens:  HH 34
4. Reward: 50 - sekuens:  7C 34

Apakah ingin melihat solusi?
1. Ya
2. Tidak
>> Masukkan pilihan:
```

- Output

```
---Solusi---
Reward: 138
Buffer: D9 7C 34 HH 34 34 HH
Koordinat:
2, 1
2, 2
6, 2
6, 1
1, 1
1, 2
7, 2

16679 ms

Apakah ingin menyimpan solusi?
1. Ya
2. Tidak
>> Masukkan pilihan:
```

3.3 Test No Solution

```
--Matrix Token--
7A 55 E9 E9 1C 55
55 7A 1C 7A E9 55
55 1C 1C 55 E9 BD
BD 1C 7A 1C 55 BD
BD 55 BD 7A 1C 1C
1C 55 55 7A 55 7A

Ukuran buffer: 7 [ _ _ _ _ _ _ _ ]

Jumlah sekuens: 3
1. Reward: 15 - sekuens: BD SA LA HH
2. Reward: 20 - sekuens: BD GA AD AA
3. Reward: 30 - sekuens: NO SO LU

Apakah ingin melihat solusi?
1. Ya
2. Tidak
>> Masukkan pilihan: 1

---Solusi---
Reward: 0
Buffer:
Koordinat:

20 ms

Apakah ingin menyimpan solusi?
1. Ya
2. Tidak
>> Masukkan pilihan:
```

DAFTAR PUSTAKA

[1] Munir, Rinaldi. 2023. Algoritma Brute Force (Bagian 1). https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf, diakses pada 11 Januari 2024.

LAMPIRAN

Link Repository:

https://github.com/nanthedom/Tucil1_13522116

Check List:

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		✓