

# **LAPORAN TUGAS KECIL 1**

## **IF2211 STRATEGIALGORITMA**

Penyelesaian Permainan Cyberpunk 2077 Breach Protocol dengan Algoritma Brute Force



Disusun oleh:

Mohammad Nugraha Eka Prawira (13522001)

**Program Studi Teknik Informatika**  
**Sekolah Teknik Elektro dan Informatika**  
**Institut Teknologi Bandung**

**2024**

# **BAB I**

## **LATAR BELAKANG**

### **Deskripsi Tugas :**

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

### **Spesifikasi Tugas Kecil 1 :**

Buatlah program sederhana dalam Bahasa C/C#/C++/Java/Python/JavaScript/Go/PHP yang mengimplementasikan algoritma BruteForce untuk mencari Solusi paling optimal permainan BreachProtocol.

### **Algoritma BruteForce**

Algoritma BruteForce sendiri adalah suatu pendekatan penyelesaian masalah yang sederhana dan langsung, di mana sistem mencoba semua solusi mungkin secara berurutan hingga menemukan solusi yang benar atau optimal. Pendekatan ini tidak menggunakan strategi atau peningkatan kecerdasan tertentu untuk mempercepat pencarian, tetapi secara sistematis memeriksa semua kemungkinan jawaban.

Pendekatan brute force cenderung mencakup beberapa karakteristik umum:

1. Eksploratif Penuh:

Brute force memeriksa semua solusi yang mungkin, tanpa meninggalkan celah atau melewatkan kemungkinan solusi yang sah. Ini termasuk pengujian setiap kombinasi yang mungkin, memeriksa semua jalur yang dapat diambil, atau mencoba semua nilai yang mungkin.

2. Kesederhanaan dan Keterbacaan:

Algoritma brute force sering kali relatif sederhana dan mudah dimengerti. Mekanisme pengujian yang langsung membuatnya mudah diimplementasikan.

3. Performa yang Kurang Efisien:

Meskipun sederhana dan jelas, algoritma brute force dapat kurang efisien dalam hal waktu dan sumber daya. Terutama pada masalah dengan ruang solusi yang besar, karena mencoba setiap kombinasi mungkin memerlukan waktu yang sangat lama.

4. Penggunaan pada Kasus Kecil:

Brute force dapat efektif pada masalah kecil atau ketika jumlah solusi yang mungkin terbatas.

Kelemahan utama dari algoritma brute force adalah kinerjanya yang kurang efisien pada masalah yang lebih besar. Oleh karena itu, pada beberapa kasus, diperlukan pendekatan algoritmik yang lebih canggih untuk mengatasi masalah dengan waktu eksekusi yang lebih cepat.

Pada persoalan kali ini kita mengimplementasikan algoritma bruteforce dalam penyelesaian Permainan Breach Protocol CyberPunk 2077. Algoritma ini digunakan untuk mencoba mencari Solusi paling optimal satu persatu. Langkah-Langkah yang dilakukan dalam implementasi yang digunakan berupa :

### Algoritma solvePuzzle

- Inisialisasi Variabel:  
Inisialisasi variabel seperti indeks, counters, point, dan buffer sementara.
- Inisialisasi Counters dan maxScore:  
Inisialisasi counters dan menghitung maxScore dari seluruh reward sequence.
- Inisialisasi tempBuffer:  
Menetapkan panjang dan ukuran buffer sementara sesuai dengan ukuran buffer input.
- Iterasi dengan Brute Force:  
Melakukan iterasi dengan brute force selama counters dalam batas matriks dan skor final belum mencapai maksimum.
- Penanganan Kasus Base:  
Jika ukuran buffer 0, set finalScore menjadi 0 dan selesaikan fungsi.
- Penanganan Level Teratas:  
Jika berada pada level teratas, tambahkan koordinat pada counters[0] ke tempBuffer dan tingkatkan index.
- Penanganan Level Tengah dan Akhir:  
Navigasi dan rekursi sesuai dengan aturan tertentu untuk level tengah atau akhir.
- Pengecekan dan Pembaruan Final Score:  
Cek skor sementara, perbarui finalScore dan buffer jika ditemukan solusi yang lebih baik.
- Inkrementasi Counters atau Backtracking:

Inkrementasi counters atau lakukan backtracking tergantung pada kondisi yang berlaku.

- Hasil Akhir:  
Setelah proses brute force selesai, perbarui finalScore dan buffer dengan solusi terbaik yang ditemukan.

Fungsi ini secara rekursif mencari solusi dengan mengeksplorasi semua kemungkinan kombinasi koordinat buffer pada matriks, dengan tujuan mencapai skor optimal.

