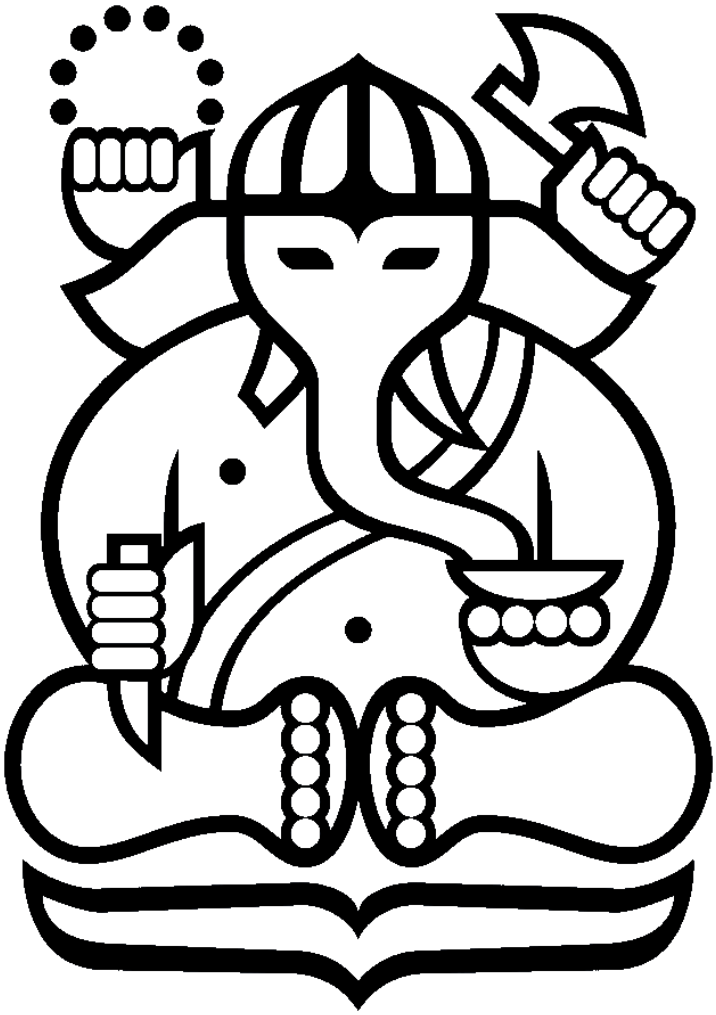
LAPORAN TUGAS KECIL I

PENYELESAIAN CYBERPUNK 2077 BREACH PROTOCOL DENGAN ALGORITMA BRUTE FORCE

IF2211 STRATEGI ALGORITMA



Rayendra Althaf Taraka Noor (13522107)

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

SEMESTER II TAHUN 2023/2024

Daftar Isi

BAB I

DESKRIPSI MASALAH

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.

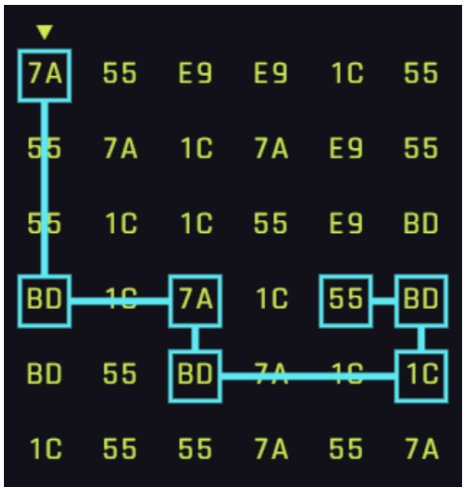
Aturan permainan Breach Protocol antara lain:
1. Pemain bergerak dengan pola horizontal, vertikal, horizontal, vertikal (bergantian) hingga semua sekuens berhasil dicocokkan atau buffer penuh.
2. Pemain memulai dengan memilih satu token pada posisi baris paling atas dari matriks.
3. Sekuens dicocokkan 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.

Ilustrasi kasus :
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 dengan hadiah berbobot 20.
3. BD 1C BD 55 dengan hadiah berbobot 30.

Maka solusi yang optimal untuk matriks dan sekuens yang diberikan adalah sebagai berikut:



Gambar 2 Contoh Solusi

(Sumber: https://cyberpunk-hacker.com/)

- Total bobot hadiah : 50 poin
- Total langkah : 6 langkah

Dalam tugas kecil kali ini, tugas kami, mahasiswa Informatika ’22, adalah untuk menemukan solusi dari **permainan Breach Protocol** yang paling optimal untuk setiap kombinasi matriks, sekuens, dan ukuran buffer dengan melalui pendekatan ***algoritma brute force***.

