Laporan Tugas Kecil 1
IF2211 Strategi Algoritma
Penyelesaian *Hacking Minigame Cyberpunk 2077* dengan Algoritma *Brute Force*

Disusun oleh:
Marvel Pangondian (13522075)

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* dalam Penyelesaian *Hacking Minigame Cyberpunk 2077*

Algoritma *Brute Force* merupakan algoritma yang termudah sebab algoritma tersebut berasal dari solusi yang dipikiran pertama kali saat mencoba menyelesaikan sebuah masalah. Algoritma *Brute Force* memecahkan masalah dengan mencoba semua kemungkinan solusi untuk menemukan solusi yang benar. Pendekatan ini sering digunakan ketika solusi tepat diperlukan dan sumber daya komputasi tidak menjadi masalah utama, atau ketika tidak ada algoritma yang lebih efisien dan spesifik untuk masalah tersebut.

Dalam mencari susunan token optimal pada permainan *Hacking Minigame Cyberpunk 2077*, digunakannya algoritma *Brute Force* untuk mencari semua kemungkinan susunan token sampai ditemukannya sebuah susunan token yang paling optimal. Implementasi Algoritma *Brute Force* ini menggunakan rekursi yang akan berjalan sampai jumlah susunan token maksimal (*buffer*) atau saat jumlah susunan token sama dengan jumlah elemen pada matrix. Setelah ditemukan sebuah susunan token, program akan menghitung berapa poin yang dihasilkan oleh susunan tersebut dan membandingkan dengan poin maksimal saat itu, jika melebihi poin maksimal maka poin maksimal akan diganti dengan poin susunan tersebut serta susunan tersebut akan disimpan pada variabel *curr_max_combination*. Rekursi ini juga menggunakan matrix *hasVisited* yang merupakan matrix boolean yang menandakan apakah sebuah koordinat sudah di hampiri oleh susunan pada saat itu. Hasil akhir dari proses ini adalah poin maksimal, susunan token optimal, koordinat pada setiap token dalam susunan optimal, serta waktu eksekusi algoritma.

.

1.2 Alur Program

Pada awal, program akan meminta user untuk memilih 2 macam input yakni pilihan pertama berupa file dengan ekstensi .txt dan pilihan 2 berupa input secara manual yang akan digunakan untuk menghasilkan matrix, *sequences*, dan poin secara otomatis oleh program. Untuk pilihan pertama, *user* akan diminta memberikan nama file dengan ekstensi .txt. Pada file tersebut harus terdapat ukuran buffer (jumlah maksimal token yang dapat disusun secara sekuensial.), ukuran matrix (jumlah kolom dan jumlah baris), matrix token, jumlah *sequences*, *sequences* dan poinnya. Pada pilihan kedua, program akan meminta *user* untuk memasukkan jumlah token unik, token, ukuran buffer, ukuran matrix (jumlah kolom dan jumlah baris), banyaknya *sequences*, dan ukuran maksimal sebuah *sequence.* Pada pilihan kedua, program akan menghasilkan *sequences*, poin, serta matrix secara otomatis berdasarkan masukkan *user.*

Setelah melalui proses *input*, program akan memulai mencari semua kombinasi susunan token dengan algoritma *Brute Force* menggunakan prosedur *allCombinations* serta *nextChoice*. Sesuai dengan aturan, program akan memulai dengan memilih sebuah token pada baris pertama, setelah itu program akan memilih token kedua yang terletak pada kolom yang sama dengan token pertama, lalu token ketiga akan berada pada baris yang sama dengan token kedua, token keempat akan berada pada kolom yang sama

dengan token ketiga, dst. Proses tersebut dilakukan secara rekursif. Ketika sebuah token telah dipilih oleh program untuk sebuah susunan token, program akan terlebih dahulu mengecek apakah token tersebut dengan koordinat *(row,col)* sudah termasuk dalam susunan token saat itu. Jika tidak, maka matrix *hasVisited* dengan index *(row,col)* diubah menjadi true menandakan bahwa token pada koordinat (*row,col*) tersebut sudah dimasukkan ke susunan token saat itu. Ketika sebuah susunan token memiliki ukuran 2 token atau lebih, program akan menghitung poin yang dihasilkan oleh susunan tersebut. Alasan proses ini dilakukan saat susuan memiliki ukuran 2 token atau lebih karena ukuran minimal sebuah *sequence* adalah 2 token. Awalnya, penulis melakukan proses penghitungan saat sebuah susunan memiliki ukuran sama dengan buffer atau saat ukuran susunan sama dengan jumlah elemen matrix, tetapi untuk mencari susunan paling **optimal** dengan jumlah token seminimal dan jumlah poin semaksimal mungkin maka proses penghitungan dilakukan saat sebuah susunan berukuran 2 token atau lebih. Hasil poin yang dihasilkan oleh sebuah susunan akan dibandingkan dengan poin maksimal yang disimpan pada variabel *curr_max_point*. Jika poin yang dihasilkan oleh susunan melebihi nilai dalam variabel *curr_max_point*, maka nilai *curr_max_point* akan diperbaharui, serta susunan token akan disimpan pada variabel *curr_max_combination*. Jika susunan tidak memiliki ukuran sama dengan buffer atau sama dengan jumlah elemen matrix, maka program akan memanggil kembali fungsi *nextChoice* sehingga proses pencarian susunan dilakukan secara rekursif. Pada akhir setiap panggilan rekursif, elemen terakhir dari *path* (susunan token dalam bentuk koordinat) dan *sequence_temp* (susunan token dalam bentuk string) dihapus serta *hasVisited* koordinat diubah kembali ke false supaya dapat kembali ke *state* sebelumnya.

Output program akan berdasarkan nilai yang terdapat pada *curr_max_point*. Jika *curr_max_point* memiliki nilai nol atau kurang maka output program adalah "*no solution"* sebab susunan token yang paling optimal adalah susunan kosong yang menghasilkan 0 poin. Sebaliknya, untuk hasil dimana *curr_max_point* lebih dari 0 maka output program akan terdiri atas bobot hadiah atau reward maksimal yang didapatkan, isi dari buffer (susunan token paling optimal), koordinat dari setiap token secara terurut, waktu eksekusi program dalam ms (tidak termasuk waktu membaca masukan dan menyimpan solusi), dan prompt untuk menyimpan solusi dalam sebuah berkas berekstensi .txt.

Bab II
Source Code

2.1 *File* Program Utama

File program utama (main.cpp) yang akan dijalankan pertama, file ini cukup singkat. Berikut *source code* program utama :

1. Module eksternal

```cpp
#include <iostream>
#include <vector>
#include <chrono>
#include <chrono>
#include "cyberpunk.hpp"
#include "IO.hpp"
using namespace std;
```

Gambar 2.1.1 Modul eksternal yang diimpor pada program utama.

2. Source code

```cpp
int main(){
    int input = -1;
    while (input != 3){
        vector<vector<string>> sequences;
        vector<vector<string>> matrix;
        vector<int> points;
        int curr_max_point = -2147483640, buffer;
        vector <pair<int,int>> curr_max_combination;
        display_menu(&input);
        switch(input){
            case(1):
            {
                if (! readFile(sequences ,matrix, points, &buffer)){
                    break;
                }
                auto start = chrono::high_resolution_clock::now(); // start timer
                allCombinatations(sequences,points,matrix,&curr_max_point,curr_max_combination,buffer);
                auto end = chrono::high_resolution_clock::now();// end timer
                auto elapsed = chrono::duration_cast<chrono::milliseconds>(end - start);

                displayResult(curr_max_point,curr_max_combination,matrix,elapsed.count());
                break;
            }
```

Gambar 2.1.2 Source code program utama - bagian (1).

```
            }
            case(2):
            {
                if (! randomInput(sequences ,matrix, points, &buffer)){
                    break;
                }
                cout << endl;
                printMatrix(matrix);
                cout << endl;
                printSequences(sequences,points);
                cout << endl;
                auto start = chrono::high_resolution_clock::now(); // start timer
                allCombinatations(sequences,points,matrix,&curr_max_point,curr_max_combination,buffer);
                auto end = chrono::high_resolution_clock::now();// end timer
                auto elapsed = chrono::duration_cast<std::chrono::milliseconds>(end - start);
                displayResult(curr_max_point,curr_max_combination,matrix,elapsed.count());
                break;
            }
        }


    }

    return 0;
}
```

Gambar 2.1.3 Source code program utama - bagian (2).

