

**LAPORAN TUGAS KECIL**  
**IF2211 Strategi Algoritma**  
**Cyberpunk 2077 Breach Protocol dengan Algoritma**  
**Brute Force**



**Disusun oleh:**

**Angelica Kierra Ninta Gurning**

**(13522048)**

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**  
**INSTITUT TEKNOLOGI BANDUNG**  
**2023**

## DAFTAR ISI

DAFTAR ISI.....	1
BAB I.....	2
Deskripsi Persoalan.....	2
BAB II.....	3
Deskripsi Program.....	3
BAB III.....	27
Lampiran.....	27

# BAB I

## Deskripsi Persoalan

**Cyberpunk 2077 Breach Protocol** adalah *minigame* meretas pada permainan video *Cyberpunk 2077*. *Minigame* ini merupakan simulasi peretasan jaringan local dari *ICE (Intrusion Countermeasures Electronics)* pada permainan *Cyberpunk 2077*. Komponen pada permainan ini antara lain adalah:

1. Token – terdiri dari dua karakter alfanumerik seperti E9, BD, dan 55.
2. Matriks – terdiri atas token-token yang akan dipilih untuk menyusun urutan kode.
3. Sekuens – sebuah rangkaian token (dua atau lebih) yang harus dicocokkan.
4. Buffer – jumlah maksimal token yang dapat disusun secara sekuensial.

Aturan permainan Breach Protocol antara lain:

1. Pemain bergerak dengan pola horizontal, vertikal, horizontal, vertikal (bergantian) hingga semua sekuens berhasil dicocokkan atau buffer penuh.
2. Pemain memulai dengan memilih satu token pada posisi baris paling atas dari matriks.
3. Sekuens dicocokkan pada token-token yang berada di buffer.
4. Satu token pada buffer dapat digunakan pada lebih dari satu sekuens.
5. Setiap sekuens memiliki bobot hadiah atau *reward* yang variatif.
6. Sekuens memiliki panjang minimal berupa dua token.

Metode yang digunakan adalah metode Brute Force. Brute force adalah suatu metode penyelesaian masalah atau pencarian solusi dengan mencoba semua kemungkinan secara sistematis. Dalam konteks algoritma, brute force algorithm adalah pendekatan di mana komputer secara kasar dan secara langsung mencoba semua solusi potensial hingga menemukan yang benar atau diinginkan.

## BAB II

### Deskripsi Program

#### 2.1. Algoritma Brute Force

Program akan mencari semua kemungkinan langkah sesuai dengan ukuran buffer yang ditentukan, lalu mencari langkah dengan jumlah hadiah terbanyak. Untuk lebih rinci langkah-langkah adalah sebagai berikut:

1. Program akan membaca masukan/*input* dari pengguna, jika pengguna memasukkan *file* berekstensi .txt, maka program akan membaca *file* tersebut lalu menyimpannya pada tipe data *Input*. Jika pengguna memilih untuk menggunakan CLI, maka matriks dan sequence akan dihasilkan secara acak oleh program
2. Selanjutnya program akan masuk untuk menyelesaikan masalah. Terdapat variabel global “currentPrize” dan “bestSolution” yang akan diganti tiap fungsi berjalan. Fungsi pencarian akan berjalan secara rekursif, untuk mencari setiap kemungkinan yang ada
3. Program akan memulai dari panjang buffer terkecil hingga maksimal panjang buffer. Untuk setiap panjang buffer, program akan mencari setiap kemungkinan yang berasal dari baris pertama dengan cara mencari setiap kemungkinan dari tiap kolom, contoh (1,1), (2,1), ...(m,1)
4. Panjang buffer akan dikurangi secara rekursif pada setiap iterasi, jika sisa langkah 0, maka program akan mencari sekuens apa saja yang ada pada langkah yang ditemukan. Sekuens disimpan pada tipe data *Sequence* yang berisi *list* hadiah dan juga *list* sekuens yang bersangkutan. Suatu langkah akan mendapatkan hadiah jika memiliki sekuens yang bersangkutan. Semua hadiah yang didapatkan suatu langkah akan diakumulasi lalu dibandingkan dengan variabel global “currentPrize”. Jika hadiah langkah lebih besar daripada “currentPrize” maka nilai “currentPrize” akan diganti dengan nilai hadiah langkah dan “bestSolution” akan diganti dengan langkah tersebut.
5. Karena program memeriksa dari panjang buffer terkecil terlebih dahulu, apabila terdapat jumlah hadiah yang sama dengan panjang yang berbeda, maka program akan **mengambil langkah dengan panjang terpendek**.

