LAPORAN
TUGAS KECIL 1 IF2211
STRATEGI ALGORITMA

Penyelesaian *Cyberpunk 2077 Breach Protocol* dengan Algoritma *Brute Force*

Emery Fathan Zwageri 13522079

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG

2023/2024

Daftar Isi

Bagian 1: Latar Belakang

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.

Dengan aturan bermain dari permainan Breach Protocol adalah sebagai berikut:

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

Contoh bermain:

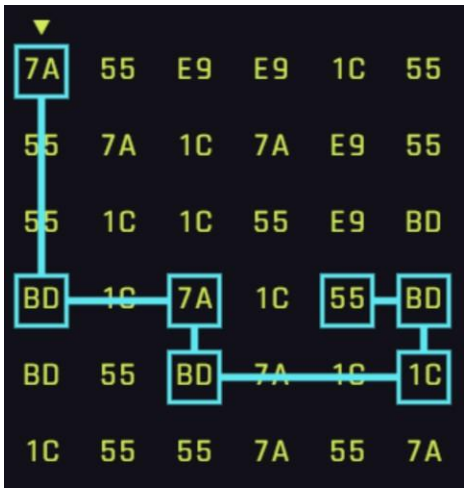
Diberikan matriks sebagai berikut dan ukuran buffernya adalah tujuh.

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

Dengan sekuens sebagai berikut:

1. BD E9 1C dengan hadiah berbobot 15.
2. BD 7A BD dengna hadiah berbobot 20.
3. BD 1C BD 55 dengan hadiah berbobot 30.

Maka solusi optimal yang bisa didapatkan untuk konfigurasi matriks dan sekuens tersebut adalah sebagai berikut



Gambar 1 Salah satu solusi optimal
(Sumber: https://cyberpunk-hacker.com/)

Dengan keterangan:

1. Pada awal mula permainan, pemain memilih baris 1 kolom 1,
2. Kemudian pada kolom 1, pemain memilih baris 4,
3. Lalu pada baris 4, pemain memilih kolom 3,
4. Selanjutnya pada kolom 3, pemain memilih baris 5,
5. Dilanjutkan dengan memilih kolom 6 pada baris 5,
6. Kemudian pada kolom 6, pemain memilih baris 4,
7. Terakhir, pemain memilih kolom 5 pada baris 4.

Sehingga, total bobot optimal adalah 50 dengan jumlah langkah sebanyak 6 (diambil dari dokumen Tugas Kecil 1 IF2211 Strategi Algoritma 2023/2024: Penyelesaian *Cyberpunk 2077 Breach Protocol* dengan Algoritma *Brute Force*). Saya, sebagai mahasiswa yang mengikuti perkuliahan Strategi Algoritma diminta untuk menemukan solusi paling optimal dari permainan ini untuk setiap kombinasi matriks, sekuens, dan ukuran buffer dengan menggunakan algoritma *brute force*.

Bagian 2: Algoritma *Brute Force*

Saya menggunakan Teknik backtracking dalam bruteforce untuk menyelesaikan masalah breach protocol ini. Solusi dimulai dari saat buffer berjumlah 0 lalu di extend hingga buffer penuh sesuai dengan aturan permainan breach protocol.

1. pertama tama reward optimal diatur menjadi 0.

2. program akan dipanggil untuk setiap kolom pada baris pertama.

3. awalmulanya buffer pada parameter kosong. Setiap fungsi dipanggil maka dia akan mengecek buffer yang dibawa apakah reward yang didapatkan melebihi reward terbesar yang sudah didapat. Jika ya maka Vector yang menyimpan buffer terbaik sejauh ini akan diganti dengan buffer yang dibawa, begitu juga dengan path.Lalu token pada matrix dipush sesuai kordinat matrix pada parameter x dan y lalu algortima secara rekursif memanggil fungsi lagi berlawanan dengan arah sebelumnya(jika sebelumnya vertikal maka dipanggil horizontal dan sebaliknya) pada fungsi solve yang saya buat 1 artinya mengecek secara horizontal sedangkan 0 vertikal. Fungsi akan berhenti backtrack saat ukuran buffer yang dibawa fungsi melebih buffer yang diizinkan.

