

LAPORAN TUGAS KECIL
STRATEGI ALGORITMA

IF 2211



Disusun oleh:

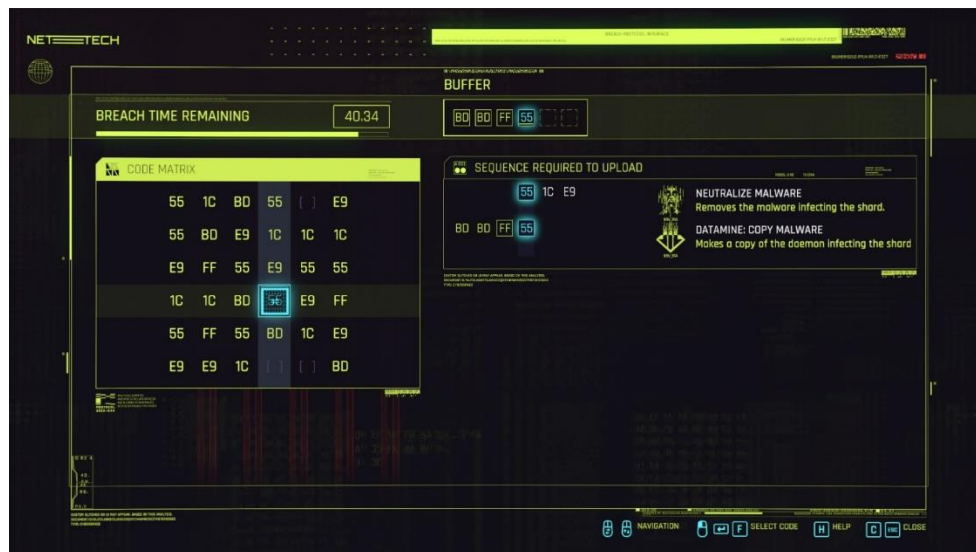
Raden Rafly Hanggaraksa Budiarto

13522014

I. ISI

II.	Deskripsi Masalah.....	3
III.	Landasan Teori.....	4
IV.	Analisis Pemecahan Masalah.....	5
	Source Code Program	5
V.	Hasil Pengujian	8
	Kasus 1.....	8
	Kasus 2.....	9
	Kasus 3.....	10
	Kasus 4.....	11
	Kasus 5.....	12
	Kasus 6.....	13
	Kasus 7.....	14
VI.	Kesimpulan	15
VII.	Lampiran	17

II. DESKRIPSI MASALAH



Gambar 1 Permainan Breach Protocol

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.

Dalam tugas ini, Anda diminta untuk menemukan solusi dari permainan Breach Protocol yang paling optimal untuk setiap kombinasi matriks, sekuens, dan ukuran buffer dengan menggunakan algoritma brute force.

III. LANDASAN TEORI

Algoritma Brute Force adalah pendekatan yang menguji setiap kemungkinan solusi secara berurutan dan sistematis, hingga ditemukan solusi yang tepat. Pendekatan ini sering digunakan dalam permasalahan optimasi atau pencarian, di mana diperlukan solusi terbaik dari sekumpulan kemungkinan solusi yang tersedia. Metode *brute force* umumnya diterapkan di berbagai bidang, termasuk pemrograman, matematika, dan ilmu komputer. Pendekatan ini terutama digunakan untuk masalah optimasi atau pencarian. Salah satu kelebihan algoritma brute force adalah kemudahan dalam implementasi dan pemahaman.

Contoh penyelesaian menggunakan pendekatan Brute Force antara lain mencari elemen terbesar atau terkecil, pencarian beruntun, menghitung factorial, menghitung perpangkatan, mengalikan dua buah matriks, menguji bilangan prima, pengurutan dengan selection sort dan bubble sort, dan lain-lain.

Algoritma *brute force* pada umumnya tidak “cerdas” dan tidak mangkus dikarenakan algoritma tersebut membutuhkan waktu dan volume komputasi yang besar dalam penyelesaiannya. Terkadang, algoritma brute force disebut dengan algoritma naif. Ini menyebabkan *brute force* lebih cocok untuk persoalan yang ukuran masukannya kecil. Algoritma brute force sering digunakan sebagai basis pembanding dengan algoritma lain yang lebih mangkus. Meskipun bukan merupakan metode penyelesaian masalah yang mangkus, hampir seluruh persoalan dapat diselesaikan melalui pendekatan ini.