2.2 *File Header* Algoritma

File algoritma (cyberpunk.cpp) memiliki fungsi - fungsi dan prosedur - prosedur yang telah didefinisikan pada file header (cyberpunk.hpp) sebagai berikut :

```
// File Header cyberpunk.hpp
#include <iostream>
#include <string>
#include <vector>
using namespace std;



void allCombinatations( const vector<vector<string>>& sequence, const vector<int>& points, const vector<vector<string>>& matrix,
    int* curr_max_point, vector <pair<int,int>>& curr_max_combination,int buffer);
    // find all possible combinations in a given matrix, restricted to the buffer with a certain patter (horizontal, vertical, horizontal, vertical, ...)

void next_choice(int curr_row, int curr_col, int row_matrix, int col_matrix, bool row_search, int buffer, vector<pair<int, int>>& path,
    int* curr_max_point, vector<pair<int,int>>& curr_max_combination, const vector<vector<string>>& sequence, const vector<int>& points,
        int max_points, const vector<vector<string>>& matrix, vector<vector<bool>>& hasVisited, int minSeqSize, vector<string>& sequence_temp);
    // find the possible next token to choose


int sequence_to_point(const vector<string>& sequence ,const vector<vector<string>>& sequences , const vector<int>& points);
    // sum up points of a certain sequence
vector<string> pathToSequence (const vector<pair<int,int>>& path, const vector<vector<string>>& matrix);
    // change from path (vector<pair<int,int>>) into a sequence (vector<string>)

bool sequenceInSequences(vector<string> sequence, vector<vector<string>> sequences);
    // return true if a sequence in a vector of sequences , for testing purposes only
```

Gambar 2.2.1 Tampilan *File Header* cyberpunk.hpp.

2.3 *File* Algoritma

File algoritma adalah file yang berisi implementasi algoritma *Brute Force*. Berikut adalah fungsi - fungsi dan prosedur - prosedur yang terdapat pada *file algoritma* (cyberpunk.cpp):

1. Fungsi sequenceInSequences

```
bool sequenceInSequences(vector<string> sequence, vector<vector<string>> sequences)
// Check if a sequence is in a vector of sequences
// to prevent duplicate sequences
{
    // KAMUS LOKAL
    bool sequenceDuplicate = false;
    int k = 0, i;

    // ALGORITMA
    for (i = 0 ; i < sequences.size() ; i++){
        k = 0;
        while (k < sequence.size() && sequence[k] == sequences[i][k]){
            k++;
        }
        if (k == sequence.size()){
            sequenceDuplicate = true;
            break;
        }
    }
    return sequenceDuplicate;
}
```

Gambar 2.3.1 Fungsi untuk mencegah duplikat *sequences*.

2. Fungsi pathToSequence

```
vector<string> pathToSequence (const vector<pair<int,int>>& path, const vector<vector<string>>& matrix)
// Change path (a sequence of coordinates) into a sequence
{
    // KAMUS LOKAL
    vector<string> sequence;
    int i;

    // ALGORITMA
    for (i = 0 ; i < path.size() ; i++){
        int row = path[i].first;
        int col = path[i].second;
        sequence.emplace_back(matrix[row][col]);
    }
    return sequence;
}
```

Gambar 2.3.2 Fungsi yang mengembalikan *sequence* (susunan token dalam bentuk string) berdasarkan sebuah path (susunan token dalam bentuk koordinat).

3. Fungsi sequenceToPoint

```
int sequence_to_point(const vector<string>& sequence ,const vector<vector<string>>& sequences , const vector<int>& points)
// Return points according to a sequence
{
    // KAMUS LOKAl
    int has_visited[points.size()];
    int total_points = 0, i, seq, k, l;

    // ALGORITMA
    for (i  = 0 ; i < points.size() ; i++){
        has_visited[i] = 0;
    }
    for (seq = 0 ; seq < sequence.size(); seq++){
        for (k = 0 ; k < points.size() ; k++){
            if (sequence[seq] == sequences[k][0] && ! has_visited[k]){
                bool valid = true;
                for (l = 1 ; l < sequences[k].size(); l++){
                    if ((seq + l) >= sequence.size()){
                        valid = false;
                        break;
                    }
                    if (sequence[seq + l] != sequences[k][l]){
                        valid = false;
                        break;
                    }
                }
                if (valid) {
                    total_points += points[k];
                    has_visited[k] = 1;
                }
            }
        }
    }
    return total_points;

}
```

Gambar 2.3.3 Fungsi yang mengembalikan total poin yang dihasilkan sebuah *sequence.*

4. Prosedur next_choice

```
void next_choice(int curr_row, int curr_col, int row_matrix, int col_matrix, bool row_search, int buffer, vector<pair<int, int>>& path,
    int* curr_max_point, vector<pair<int,int>>& curr_max_combination, const vector<vector<string>>& sequence, const vector<int>& points,
        int max_points, const vector<vector<string>>& matrix, vector<vector<bool>>& hasVisited, int minSeqSize, vector<string>& sequence_temp)
// find the possible next token to choose
{
    // KAMUS LOKAL
    int next_col, temp_point, next_row;

    if (row_search){ // make sure if program is looking for a token in the same row or not
        for (int next_col = 0 ; next_col < col_matrix ; ++next_col){
            if (next_col != curr_col){
                if (!hasVisited[curr_row][next_col]){
                    hasVisited[curr_row][next_col] = true;
                    path.emplace_back(curr_row, next_col);
                    sequence_temp.push_back(matrix[curr_row][next_col]);

                    temp_point = sequence_temp.size() >= minSeqSize ? sequence_to_point(sequence_temp, sequence, points) : 0;
                    if (temp_point > *curr_max_point || (temp_point == *curr_max_point && path.size() < curr_max_combination.size())) {
                    *curr_max_point = temp_point;
                    curr_max_combination = path;
                    }
                    if (temp_point < max_points && path.size() != buffer && path.size() != row_matrix * col_matrix) {
                    next_choice(curr_row, next_col, row_matrix, col_matrix, !row_search, buffer, path,
                        curr_max_point, curr_max_combination, sequence, points, max_points, matrix, hasVisited, minSeqSize, sequence_temp);
                    }
                    // return to previous state
                    path.pop_back(); // pop the current coordinate from path
                    sequence_temp.pop_back(); // pop the current coordinate from sequence_temp
                    hasVisited[curr_row][next_col] = false; // revert back to false for the current coordinate
                }
            }
        }
    }
```

Gambar 2.3.4 Prosedur untuk mencari token berikutnya untuk dimasukkan pada sebuah path, Fungsi ini akan dipanggil secara rekursif - bagian (1).

```
    else{
        for (next_row = 0 ; next_row < row_matrix ; ++next_row){
            if (next_row != curr_row){
                if (!hasVisited[next_row][curr_col]){
                    hasVisited[next_row][curr_col] = true;
                    path.emplace_back(next_row, curr_col);
                    sequence_temp.push_back(matrix[next_row][curr_col]);
                    temp_point = sequence_temp.size() >= minSeqSize ? sequence_to_point(sequence_temp, sequence, points) : 0;

                    if (temp_point > *curr_max_point || (temp_point == *curr_max_point && path.size() < curr_max_combination.size())) {
                    *curr_max_point = temp_point;
                    curr_max_combination = path;
                    }
                    if (temp_point < max_points && path.size() != buffer && path.size() != row_matrix * col_matrix) {
                    next_choice(next_row, curr_col, row_matrix, col_matrix, !row_search, buffer, path,
                        curr_max_point, curr_max_combination, sequence, points, max_points, matrix, hasVisited, minSeqSize, sequence_temp);
                    }
                    // return to previous state
                    path.pop_back(); // pop the current coordinate from path
                    sequence_temp.pop_back(); // pop the current coordinate from sequence_temp
                    hasVisited[next_row][curr_col] = false; // revert back to false for the current coordinate
                }
            }
        }
    }
}
```

Gambar 2.3.5 Prosedur untuk mencari token berikutnya untuk dimasukkan pada sebuah path, Fungsi ini akan dipanggil secara rekursif - bagian(2).