Bagian 3: Source code C++

Program diimplementasikan dalam 1 file main.cpp pada folder src.

```
#include<bits/stdc++.h>
#include<fstream>
#include<ctime>
#include<chrono>
#include<sstream>
#include<string>
using namespace std;
using namespace std :: chrono;
vector<vector<string>> matrix;
vector<pair<vector<string>,int>> seq;
int token_size,seq_length;
vector<string> tokens;
vector<vector<int>> checked;
int buffer,m ,n,n_seq;
int optimum_reward=0,rew;
vector<string> buff1;
vector<pair<int,int>>path1;
vector<string> optimal_buff;
vector<pair<int,int>> optimal_path;
void solve(vector<string> buff,vector<pair<int,int>> path,int x,int y,int dir){
    if((int)buff.size()>buffer){
        return ;
    }
        int current_reward=0;
        for(int i=0;i<n_seq;i++){
            for(int  j=0;j<(int)buff.size()-(int)seq[i].first.size()+1;j++){
                vector<string> t =
{buff.begin()+j,buff.begin()+j+(int)seq[i].first.size()};
```

```
                if(t==seq[i].first){
                    current_reward+= seq[i].second;
                }
            }
        }
        if(current_reward>rew){
             rew= current_reward;
            buff1 = buff;
            path1 = path;
        }
    string token = matrix[y][x];
    if(dir==0){
        for(int i=0;i<n;i++){
            buff.push_back(token);
        path.push_back({x,y});
        checked[y][x]= 1;
            if(checked[i][x]==0){
            solve(buff,path,x,i,1);
            }
            checked[y][x]=0;
            buff.pop_back();
    path.pop_back();
            }
    }else{
        for(int i =0;i<m;i++){
            buff.push_back(token);
        path.push_back({x,y});
        checked[y][x]= 1;
            if(checked[y][i]==0){
                solve(buff,path,i,y,0);
            }
            checked[y][x]= 0;
            buff.pop_back();
    path.pop_back();
        }
    }
}
int main(){
    string choice;
    cout << "Masukan tipe input(cli/txt): "<<"\n";
    cin >> choice;
    if(choice=="cli"){
        cin >> token_size;
        srand(time(0));
        tokens.resize(token_size);
        for(int i=0;i<token_size;i++){
            cin >> tokens[i];
        }
        cin >> buffer;
        cin >> m >> n;
        checked.resize(n);
        matrix.resize(n);
        for(int i=0;i<n;i++){
```

```
            for(int j=0;j<m;j++){
                int random = rand()%token_size;
                matrix[i].push_back(tokens[random]);
                checked[i].push_back(0);
            }
        }
        cin >> n_seq;
        seq.resize(n_seq);
        cin >> seq_length;
        for(int i=0;i<n_seq;i++){
            int range= seq_length-1;
            int r = rand()% range + 2;
            while(r--){
                int random = rand()%token_size;
                seq[i].first.push_back(tokens[random]);
            }

            int reward = rand()%100+1;
            seq[i].second = reward;
        }
        auto start = high_resolution_clock::now();
        for(int i=0;i<m;i++){
            rew = 0;
            solve(buff1,path1,i,0,0);
            if(rew>optimum_reward){
                optimum_reward = rew;
                optimal_buff = buff1;
                optimal_path = path1;
            }
        }
        cout << "matriks permainan:\n";
        for(int i=0;i<n;i++){
            for(int j=0;j<m ;j++){
                cout << matrix[i][j]<< " ";
            }
            cout << "\n";
        }
        cout << "sequence:\n";
        for(int i=0;i<n_seq;i++){
            for(auto s: seq[i].first){
                cout << s<< " ";

            }
            cout << "\n"<<seq[i].second;
            cout <<"\n";
        }cout<<"\n";
        cout << optimum_reward<<"\n";
        if(optimal_buff.size()!=0){

        for(auto s: optimal_buff){
            cout << s<< " ";
        }cout << "\n";
        for(auto s : optimal_path){
```

```
            cout <<s.first+1<<", "<<s.second+1;
            cout << "\n";
        }
        }
        cout <<"\n\n";
        auto end = high_resolution_clock::now();
        auto duration = duration_cast<milliseconds>(end-start);
        cout << duration.count()<< " ms"<<"\n";