Exhaustive Search merupakan teknik pencarian solusi brute force untuk persoalan kombinatorik seperti permutasi, kombinasi, atau himpunan bagian dari suatu himpunan. Teknik ini menghasilkan seluruh kemungkinan yang ada lalu mengevaluasi dan menyimpan solusi yang terbaik. Meskipun exhaustive search secara teoritis menghasilkan solusi, namun waktu atau sumberdaya yang dibutuhkan dalam pencarian solusinya sangat besar.

IV. ANALISIS PEMECAHAN MASALAH

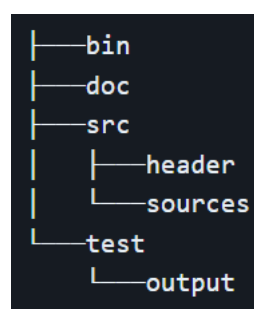
Dalam menyelesaikan permasalahan ini, digunakan pendekatan brute force dengan cara exhaustive search. Program membuat setiap kemungkinan buffer dari berukuran satu hingga ketentuan pengguna. Kombinasi diciptakan melalui serangkaian looping dan rekursi agar setiap langkah dapat dilalui sesuai dengan batasan tertentu. Base case dari rekursi tersebut adalah jika nilai reward yang diproses lebih besar dari nilai reward yang disimpan atau panjang buffer sudah sama dengan panjang maksimum buffer yang ditentukan oleh pengguna. Kompleksitas dalam menciptakan semua kemungkinan ialah $O(n^n)$.

Nilai reward secara aktif dihitung pada saat rekursi. Nilai buffer dicocokkan dengan sequence pada program dengan cara string matching pada tiap tokennya. Nilai reward akan diakumulasi dan dibandingkan dengan nilai reward sebelumnya. Kompleksitas menghitung nilai reward suatu buffer merupakan $O(n^2)$.

SOURCE CODE PROGRAM

Program ini dibuat menggunakan pendekatan brute force dalam menghasilkan setiap kombinasinya dan menggunakan bahasa C++ dengan algoritma exhaustive search. Pendekatan brute force berarti program secara sistematis mencoba semua kemungkinan solusi tanpa memperhatikan efisiensi atau optimasi. Dalam hal ini, program akan menguji setiap kombinasi secara berurutan hingga menemukan solusi yang sesuai. Algoritma exhaustive search memastikan bahwa tidak ada solusi yang terlewatkan, meskipun metodenya mungkin memakan waktu lebih lama. Dengan menggunakan bahasa C++, program ini dapat dijalankan dengan cepat dan efisien.

Program ini memiliki struktur sebagai berikut



Gambar 2 Struktur Projek

- “bin” merupakan folder berisi executable yang bisa dijalankan di Windows dan Linux.
- “doc” merupakan folder berisi laporan tugas kecil (dalam bentuk PDF).
- “src” merupakan folder berisi *source code*.

- “header” merupakan folder berisi header files.
- “sources” merupakan folder berisi file implementasi.
- “test” merupakan folder berisi *input* dan *output* testcase dalam bentuk “.txt”.
 - “output” merupakan folder berisi solusi jawaban dari data uji yang digunakan dalam laporan.