5. Prosedur allCombinations

```
void allCombinatations(const vector<vector<string>>& sequence, const vector<int>& points,
    const vector<vector<string>>& matrix, int* curr_max_point, vector <pair<int,int>>& curr_max_combination,int buffer){
// find all possible combinations in the matrix

    // KAMUS LOKAL
    int row = matrix.size(), col = matrix[0].size(), max_points = 0;
    int k, i ;
    int minSeqSize;
    vector<vector<bool>> hasVisited(row, vector<bool>(col, false));
    vector<string> sequence_temp;

    // ALGORITMA
    for (i = 0 ; i < points.size(); i++){
        if (points[i] >= 0){
            max_points += points[i];
            }
    }
    // minimum sequence size :

    for (k = 0 ; k < sequence.size(); k++){
        if (k == 0){
            minSeqSize = sequence[0].size();
        }
        else{
            if (sequence[k].size() < minSeqSize){
                minSeqSize = sequence[k].size();
            }
        }
        if (minSeqSize == 2){
            break;
        }
    }
```

Gambar 2.3.6 Prosedur untuk mencari kombinasi susunan token pada sebuah matrix token - bagian (1).

```
    for (int top_col = 0 ; top_col < col ; ++top_col){
        hasVisited[0][top_col] = true;
        sequence_temp.emplace_back(matrix[0][top_col]);
        vector<pair<int, int>> path = {{0, top_col}};
        next_choice(0,top_col,row,col,false,buffer,path,curr_max_point,curr_max_combination,sequence,points,
            max_points,matrix,hasVisited, minSeqSize,sequence_temp);
        hasVisited[0][top_col] = false;
        sequence_temp.pop_back();
    }
}
```

Gambar 2.3.7 Prosedur untuk mencari kombinasi susunan token pada sebuah matrix token - bagian (2).

2.4 *File Header Input/Output*

Pada *file input /output* (IO.cpp) terdapat fungsi - fungsi dan prosedur - prosedur yang telah didefinisikan pada file header (IO.hpp) sebagai berikut :

```
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <sstream>
#include <random>
using namespace std;

bool areTokensUnique(vector<string> tokens);
    // check if all tokens are unique
void displayResult(int curr_max_point, vector<pair<int,int>> curr_max_combination, vector<vector<string>> matrix, int time);
    // Display final result
void printPath(vector<pair<int, int>> path);
    // display path (coordinates of each token in said path)
void printSequence(vector<string> sequence);
    // Display sequence
void printMatrix(const vector<vector<string>> matrix);
    // Display matrix
void printSequences(vector<vector<string>> sequences, vector<int> points);
    // Display sequences and each of their points
string removeNewLine(string line);
    // remove new line in a string
vector<string> stringSeperator(string sentence);
    // Seperate string with " " as the delimeter
void display_menu(int* input);
    // Display menui (options the user can choose)
bool randomInput(vector<vector<string>>& sequences ,vector<vector<string>>& matrix, vector<int>& points, int* buffer);
    // Menu if user wants to automatically create matrix, sequences, and points
bool readFile(vector<vector<string>>& sequences ,vector<vector<string>>& matrix, vector<int>& points, int* buffer );
    // read .txt file
void write_file(int curr_max_point, vector <pair<int,int>> curr_max_combination, vector<vector<string>> matrix, int time);
    // write results to a txt file
```

Gambar 2.4.1 Tampilan *file header input/output.*

2.5 *File Input/Output*

File input/output (IO.cpp) merupakan file yang berisi fungsi - fungsi dan prosedur - prosedur yang berhubungan dengan *input* pengguna dan *output* hasil. Berikut implementasi file tersebut :

1. Fungsi areTokensUnique

```
bool areTokensUnique(vector<string> tokens)
// To make sure all tokens are unique
{
    // KAMUS LOKAL
    int i, k;

    // ALGORITMA
    for (i = 0 ; i < tokens.size() ; i++){
        for (k = i + 1; k < tokens.size() ; k++){
            if (tokens[i] == tokens[k]){
                return false;
            }
        }
    }
    return true;
}
```

Gambar 2.5.1 Fungsi untuk memastikan semua token masukkan user unik.

2. Prosedur displayResult

```
void displayResult(int curr_max_point, vector <pair<int,int>> curr_max_combination, vector<vector<string>> matrix, int time)
// Function to display final result
{
    // KAMUS LOKAL
    char input;
    vector<string> sequence;

    // ALGORITMA
    if (curr_max_point <= 0){
        cout << "Result: " << endl;
        cout << "No Solution" << endl;
        cout << endl;
        cout << time << "ms" << endl << endl;
        cout << "Apakah ingin menyimpan solusi? (y/n) : ";
        cin >> input;
        if (input == 'y'){
            write_file(curr_max_point,curr_max_combination,matrix,time);
        }
    }
    else{
        cout << "Result: " << endl;
        cout << curr_max_point << endl;
        sequence = pathToSequence(curr_max_combination,matrix);
        printSequence(sequence);
        printPath(curr_max_combination);
        cout << endl;
        cout << time << "ms" << endl << endl;
        cout << "Apakah ingin menyimpan solusi? (y/n) : ";
        cin >> input;
        if (input == 'y'){
            write_file(curr_max_point,curr_max_combination,matrix,time);
        }
    }
}
```

Gambar 2.5.2 Prosedur yang menampilkan hasil akhir.

3. Prosedur printPath

```
void printPath(vector<pair<int, int>> path)
// Function to print path (coordinate)
{
    // KAMUS LOKAL
    int i;

    // ALGORITMA
    cout << "Coordinates (col,row): " << endl;
    for (i = 0 ; i < path.size() ; i++){
        cout << path[i].second  + 1 << ", " << path[i].first  + 1 << endl;
    }
}
```

Gambar 2.5.3 Prosedur yang menampilkan *path* (susunan token dalam bentuk kooridnat).

4. Prosedur printSequence

```
void printSequence(vector<string> sequence)
// Function to print sequence
{
    // KAMUS LOKAL
    int i;

    // ALGORITMA
    cout << "Optimal Sequence : ";
    for (i = 0 ; i < sequence.size() ; i++){
        cout << sequence[i] << " ";

    }
    cout << endl;
}
```

Gambar 2.5.4 Prosedur yang menampilkan *sequence* (susunan token dalam bentuk string).

5. Prosedur printMatrix

```
void printMatrix(const vector<vector<string>> matrix)
// Function to print matrix
{
    // KAMUS LOKAL
    int row = matrix.size();
    int col = matrix[0].size();
    int curr_row, curr_col;
    // ALGORITMA
    cout << "Matrix : " << endl;
    for (curr_row = 0 ; curr_row < row ; curr_row++){
        for (curr_col = 0 ; curr_col < col; curr_col++){
            cout << matrix[curr_row][curr_col] << " ";
        }
        cout << endl;
    }
}
```

Gambar 2.5.5 Prosedur menampilkan matrix.

6. Prosedur printSequences

```
void printSequences(vector<vector<string>> sequences, vector<int> points)
// Print each sequence in sequences
{
    // KAMUS LOKAL
    int rowseq = sequences.size();
    int curr_row, curr_col;

    // ALGORITMA
    for (curr_row = 0 ; curr_row < rowseq ; curr_row++){
        cout << "Sequence " << curr_row + 1 << "    : ";
        for (curr_col = 0 ; curr_col < sequences[curr_row].size(); curr_col++){
            cout << sequences[curr_row][curr_col] << " ";
        }
        cout << endl;
        cout << "Point " << curr_row +1 << " : " << points[curr_row] << endl;
    }

}
```

Gambar 2.5.6 Prosedur menampilkan *sequences* yang telah dibuat secara otomatis oleh program.

7. Fungsi removeNewLine

```
string removeNewLine(string line)
// Function to remove new line in a string
{
    // KAMUS LOKAL

    // ALGORITMA
    while(line[line.size()-1] == '\n' ||line[line.size()-1] == '\r' ){
            line = line.substr(0,line.size()-1);
        }
    return line;
}
```

Gambar 2.5.7 Fungsi yang menghapus *new line* pada sebuah string.