        cout<< "\n\n";
        cout << "apakah ingin menyimpan solusi? (y/n)";
        char c;
        cin >>c;
        if(c=='y'){
            string filepath;
            cout << "Masukan nama file output:(.txt) ";
            cin >> filepath;
            ofstream outputFile(filepath);
            outputFile << "matriks permainan:\n";
        for(int i=0;i<n;i++){
            for(int j=0;j<m ;j++){
                outputFile << matrix[i][j]<< " ";
            }
            outputFile << "\n";
        }
        outputFile << "sequence:\n";
        for(int i=0;i<n_seq;i++){
            for(auto s: seq[i].first){
                outputFile << s<< " ";
            }
            outputFile << "\n"<<seq[i].second;
            outputFile <<"\n";
        }outputFile<<"\n";
        outputFile << optimum_reward<<"\n";
        if(optimal_buff.size()!=0){
        for(auto s: optimal_buff){
            outputFile << s<< " ";
        }outputFile << "\n";
        for(auto s : optimal_path){
            outputFile <<s.first+1<<", "<<s.second+1;
            outputFile << "\n";
        }
        outputFile<<"\n\n"<< duration.count()<<" ms"<<"\n";
        outputFile.close();
        }
        }
        }else if(choice=="txt"){
            string inputPath ;
            cout << "Masukan alamat file: "<<"\n";
            cin >> inputPath;
            ifstream inp(inputPath);
            if(!inp){
                cerr << "Gagal membuka file" << endl;
                return 1;
```

```
        }
        string line;
        getline(inp,line);
        buffer = stoi(line);
        getline(inp,line);
        istringstream iss(line);
        iss >> m >> n;
        matrix.resize(n);
        checked.resize(n);
        for(int i=0;i<n;i++){
            getline(inp,line);
            istringstream iss(line);
            for(int j=0;j<m;j++){
                string token;
                iss >> token;
                if(token.size()!=2){
                    cerr << "panjang token harus 2"<<endl;
                    return 1;
                }
                matrix[i].push_back(token);
                checked[i].push_back(0);
                }
        }
        getline(inp,line);
        n_seq = stoi(line);
        seq.resize(n_seq);
        for(int i=0;i<n_seq;i++){
            getline(inp,line);
            string token ;
            stringstream ss(line);
            while(ss >> token){
                seq[i].first.push_back(token);
            }
            getline(inp,line);
            int reward;
            reward = stoi(line);
            seq[i].second = reward;
        }
        inp.close();
        auto start = high_resolution_clock::now();
        for(int i=0;i<m;i++){
        rew = 0;
        solve(buff1,path1,i,0,0);
        if(rew>optimum_reward){
            optimum_reward = rew;
            optimal_buff = buff1;
            optimal_path = path1;
        }

    }
    cout << optimum_reward<<"\n";
    cout << "hello"<<"\n";
    if(optimal_buff.size()!=0){
```

```
        for(auto s: optimal_buff){
            cout << s<< " ";
        }cout << "\n";
        for(auto s : optimal_path){
            cout <<s.first+1<<", "<<s.second+1;
            cout << "\n";
        }
        }
        cout <<"\n\n";
        auto end = high_resolution_clock::now();
        auto duration = duration_cast<milliseconds>(end-start);
        cout << duration.count()<< " ms"<<"\n";
        cout<< "\n\n";
        cout << "apakah ingin menyimpan solusi? (y/n)";
        char c;
        cin >>c;
        if(c=='y'){
            string filepath;
            cout << "Masukan nama file output:(.txt) ";
            cin >> filepath;
            ofstream outputFile(filepath);
        outputFile << optimum_reward<<"\n";
        if(optimal_buff.size()!=0){

        for(auto s: optimal_buff){
            outputFile << s<< " ";
        }outputFile << "\n";
        for(auto s : optimal_path){
            outputFile <<s.first+1<<", "<<s.second+1;
            outputFile << "\n";
        }
        outputFile<<"\n\n"<< duration.count()<<" ms"<<"\n";
        outputFile.close();
        }
        }
        }
}
```