BAB II

TEORI SINGKAT

Brute force merupakan algoritma yang mudah dipahami karena pendekatannya yang *straightforward*. Ide dasar dari algoritma sederhana yakni dengan mencari dan menguji semua kemungkinan yang ada. Kelebihan pada algoritma ini ada pada kesederhanaannya dan kemampuannya untuk diterapkan dalam berbagai macam permasalahan. Namun, ia juga memiliki kekurangan yang cukup besar, yaitu memakan waktu dan memori yang banyak.

BAB III

STRATEGI PENYELESAIAN DAN KODE IMPLEMENTASI

Sebagaimana dengan pendekatan brute force lainnya, pada penyelesaian permasalahan ini akan dicoba semua kemungkinan rangkaian token yang bisa dibentuk. Untuk memudahkan pembuatan algoritma brute force, sebelumnya kita perlu menyimpan beberapa informasi penting yang sudah ada pada persoalan dengan merepresentasikannya dalam bentuk data yang tepat. Selain itu, diperlukan juga informasi tambahan yakni penanda untuk setiap elemen matriks terkait apakah elemen tersebut sudah ada dalam rangkaian token yang sudah pernah diambil pada rangkaian yang sedang dijelajahi. Berikutnya ketika data-data sudah siap, akan dilakukan langkah-langkah untuk menjelajahi setiap kemungkinan sambil menghitung nilai maksimum sementara sebagai berikut:

1. Menandai nilai maksimum sementara awal yakni 0 sebagai poin ketika tidak diambil token apapun
2. Mengambil salah satu titik pada baris pertama sebagai titik awal memasukannya ke dalam rangkaian sembari menandai bahwa elemen terus sudah diambil. Kemudian hitung poin total rangkaian terkini dengan memeriksa keberadaan sekuens-sekuens yang ada di dalamnya lalu bandingkan poin tersebut dengan poin maksimum sementara.
3. Selanjutnya bergerak dari titik pengambilan ke semua titik yang berada pada kolom yang sama dan belum dikunjungi (bergerak secara vertikal), ambil dan tandai titik tersebut. Setelah itu kembali cek poin sementara dan bandingkan dengan poin maksimum sementara.
4. Ulangi langkah ke-3 namun kali ini ke titik dengan baris yang sama (bergerak horizontal).
5. Ulangi terus langkah ke-3 dan ke-4 sampai tidak ada lagi titik yang belum diambil atau sampai panjang rangkaian memenuhi batas buffer.
6. Ketika sudah tidak ada lagi langkah yang bisa diambil, titik yang terakhir diambil akan dikeluarkan dari rangkaian dan dilanjutkan dengan titik selanjutnya. Langkah ini akan terus berulang hingga semua titik pada baris pertama sudah menjadi titik awal dan sudah dikeluarkan karena semua percobaan yang memungkinkan sudah dilakukan.
7. Ketika semua kemungkinan rangkaian sudah dicoba, poin maksimum sementara sudah menjadi poin maksimum akhir.

Dalam penerapannya, ide tersebut saya wujudkan menjadi sebuah program dalam bahasa c++, dengan lengkapnya sebagai berikut

1. Variable Global

```
typedef pair <char,char> token;
struct sequence{
    vector <token> tokSequence;
    int valSequence,lenSequence;
};

struct result{
    int valRes,lenRes;
    vector <token> tokRes;
    vector <pair <int,int> > posRes;
};

short int tipeInp;
string stringp,ans;
result curRes,finalRes;
int buffer, nSequence, row, col;
double execTime;
vector <vector <token> > dataToken;
vector <vector <bool> > visToken;
vector <sequence> dataSequence;

token stringToToken(string s){
    token res;
    if(s.length()!=2){
        cout<<"Masukan token tidak valid, program diberhentikan";
        exit(0);
    }
    res.first = s[0];
    res.second = s[1];
    return res;
}
```