8. Fungsi stringSeperator

```
vector<string> stringSeperator(string sentence)
// Function to seperate string with " " as delimeter
{
    // KAMUS LOKAL
    string token;
    istringstream iss(sentence);
    vector<string> result;

    // ALGORITMA
    while (iss >> token) {
        result.push_back(token);
    }
    return result;
}
```

Gambar 2.5.8 Fungsi yang mengubah sebuah string menjadi vector string dengan spasi sebagai delimiter.

9. Prosedur display_menu

```
void display_menu(int* input)
// Function to display menu
{
    // KAMUS LOKAL
    char clearBuff[80];
    bool valid = false;

    // ALGORITMA
    cout << "================================================================" << endl;
    cout << "INPUT OPTIONS : " << endl;
    cout << "1.Text file" << endl;
    cout << "2.Command Line" << endl;
    cout << "3.Exit" << endl;
    *input = 0;
    do {
        cout << "input : " ;
        cin >> *input;
        if (cin.fail()){
            cout << "That's not an integer !" << endl;
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
        }
```

Gambar 2.5.9 Prosedur untuk menampilkan menu utama program - bagian (1)

```
        else {
        }
        if (*input == 1){
            valid = true;
        }
        else if (*input == 2){
            valid = true;
        }
        else if (*input == 3){
            valid = true;
        }
        else {
            cout << "Please enter the correct input type !" << endl;
        }
    }while (!valid);
}
```

Gambar 2.5.10 Prosedur untuk menampilkan menu utama program - bagian (2)

10. Fungsi randomInput

```
bool randomInput(vector<vector<string>>& sequences ,vector<vector<string>>& matrix, vector<int>& points, int* buffer){

    // KAMUS LOKAL
    int total_token, row, col, total_sequences,maximum_sequence_size;
    string token_temp;
    vector<string> tokens;
    bool unique = false;

    // ALGORITMA
    cout << "================================================================" << endl;
    try {
    cout << "Command Line Input" << endl;
    cout << "number of unique token     : ";
    cin >> total_token;
    if (cin.fail()){
        cout << "Something Went Wrong !" << endl;
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        return false;
    }
```

Gambar 2.5.11 Fungsi untuk menghasilkan matrix, *sequences*, dan poin secara otomatis berdasarkan masukkan *user*, mengembalikan *true* jika fungsi berjalan dengan benar - bagian (1).

```
do {
    vector<string> tokensTemp;
    cout << "Enter all tokens           : ";
    string temp_token;
    for (int token = 0 ; token < total_token ; token++){
    cin >> temp_token;
    if (cin.fail()){
    cout << "Something Went Wrong !" << endl;
    cin.clear();
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
    return false;
    }
    tokensTemp.emplace_back(temp_token);
}
unique = areTokensUnique(tokensTemp);
if (!unique){
    cout << "Please make sure all tokens are unique !" << endl;
}
else {
    tokens = tokensTemp;
}
} while (!unique);
```

Gambar 2.5.12 Fungsi untuk menghasilkan matrix, *sequences*, dan poin secara otomatis berdasarkan masukkan *user*, mengembalikan *true* jika fungsi berjalan dengan benar - bagian (2).

```
cout << "Buffer Size               : ";
cin >> *buffer;
if (cin.fail()){
    cout << "Something Went Wrong !" << endl;
    cin.clear();
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
    return false;
}

cout << "Matrix Size (column row)   : ";
cin >> col;
if (cin.fail()){
    cout << "Something Went Wrong !" << endl;
    cin.clear();
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
    return false;
}

cin >> row;
if (cin.fail()){
    cout << "Something Went Wrong !" << endl;
    cin.clear();
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
    return false;
}
```

Gambar 2.5.13 Fungsi untuk menghasilkan matrix, *sequences*, dan poin secara otomatis berdasarkan masukkan *user*, mengembalikan *true* jika fungsi berjalan dengan benar - bagian (3).

```
cout << "Number of Sequences        : ";
cin >> total_sequences;
if (cin.fail()){
    cout << "Something Went Wrong !" << endl;
    cin.clear();
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
    return false;
}
cout << "Maximum sequence size      : ";
cin >> maximum_sequence_size;
if (cin.fail()){
    cout << "Something Went Wrong !" << endl;
    cin.clear();
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
    return false;
}

random_device rd;
mt19937 engine(rd());
uniform_int_distribution<int> dist(0, total_token-1);
// generating matrix
for (int curr_row = 0 ; curr_row < row; curr_row++){
    vector<string> aRow;
    for (int curr_col = 0 ; curr_col < col ; curr_col++){
        aRow.emplace_back(tokens[dist(engine)]);
    }
    matrix.emplace_back(aRow);
}
```

Gambar 2.5.14 Fungsi untuk menghasilkan matrix, *sequences*, dan poin secara otomatis berdasarkan masukkan *user*, mengembalikan *true* jika fungsi berjalan dengan benar - bagian (4).

```
// generating sequence and points
random_device rd2;
mt19937 engine2(rd2());
uniform_int_distribution<int> dist2(2, maximum_sequence_size);

random_device rd3;
mt19937 engine3(rd3());
uniform_int_distribution<int> dist3(3, 10);
int repeats;

for (int curr_sequence = 0 ; curr_sequence < total_sequences ; curr_sequence++){
    vector<string> aSeq1;
    repeats = 0;
    do {
    vector<string> aSeq;
    int random_size_sequence = dist2(engine2);
    for (int size = 0 ;size <  random_size_sequence; size++ ){
        int random_token_index = dist(engine);
        aSeq.emplace_back(tokens[random_token_index]);
    aSeq1 = aSeq;
    }
    repeats++;
    if (repeats > 100){
        cout << "Something Went Wrong !" << endl;
        cout << "With the current inputs, creating unique sequences is impossible !" << endl;
        return false;
    }
        } while (sequenceInSequences(aSeq1,sequences));
    sequences.emplace_back(aSeq1);
    points.emplace_back(dist3(engine3) * 5);
}
```

Gambar 2.5.15 Fungsi untuk menghasilkan matrix, *sequences*, dan poin secara otomatis berdasarkan masukkan *user*, mengembalikan *true* jika fungsi berjalan dengan benar - bagian (5).

```
    } catch(const runtime_error &e){
        cerr << "Caught a runtime error: " << e.what() << endl ;
        return false;
    }
    catch (...){
        cerr << "Something Went Wrong" << endl;
        return false;
    }
    return true;
}
```

Gambar 2.5.16 Fungsi untuk menghasilkan matrix, *sequences*, dan poin secara otomatis berdasarkan masukkan *user*, mengembalikan *true* jika fungsi berjalan dengan benar - bagian (6).

11. Fungsi readFile

```
bool readFile(vector<vector<string>>& sequences ,vector<vector<string>>& matrix, vector<int>& points, int* buffer )
// Function to read file
{
    // KAMUS LOKAL
    string fileNameTemp, line;
    string fileName = "test/";
    fstream myFile;
    int row,col,total_sequence, k, point,i;
    vector<string>row_col ;
    vector<string> aRow, aSequence;

  // ALGORITMA
    cout << "================================================================" << endl;
    cout << "Enter File Name : ";
    cin >> fileNameTemp;
    fileName.append(fileNameTemp);
    ifstream file(fileName);


    while (!file) {
        cout << "File doesn't exist!, make sure the file is in the test folde and the working folder is correct !" << endl;
        file.close();
        cout << "Enter File Name : ";
        cin >> fileNameTemp;
        fileName = "test/";
        fileName.append(fileNameTemp);
        file.open(fileName);
    }
    file.close();
```

Gambar 2.5.17 Fungsi memproses file, mengembalikan *true* jika fungsi berjalan dengan benar - bagian (1).

```
    try{
        myFile.open(fileName,ios::in);
    if (myFile.is_open()){
        getline(myFile,line);
        line = removeNewLine(line);
        *buffer = stoi(line);
        getline(myFile,line);
        row_col = stringSeperator(line);
        col = stoi(removeNewLine(row_col[0]));
        row = stoi(removeNewLine(row_col[1]));
        for (i = 0 ; i < row; i++){
            getline(myFile,line);
            aRow = stringSeperator(line);
            matrix.emplace_back(aRow);
        }
        getline(myFile,line);
        total_sequence = stoi(removeNewLine(line));
        for (k = 0 ; k < total_sequence ; k++){
            point;
            getline(myFile,line);
            aSequence = stringSeperator(line);
            sequences.push_back(aSequence);
            getline(myFile,line);
            points.push_back(stoi(removeNewLine(line)));
        }
        myFile.close();
        }
    }
    catch (const std::runtime_error &e){
        cerr << "Caught a runtime error: " << e.what() << endl;
        return false;
    }
    catch (...){
        cerr << "Something went wrong" << endl;
        return false;
        }
    return true;
}
```

Gambar 2.5.18 Fungsi memproses file, mengembalikan *true* jika fungsi berjalan dengan benar - bagian (2).