Gambar 2. Source code program

Bagian 4: Uji Coba

A. Cara menjalankan program:

1. Compile dan run aplikasi sesuai yang ada di readme, atau boleh langsung run dgn ./bin/main.exe di windows dan ./bin/mainlinux.exe di linux
2. Program ini memiliki 2 opsi:
   a. Pengguna memasukkan masukan menggunakan file,
   b. Pengguna memasukkan masukan melalui *Command Line Interface*,
3. Untuk masukan menggunakan file, pengguna harus mengikuti format:

```
buffer_size
matrix_width matrix_height
matrix
number_of_sequences
sequences_1
sequences_1_reward
sequences_2
sequences_2_reward
…
sequences_n
sequences_n_reward
```

Sementara untuk masukkan melalui *CLI*, berupa jumlah_token_unik, token, ukuran_buffer, ukuran_matriks, jumlah_sekuens, dan ukuran_maksimal_sekuens.

Contoh:

```
5
BD 1C 7A 55 E9
7
6 6
3
4
```

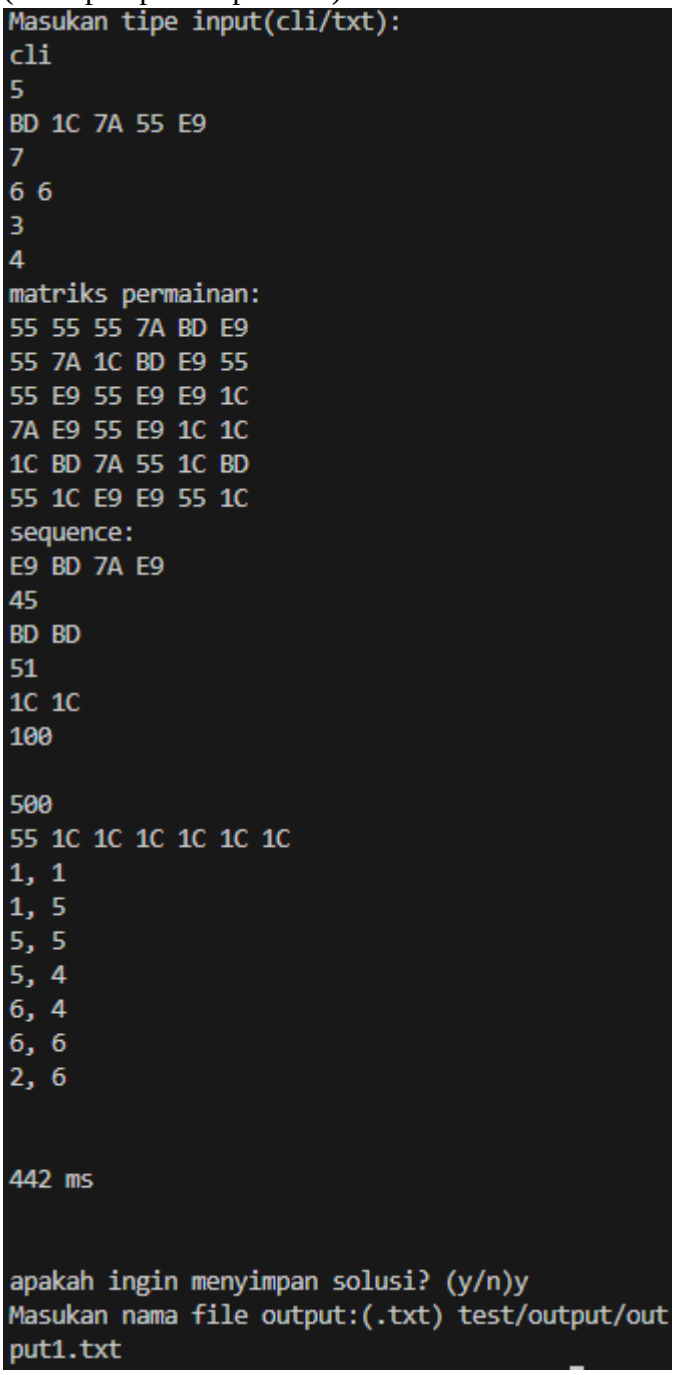
4. Saat berhasil memasukkan masukan ke dalam program, program akan memberikan solusi optimal yang kemudian dapat disimpan pada file (.txt) tergantung kemauan pengguna.