2. Main

```
int main (){
    //inisialisasi
    ans="";
    finalRes.valRes=0;
    finalRes.lenRes=0;
    curRes.valRes=0;
    curRes.lenRes=0;

    //mengambil masukan
    getInpType();
    if(tipeInp==2){
        type2();
        displayToken();
        displaySequence();
    }
    else{
        type1();
    }

    //menghitung poin maksimal
    auto start = chrono::high_resolution_clock::now();
    for(int i=0;i<col;i++){
        generatePos(0,i,true,buffer-1);
    }
    auto stop = chrono::high_resolution_clock::now();
    auto duration = chrono::duration_cast<chrono::milliseconds>(stop-start);
    execTime = duration.count();
    finalResToOp();
    cout<<ans<<endl;
    cout<<"Apakah ingin menyimpan solusi? (y/n)";
    stringp="";
    while(stringp!="y"&&stringp!="n"){
        cin>>stringp;
        if(stringp!="y"&&stringp!="n"){
            cout<<"masukan tidak valid, silahkan ulangi"<<endl;
        }
    }
    if(stringp=="y"){
        cout<<"masukkan nama file tujuan: ";
        cin>>stringp;
        string p="test/output/";
        p+=stringp;
        p+=".txt";
        ofstream ofile(p);
        ofile<<ans;
        ofile.close();
    }
}
```

3. Menyusuri dan Menghitung Semua Kemungkinan

```
bool checkExist(int idxSeq){
    for(int i=0;i<curRes.lenRes;i++){
        if(dataSequence[idxSeq].lenSequence>curRes.lenRes-i) return 0;
        else{
            for(int j=0;j<=dataSequence[idxSeq].lenSequence;j++){
                if(j==dataSequence[idxSeq].lenSequence) return 1;
                if(dataSequence[idxSeq].tokSequence[j]!=curRes.tokRes[i+j]) break;
            }
        }
    }
    return 0;
}

void calculatePoint(){
    curRes.valRes=0;
    for(int i=0;i<nSequence;i++){
        curRes.valRes+=(checkExist(i)*dataSequence[i].valSequence);
    }
    if(curRes.valRes>finalRes.valRes||(curRes.valRes==finalRes.valRes && curRes.lenRes<finalRes.lenRes)){
        finalRes=curRes;
    }
}

void generatePos(int posR, int posC,bool vertikal,int reBuff){
    if(reBuff==-1) return;
    visToken[posR][posC]=true;
    curRes.lenRes+=1;
    curRes.tokRes.push_back(dataToken[posR][posC]);
    curRes.posRes.push_back(make_pair(posR,posC));
    calculatePoint();
    if(vertikal){
        for(int i=0;i<row;i++){
            if(visToken[i][posC]==false){
                generatePos(i,posC,false,reBuff-1);
            }
        }
    }
    else{
        for(int i=0;i<col;i++){
            if(visToken[posR][i]==false){
                generatePos(posR,i,true,reBuff-1);
            }
        }
    }
    visToken[posR][posC]=false;
    curRes.lenRes-=1;
    curRes.tokRes.pop_back();
    curRes.posRes.pop_back();
}
```

4. Input Tipe 1 (file.txt)

```
void type1(){
    cout<<"masukkan nama file yang ingin dibaca (tanpa menulis.txt): ";
    cin>>strinp;
    string p="test/input/";
    p+=strinp;
    p+=".txt";
    ifstream in(p);
    in>>buffer>>col>>row;
    dataToken.resize(row,vector<token> (col));
    visToken.resize(row,vector<bool> (col));
    for(int i=0;i<row;i++){
        for(int j=0;j<col;j++){
            in>>strinp;
            dataToken[i][j]=stringToToken(strinp);
            visToken[i][j]=false;
        }
    }
    in>>nSequence;
    dataSequence.resize(nSequence);
    strinp="";
    for(int i=0;i<nSequence;i++){
        getline(in,strinp);
        getline(in,strinp);
        separateToken(strinp,i);
        in>>dataSequence[i].valSequence;
    }
}
```