6. Saat program telah menentukan *starting point* maka token tersebut akan dimasukkan sebagai langkah pertama.
7. Langkah akan bergantian secara vertikal maupun horizontal, ditandai dengan variabel boolean *isVertical*, ketika bergerak vertikal, program akan mencari dengan arah atas maupun bawah. Gerakan ke atas ditandai dengan mengurangi baris token, sebaliknya untuk gerakan ke bawah. Setelah melakukan gerakan vertikal program akan memanggil fungsi secara rekursif dengan variabel *isVertical* bernilai *false* untuk melakukan gerak horizontal. Program akan bergerak secara horizontal kanan dan kiri, jika bergerak ke kanan maka kolom akan ditambahkan, sebaliknya untuk gerakan ke kiri.
8. Setelah mendapatkan token yang baru, token akan dievaluasi terlebih dahulu (dengan membandingkan dengan variabel *visitedToken*. Jika token pernah dilewati sebelumnya maka token tidak akan dicek untuk langkah tersebut, jika belum token akan ditambahkan kepada langkah dan koordinat token akan dicatat oleh program untuk pengecekan token selanjutnya
9. Jika ingin mengganti *starting point* untuk langkah yang baru, *visitedPath* akan *reset* ulang.
10. Program akan berjalan sampai mencapai batas buffer maksimum

## 2.2. Source Program

### 2.2.1 Struktur Data

```
struct Coordinates
{
    int x;
    int y;
};

struct TokenItems
{
    Coordinates position;
    string token;
};

struct Matrix {
    int rows;
    int cols;
    vector<vector<TokenItems>> item;
};
```

Gambar 1 Struktur Data I

```
struct Tokens {
    int jumlah;
    vector<string> item;
};

struct Sequences {
    int number;
    vector<int> reward;
    vector<vector<TokenItems>> item;
};

struct Path {
    vector<TokenItems> item;
};

struct Inputs {
    int bufferSize;
    Tokens tokens;
    Sequences sequence;
    Matrix matrix;
};
```

Gambar 2 Struktur Data II

### 2.2.2 Variabel Global

```
extern Inputs input;  
extern Path bestSolution;  
extern int currentReward;
```

Gambar 3 Struktur Variabel-variabel global

### 2.2.3 Algoritma Bruteforce

```
1 bool hasSequence(const vector<TokenItems>& path, const vector<TokenItems>& sequence) {  
2     if (path.size() >= sequence.size()) {  
3         int lenDiff = path.size() - sequence.size();  
4         for (int i = 0; i <= lenDiff; i++) {  
5             int j = 0;  
6             while (j < sequence.size() && path[i + j].token == sequence[j].token) {  
7                 j++;  
8             }  
9             if (j == sequence.size()) {  
10                return true;  
11            }  
12        }  
13        return false;  
14    } else {  
15        return false;  
16    }  
17 }
```

Gambar 4 Fungsi hasSequence

Fungsi untuk memeriksa apakah sebuah langkah memiliki sequence tertentu atau tidak, fungsi akan mencari elemen pertama yang sama jika ditemukan lalu akan mencari elemen yang lainnya. Apabila sampai akhir langkah tidak ditemukan elemen pertama yang sama maka fungsi akan langsung mengembalikan nilai *false*

```
1 bool hasVisited(vector<Coordinates>& visitedTokens, TokenItems& currentPath) {  
2     Coordinates position = currentPath.position;  
3     for (const auto& pos : visitedTokens) {  
4         if (position.x == pos.x && position.y == pos.y) {  
5             return true;  
6         }  
7     }  
8     return false;  
9 }
```

Gambar 5 Fungsi hasVisited


Fungsi untuk memeriksa apakah sebuah token dengan koordinat tertentu terdapat di *list* kumpulan koordinat, jika belum ada akan mengembalikan *true*



```
1 void insertCoor(TokenItems& token, int x, int y){  
2     token.position.x = x;  
3     token.position.y = y;  
4 }
```

Gambar 6 Fungsi insertCoor

Fungsi untuk memasukkan koordinat pada tipe data *TokenItem*



```
1 Path addToken(Path path, TokenItems item) {  
2     path.item.push_back(item);  
3     return path;  
4 }
```

Gambar 7 Fungsi addToken

Fungsi untuk memasukkan TokenItems ke tipe struktur data *Path*

```

1 void findPaths(TokenItems token,
2               Path currentSolution,
3               bool isVertical,
4               vector<Coordinates> visitedToken,
5               int remainingMove) {
6
7     if (remainingMove == 0) {
8         int reward = 0;
9         for (size_t i = 0; i < input.sequence.item.size(); ++i) {
10             const auto& seq = input.sequence.item[i];
11             if (hasSequence(currentSolution.item, seq)) {
12                 reward += input.sequence.reward[i];
13             }
14         }
15         if (reward > currentReward) {
16             currentReward = reward;
17             bestSolution = currentSolution;
18         }
19     } else {
20         if (isVertical) {
21             // Moves Downward
22             for (int i = token.position.x + 1; i < input.matrix.rows; i++) {
23                 TokenItems next = input.matrix.item[i][token.position.y];
24                 if (!hasVisited(visitedToken, next)) {
25                     Path path = addToken(currentSolution, next);
26                     visitedToken.push_back(next.position);
27                     findPaths(next, path, false, visitedToken, remainingMove - 1);
28                 }
29             }
30
31             // Moves upwards
32             for (int i = token.position.x - 1; i >= 0; i--) {
33                 TokenItems next = input.matrix.item[i][token.position.y];
34                 if (!hasVisited(visitedToken, next)) {
35                     Path path = addToken(currentSolution, next);
36                     visitedToken.push_back(next.position);
37                     findPaths(next, path, false, visitedToken, remainingMove - 1);
38                 }
39             }
40         } else {
41
42             // Moves right
43             for (int j = token.position.y + 1; j < input.matrix.cols; j++) {
44                 TokenItems next = input.matrix.item[token.position.x][j];
45                 if (!hasVisited(visitedToken, next)) {
46                     Path path = addToken(currentSolution, next);
47                     visitedToken.push_back(next.position);
48                     findPaths(next, path, true, visitedToken, remainingMove - 1);
49                 }
50             }
51
52             // Moves left
53             for (int j = token.position.y - 1; j >= 0; j--) {
54                 TokenItems next = input.matrix.item[token.position.x][j];
55                 if (!hasVisited(visitedToken, next)) {
56                     Path path = addToken(currentSolution, next);
57                     visitedToken.push_back(next.position);
58                     findPaths(next, path, true, visitedToken, remainingMove - 1);
59                 }
60             }
61         }
62     }
63 }
64 }