12. Prosedur write_file

```
void write_file(int curr_max_point, vector <pair<int,int>> curr_max_combination, vector<vector<string>> matrix, int time)
// Write result to a .txt file
{
    // KAMUS LOKAL
    string fileNameTemp;
    string fileName = "test/";
    vector<string> sequence;
    int i, k;

    // ALGORTIMA
    cout << "============================================================" << endl;
    cout << "SAVE SOLUTION" << endl;
    cout << "File Name (With extension) : ";
    cin >> fileNameTemp;
    fileName.append(fileNameTemp);
    ifstream file(fileName);

    while (file) {
        cout << "File already exist, please rename the file !" << endl;
        file.close();
        cout << "File Name (With extension) : ";
        cin >> fileNameTemp;
        fileName = "test/";
        fileName.append(fileNameTemp);
        file.open(fileName);
    }
    file.close();
    fstream myFile;
    myFile.open(fileName,ios::out);
```

Gambar 2.5.19 Prosedur menulis hasil akhir ke dalam file .txt - bagian (1).

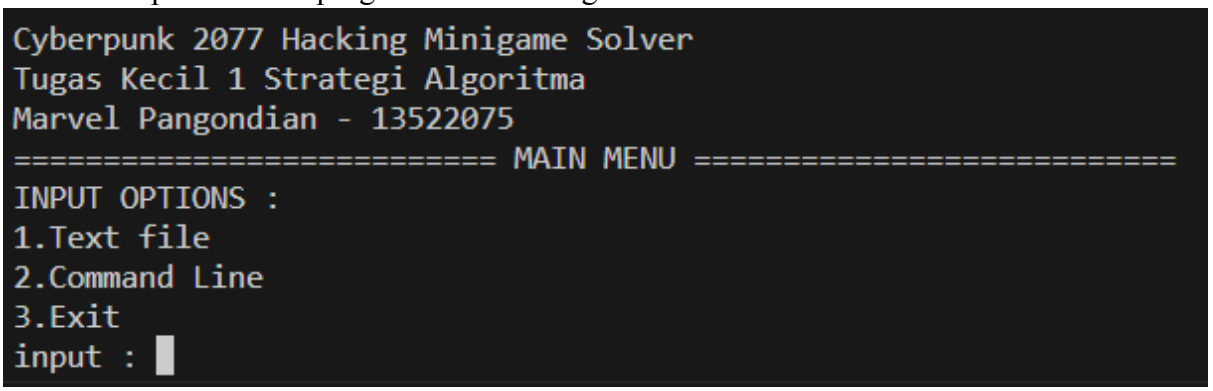
```
    if (myFile.is_open()){
        if (curr_max_point <= 0){
            myFile << "No Solution\n";
            myFile << "\n";
            myFile << time <<"ms";
        }
        else{
            myFile << curr_max_point << "\n";
            sequence = pathToSequence(curr_max_combination,matrix);
            for(i = 0 ; i < sequence.size(); i++){
                myFile << sequence[i] << " ";
            }
            myFile << "\n";
            for (k = 0 ; k < curr_max_combination.size() ; k++){
                myFile << curr_max_combination[k].second +1 << ", " << curr_max_combination[k].first + 1;
                myFile << "\n";
            }
            myFile << "\n";
            myFile << time <<"ms";
        }
        myFile.close();
    }
}
```

Gambar 2.5.20 Prosedur menulis hasil akhir ke dalam file .txt - bagian (2).

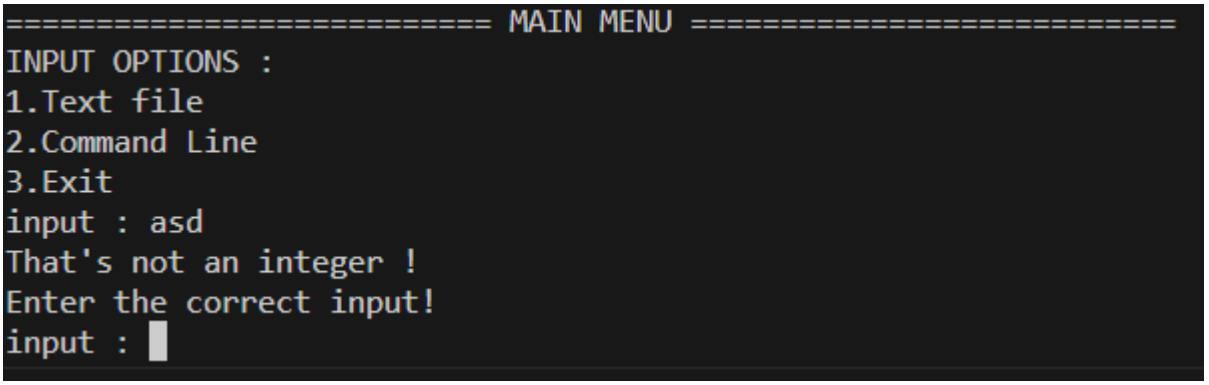
Bab III
TEST

3.1 Tampilan Utama

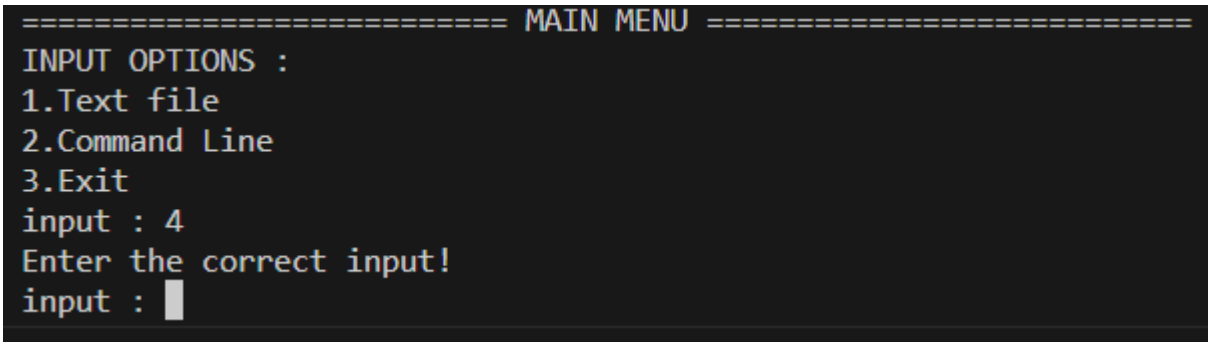
Tampilan Utama program adalah sebagai berikut :



Gambar 3.1.1 Tampilan Utama Program.



Gambar 3.1.2 Tampilan program jika input tidak integer.



Gambar 3.1.3 Tampilan program jika input tidak ada yang sesuai.

3.2 Tampilan dan Skema Validasi Input Text File

```
Cyberpunk 2077 Hacking Minigame Solver
Tugas Kecil 1 Strategi Algoritma
Marvel Pangondian - 13522075
========================== MAIN MENU ==========================
INPUT OPTIONS :
1.Text file
2.Command Line
3.Exit
input : 1
========================== TEXT FILE ==========================
Enter File Name : testWHAT.txt
File doesn't exist!, make sure the file is in the test folder and the working folder is correct !

Enter File Name :
```

Gambar 3.2.1 Tampilan ketika file yang dicari tidak ada.

```
========================== MAIN MENU ==========================
INPUT OPTIONS :
1.Text file
2.Command Line
3.Exit
input : 1
========================== TEXT FILE ==========================
Enter File Name : testempty.txt
Something went wrong
========================== MAIN MENU ==========================
INPUT OPTIONS :
1.Text file
2.Command Line
3.Exit
input :
```

Gambar 3.2.2 Tampilan ketika file kosong.

```
========================== MAIN MENU ==========================
INPUT OPTIONS :
1.Text file
2.Command Line
3.Exit
input : 1
========================== TEXT FILE ==========================
Enter File Name : testWrong.txt
Something is wrong with the matrix in the .txt file
========================== MAIN MENU ==========================
INPUT OPTIONS :
1.Text file
2.Command Line
3.Exit
input :
```

Gambar 3.2.3 Tampilan ketika ukuran matrix tidak sesuai.