## BAB II

## SOURCE CODE PROGRAM

Program ini dibuat dalam bahasa C dengan menggunakan library:

```

2 // main.c
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <time.h>
5 #include <stdbool.h>
6 #include <math.h>
7 #include "algorithm.h"
8
9 int main()
10 {
11     clock_t t;
12     int bufferSize;
13     int matrixWidth, matrixHeight;
14     Matrix matrixs;
15     int numSequences;
16     int totalToken;
17     int buffers;
18     int sequenceSize;
19     sequence seqnums[numSequences];
20     int totalscore = 0;
21     Buffer buffer;
22     char answer;
23
24     printf("Cyberpunk 2077 Breach Protocol\n");
25     printf("\n\nPILIH MODE PERMAINAN :\n\n");
26     printf("1. Inputan text.\n");
27     printf("2. Inputan Remain.\n");
28
29     printf("Input: ");
30     scanf("%c", &answer);
31     while (answer != '1' && answer != '2')
32     {
33         printf("Hanya ada dua pilihan.\n");
34         scanf("Input: %c", &answer);
35     }
36
37     if (answer == '1')
38     {
39         FILE *file = fopen("input.txt", "r");
40         if (file == NULL) {
41             printf("Gagal membuka berkas\n");
42             return 1;
43         }
44         // Baca buffer size
45         fscanf(file, "%d", &bufferSize);
46         // Baca jumlah kolom dan baris
47         fscanf(file, "%d %d", &matrixWidth, &matrixHeight);
48         // Baca Matrix
49         readMatrix(file, &matrixs, matrixWidth, matrixHeight);
50         printf("Matrix :\n");
51         displayMatrix(matrixs, matrixHeight, matrixWidth);
52
53         // baca sequence
54         fscanf(file, "%d", &numSequences);
55
56         for (int i = 0; i < numSequences; ++i) {
57             int long=0;
58             readSequences(file, &seqnums[i], &long);
59             int reward;
60             fscanf(file, "%d", &reward);
61             seqnums[i].reward=reward;
62             printf("Sequences %d:\n", i + 1);
63             for (int j = 0; j < long; ++j) {
64                 printf("%X ", seqnums[i].seqs[j]);
65             }
66             printf("\nReward: %d\n", reward);
67         }
68         qsort(seqnums, numSequences, sizeof(sequence), compareRewards);
69         buffer.lengths = 0;
70         buffer.bufferSize = bufferSize;
71         t = clock(); // Temukan sekuens optimal
72         solvePuzzle(matrixs, seqnums, numSequences, &buffer, &totalscore);
73
74         // Cetak hasil optimal
75         printf("\n\nSkor optimal: %d\n", totalscore);
76         if (totalscore == 0) {
77             printf("Tidak ada solusi\n\n");
78         } else {
79             printf("Buffer: "); displayBuffMat(buffer, matrixs); printf("\n");
80             printf("Lokasi pada matriks:\n"); displayBuff(buffer); printf("\n");
81         }
82         t = clock() - t;
83         int totaltime = (int)((double)(t / CLOCKS_PER_SEC)*1000);
84         printf("%d ms \n", totaltime);
85         saveText( buffer, totaltime, totalscore, matrixs);
86     }
87     else{
88         clock_t t;
89         int matrixWidth, matrixHeight;
90         Matrix matrixs;
91         int numSequences;
92         int totalToken;
93         int buffers;
94         int sequenceSize;
95         int totalscore = 0;
96         Buffer buffer;
97
98         scanf("%d", &totalToken);
99         int token[totalToken];
100         for (int i = 0; i < totalToken; i++)
101         {
102             scanf("%X", &token[i]);
103         }
104         scanf("%d", &bufferSize);
105         scanf("%d %d", &matrixWidth, &matrixHeight);
106         scanf("%d", &numSequences);
107         sequence seqnums[numSequences];
108         scanf("%d", &sequenceSize);
109         buildMatrix(&matrixs, matrixHeight, matrixWidth, totalToken);
110         displayMatrix(matrixs, matrixHeight, matrixWidth);
111         for (int i = 0; i < numSequences; i++)
112         {
113             buildSequence(&seqnums[i], sequenceSize, token, totalToken);
114         }
115         for (int i = 0; i < numSequences; i++)
116         {
117             printf("Sequences %d:\n", i + 1);
118             for (int j = 0; j < (seqnums[i].length); j++)
119             {
120                 printf("%X ", seqnums[i].seqs[j]);
121             }
122             printf("\nReward: %d\n", seqnums[i].reward);
123         }
124         qsort(seqnums, numSequences, sizeof(sequence), compareRewards);
125         buffer.lengths = 0;
126         buffer.bufferSize = buffers;
127         t = clock(); // Temukan sekuens optimal
128         solvePuzzle(matrixs, seqnums, numSequences, &buffer, &totalscore);
129
130         // Cetak hasil optimal
131         printf("\n\nSkor optimal: %d\n", totalscore);
132         if (totalscore == 0) {
133             printf("Tidak ada solusi\n\n");
134         } else {
135             printf("Buffer: "); displayBuffMat(buffer, matrixs); printf("\n");
136             printf("Lokasi pada matriks:\n"); displayBuff(buffer); printf("\n");
137         }
138         t = clock() - t;
139         int totaltime = (int)((double)(t / CLOCKS_PER_SEC)*1000);
140         printf("%d ms \n", totaltime);
141         saveText( buffer, totaltime, totalscore, matrixs);
142     }
143
144     return 0;
145 }