```

Gambar 8 Fungsi findPaths



```

1 void solve(Matrix matrix, Sequences sequences, int bufferSize, int& currentReward, Path& bestSolution){
2     for (int i=2;i <=bufferSize;i++){
3         for (int j = 0 ; j < matrix.cols;j++){
4             Path currentSolution;
5             vector<Coordinates> visitedToken;
6             TokenItems start = matrix.item[0][j];
7             currentSolution = addToken(currentSolution, start);
8             visitedToken.clear();
9             visitedToken.push_back(start.position);
10            findPaths(start,currentSolution,true,visitedToken,i);
11        }
12    }
13 }

```

Gambar 9 Fungsi solve

Fungsi algoritma utama untuk program bruteforce (penjelasan lengkap pada 2.1)

## 2.2.4 Print Struktur Data

```

1 void printMatrix(Matrix matrix) {
2     for (int i = 0; i < matrix.rows; i++) {
3         for (int j = 0; j < matrix.cols; j++) {
4             cout << matrix.item[i][j].token << " ";
5         }
6         cout << endl;
7     }
8 }

```

Gambar 10 Fungsi printMatrix

Fungsi untuk mencetak struktur *Matrix*

```

1 void printSequence(Sequences sequence) {
2     for (int i = 0; i < sequence.number; i++) {
3         cout << "Sequence " << i + 1 << " : ";
4         for (size_t j = 0; j < sequence.item[i].size(); j++) {
5             cout << sequence.item[i][j].token;
6             if (j < (sequence.item[i].size() - 1)) {
7                 cout << " ";
8             }
9         }
10        cout << ", Reward: " << sequence.reward[i] << endl;
11    }
12 }

```

Gambar 11 Fungsi printSequence

Fungsi untuk mencetak struktur *Sequence*

```

1 void printInputs(Inputs input) {
2     cout << "Buffer: " << input.bufferSize << endl;
3     cout << "Matrix-width: " << input.matrix.cols << " Matrix-height: " << input.matrix.rows << endl;
4     printMatrix(input.matrix);
5     cout << "Number of sequence: " << input.sequence.number << endl;
6     printSequence(input.sequence);
7 }

```

Gambar 12 Fungsi printInputs

Fungsi untuk mencetak struktur *Inputs*

```

1 void printPath(const Path& path) {
2     for (const auto& tokenItem : path.item) {
3         cout << tokenItem.token << " ";
4     }
5     cout << endl;
6     cout << "Coordinates: " << endl;
7     for (const auto& token : path.item) {
8         cout << "(" << token.position.y + 1 << ", " << token.position.x + 1 << ")" << endl;
9     }
10 }

```

Gambar 13 Fungsi printPath

Fungsi untuk mencetak struktur *Path*

### 2.2.5 Random Generator

```
1  int randomize(int min,int max){
2      random_device rd;
3      mt19937 gen(rd());
4      uniform_int_distribution<> range(min,max);
5
6      return range(gen);
7  }
```

Gambar 14 Fungsi randomize

Fungsi untuk menghasilkan angka random dari range angka tertentu

```
1  Tokens getTokens() {
2      Tokens tokens;
3      cout << "Masukkan jumlah token:" << endl;
4      cin >> tokens.jumlah;
5
6      cin.ignore(numeric_limits<streamsize>::max(), '\n');
7
8      string line;
9      cout << "Masukkan token dalam satu baris:" << endl;
10
11     getline(cin,line);
12     istream iss(line);
13
14     string token;
15     while (iss >> token)
16     {
17         tokens.item.push_back(token);
18     }
19     return tokens;
20 }
```

Gambar 15 Fungsi getTokens

Fungsi untuk mengambil token dari masukkan CLI dan mengembalikan tipe data *Token*



```
1  Matrix getRandomMatrix(Tokens token) {
2      Matrix matrix;
3      string line;
4      cout << "Masukkan ukuran matrix (lebar<spasi>tinggi): " << endl;
5      cin >> matrix.cols >> matrix.rows;
6
7
8      // Seed
9      shuffle(token.item.begin(), token.item.end(), default_random_engine(time(0)));
10
11     matrix.item.resize(matrix.rows, vector<TokenItems>(matrix.cols));
12
13     for (int i =0; i< matrix.rows;i++){
14         for (int j=0; j<matrix.cols;j++){
15             int index = randomize(0,token.jumlah-1);
16             matrix.item[i][j].token = token.item[index];
17             matrix.item[i][j].position.x = i;
18             matrix.item[i][j].position.y = j;
19         }
20     }
21     return matrix;
22 }
```