5. Tipe 2(CLI)

```
void type2(){
    srand((int)time(0));
    int nUniqueToken, maxSequenceLen;
    //ambil jumlah dan data token unik
    cout<<"masukkan jumlah token unik: ";
    cin>>nUniqueToken;
    token jenisToken[nUniqueToken];
    cout<<"masukkan token unik: "<<endl;
    for(int i=0;i<nUniqueToken;i++){
        cin>>strinp;
        jenisToken[i]=stringToToken(strinp);
    }
    //ambil masukan lain
    cout<<"masukkan ukuran buffer: ";
    cin>>buffer;
    cout<<"masukkan ukuran matriks: ";
    cin>>col>>row;
    cout<<"masukkan jumlah sekuens: ";
    cin>>nSequence;
    cout<<"masukkan panjang maksimum sekuens: ";
    cin>>maxSequenceLen;
    //bentuk matriks
    dataToken.resize(row,vector<token> (col));
    visToken.resize(row,vector<bool> (col));
    for(int i=0;i<row;i++){
        for(int j=0;j<col;j++){
            dataToken[i][j]=jenisToken[rand()%nUniqueToken];
            visToken[i][j]=false;
        }
    }
    //bentuk sequence
    dataSequence.resize(nSequence);
    for(int i=0;i<nSequence;i++){
        dataSequence[i].lenSequence = rand()%(maxSequenceLen-1)+1;
        for(int j=0;j<dataSequence[i].lenSequence;j++){
            dataSequence[i].tokSequence.push_back(jenisToken[rand()%nUniqueToken]);
        }
        dataSequence[i].valSequence=rand()%200-50;
    }
    cout<<endl;
}
```

6. Lainnya

```
token stringToToken(string s){
    token res;
    if(s.length()!=2){
        cout<<"Masukan token tidak valid, program diberhentikan";
        exit(0);
    }
    res.first = s[0];
    res.second = s[1];
    return res;
}

void finalResToOp(){
    ans+="Result:\n";
    ans+=to_string(finalRes.valRes);
    ans+="\n";
    for(int i=0;i<finalRes.LenRes;i++){
        ans+=finalRes.tokRes[i].first;
        ans+=finalRes.tokRes[i].second;
        if(i!=finalRes.LenRes-1) ans+=" ";
        else ans+="\n";
    }
    for(int i=0;i<finalRes.LenRes;i++){
        ans+=to_string(finalRes.posRes[i].second+1);
        ans+=", ";
        ans+=to_string(finalRes.posRes[i].first+1);
        ans+="\n";
    }
    ans+="\n";
    ans+=to_string(execTime);
    ans+=" ms\n";
}

void separateToken(string s,int idx){
    stringstream full(s);
    string word;
    dataSequence[idx].LenSequence=0;
    while(full>>word){
        dataSequence[idx].LenSequence++;
        dataSequence[idx].tokSequence.push_back(stringToToken(word));
    }
}
```

```
void getInpType(){
    tipeInp = -1;
    while(tipeInp == -1){
        cout<<"Pilih Tipe Masukan: \n 1.File txt \n 2.Command Line"<<endl;
        cin>>strinp;
        if(strinp == "1")tipeInp = 1;
        else{
            if(strinp == "2")tipeInp = 2;
            else{
                cout<<"Tipe masukan tidak valid silahkan masukkan kembali tipe masukan"<<endl;
            }
        }
    }
}

void displayToken(){
    ans+= "Dihasilkan Matriks:\n";
    for(int i=0;i<row;i++){
        for(int j=0;j<col;j++){
            ans+=dataToken[i][j].first;
            ans+=dataToken[i][j].second;
            if(j!=row-1) ans+=" ";
            else ans+="\n";
        }
    }
    ans+="\n";
}

void displaySequence(){
    ans+= "Dihasilkan ";
    ans+= to_string(nSequence);
    ans+= " Sequence:\n";
    for(int i=0;i<nSequence;i++){
        for(int j=0;j<dataSequence[i].LenSequence;j++){
            ans+=dataSequence[i].tokSequence[j].first;
            ans+=dataSequence[i].tokSequence[j].second;
            if(j!=dataSequence[i].LenSequence-1) ans+=" ";
            else ans+="\n";
        }
        ans+=to_string(dataSequence[i].valSequence);
        ans+="\n";
    }
    ans+="\n";
}
```

BAB IV
UJI KASUS

Kasus 1

```
Pilih Tipe Masukan:
 1.File txt
 2.Command Line
2
masukkan jumlah token unik: 1
masukkan token unik:
55
masukkan ukuran buffer: 7
masukkan ukuran matriks: 5 5
masukkan jumlah sekuens: 4
masukkan panjang maksimum sekuens: 5
```

Gambar 4.1.1 Input1

```
Dihasilkan Matriks:
55 55 55 55 55
55 55 55 55 55
55 55 55 55 55
55 55 55 55 55
55 55 55 55 55

Dihasilkan 4 Sequence:
55 55
55
55 55 55 55
-37
55 55
-16
55 55 55 55
117

Result:
119
55 55 55 55
1, 1
1, 2
2, 2
2, 1

9.000000 ms

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

Gambar 4.1.2 Output Kasus 1

Kasus 2