Algoritma brute force pada program ini terdapat dalam “algorithm.cpp” dan pada prosedur findOptimum, sementara file implementasi lainnya merupakan penyokong dari file tersebut.

```
void findOptimum(Matrix &dataMatrix,int &rewardMaks, std::vector<Langkah> &tLangkah, bool
&isHorizontal, std::vector<std::string> &buffer,int &bufferSize,std::vector<Sequence>
&dataSequence, int &idx, hasil &result){
    int tempReward = countReward(buffer,dataSequence);
    if((buffer.size() == bufferSize) || tempReward > rewardMaks){
        if (tempReward > rewardMaks){
            rewardMaks = tempReward;
            result.hasilLangkah = currentLangkah;
            result.hasilBuffer = buffer;
            result.reward = rewardMaks;
        }
    }
    else {
        if (isHorizontal){
            for (int ptr=0; ptr < dataMatrix.getWidth();ptr++){
                Langkah currentTilePtr;
                currentTilePtr.i = idx;
                currentTilePtr.j = ptr;

                if(isSafe(currentTilePtr,currentLangkah)){
                    bool check = !isHorizontal;
                    std::vector<Langkah> tempCurrentLangkah = currentLangkah;
                    std::vector<std::string> tempBuffer = buffer;
                    tempCurrentLangkah.push_back(currentTilePtr);

                    tempBuffer.push_back(dataMatrix.getValue(currentTilePtr.i,currentTilePtr.j));

                    findOptimum(dataMatrix,rewardMaks,tempCurrentLangkah,check,tempBuffer,bufferSize,dataSequence,
                    ptr,result);
                }
            }
        }
        else {
            for (int ptr=0; ptr < dataMatrix.getHeight();ptr++){
                Langkah currentTilePtr;
                currentTilePtr.i = ptr;
                currentTilePtr.j = idx;
```

```

        if(isSafe(currentTilePtr,currentLangkah) ){
            bool check = !isHorizontal;
            std::vector<Langkah> tempCurrentLangkah = currentLangkah;
            std::vector<std::string> tempBuffer = buffer;
            tempCurrentLangkah.push_back(currentTilePtr);

tempBuffer.push_back(dataMatrix.getValue(currentTilePtr.i,currentTilePtr.j));

findOptimum(dataMatrix,rewardMaks,tempCurrentLangkah,check,tempBuffer,bufferSize,dataSequence,
ptr,result);
        }
    }
}
}
}
}

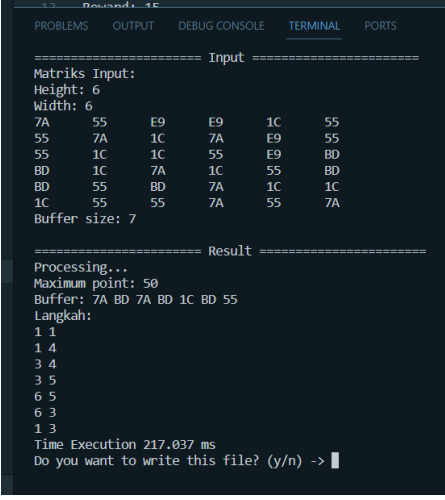
```

Untuk source code selengkapnya, silahkan rujuk [source code](#).

V. HASIL PENGUJIAN

KASUS 1

```
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
```



```
===== Input =====
Matrics Input:
Height: 6
Width: 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
Buffer size: 7

===== Result =====
Processing...
Maximum point: 50
Buffer: 7A BD 7A BD 1C BD 55
Langkah:
1 1
1 4
3 4
3 5
6 5
6 3
1 3
Time Execution 217.037 ms
Do you want to write this file? (y/n) -> |
```

Gambar 3 Test Case 1

KASUS 2

7

7 7

PG D4 PG 7C 7C 7C 7C

PG PG D4 7C 7C 1A 7C

PG D4 1A 1A 1A D4 D4

7C 7C D4 7C D4 7C 7C

D4 7C PG PG 7C 1A 1A

D4 1A PG 1A D4 D4 1A

D4 7C 7C 1A 7C PG 7C

6

D4 7C 1A

211

PG D4 7C

125

7C PG D4 PG

220

1A PG

272

1A 7C 1A

282

PG 1A 7C

253

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
===== Input =====
Matriks Input:
Height: 7
Width: 7
PG D4 PG 7C 7C 7C 7C
PG PG D4 7C 7C 1A 7C
PG D4 1A 1A 1A D4 D4
7C 7C D4 7C D4 7C 7C
D4 7C PG PG 7C 1A 1A
D4 1A PG 1A D4 D4 1A
D4 7C 7C 1A 7C PG 7C
Buffer size: 7

===== Result =====
Processing...
Maximum point: 1018
Buffer: D4 7C 1A PG 1A 7C 1A
Langkah:
2 1
2 5
6 5
6 7
4 7
4 2
6 2
Time Execution 1154.77 ms
Do you want to write this file? (y/n) -> |
```

Gambar 4 Test Case 2

KASUS 3

7

6 6

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 55 55 55 55 55

55 55 55 55 55 55

3

55 55 55 55

-100

55 55 55

15

55 55 55 55 55

30

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
===== Input =====
Matriks Input:
Height: 6
Width: 6
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 55 55 55 55 55
55 55 55 55 55 55
Buffer size: 7

===== Result =====
Processing...
Maximum point: 15
Buffer: 55 55 55
Langkah:
1 1
1 2
2 2
Time Execution 211.866 ms
Do you want to write this file? (y/n) -> |
```

Gambar 5 Test Case 3

KASUS 4

7

4 4

AB AB AB AB

AB AB AB AB

AB AB AB AB

AB AB AB AB

2

BB BB

10

CC CC

20

```
===== Input =====
Matriks Input:
Height: 4
Width: 4
AB      AB      AB      AB
AB      AB      AB      AB
AB      AB      AB      AB
AB      AB      AB      AB
Buffer size: 7

===== Result =====
Processing...
There is no solution
Time Execution 11.294 ms
Do you want to write this file? (y/n) -> █
```