Gambar 16 Fungsi getRandomMatrix

Fungsi untuk menghasilkan matrix dengan isi token *random*, *randomize()* akan menghasilkan angka acak yang berkorelasi dengan indeks token. Token yang terpilih akan dimasukkan ke dalam matrix

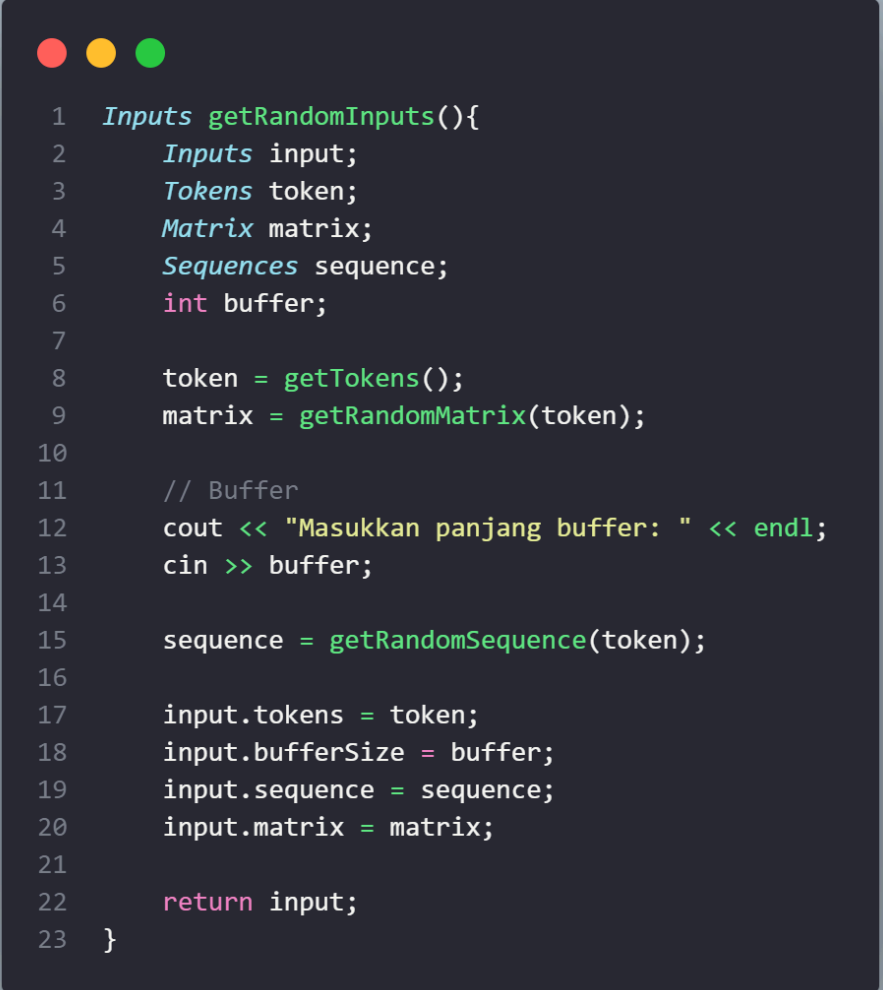
```

1 Sequences getRandomSequence(Tokens token){
2     Sequences sequence;
3     int maxSeq;
4     cout << "Masukkan jumlah sequence: " << endl;
5     cin >> sequence.number;
6
7     cout << "Masukkan ukuran maksimal sequence: " << endl;
8     cin >> maxSeq;
9
10    // Seed
11    shuffle(token.item.begin(), token.item.end(), default_random_engine(time(0)));
12
13
14    for (int i=0;i<sequence.number;i++){
15        vector<TokenItems> list;
16        int length = randomize(1,maxSeq-1);
17        for (int j=0; j < length ; j++){
18            int index = randomize(0,token.jumlah-1);
19            TokenItems temp;
20            temp.token = token.item[index];
21            list.push_back(temp);
22        }
23        sequence.item.push_back(list);
24
25        int points = randomize(1,50);
26        sequence.reward.push_back(points); // range masih ngasal
27    }
28
29    return sequence;
30 }

```

Gambar 17 Fungsi getRandomSequence

Fungsi untuk menghasilkan sequence dengan isi token *random*, *randomize()* akan menghasilkan angka acak yang berkorelasi dengan indeks token. Token yang terpilih akan dimasukkan ke dalam sequence. Hadiah tiap sekuens akan didapatkan dari angka acak yang dihasilkan fungsi *randomize()* lainnya.



```

1  Inputs getRandomInputs(){
2      Inputs input;
3      Tokens token;
4      Matrix matrix;
5      Sequences sequence;
6      int buffer;
7
8      token = getTokens();
9      matrix = getRandomMatrix(token);
10
11     // Buffer
12     cout << "Masukkan panjang buffer: " << endl;
13     cin >> buffer;
14
15     sequence = getRandomSequence(token);
16
17     input.tokens = token;
18     input.bufferSize = buffer;
19     input.sequence = sequence;
20     input.matrix = matrix;
21
22     return input;
23 }

```

Gambar 18 Fungsi getRandomInputs

Fungsi untuk memasukkan tipe-tipe data yang sudah diacak dan mengembalikan dalam bentuk tipe data *Inputs* untuk diproses