```

```

2 - algorithm.h

1  #ifndef PUZZLE_SOLVER_H
2  #define PUZZLE_SOLVER_H
3
4  #include <stdio.h>
5  #include <stdlib.h>
6  #include <string.h>
7  #include <time.h>
8  #include <stdbool.h>
9  #include <math.h>
10
11 #define MAXS 100
12 #define ELMT(M, i, j) (M).mem[(i)][(j)]
13
14 typedef struct
15 {
16     int seqs[MAXS];
17     int lengs;
18     int rewards;
19 } sequence;
20
21
22 typedef struct
23 {
24     int x;
25     int y;
26 } Coordinate;
27
28 typedef struct
29 {
30     int mem[MAXS][MAXS];
31     int height;
32     int width;
33 } Matrix;
34
35 typedef struct
36 {
37     int lengths;
38     int buffSize;
39     Coordinate container[MAXS];
40 } Buffer;
41
42 int compareRewards(const void *a, const void *b);
43
44 void readMatrix(FILE *file, Matrix *matrix, int width, int height);
45
46 void readSequences(FILE *file, sequence *sequences, int *leng);
47
48 void displayMatrix(Matrix matrix, int row, int col);
49
50 bool sameCoordinateToken(int token, Matrix m, Coordinate p);
51
52 bool isValidBuffer(Buffer b, Matrix m, sequence s);
53
54 int bufferScore(Buffer b, Matrix m, sequence *l, int numSequences);
55
56
57 void createCoordinate(Coordinate *coord, int a, int b);
58
59 void push(Buffer *b, Coordinate p);
60
61 void pop(Buffer *b);
62
63 bool isValidPoint(Buffer b, int x, int y);
64
65 void solvePuzzle(Matrix matrix, sequence *sequences, int numSequences, Buffer *buffer, int *finalScore);
66
67 void displayCoordinate(Coordinate coord);
68
69 void displayBuff(Buffer b);
70
71 void displayBuffMat(Buffer b, Matrix m);
72
73 void saveText(Buffer buffer, int t, int totalscore, Matrix matrixs);
74
75 unsigned int generateSeed();
76
77 // Function to build a random matrix
78 void buildMatrix(Matrix *matrixs, int row, int col, int *token, int total);
79
80 // Function to build a random sequence
81 void buildSequence(sequence *sequences, int maxSeq, int *token, int total);
82
83
84 #endif // PUZZLE_SOLVER_H
85

```

## BAB III

### SCREENSHOT HASIL TEST

#### 1. Test Case I

Masukan diterima melalui teks.

```
PROBLEMS TERMINAL DEBUG CONSOLE PORTS OUTPUT
PS C:\ITB\SEM 4\STIPA\2> gcc -o test main.c algorithm.c
PS C:\ITB\SEM 4\STIPA\2> ./test
Cyberpunk 2077 Breach Protocol

PILIH MODE PERMAINAN :
1. Inputan text.
2. Inputan Pemain.
Input: 1
Matrix :
7A 55 E9 E9 1C 55
55 7A 1C 7A E9 55
55 1C 1C 55 E9 8D
8D 1C 7A 1C 55 8D
8D 55 8D 7A 1C 1C
1C 55 55 7A 55 7A
Sequences 1:
8D E9 1C
Reward: 15
Sequences 2:
8D 7A 8D
Reward: 20
Sequences 3:
8D 1C 8D 55
Reward: 30

Skor optimal: 50
Buffer: 7A 8D 7A 8D 1C 8D 55
Lokasi pada matriks:
1, 1
1, 4
3, 4
3, 5
6, 5
6, 3
1, 3

4365 ms
Apakah ingin menyimpan solusi? (y/n): y
Masukkan nama berkas (.txt): 1
Hasil disimpan ke dalam berkas 1
PS C:\ITB\SEM 4\STIPA\2>
```