B. Uji coba program:

File hasil uji coba disimpan di folder test.

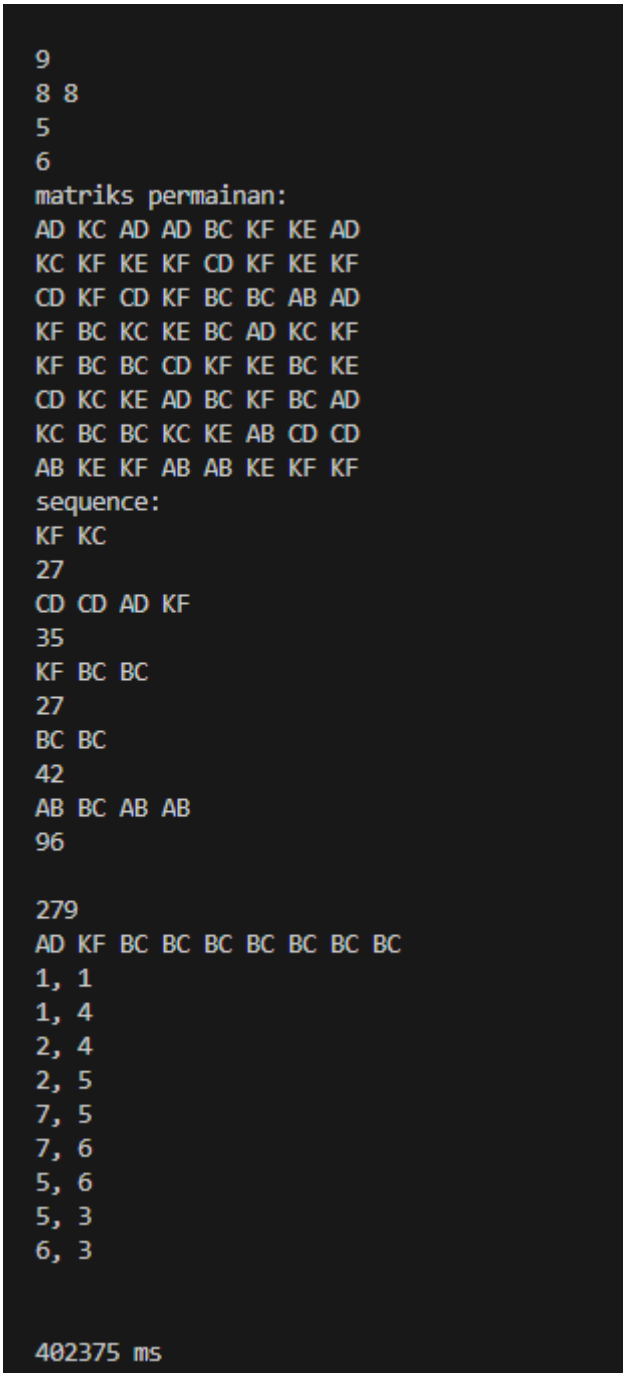
1. Opsi cli

   a.(disimpan pada input1.txt)