### 2.2.6 File Handling

```

1 Inputs fileParser() {
2     string fileName;
3     cout << "Silahkan masukkan nama file dengan .txt, contoh 'hi.txt': " << endl;
4     cin >> fileName;
5
6     string filepath = "../test/input/" + fileName;
7
8     Inputs input;
9
10    ifstream inputFile(filepath, ios::in);
11
12    // Error handling
13    if (!inputFile.is_open()) {
14        cerr << "File Error" << filepath << endl;
15        return input;
16    }
17
18    string line;
19
20    // Read buffer length
21    getline(inputFile, line);
22    istream(line) >> input.bufferSize;
23
24    // Read matrix rows and columns
25    getline(inputFile, line);
26    istream(line) >> input.matrix.cols >> input.matrix.rows;
27
28    // Initialize matrix item vector
29    input.matrix.item.resize(input.matrix.rows, vector<TokenItems>(input.matrix.cols));
30
31    // Read matrix
32    for (int i = 0; i < input.matrix.rows; i++) {
33        getline(inputFile, line);
34        istream issMatrix(line);
35        for (int j = 0; j < input.matrix.cols; j++) {
36            issMatrix >> input.matrix.item[i][j].token;
37            input.matrix.item[i][j].position.x = i;
38            input.matrix.item[i][j].position.y = j;
39        }
40    }
41
42    // Read number of sequences
43    getline(inputFile, line);
44    istream(line) >> input.sequence.number;
45
46    input.sequence.reward.resize(input.sequence.number);
47
48    // Read sequence and reward
49    for (int i = 0; i < input.sequence.number; i++) {
50        getline(inputFile, line);
51        istream iss(line);
52        vector<TokenItems> list;
53        string code;
54        while (iss >> code){
55            TokenItems temp;
56            temp.token = code;
57            list.push_back(temp);
58        }
59        input.sequence.item.push_back(list);
60        getline(inputFile, line);
61        istream(line) >> input.sequence.reward[i];
62    }
63
64    inputFile.close();
65    return input;
66 }

```

Gambar 19 Fungsi fileParser

Fungsi untuk membaca *file* .txt dan menyimpan hasil ke dalam struktur data  
*Inputs*

```
1 void txtWrite(int reward, Path bestSolution, int duration){
2     string fileName;
3     cout << endl;
4     cout << BLUE << "Silahkan masukkan nama file dengan .txt, contoh \'hi.txt\': " << RESET;
5     cin >> fileName;
6
7     string filepath = "../test/output/" + fileName;
8
9     ofstream outputFile(filepath);
10
11     if (!outputFile.is_open()){
12         cerr << RED << "File error" << filepath << endl;
13     } else {
14         outputFile << customArt << endl << endl;
15
16         outputFile << "-- MASUKAN DATA -- " << endl;
17         outputFile << "Matriks: " << endl;
18         for (int i = 0; i < input.matrix.rows; i++) {
19             for (int j = 0; j < input.matrix.cols; j++) {
20                 outputFile << input.matrix.item[i][j].token << " ";
21             }
22             outputFile << endl;
23         }
24
25         outputFile << "Sekuens-sekuens: " << endl;
26         for (int i = 0; i < input.sequence.number; i++) {
27             outputFile << "Sequence " << i + 1 << " : ";
28             for (size_t j = 0; j < input.sequence.item[i].size(); j++) {
29                 outputFile << input.sequence.item[i][j].token;
30                 if (j < (input.sequence.item[i].size() - 1)) {
31                     outputFile << " ";
32                 }
33             }
34             outputFile << ", Reward: " << input.sequence.reward[i] << endl;
35         }
36         outputFile << endl;
37         outputFile << "----- HASIL ALGORITMA ---- " << endl;
38         outputFile << "Total hadiah: " << reward << endl;
39         outputFile << "Solusi: ";
40
41         for (size_t i = 0; i < bestSolution.item.size(); i++){
42             outputFile << bestSolution.item[i].token;
43             outputFile << " ";
44         }
45         outputFile << endl;
46
47         outputFile << "Koordinat: " << endl;
48         for (size_t i = 0; i < bestSolution.item.size(); i++){
49             outputFile << bestSolution.item[i].position.y + 1 << " ";
50             outputFile << bestSolution.item[i].position.x + 1 << endl;
51         }
52
53         outputFile << "Waktu eksekusi: ";
54         outputFile << duration << " ms" << endl;
55
56         outputFile.close();
57         cout << endl;
58         cout << UNDERLINE << GREEN << "File berhasil ditulis" << RESET << endl;
59     }
60 }
```