```
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
```

Gambar 4.2.1 Input Kasus 2

```
Pilih Tipe Masukan:
 1.File txt
 2.Command Line
1
masukkan nama file yang ingin dibaca (tanpa menulis.txt): data1
Result:
50
7A BD 7A BD 1C BD 55
1, 1
1, 4
3, 4
3, 5
6, 5
6, 3
1, 3

50.000000 ms
```

Gambar 4.2.2 Output Kasus 2

Kasus 3

```
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
-100
BD 7A BD
-100
BD 1C BD 55
-100
```

Gambar 4.3.1 Input Kasus 3

```
Pilih Tipe Masukan:
 1.File txt
 2.Command Line
1
masukkan nama file yang ingin dibaca (tanpa menulis.txt): data2
Result:
0

52.000000 ms
```

Gambar 4.3.2 Output Kasus 3

Kasus 4

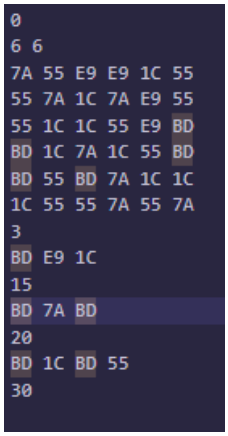
```
Pilih Tipe Masukan:
 1.File txt
 2.Command Line
2
masukkan jumlah token unik: 5
masukkan token unik:
AA BB CC DD EE
masukkan ukuran buffer: 8
masukkan ukuran matriks: 7 7
masukkan jumlah sekuens: 3
masukkan panjang maksimum sekuens: 4
```

Gambar 4.4.1 Input Kasus 4

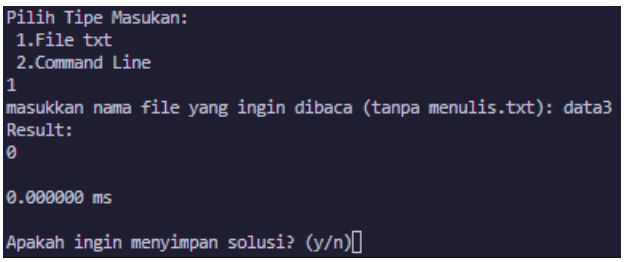
```
Dihasilkan Matriks:
BB DD AA DD DD BB BB
DD DD DD EE EE BB BB
CC DD AA DD CC EE CC
CC AA EE AA BB BB BB
BB BB DD BB EE BB AA
DD BB AA BB DD AA DD
DD AA DD DD CC AA DD

Dihasilkan 3 Sequence:
DD EE CC
71
EE BB
-46
DD BB
12

Result:
83
DD BB DD EE CC
2, 1
2, 5
3, 5
3, 4
1, 4

1303.000000 ms

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

Gambar 4.4.2 Output Kasus 4

Kasus 5

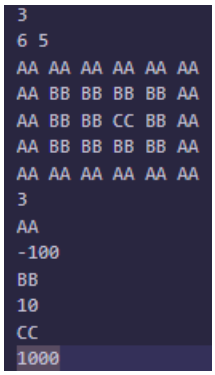


Gambar 4.5.1 Input Kasus 5

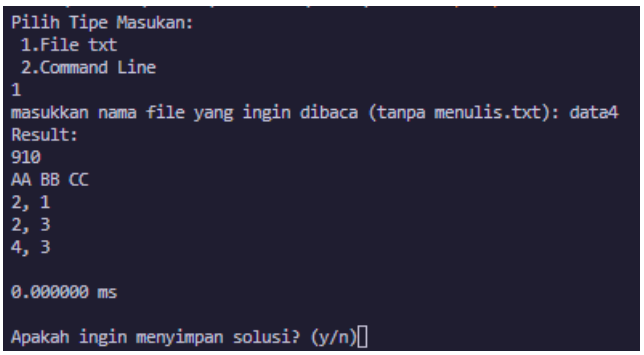


Gambar 4.5.2 Output Kasus 5

Kasus 6



Gambar 4.6.1 Input Kasus 6



Gambar 4.6.2 Output Kasus 6

LAMPIRAN

GitHub Repository: https://github.com/RayNoor0/Tucil1_13522107

Tabel 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