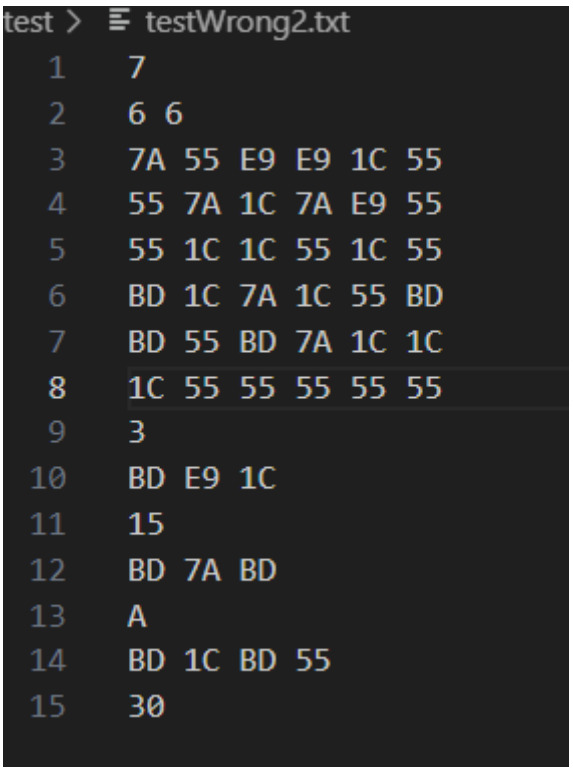
test > testWrong.txt

```
7
6 6
7A 55 E9 E9 1C 55
55 7A 1C 7A E9 55
55 1C 1C 55
BD 1C 7A 1C 55 BD
BD 55 BD 7A 1C 1C
1C 55 55
3
BD E9 1C
15
BD 7A BD
20
BD 1C BD 55
30
```

Gambar 3.2.4 .txt file yang digunakan untuk percobaan pada gambar 3.2.3.

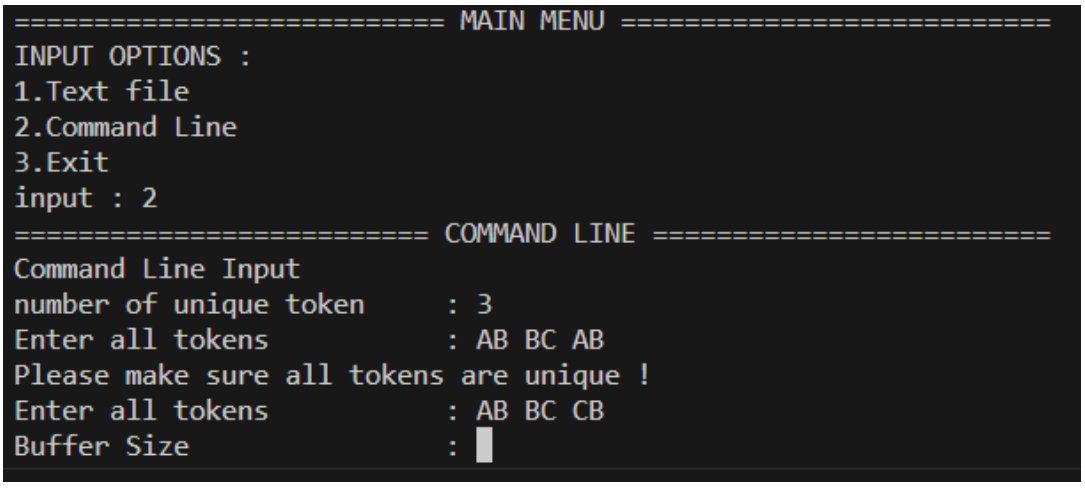
```
========================== MAIN MENU ==========================
INPUT OPTIONS :
1.Text file
2.Command Line
3.Exit
input : 1
========================== TEXT FILE ==========================
Enter File Name : testWrong2.txt
Something went wrong
========================== MAIN MENU ==========================
INPUT OPTIONS :
1.Text file
2.Command Line
3.Exit
input :
```

Gambar 3.2.5 Tampilan ketika isi dari .txt ada yang salah.



Gambar 3.2.6 .txt file yang digunakan untuk percobaan pada gambar 3.2.5 (pada line 14, seharusnya integer bukan character).

3.3 Tampilan dan Skema Validasi Input Command Line



Gambar 3.3.1 Tampilan ketika input token tidak unik

```
========================== COMMAND LINE ==========================
Command Line Input
number of unique token      : ABCC
Something Went Wrong !
============================ MAIN MENU ============================
INPUT OPTIONS :
1.Text file
2.Command Line
3.Exit
input :
```

Gambar 3.3.2 Tampilan ketika input tidak integer

```
============================ MAIN MENU ============================
INPUT OPTIONS :
1.Text file
2.Command Line
3.Exit
input : 2
========================== COMMAND LINE ==========================
Command Line Input
number of unique tokens     : 0
Please input a number above 0
```

Gambar 3.3.3 Tampilan ketika input tidak sesuai

```
Cyberpunk 2077 Hacking Minigame Solver
Tugas Kecil 1 Strategi Algoritma
Marvel Pangondian - 13522075
============================ MAIN MENU ============================
INPUT OPTIONS :
1.Text file
2.Command Line
3.Exit
input : 2
========================== COMMAND LINE ==========================
Command Line Input
number of unique tokens     : 2
Enter all tokens            : AB BC
Buffer Size                 : 5
Matrix Size (column row)    : 5 6
Number of Sequences         : 10
Maximum sequence size       : 3
Something Went Wrong !
With the current inputs, creating unique sequences is impossible !
```

Gambar 3.3.4 Tampilan ketika input tidak dapat menghasilkan *sequences* yang unik

3.4 *Test Case* dengan *File .txt*

1.

Input

```
7
6 6
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
3
BD E9 1C
15
BD 7A BD
20
BD 1C BD 55
30
```

Output In Terminal

```
============================ TEXT FILE ============================
Enter File Name : test1.txt
============================ RESULT ============================
Result:
50
Optimal Sequence : 7A BD 7A BD 1C BD 55
Coordinates (col,row):
1, 1
1, 4
3, 4
3, 5
6, 5
6, 3
1, 3

49ms
```

Output In File

```
50
7A BD 7A BD 1C BD 55
1, 1
1, 4
3, 4
3, 5
6, 5
6, 3
1, 3

49ms
```

2.

Input

```
4
3 4
7A 55 E9
55 7A 1C
55 1C 1C
BD 1C 7A
3
BD E9 1C
15
BD 7A BD
20
BD 1C BD 55
30
```

Output In Terminal

```
============================ TEXT FILE ============================
Enter File Name : test2.txt
============================ RESULT ============================
No Solution

0ms
```

Output In File

```
No Solution

0ms
```

3.

Input

```
4
5 5
1C 1C 55 E9 55
1C 1C E9 55 55
BD 55 55 1C BD
1C E9 55 BD BD
55 1C 55 E9 55
3
1C 55 55
15
1C 55
20
55 BD E9 1C
30
```

Output In Terminal

```
============================ TEXT FILE ============================
Enter File Name : test3.txt
============================ RESULT ============================
Result:
35
Optimal Sequence : 1C 55 55
Coordinates (col,row):
1, 1
1, 5
3, 5
```

Output In File

```
35
1C 55 55
1, 1
1, 5
3, 5

0ms
```

4.