Gambar 20 Fungsi txtWrite

Fungsi untuk menuliskan hasil dan menyimpannya dalam bentuk .txt



```

1  #include "utilities.h"
2  #include <chrono>
3
4  using namespace chrono;
5
6
7  int main(){
8      int type;
9      cout << BOLD << YELLOW << customArt << RESET << endl;
10     cout << BOLD << CYAN << "Selamat datang di permainan!" << RESET << endl << endl;
11
12
13     cout << RED << "Silahkan pilih mode input: " << RESET << endl;
14     cout << "1. Menggunakan file txt" << endl;
15     cout << "2. Menggunakan CLI/Randomize" << endl << endl;
16
17     do {
18         cout << GREEN << "Masukkan angka yang dipilih (1/2): " << RESET;
19         cin >> type;
20     } while (type != 1 and type != 2);
21
22     cout << endl;
23
24     if (type == 1){
25         cout << MAGENTA << "Anda memilih: " << BOLD << "INPUT DARI TXT" << RESET << endl;
26         input = fileParser();
27     } else {
28         cout << MAGENTA << "Anda memilih: " << BOLD << "INPUT DARI CLI" << RESET << endl;
29         cout << "Matrix dan Sequence akan dihasilkan secara acak" << endl << endl;
30         input = getRandomInputs();
31     }
32
33     if (input.bufferSize != 0){
34
35         printInputs(input);
36
37         cout << endl;
38         cout << BOLD << RED << "CALCULATING....." << RESET << endl;
39
40         // ALGORITHM
41         auto start = high_resolution_clock::now();
42         solve(input.matrix,input.sequence,input.bufferSize,currentReward,bestSolution);
43         auto stop = high_resolution_clock::now();
44         auto duration = duration_cast<milliseconds>(stop - start);
45
46         cout << endl;
47         cout << BOLD << BLUE << "Reward: " << RESET << currentReward << endl;
48         cout << BOLD << BLUE << "Solusi: " << RESET;
49         printPath(bestSolution);
50         cout << endl;
51         cout << UNDERLINE << RED << duration.count() << " ms" << RESET << endl << endl;
52
53         string ans;
54         do {
55             cout << GREEN << "Apakah ingin menyimpan solusi? (y/n): " << RESET;
56             cin >> ans;
57         } while (ans != "y" and ans != "n" and ans != "Y" and ans != "N");
58
59         if (ans == "y" || ans == "Y"){
60             txtWrite(currentReward,bestSolution,duration.count());
61         }
62         cout << endl;
63         cout << BOLD << CYAN << "Terima kasih sudah melakukan permainan" << RESET << endl;
64
65     } else {
66         cout << endl;
67         cout << BOLD << RED << "Panjang Buffer 0, tidak ada solusi yang memungkinkan" << RESET << endl;
68         cout << BOLD << CYAN << "Terima kasih sudah melakukan permainan" << RESET << endl;
69     }
70
71 }

```

Gambar 21 Fungsi main

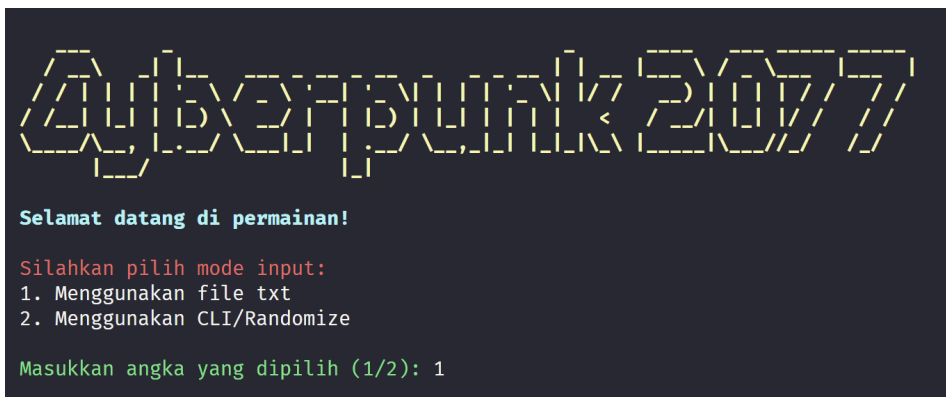
Fungsi utama untuk menjalankan program, terdapat beberapa “styling” untuk output pada CLI

### 2.2.7 Folder Structure

```
project_root
├── bin
├── doc
├── src
│   ├── main.cpp
│   ├── utilities.cpp
│   └── utilities.h
├── test
│   ├── input
│   └── output
├── readme
├── run.bat
└── run.sh
```

1. Bin : folder memuat hasil file .exe
2. Doc : berisi laporan
3. Src : berisi source code
  - a. Main.cpp : kode utama
  - b. Utilities.cpp : berisi fungsi fungsi pembantu
  - c. Utilities.h : header file untuk utilities.cpp
4. Test : folder untuk txt
  - a. Input : folder untuk menyimpan masukkan berupa txt
  - b. Output : folder untuk menyimpan hasil keluaran berupa txt
5. Readme : berisi *overview* project
6. Run.bat : file untuk membantu menjalankan program di Windows
7. Run.sh : file untuk membantu menjalankan program di Linux

### 2.3. Hasil Input Output



```
Cyberpunk 2077

Selamat datang di permainan!

Silahkan pilih mode input:
1. Menggunakan file txt
2. Menggunakan CLI/Randomize

Masukkan angka yang dipilih (1/2): 1
```