#### 2. Test Case II

Masukan diterima melalui teks

```
PS C:\ITB\SEM 4\STIPA\2> ./test
Cyberpunk 2077 Breach Protocol

PILIH MODE PERMAINAN :
1. Inputan text.
2. Inputan Pemain.
Input: 1
Matrix :
44 22 44 44
22 44 22 44
33 33 44 33
22 33 44 22
Sequences 1:
22 11 22
Reward: 10
Sequences 2:
44 33 22
Reward: 35
Sequences 3:
22 11
Reward: 40

Skor optimal: 35
Buffer: 44 44 33 22
Lokasi pada matriks:
3, 1
3, 3
1, 3
1, 2

9 ms
Apakah ingin menyimpan solusi? (y/n): y
Masukkan nama berkas (.txt): 2.txt
Hasil disimpan ke dalam berkas 2.txt
```





### 3. Test Case III

Masukan diterima melalui berkas.

```
Cyberpunk 2077 Breach Protocol

PILIH MODE PERMAINAN :
1. Inputan text.
2. Inputan Pemain.
Input: 1
Matrix :
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
Sequences 1:
55 55 55 55
Reward: -100
Sequences 2:
55 55 55
Reward: 15
Sequences 3:
55 55 55 55 55
Reward: 30

Skor optimal: 15
Buffer: 55 55 55
Lokasi pada matriks:
1, 1
1, 2
2, 2

4386 ms
Apakah ingin menyimpan solusi? (y/n): y
Masukkan nama berkas (.txt): 3.txt
Hasil disimpan ke dalam berkas 3.txt
```

### 4. Test Case IV

Masukan memiliki *reward* diterima melalui *keyboard*.

```
Cyberpunk 2077 Breach Protocol

PILIH MODE PERMAINAN :
1. Inputan text.
2. Inputan Pemain.
Input: 2
3
AB AC AD
4
6 6
3
4
AD AB AC AD AC AB
AC AD AC AB AB AD
AB AC AC AB AC AC
AC AB AC AB AC AD
AD AB AB AB AB AC
AD AC AC AD AD AD
Sequences 1:
AD AD
Reward: 50
Sequences 2:
AD AC AB AD
Reward: 5
Sequences 3:
AB AC AC
Reward: 98

Skor optimal: 98
Buffer: AD AB AC AC
Lokasi pada matriks:
1, 1
1, 3
2, 3
2, 6

43 ms
Apakah ingin menyimpan solusi? (y/n): Y
Masukkan nama berkas (.txt): 4.txt
Hasil disimpan ke dalam berkas 4.txt
```



## 5. Test Case V

Masukan diterima melalui keyboard.

```
Cyberpunk 2077 Breach Protocol

PILIH MODE PERMAINAN :
1. Inputan text.
2. Inputan Pemain.
Input: 2
3
AC AB AD
5
5 5
3
4
AC AB AD AB AD
AC AC AC AB AC
AC AB AD AC AC
AC AB AB AC AB
AC AC AC AB AD
Sequences 1:
AB AB AD
Reward: 4
Sequences 2:
AD AD AB
Reward: 57
Sequences 3:
AD AB AB AC
Reward: 15

Skor optimal: 72
Buffer: AD AD AB AB AC
Lokasi pada matriks:
3, 1
3, 3
2, 3
2, 1
1, 1

72 ms
Apakah ingin menyimpan solusi? (y/n): Y
Masukkan nama berkas (.txt): 5.txt
Hasil disimpan ke dalam berkas 5.txt
```

## 6. Test Case VI

Masukan diterima melalui *keyboard*.

```
PS C:\ITB\SEM 4\SIIMA\>> ./test
Cyberpunk 2077 Breach Protocol

PILIH MODE PERMAINAN :
1. Inputan text.
2. Inputan Pemain.
Input: 2
3
7A 8D AC
5
2
4
4
4
8D 7A
8D 8D
8D AC
7A 7A
Sequences 1:
8D 7A 7A 8D
Reward: 92
Sequences 2:
7A 8D 7A
Reward: 98
Sequences 3:
7A 8D
Reward: 91
Sequences 4:
AC 7A AC AC
Reward: 56

Skor optimal: 183
Buffer: 8D 7A 7A 8D
Lokasi pada matriks:
1, 1
1, 4
2, 4
2, 2

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



## LINK REPOSITORY

Pranala ke *repository*: <https://github.com/EkaaPrawiraa/Tucil-Stima>

## LAMPIRAN

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