Input

```
8
6 6
E9 1C E9 E9 1C BD
55 55 E9 E9 1C 55
55 7A 1C BD 55 55
E9 7A 1C BD E9 BD
1C 55 E9 1C 1C BD
55 1C 7A 1C E9 7A
3
1C E9 1C
15
E9 E9 BD
20
BD 7A E9 1C
30
```

Output In Terminal

```
========================== TEXT FILE ==========================
Enter File Name : test4.txt
========================== RESULT ==========================
Result:
65
Optimal Sequence : E9 E9 BD 7A E9 1C E9 1C
Coordinates (col,row):
1, 1
1, 4
6, 4
6, 6
5, 6
5, 1
3, 1
3, 3

266ms
```

Output In File

```
65
E9 E9 BD 7A E9 1C E9 1C
1, 1
1, 4
6, 4
6, 6
5, 6
5, 1
3, 1
3, 3

266ms
```

5.

Input

```
5
6 6
BD 1C 1C 1C BD 1C
55 55 7A 55 55 BD
E9 55 BD 1C 7A 1C
BD 7A BD 55 BD 55
7A E9 7A E9 E9 1C
55 1C 7A 1C 55 7A
3
55 1C
15
55 55 BD
20
55 7A 1C BD
30
```

Output In Terminal

```
============================ TEXT FILE ============================
Enter File Name : test5.txt
============================ RESULT ============================
Result:
30
Optimal Sequence : BD 55 7A 1C BD
Coordinates (col,row):
1, 1
1, 2
3, 2
3, 1
5, 1

1ms
```

Output In File

```
30
BD 55 7A 1C BD
1, 1
1, 2
3, 2
3, 1
5, 1

1ms
```

6.

Input

```
10
6 6
BD 1C BD E9 55 BD
1C 7A 1C 1C 1C 55
1C 1C 1C E9 55 1C
7A 55 BD 7A 55 E9
55 7A 55 E9 55 7A
7A BD BD 55 1C 55
3
1C BD E9
45
7A BD 1C
15
7A 1C E9 55
50
```

Output In Terminal

```
========================== TEXT FILE ==========================
Enter File Name : test6.txt
========================== RESULT ==========================
Result:
110
Optimal Sequence : BD 7A BD 1C BD E9 7A 1C E9 55
Coordinates (col,row):
1, 1
1, 6
2, 6
2, 1
6, 1
6, 4
1, 4
1, 3
4, 3
4, 6

5596ms
```

Output In File

```
110
BD 7A BD 1C BD E9 7A 1C E9 55
1, 1
1, 6
2, 6
2, 1
6, 1
6, 4
1, 4
1, 3
4, 3
4, 6

5596ms
```

7.

Input

```
10
6 6
BD 55 1C 7A E9 7A
E9 E9 1C 7A 55 E9
BD E9 E9 BD 1C BD
1C 1C BD 7A BD E9
BD BD BD BD 7A BD
55 1C E9 1C 55 55
3
BD 7A 1C 1C
35
7A 1C E9
45
E9 7A 1C 7A
50
```

Output In Terminal

```
============================ TEXT FILE ============================
Enter File Name : test7.txt
============================ RESULT ============================
Result:
80
Optimal Sequence : 7A 7A 1C E9 BD 7A 1C 1C
Coordinates (col,row):
4, 1
4, 2
3, 2
3, 3
4, 3
4, 4
2, 4
2, 6

5736ms
```

Output In File

```
80
7A 7A 1C E9 BD 7A 1C 1C
4, 1
4, 2
3, 2
3, 3
4, 3
4, 4
2, 4
2, 6

5736ms
```

8.

Input

```
8
7 6
CD 1F BC TR AB CD TR
CD TR TR AB 1F BC CD
1F 1F BC AB CD BC BC
CD BC AB AB TR TR AB
1F 1F CD BC AB BC AB
CD TR TR BC 1F TR CD
4
TR AB CD BC
35
TR 1F AB 1F
40
BC CD BC
25
BC TR TR CD 1F
50
```

Output In Terminal

```
========================== TEXT FILE ==========================
Enter File Name : test8.txt
============================ RESULT ============================
Result:
75
Optimal Sequence : BC CD BC TR TR CD 1F
Coordinates (col,row):
3, 1
3, 5
4, 5
4, 1
7, 1
7, 2
5, 2

696ms
```

Output In File

```
75
BC CD BC TR TR CD 1F
3, 1
3, 5
4, 5
4, 1
7, 1
7, 2
5, 2

696ms
```

9.

Input

```
8
10 10
BC CD BC AB AB GH BC GH AB CD
BC AB CD CD GH CD GH BC AB EF
CD AB BC BC BC BC AB GH BC AB
CD GH GH GH EF CD BC EF EF EF
EF BC GH AB CD CD BC AB EF BC
BC CD AB EF BC GH CD EF AB EF
GH BC EF CD AB AB AB CD GH GH
AB BC AB GH AB BC CD GH GH CD
BC CD CD GH AB CD GH CD BC AB
BC BC AB GH AB CD CD AB CD EF
5
BC GH EF GH AB EF
20
EF AB EF GH EF
30
GH AB AB EF CD EF
40
BC EF
50
BC AB EF BC
55
```

Output In Terminal

```
============================ TEXT FILE ============================
Enter File Name : brutal.txt
============================ RESULT ============================
Result:
105
Optimal Sequence : BC AB EF BC EF
Coordinates (col,row):
3, 1
3, 6
8, 6
8, 2
10, 2

45653ms
```

Output In File

```
105
BC AB EF BC EF
3, 1
3, 6
8, 6
8, 2
10, 2

45653ms
```

3.5 *Test Case* dengan *Command Line*

1.

Input

```
============================ MAIN MENU ============================
INPUT OPTIONS :
1.Text file
2.Command Line
3.Exit
input : 2
========================= COMMAND LINE =========================
Command Line Input
number of unique tokens    : 4
Enter all tokens           : AB BC CD EF
Buffer Size                : 5
Matrix Size (column row)   : 5 5
Number of Sequences        : 5
Maximum sequence size      : 5
```

Output In Terminal

```
MATRIX :
BC EF EF EF AB
EF EF AB BC AB
CD EF AB CD CD
AB CD AB AB CD
EF BC AB CD AB

 SEQUENCES :
Sequence 1   : EF CD
Point 1 : 629
Sequence 2   : AB CD AB EF EF
Point 2 : 284
Sequence 3   : CD AB EF CD
Point 3 : 977
Sequence 4   : AB BC CD EF
Point 4 : 307
Sequence 5   : EF EF EF AB
Point 5 : 251

============================== RESULT ==============================
Result:
1606
Optimal Sequence : EF CD AB EF CD
Coordinates (col,row):
2, 1
2, 4
1, 4
1, 5
4, 5

1ms
```

Output In File

```
1606
EF CD AB EF CD
2, 1
2, 4
1, 4
1, 5
4, 5

1ms
```

2.

Input

```
========================== MAIN MENU ==========================
INPUT OPTIONS :
1.Text file
2.Command Line
3.Exit
input : 5
Enter the correct input!
input : 2
========================= COMMAND LINE =========================
Command Line Input
number of unique tokens    : 6
Enter all tokens           : AB BC CD EF GH TY
Buffer Size                : 7
Matrix Size (column row)   : 7 8
Number of Sequences        : 8
Maximum sequence size      : 6
```

Output In Terminal

```
MATRIX :
TY AB GH TY BC AB CD
GH BC CD CD TY EF CD
GH GH TY CD AB CD TY
TY GH AB TY CD AB BC
GH BC EF AB GH BC BC
AB EF EF CD BC BC EF
TY BC BC EF TY TY AB
EF AB EF TY EF BC TY

 SEQUENCES :
Sequence 1   : AB CD EF AB
Point 1 : 546
Sequence 2   : GH EF TY
Point 2 : 528
Sequence 3   : CD BC CD BC TY GH
Point 3 : 899
Sequence 4   : TY GH TY CD EF
Point 4 : 499
Sequence 5   : TY EF CD
Point 5 : 900
Sequence 6   : EF GH CD EF AB
Point 6 : 603
Sequence 7   : BC BC AB
Point 7 : 886
Sequence 8   : CD BC AB
Point 8 : 515

============================== RESULT ==============================
Result:
1786
Optimal Sequence : BC BC AB TY EF CD
Coordinates (col,row):
5, 1
5, 6
1, 6
1, 7
4, 7
4, 2

699ms
```

Output In File

```
1786
BC BC AB TY EF CD
5, 1
5, 6
1, 6
1, 7
4, 7
4, 2

699ms
```

3.