Gambar 22 Tampilan awal permainan

```

Masukkan angka yang dipilih (1/2): 1

Anda memilih: INPUT DARI TXT

Silahkan masukkan nama file dengan .txt, contoh 'hi.txt': yoi.txt

Data yang akan digunakan
Buffer: 4
Matrix-width: 4 Matrix-height: 5
Matrix:
CD AB EF CD
EF EF AB BC
EF AB BC CD
AB CD CD BC
CD BC EF AB
Number of sequence: 3
Sequence 1 : CD BC AB, Reward: 42
Sequence 2 : EF EF BC, Reward: 5
Sequence 3 : AB AB, Reward: 17

CALCULATING.....

Reward: 59
Solusi: AB AB CD BC AB
Coordinates:
(2, 1)
(2, 3)
(4, 3)
(4, 4)
(1, 4)

3 ms

```

Gambar 23 Tampilan masukan dengan txt

Tampilan masukkan jika pengguna memilih nomor 1, yaitu *input* didapatkan dengan membaca txt

```
Silahkan masukkan nama file dengan .txt, contoh 'hi.txt': test2.txt
```

**Data yang akan digunakan**

Buffer: 7

Matrix-width: 6 Matrix-height: 6

Matrix:

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

Number of sequence: 3

Sequence 1 : BD E9 1C, Reward: 15

Sequence 2 : BD 7A BD, Reward: 20

Sequence 3 : BD 1C BD 55, Reward: 30

**CALCULATING.....**

**Reward: 50**

**Solusi: 7A BD 7A BD 1C BD 55**

**Coordinates:**

(1, 1)

(1, 4)

(3, 4)

(3, 5)

(6, 5)

(6, 4)

(5, 4)

569 ms

Gambar 24 Tampilan masukan dengan txt II

```
Apakah ingin menyimpan solusi? (y/n): y
```

```
Silahkan masukkan nama file dengan .txt, contoh 'hi.txt': test2-solution.txt
```

File berhasil ditulis

**Terima kasih sudah melakukan permainan**

Gambar 25 Tampilan prompt menyimpan hasil

```
test > input > test2.txt
```

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

Gambar 26 Tampilan txt "test2.txt" (masukan)

```

test > output > test2-solution.txt
1
2
3
4
5
6
7
8
9
10 -- MASUKAN DATA --
11 Matriks:
12 7A 55 E9 E9 1C 55
13 55 7A 1C 7A E9 55
14 55 1C 1C 55 E9 BD
15 BD 1C 7A 1C 55 BD
16 BD 55 BD 7A 1C 1C
17 1C 55 55 7A 55 7A
18 Sekuens-sekuens:
19 Sequence 1 : BD E9 1C, Reward: 15
20 Sequence 2 : BD 7A BD, Reward: 20
21 Sequence 3 : BD 1C BD 55, Reward: 30
22
23 ----- HASIL ALGORITMA -----
24 Total hadiah: 50
25 Solusi: 7A BD 7A BD 1C BD 55
26 Koordinat:
27 1,1
28 1,4
29 3,4
30 3,5
31 6,5
32 6,4
33 5,4
34 Waktu eksekusi: 570 ms
35

```

Gambar 27 Tampilan txt “test2-solution.txt” (keluaran)

Hasil program dengan masukkan txt dan melakukan penyimpanan dalam bentuk txt

Anda memilih: **INPUT DARI TXT**

Silahkan masukkan nama file dengan .txt, contoh 'hi.txt': hello.txt

**Data yang akan digunakan**

Buffer: 7

Matrix-width: 5 Matrix-height: 8

Matrix:

7A 55 E9 E9 1C

55 7A 1C 7A E9

55 1C 1C 55 E9

BD 1C 7A 1C 55

BD 55 BD 7A 1C

1C 55 55 7A 55

BD 1C 7A 1C 55

7A 55 E9 E9 1C

Number of sequence: 3

Sequence 1 : BD E9 1C, Reward: 40

Sequence 2 : BD 7A BD, Reward: 21

Sequence 3 : BD 1C BD 55, Reward: 15

**CALCULATING.....**

**Reward:** 40

**Solusi:** 7A BD BD E9 1C

**Coordinates:**

(1, 1)

(1, 5)

(3, 5)

(3, 8)

(5, 8)

876 ms

Gambar 28 Tampilan masukan dengan txt III

Apakah ingin menyimpan solusi? (y/n): n

**Terima kasih sudah melakukan permainan**

Gambar 29 prompt menyimpan hasil II

Hasil dari eksekusi program menggunakan input dari txt dan tidak disimpan

```
Anda memilih: INPUT DARI CLI
Matrix dan Sequence akan dihasilkan secara acak

Masukkan jumlah token: 5
Masukkan token dalam satu baris: AA BB CC DD EE
Masukkan ukuran matrix (lebar<spasi>tinggi): 6 6
Masukkan panjang buffer: 4
Masukkan jumlah sequence: 3
Masukkan ukuran maksimal sequence: 5
Data yang akan digunakan
Buffer: 4
Matrix-width: 6 Matrix-height: 6
Matrix:
EE CC DD AA BB DD
CC AA BB DD EE DD
DD AA AA EE DD DD
DD CC CC CC DD BB
CC EE CC BB DD DD
BB BB AA BB EE EE
Number of sequence: 3
Sequence 1 : DD EE CC DD, Reward: 16
Sequence 2 : AA BB DD DD, Reward: 43
Sequence 3 : BB AA BB, Reward: 32

CALCULATING.....

Reward: 43
Solusi: AA BB DD DD
Coordinates:
(4, 1)
(4, 5)
(5, 5)
(5, 4)

9 ms
```