Gambar 3 Input dan
output dari masukan cli

b. disimpan pada input3.txt dan output3.txt

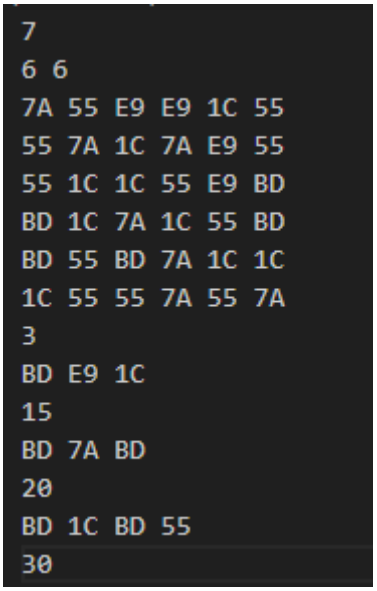


Gambar 4. Input dan output dari masukan 2 cli

2. opsi txt

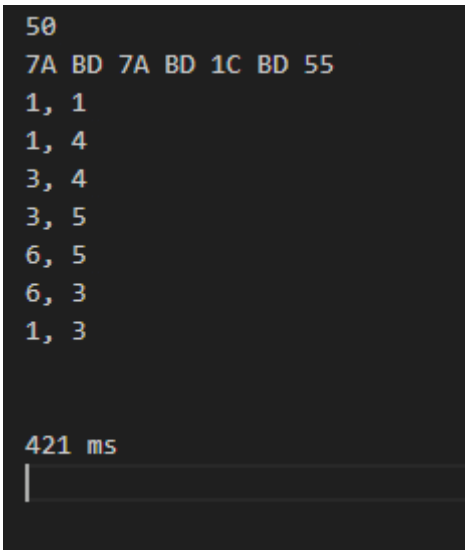
a.