Gambar 6 Test Case 4

KASUS 5

```
Token Amount (integer) -> 7
Token (ex. A1 BB) -> AB E2 1R T5 P2 TE ER
Buffer Size (integer) -> 9
Matrices Width (integer) -> 6
Matrices Height (integer) -> 8
Sequence Amount (integer) -> 5
Maximum Sequence Length (integer) -> 7
Maximum reward value (integer) -> 30
```

Gambar 7 Test Case 5 Input

```
===== Input =====
Matriks Input:
Height: 8
Width: 6
TE      P2      ER      E2      TE      P2
AB      P2      P2      AB      ER      1R
AB      TE      E2      TE      ER      ER
P2      1R      TE      1R      T5      P2
TE      TE      ER      1R      P2      1R
AB      E2      AB      E2      TE      TE
P2      ER      TE      T5      TE      1R
AB      TE      T5      AB      E2      AB
Buffer size: 9

===== Result =====
Processing...
Maximum point: 28
Buffer: TE AB ER ER TE TE AB T5
Langkah:
1 1
1 2
5 2
5 3
2 3
2 5
1 5
1 8
3 8
Time Execution 29457.1 ms
Do you want to write this file? (y/n) -> 
```

Gambar 8 Test Case 5 Output

KASUS 6

```
Token Amount (integer) -> 5
Token (ex. A1 BB) -> 2W E4 T2 PO M7
Buffer Size (integer) -> 6
Matrices Width (integer) -> 5
Matrices Height (integer) -> 6
Sequence Amount (integer) -> 4
Maximum Sequence Length (integer) -> 8
Maximum reward value (integer) -> 40
```

Gambar 9 Input Test Case 6

```
===== Input =====
Matriks Input:
Height: 6
Width: 5
M7      T2      2W      2W      M7
E4      2W      2W      E4      PO
M7      T2      PO      T2      M7
PO      T2      PO      2W      M7
2W      2W      PO      2W      T2
2W      E4      M7      T2      M7
Buffer size: 6

===== Result =====
Processing...
Maximum point: 20
Buffer: M7 E4 PO M7 T2
Langkah:
1 1
1 2
5 2
5 1
2 1
Time Execution 25.493 ms
Do you want to write this file? (y/n) ->
```

Gambar 10 Output Test Case 6

KASUS 7

```
Token Amount (integer) -> 7
Token (ex. A1 BB) -> AB E2 T8 P1 B5 U2 E4
Buffer Size (integer) -> 8
Matrices Width (integer) -> 6
Matrices Height (integer) -> 4
Sequence Amount (integer) -> 7
Maximum Sequence Length (integer) -> 6
Maximum reward value (integer) -> 30
```

Gambar 11 Input Test Case 7

```
===== Input =====
Matriks Input:
Height: 4
Width: 6
E2      AB      T8      E4      AB      U2
U2      T8      E4      E2      B5      E2
E2      T8      P1      AB      E4      T8
T8      U2      E4      U2      E4      U2
Buffer size: 8

===== Result =====
Processing...
Maximum point: 7
Buffer: E2 E2 T8 T8
Langkah:
1 1
1 3
2 3
2 2
Time Execution 210.365 ms
Do you want to write this file? (y/n) -> 
```

Gambar 12 Output Input Test Case 7

VI. KESIMPULAN

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.

Algoritma Brute Force adalah pendekatan yang menguji setiap kemungkinan solusi secara berurutan dan sistematis, hingga ditemukan solusi yang tepat. Pendekatan ini sering digunakan dalam permasalahan optimasi atau pencarian, di mana diperlukan solusi terbaik dari sekumpulan kemungkinan solusi yang tersedia.

Dalam menyelesaikan permasalahan ini, digunakan pendekatan brute force dengan cara exhaustive search. Program membuat setiap kemungkinan buffer dari berukuran satu hingga ketentuan pengguna. Kombinasi diciptakan melalui serangkaian looping dan rekursi agar setiap langkah dapat dilalui sesuai dengan batasan tertentu. Base case dari rekursi tersebut adalah jika nilai reward yang diproses lebih besar dari nilai reward yang disimpan atau panjang buffer sudah sama dengan panjang maksimum buffer yang ditentukan oleh pengguna. Dalam menyelesaikan permasalahan ini, digunakan pendekatan brute force dengan cara exhaustive search.

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
-------------------------	--	---

VII. LAMPIRAN

<https://github.com/raflyhanga/tugas-kecil-stima>