Input

```
========================== MAIN MENU ===========================
INPUT OPTIONS :
1.Text file
2.Command Line
3.Exit
input : 2
======================== COMMAND LINE ========================
Command Line Input
number of unique tokens    : 4
Enter all tokens           : AB BC CD EF
Buffer Size                : 8
Matrix Size (column row)   : 8 8
Number of Sequences        : 6
Maximum sequence size      : 7
```

Output In Terminal

```
MATRIX :
BC CD BC CD BC BC CD BC
AB AB BC BC EF AB AB BC
EF EF CD CD BC EF EF EF
BC CD CD CD CD EF AB CD
EF CD BC CD BC BC AB EF
BC CD AB AB AB CD CD BC
BC BC AB CD EF BC CD BC
EF CD AB BC AB AB BC EF

 SEQUENCES :
Sequence 1   : BC CD AB BC CD AB BC
Point 1 : 771
Sequence 2   : EF AB BC AB
Point 2 : 767
Sequence 3   : BC EF EF CD EF CD EF
Point 3 : 112
Sequence 4   : AB BC EF BC EF EF BC
Point 4 : 607
Sequence 5   : BC CD EF CD BC BC
Point 5 : 935
Sequence 6   : EF BC CD AB EF
Point 6 : 528

============================== RESULT ==============================
Result:
935
Optimal Sequence : BC CD EF CD BC BC
Coordinates (col,row):
3, 1
3, 3
2, 3
2, 1
1, 1
1, 4

8192ms
```

Output In File

```
935
BC CD EF CD BC BC
3, 1
3, 3
2, 3
2, 1
1, 1
1, 4

8192ms
```

4.

Input

```
============================ MAIN MENU ============================
INPUT OPTIONS :
1.Text file
2.Command Line
3.Exit
input : 2
========================= COMMAND LINE =========================
Command Line Input
number of unique tokens    : 4
Enter all tokens           : AB BC CD EF
Buffer Size                : 3
Matrix Size (column row)   : 10 10
Number of Sequences        : 5
Maximum sequence size      : 6
```

Output In Terminal

```
MATRIX :
CD BC CD CD AB EF BC BC CD AB
CD AB CD BC EF AB AB AB EF EF
BC EF AB EF BC CD EF AB CD AB
AB EF AB AB EF AB CD EF EF EF
BC BC BC EF CD EF BC CD AB CD
CD CD BC EF BC AB CD CD CD AB
CD BC CD CD AB BC CD CD CD CD
AB BC AB AB EF BC CD EF CD BC
AB CD AB EF AB BC BC BC CD BC
AB EF BC EF AB AB CD BC EF EF

 SEQUENCES :
Sequence 1   : AB EF EF CD
Point 1 : 944
Sequence 2   : BC EF CD BC BC EF
Point 2 : 378
Sequence 3   : BC CD
Point 3 : 409
Sequence 4   : AB BC BC AB BC
Point 4 : 179
Sequence 5   : AB BC BC BC AB AB
Point 5 : 821

============================== RESULT ==============================
Result:
409
Optimal Sequence : BC CD
Coordinates (col,row):
2, 1
2, 6

0ms
```

Output In File

```
409
BC CD
2, 1
2, 6

0ms
```

5.

Input

```
========================== MAIN MENU ==========================
INPUT OPTIONS :
1.Text file
2.Command Line
3.Exit
input : 2
======================== COMMAND LINE ========================
Command Line Input
number of unique tokens    : 5
Enter all tokens           : AB BC CD EF GH
Buffer Size                : 6
Matrix Size (column row)   : 6 6
Number of Sequences        : 6
Maximum sequence size      : 6
```

Output In Terminal

```
MATRIX :
GH EF AB CD BC AB
EF EF BC CD GH CD
GH EF AB AB BC CD
CD AB EF EF GH CD
GH BC BC CD BC CD
EF AB CD BC CD CD

 SEQUENCES :
Sequence 1   : AB CD
Point 1 : 582
Sequence 2   : GH EF EF BC CD
Point 2 : 419
Sequence 3   : CD GH CD
Point 3 : 868
Sequence 4   : CD CD CD EF AB GH
Point 4 : 299
Sequence 5   : EF GH AB BC CD EF
Point 5 : 439
Sequence 6   : AB BC
Point 6 : 944

============================ RESULT ============================
Result:
2394
Optimal Sequence : AB CD GH CD AB BC
Coordinates (col,row):
6, 1
6, 2
5, 2
5, 6
2, 6
2, 5

21ms
```

Output In File

```
2394
AB CD GH CD AB BC
6, 1
6, 2
5, 2
5, 6
2, 6
2, 5

21ms
```

6.

Input

```
========================== MAIN MENU ==========================
INPUT OPTIONS :
1.Text file
2.Command Line
3.Exit
input : 2
========================= COMMAND LINE =========================
Command Line Input
number of unique tokens    : 10
Enter all tokens           : AB BC CD EF GH HI JK LM NO ZQ
Buffer Size                : 8
Matrix Size (column row)   : 10 10
Number of Sequences        : 10
Maximum sequence size      : 5
```

Output In Terminal

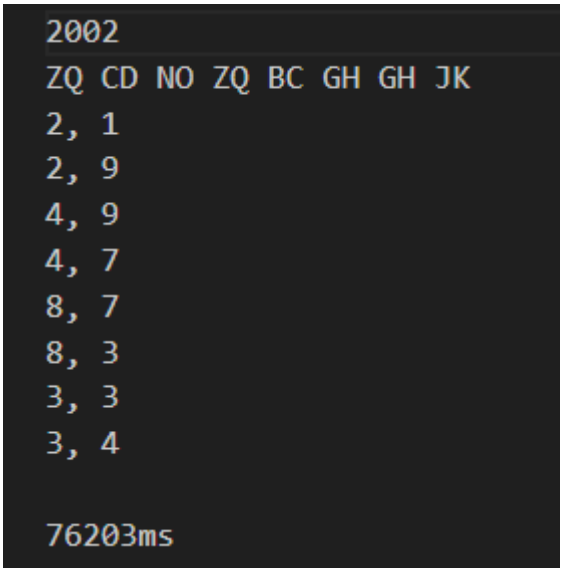
```
MATRIX :
LM ZQ AB GH NO AB EF HI CD NO
AB EF LM GH CD NO ZQ AB ZQ CD
EF EF GH HI BC JK CD GH AB CD
HI GH JK EF JK HI BC BC AB ZQ
HI EF ZQ GH ZQ NO HI BC AB EF
LM EF CD CD HI AB EF HI LM LM
EF ZQ JK ZQ CD GH AB BC NO AB
LM GH ZQ AB HI EF GH GH LM LM
NO CD AB NO NO HI EF EF EF CD
BC AB EF CD LM AB HI LM LM CD

 SEQUENCES :
Sequence 1   : EF GH CD
Point 1 : 177
Sequence 2   : CD NO
Point 2 : 175
Sequence 3   : BC ZQ ZQ GH BC
Point 3 : 798
Sequence 4   : NO ZQ BC
Point 4 : 832
Sequence 5   : ZQ HI JK AB ZQ
Point 5 : 820
Sequence 6   : ZQ HI CD
Point 6 : 546
Sequence 7   : LM ZQ ZQ EF ZQ
Point 7 : 973
Sequence 8   : CD JK
Point 8 : 143
Sequence 9   : CD JK HI CD JK
Point 9 : 420
Sequence 10   : GH GH JK
Point 10 : 995
```

```
============================ RESULT ============================
Result:
2002
Optimal Sequence : ZQ CD NO ZQ BC GH GH JK
Coordinates (col,row):
2, 1
2, 9
4, 9
4, 7
8, 7
8, 3
3, 3
3, 4

76203ms
```

Output In File



Bab IV
LAMPIRAN

4.1 *Link Repository*

https://github.com/MarvelPangondian/Tucil1_13522075

4.2 Tabel *Checklist*

Status : *Completed*

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 (**Bonus**)		✓