Input : input2.txt



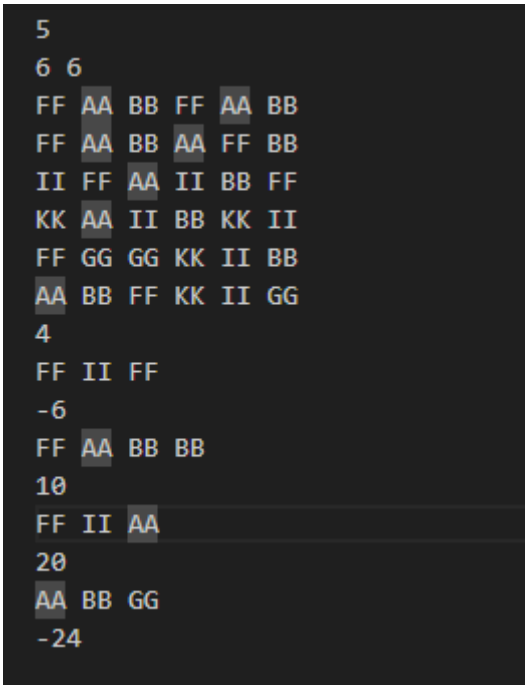
Gambar 5. Masukan sesuai tc dari spek

Output: output2.txt



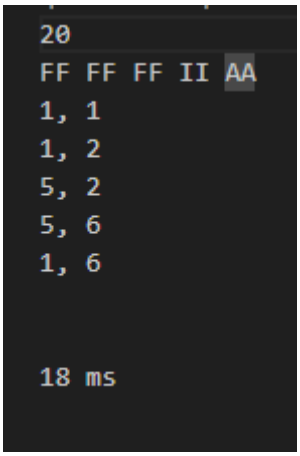
Gambar 6. Output

b. Input: input4.txt



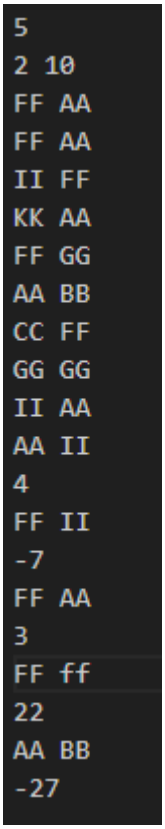
Gambar 7. Input4.txt

Output: output4.txt



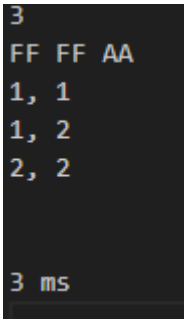
Gambar 8. Output4.txt

c. input: input5.txt



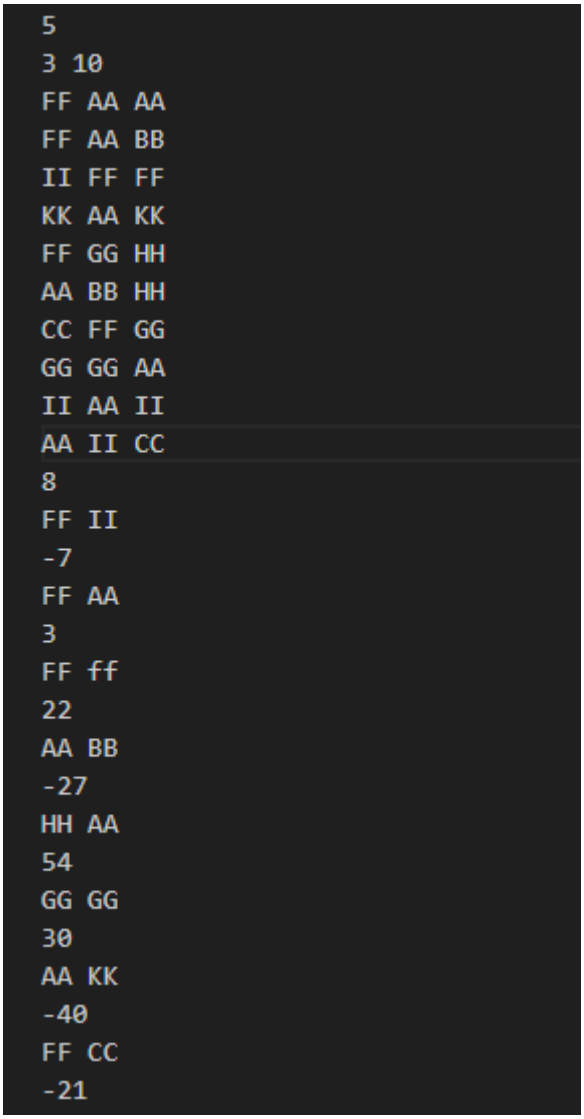
Gambar 9. Input5.txt

Output:output5.txt



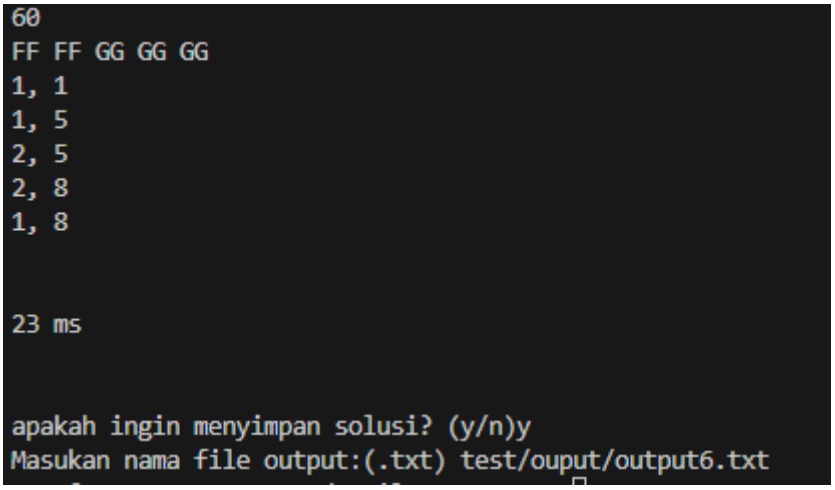
Gambar 10. Output5.txt

d. input: input6.txt



Gambar 10. Input6.txt

Output: terminal(karena typo pada path)



Gambar 11. Output pada termina

Lampiran

Github *Repository:*

https://github.com/mrsuiii/Tucil1_13522079

***Checklist*:**

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa kesalahan	**V**	
2. Program berhasil dijalankan	**V**	
3. Program dapat membaca masukan berkas .txt	**V**	
4. Program dapat menghasilkan masukan secara acak	**V**	
5. Solusi yang diberikan program optimal	**V**	
6. Program dapat menyimpan solusi dalam berkas .txt	**V**	
7. Program memiliki GUI		**V**