Gambar 30 Tampilan masukan dengan CLI (randomizer)

```

Anda memilih: INPUT DARI CLI
Matrix dan Sequence akan dihasilkan secara acak

Masukkan jumlah token: 5
Masukkan token dalam satu baris: AA BB CC DD EE
Masukkan ukuran matrix (lebar<spasi>tinggi): 6 6
Masukkan panjang buffer: 7
Masukkan jumlah sequence: 3
Masukkan ukuran maksimal sequence: 6
Data yang akan digunakan
Buffer: 7
Matrix-width: 6 Matrix-height: 6
Matrix:
CC DD CC DD CC EE
DD AA EE CC CC BB
DD EE CC CC EE EE
AA AA CC DD EE AA
AA AA CC BB AA EE
AA AA BB CC EE CC
Number of sequence: 3
Sequence 1 : DD BB BB AA, Reward: 11
Sequence 2 : EE BB AA, Reward: 47
Sequence 3 : AA BB DD, Reward: 27

CALCULATING.....

Reward: 74
Solusi: EE BB AA AA BB DD
Coordinates:
(6, 1)
(6, 2)
(2, 2)
(2, 5)
(4, 5)
(4, 4)

596 ms

```

Gambar 31 Tampilan masukan dengan CLI (randomizer) II

```

Apakah ingin menyimpan solusi? (y/n): y
Silahkan masukkan nama file dengan .txt, contoh 'hi.txt': random-solution.txt
File berhasil ditulis
Terima kasih sudah melakukan permainan

```

Gambar 32 prompt menyimpan hasil III



```

test > output > random-solution.txt
1
2
3
4
5
6
7
8
9
10 -- MASUKAN DATA --
11 Matriks:
12 CC DD CC DD CC EE
13 DD AA EE CC CC BB
14 DD EE CC CC EE EE
15 AA AA CC DD EE AA
16 AA AA CC BB AA EE
17 AA AA BB CC EE CC
18 Sekuens-sekuens:
19 Sequence 1 : DD BB BB AA, Reward: 11
20 Sequence 2 : EE BB AA, Reward: 47
21 Sequence 3 : AA BB DD, Reward: 27
22
23 ----- HASIL ALGORITMA -----
24 Total hadiah: 74
25 Solusi: EE BB AA AA BB DD
26 Koordinat:
27 6,1
28 6,2
29 2,2
30 2,5
31 4,5
32 4,4
33 Waktu eksekusi: 596 ms
34

```

Gambar 33 Tampilan txt “random-solution.txt” (keluaran)

Hasil program dengan masukan CLI dan melakukan penyimpanan dalam bentuk txt

```

Masukkan jumlah token: 4
Masukkan token dalam satu baris: QW WE ER RT
Masukkan ukuran matrix (lebar<spasi>tinggi): 5 7
Masukkan panjang buffer: 7
Masukkan jumlah sequence: 3
Masukkan ukuran maksimal sequence: 6
Data yang akan digunakan
Buffer: 7
Matrix-width: 5 Matrix-height: 7
Matrix:
ER ER QW WE WE
QW ER QW RT RT
RT ER ER ER QW
WE QW RT QW QW
QW RT QW ER ER
ER RT QW WE RT
RT QW ER RT RT
Number of sequence: 3
Sequence 1 : RT ER QW, Reward: 44
Sequence 2 : WE QW WE WE, Reward: 44
Sequence 3 : QW WE WE, Reward: 15

```

**CALCULATING.....**

```

Reward: 103
Solusi: ER WE QW WE WE RT ER QW
Coordinates:
(1, 1)
(1, 4)
(4, 4)
(4, 1)
(5, 1)
(5, 2)
(2, 2)
(2, 7)

```

466 ms

Gambar 34 Tampilan masukan dengan CLI (randomizer) III

```

Apakah ingin menyimpan solusi? (y/n): n

```

```

Terima kasih sudah melakukan permainan

```

Gambar 35 prompt menyimpan hasil IIII

Hasil dari eksekusi program menggunakan input dari CLI dan tidak disimpan

# Atanarink 2022

Selamat datang di permainan!

Silahkan pilih mode input:

1. Menggunakan file txt
2. Menggunakan CLI/Randomize

Masukkan angka yang dipilih (1/2): 5

Masukkan angka yang dipilih (1/2): 4

Masukkan angka yang dipilih (1/2): 2

Anda memilih: **INPUT DARI CLI**

Matrix dan Sequence akan dihasilkan secara acak

Masukkan jumlah token: 3

Masukkan token dalam satu baris: AA BB CC

Masukkan ukuran matrix (lebar<spasi>tinggi): 4 3

Masukkan panjang buffer: 0

Masukkan jumlah sequence: 3

Masukkan ukuran maksimal sequence: 4

**Panjang Buffer 0, tidak ada solusi yang memungkinkan**

**Terima kasih sudah melakukan permainan**

Gambar 36 Tampilan saat buffer 0

Tampilan saat panjang buffer 0 dan tidak ada solusi, dan dilakukan beberapa validasi input

## BAB III

### Lampiran

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		✓

**Link Repository GitHub :** [https://github.com/angiekierra/Tucil1\\_13522048.git](https://github.com/angiekierra/Tucil1_13522048.git)